Reach Set Approximation through Decomposition with Low-dimensional Sets and High-dimensional Matrices

Sergiy Bogomolov

Marcelo Forets
Goran Frehse
Frédéric Viry

Andreas Podelski
Christian Schilling

Australian National University

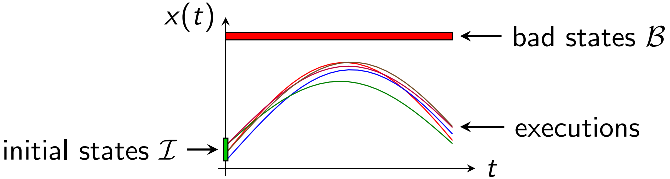
Université Grenoble-Alpes

University of Freiburg

HSCC 2018 — April 11, 2018 — Porto, Portugal

Linear time-invariant (LTI) systems

$$\dot{x}(t) \;=\; A \cdot x(t) \;+\; \underbrace{C \cdot u(t)}_{\text{nondeterministic inputs}}, \qquad u(t) \in \mathcal{U}$$



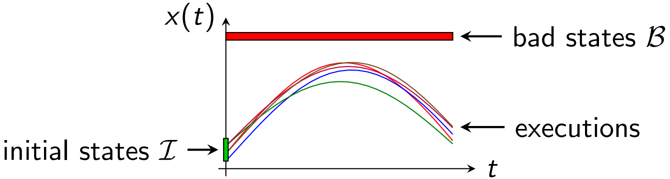
Task (Safety verification)

Verify that **no execution** leads to a bad state

Linear time-invariant (LTI) systems

$$\dot{x}(t) \;=\; A \cdot x(t) \;+\; \cancel{C \cdot u(t), \quad u(t) \in \mathcal{U}}$$

nondeterministic inputs omitted for now



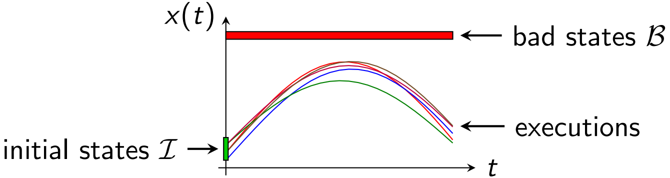
Task (Safety verification)

Verify that **no execution** leads to a bad state

Safety verification

Task (Safety verification)

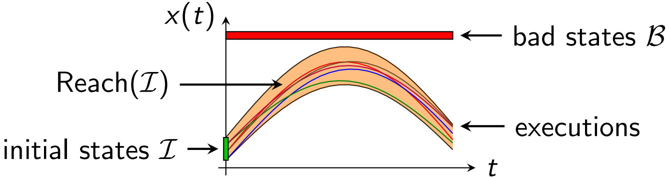
Verify that **no execution** leads to a bad state



Task (Safety verification)

Verify that **no execution** leads to a bad state

$\hat{=}$ Reach($\mathcal{I}$) $\cap\ \mathcal{B} = \emptyset$



Safety verification

Task (Safety verification)

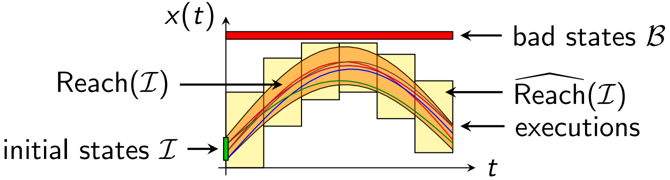
Verify that **no execution** leads to a bad state

$\hat{=}$ $\mathrm{Reach}(\mathcal{I}) \cap \mathcal{B} = \emptyset$

- Undecidable
- Showing $\widehat{\mathrm{Reach}}(\mathcal{I}) \cap \mathcal{B} = \emptyset$ is sufficient

  $\uparrow$

  overapproximation of $\mathrm{Reach}(\mathcal{I})$



Scalability

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

- Determines node voltage and branch currents in a circuit
- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

Scalability

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

- Determines node voltage and branch currents in a circuit
- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

State-of-the-art tool SpaceEx does not scale to such systems

"max. number of allocatable variables exceeded"

Scalability

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

- Determines node voltage and branch currents in a circuit
- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

State-of-the-art tool SpaceEx does not scale to such systems

"max. number of allocatable variables exceeded"

Task (Scalability)

Find a **sweet spot** between **precision** and **speed**

Decomposition

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

Decomposition use case

- Large systems

Decomposition

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

Decomposition use case

- Large systems

Observation

Property only depends on **two dimensions**

Decomposition

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

Decomposition use case

- Large systems

Observation

Property only depends on **two dimensions**

- Can we just look at x_1 and x_2?

Decomposition

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

Decomposition use case

- Large systems

Observation

Property only depends on **two dimensions**

- Can we just look at x_1 and x_2?
  No, all dimensions are **coupled**

Decomposition

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

Decomposition use cases

- Large systems
- "Sparse" properties

Decomposition

Example: MNA5

10,913-dimensional Modified Nodal Analysis model

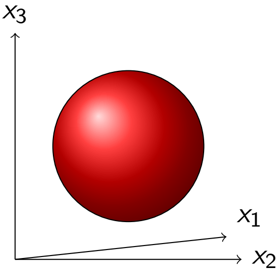
- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

Decomposition use cases

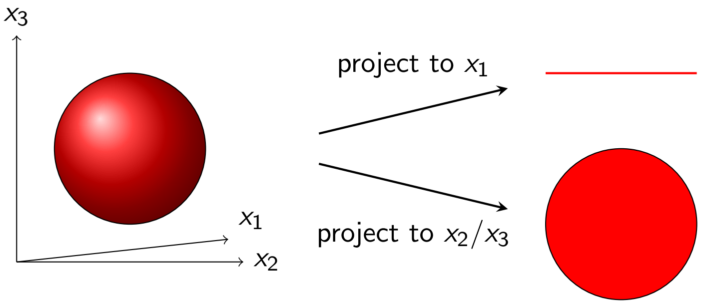
- Large systems
- "Sparse" properties

What can we decompose?

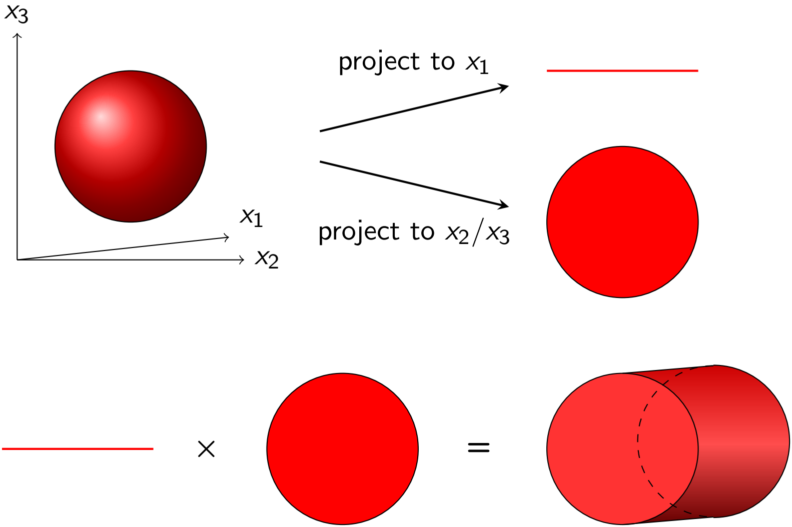
Cartesian decomposition



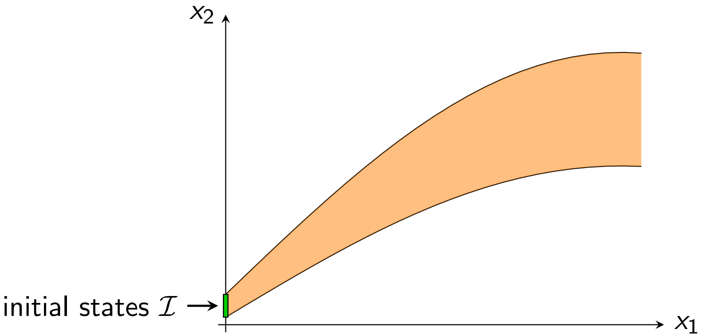
Cartesian decomposition



Cartesian decomposition

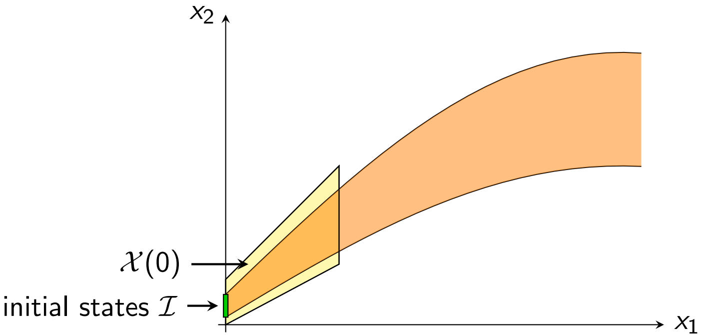


Reachability algorithm by Le Guernic and Girard (LGG)



Reachability algorithm by Le Guernic and Girard (LGG)

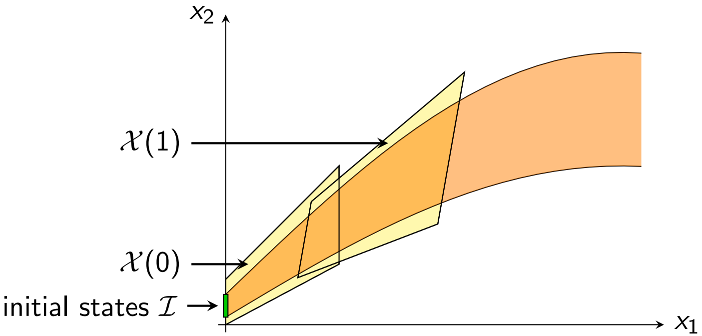
Discretize time
Compute overapproximation $\mathcal{X}(0)$ up to time step



Reachability algorithm by Le Guernic and Girard (LGG)

Compute successors

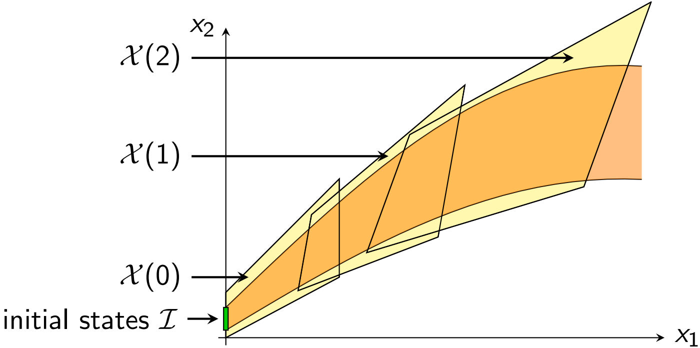
$$\mathcal{X}(1) = \Phi \cdot \mathcal{X}(0)$$



Reachability algorithm by Le Guernic and Girard (LGG)

Compute successors

$$\mathcal{X}(1) = \Phi \cdot \mathcal{X}(0)$$
$$\mathcal{X}(2) = \Phi \cdot \mathcal{X}(1) = \Phi^2 \cdot \mathcal{X}(0)$$



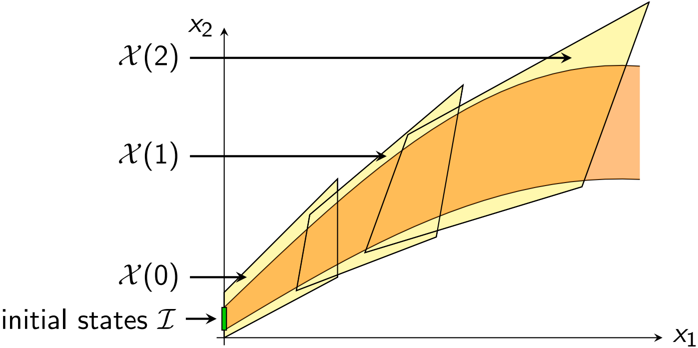
Reachability algorithm by Le Guernic and Girard (LGG)

Compute successors

$$\mathcal{X}(1) = \Phi \cdot \mathcal{X}(0)$$

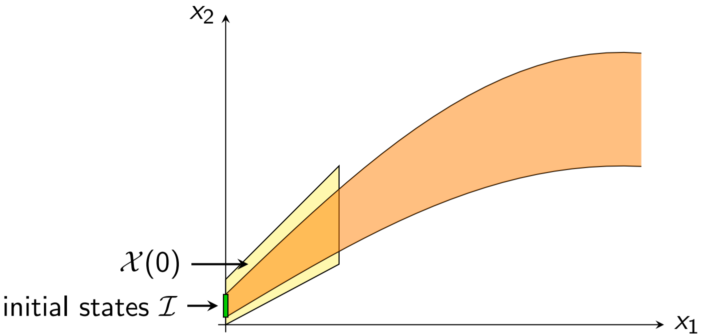
$$\mathcal{X}(2) = \Phi \cdot \mathcal{X}(1) = \Phi^2 \cdot \mathcal{X}(0)$$

$$\mathcal{X}(k) = \Phi^k \cdot \mathcal{X}(0)$$



LGG algorithm with Cartesian decomposition

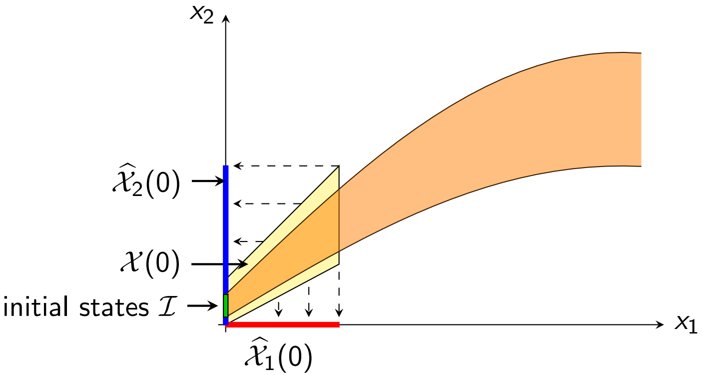
Compute **high-dimensional** set $\mathcal{X}(0)$ (as before)



LGG algorithm with Cartesian decomposition

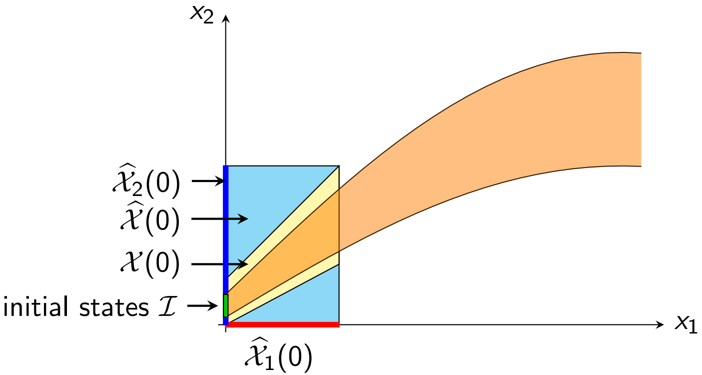
Decompose $\mathcal{X}(0)$ into **low-dimensional** sets $\widehat{\mathcal{X}}_1(0)$ and $\widehat{\mathcal{X}}_2(0)$

(Note: In general we do not need to go down to 1D)



LGG algorithm with Cartesian decomposition

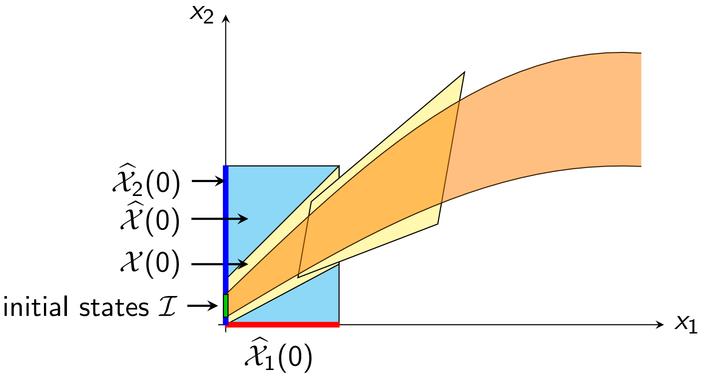
Define $\widehat{\mathcal{X}}(k) := \widehat{\mathcal{X}}_1(k) \times \widehat{\mathcal{X}}_2(k)$



LGG algorithm with Cartesian decomposition

Define $\widehat{\mathcal{X}}(k) := \widehat{\mathcal{X}}_1(k) \times \widehat{\mathcal{X}}_2(k)$

original: $\mathcal{X}(k) = \Phi^k \cdot \mathcal{X}(0)$

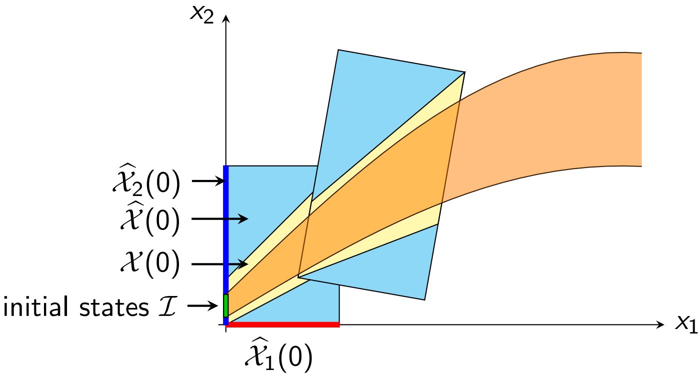


LGG algorithm with Cartesian decomposition

Define $\widehat{\mathcal{X}}(k) := \widehat{\mathcal{X}}_1(k) \times \widehat{\mathcal{X}}_2(k)$

original: $\mathcal{X}(k) = \Phi^k \cdot \mathcal{X}(0)$

decomposed: $\widehat{\mathcal{X}}(k) = \Phi^k \cdot \widehat{\mathcal{X}}(0)$?

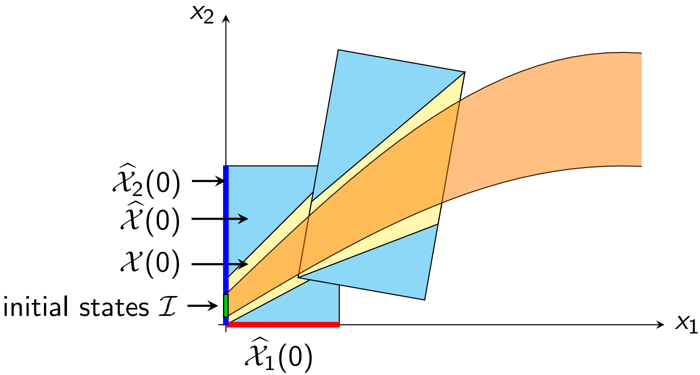


LGG algorithm with Cartesian decomposition

Define $\widehat{\mathcal{X}}(k) := \widehat{\mathcal{X}}_1(k) \times \widehat{\mathcal{X}}_2(k)$

original: $\mathcal{X}(k) = \Phi^k \cdot \mathcal{X}(0)$

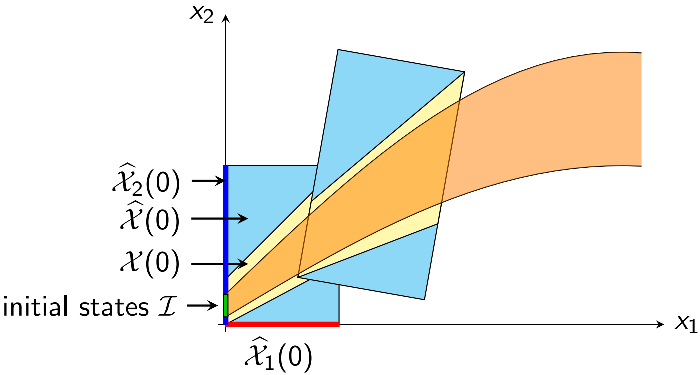
decomposed: $\widehat{\mathcal{X}}_i(k) = \bigoplus_j \Phi^k_{i,j} \cdot \widehat{\mathcal{X}}_j(0)$



Example

$$\widehat{\mathcal{X}}_i(k) = \bigoplus_j \Phi^k_{i,j} \cdot \widehat{\mathcal{X}}_j(0)$$

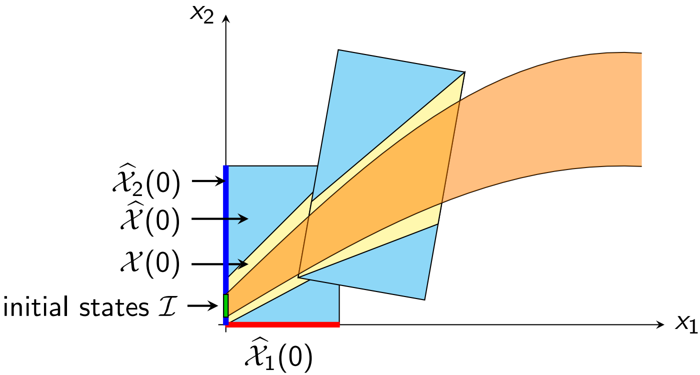
$$\Phi = \left(\begin{array}{c|c} a & b \\ \hline c & 0 \end{array}\right)$$



Example

$$\widehat{\mathcal{X}}_i(k) = \bigoplus_j \Phi^k_{i,j} \cdot \widehat{\mathcal{X}}_j(0)$$
$$\widehat{\mathcal{X}}_1(1) = a \cdot \widehat{\mathcal{X}}_1(0)$$

$$\Phi = \left(\begin{array}{c|c} a & b \\ \hline c & 0 \end{array}\right)$$

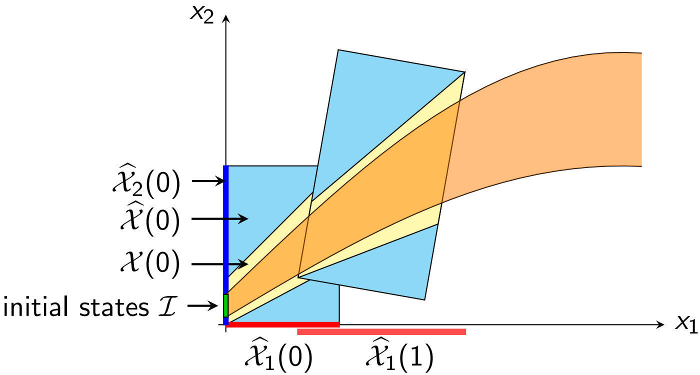


Example

$$\widehat{\mathcal{X}}_i(k) = \bigoplus_j \Phi^k_{i,j} \cdot \widehat{\mathcal{X}}_j(0)$$

$$\widehat{\mathcal{X}}_1(1) = a \cdot \widehat{\mathcal{X}}_1(0) \oplus b \cdot \widehat{\mathcal{X}}_2(0)$$

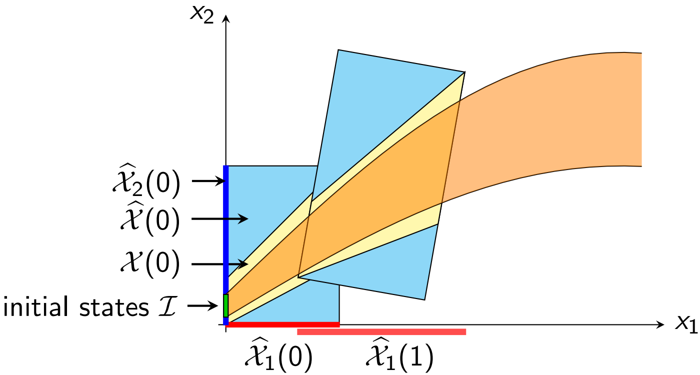
$$\Phi = \left(\begin{array}{c|c} a & b \\ \hline c & 0 \end{array}\right)$$



Example

$$\widehat{\mathcal{X}}_i(k) = \bigoplus_j \Phi^k_{i,j} \cdot \widehat{\mathcal{X}}_j(0)$$
$$\widehat{\mathcal{X}}_1(1) = a \cdot \widehat{\mathcal{X}}_1(0) \oplus b \cdot \widehat{\mathcal{X}}_2(0)$$
$$\widehat{\mathcal{X}}_2(1) = c \cdot \widehat{\mathcal{X}}_1(0)$$

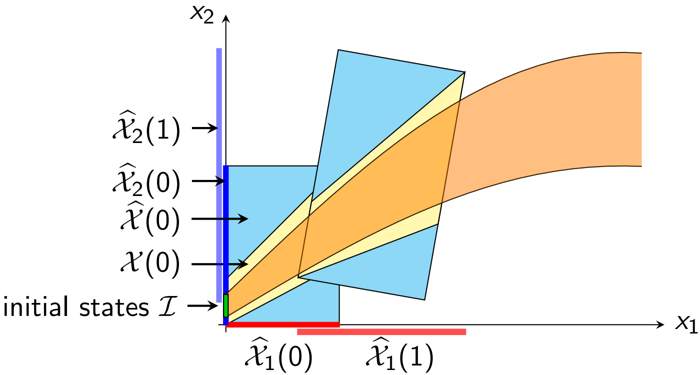
$$\Phi = \left(\begin{array}{c|c} a & b \\ \hline c & 0 \end{array}\right)$$



Example

$$\widehat{\mathcal{X}}_i(k) = \bigoplus_j \Phi^k_{i,j} \cdot \widehat{\mathcal{X}}_j(0)$$
$$\widehat{\mathcal{X}}_1(1) = a \cdot \widehat{\mathcal{X}}_1(0) \oplus b \cdot \widehat{\mathcal{X}}_2(0)$$
$$\widehat{\mathcal{X}}_2(1) = c \cdot \widehat{\mathcal{X}}_1(0) \oplus 0 \cdot \widehat{\mathcal{X}}_2(0)$$

$$\Phi = \left(\begin{array}{c|c} a & b \\ \hline c & 0 \end{array} \right)$$



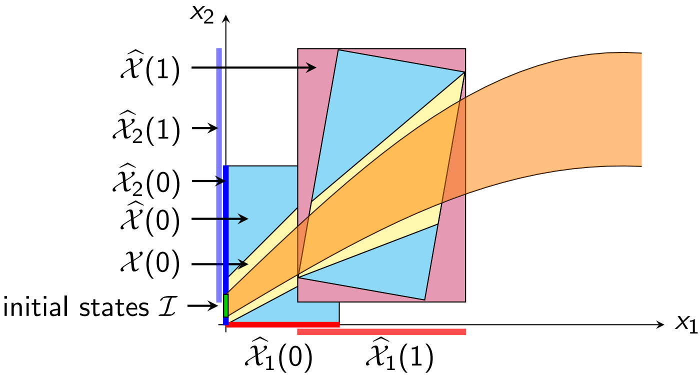
Example

$$\widehat{\mathcal{X}}_i(k) = \bigoplus_j \Phi^k_{i,j} \cdot \widehat{\mathcal{X}}_j(0)$$

$$\widehat{\mathcal{X}}_1(1) = a \cdot \widehat{\mathcal{X}}_1(0) \oplus b \cdot \widehat{\mathcal{X}}_2(0)$$

$$\widehat{\mathcal{X}}_2(1) = c \cdot \widehat{\mathcal{X}}_1(0) \oplus 0 \cdot \widehat{\mathcal{X}}_2(0)$$

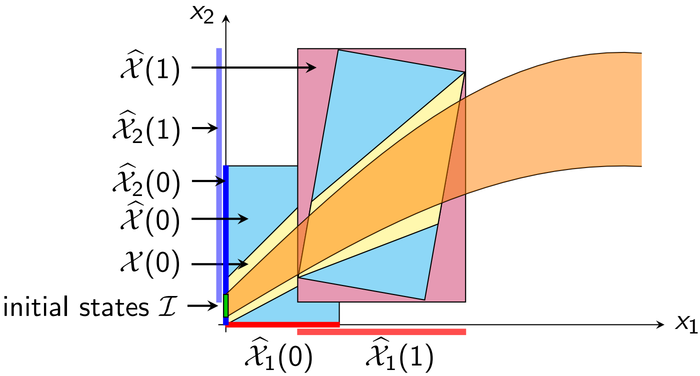
$$\Phi = \left(\begin{array}{c|c} a & b \\ \hline c & 0 \end{array}\right)$$



Example

$$\widehat{\mathcal{X}}_i(k) = \bigoplus_j \Phi^k_{i,j} \cdot \widehat{\mathcal{X}}_j(0)$$
$$\widehat{\mathcal{X}}_1(1) = a \cdot \widehat{\mathcal{X}}_1(0) \oplus b \cdot \widehat{\mathcal{X}}_2(0)$$
$$\widehat{\mathcal{X}}_2(1) = c \cdot \widehat{\mathcal{X}}_1(0) \cancel{\oplus\, 0 \cdot \widehat{\mathcal{X}}_2(0)}$$

$$\Phi = \left(\begin{array}{c|c} a & b \\ \hline c & 0 \end{array}\right)$$



Decomposed reachability algorithm – Summary

Classical LGG algorithm is a special case (with one block)

Precision

- Sacrifice precision due to **inter-block dependencies**
- Preserve dependencies between **intra-block dimensions**

Speed

- Perform **set operations in decomposed dimensions**
- Skip computations for **irrelevant dimensions**
- Exploit **sparsity** of matrices Φ^k

Implementation & evaluation

Implementation

- JuliaReach[1], written in Julia

Benchmark settings

- **1D blocks** (worst case precision)
- High-dimensional benchmark suite, with inputs
- 1st setting: evaluate **speed** in reach set computation
  - Comparison to state-of-the-art tool SpaceEx
  - Time step 10^{-3}, one dimension
- 2nd setting: evaluate **precision** in safety verification

[1] https://github.com/JuliaReach

Reachability in dense time

Model	Dim	JuliaReach	SpaceEx	Speedup
Motor	8	1.1 s	1.9 s	1.8
Building	48	4.5 s	9.5 s	2.1
PDE	84	4.4 s	61.7 s	13.9
Heat	200	24.7 s	102.8 s	4.1
ISS*	270	2.5 s	79.1 s	32.1
Beam	348	54.0 s	332.1 s	6.1
MNA1	578	140.0 s	*crashed*	*n/a*
FOM*	1006	10.6 s	*crashed*	*n/a*
MNA5*	10913	1650.3 s	*crashed*	*n/a*

*sparse matrix

Reachability in dense time

Model	Dim	JuliaReach	SpaceEx	Speedup
Motor	8	1.1 s	1.9 s	1.8
Building	48	4.5 s	9.5 s	2.1
PDE	84	4.4 s	61.7 s	**13.9**
Heat	200	24.7 s	102.8 s	4.1
ISS*	270	2.5 s	79.1 s	32.1
Beam	348	54.0 s	332.1 s	6.1
MNA1	578	140.0 s	*crashed*	*n/a*
FOM*	1006	10.6 s	*crashed*	*n/a*
MNA5*	10913	1650.3 s	*crashed*	*n/a*

*sparse matrix

Reachability in dense time

Model	Dim	JuliaReach	SpaceEx	Speedup
Motor	8	1.1 s	1.9 s	1.8
Building	48	4.5 s	9.5 s	2.1
PDE	84	4.4 s	61.7 s	13.9
Heat	200	24.7 s	102.8 s	4.1
ISS*	270	2.5 s	79.1 s	**32.1**
Beam	348	54.0 s	332.1 s	6.1
MNA1	578	140.0 s	*crashed*	*n/a*
FOM*	1006	10.6 s	*crashed*	*n/a*
MNA5*	10913	1650.3 s	*crashed*	*n/a*

* **sparse matrix**

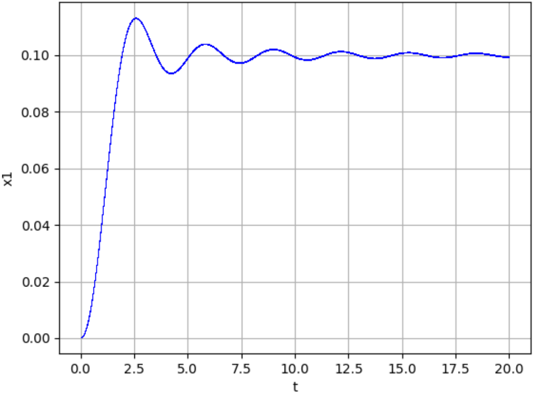
Reachability in dense time

Model	Dim	JuliaReach	SpaceEx	Speedup
Motor	8	1.1 s	1.9 s	1.8
Building	48	4.5 s	9.5 s	2.1
PDE	84	4.4 s	61.7 s	13.9
Heat	200	24.7 s	102.8 s	4.1
ISS*	270	2.5 s	79.1 s	32.1
Beam	348	54.0 s	332.1 s	6.1
MNA1	**578**	140.0 s	**crashed**	*n/a*
FOM*	**1006**	10.6 s	**crashed**	*n/a*
MNA5*	**10913**	1650.3 s	**crashed**	*n/a*

*sparse matrix

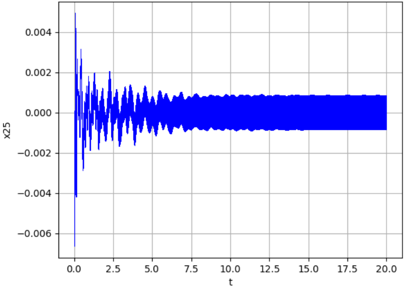
Reach set comparison – MNA5 model

- Bad states $\mathcal{B}$: $x_1 \geq 0.2 \vee x_2 \geq 0.15$

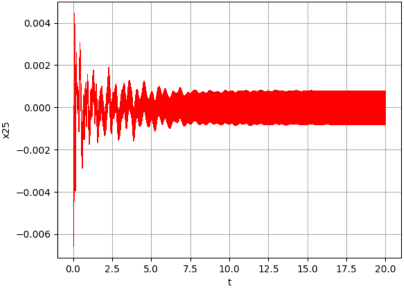


Reach set comparison – Building model

- Bad states $\mathcal{B}$: $x_{25} \geq 0.006$



JuliaReach



SpaceEx

Safety property verification in dense time

Model	Dim	#Var	Time step	JuliaReach
Motor	8	2	1×10^{-3}	1.6 s
Building	48	1	2×10^{-3}	1.1 s
PDE	84	84	3×10^{-4}	1030.0 s
Heat	200	1	1×10^{-3}	14.8 s
Beam	348	1	5×10^{-5}	857.1 s
MNA1	578	1	4×10^{-4}	287.2 s
MNA5*	10913	2	3×10^{-1}	719.1 s

*sparse matrix

Safety property verification in dense time

Model	Dim	**#Var**	Time step	JuliaReach
Motor	8	2	1×10^{-3}	1.6 s
Building	48	1	2×10^{-3}	1.1 s
PDE	84	84	3×10^{-4}	1030.0 s
Heat	200	1	1×10^{-3}	14.8 s
Beam	348	1	5×10^{-5}	857.1 s
MNA1	578	1	4×10^{-4}	287.2 s
MNA5*	10913	2	3×10^{-1}	719.1 s

*sparse matrix

Safety property verification in dense time

Model	Dim	#Var	Time step	JuliaReach
Motor	8	**2**	1×10^{-3}	1.6 s
Building	48	**1**	2×10^{-3}	1.1 s
PDE	84	84	3×10^{-4}	1030.0 s
Heat	200	**1**	1×10^{-3}	14.8 s
Beam	348	**1**	5×10^{-5}	857.1 s
MNA1	578	**1**	4×10^{-4}	287.2 s
MNA5*	10913	**2**	3×10^{-1}	719.1 s

*sparse matrix

Safety property verification in dense time

Model	Dim	#Var	Time step	JuliaReach
Motor	8	2	1×10^{-3}	1.6 s
Building	48	1	2×10^{-3}	1.1 s
PDE	84	**84**	3×10^{-4}	1030.0 s
Heat	200	1	1×10^{-3}	14.8 s
Beam	348	1	5×10^{-5}	857.1 s
MNA1	578	1	4×10^{-4}	287.2 s
MNA5*	10913	2	3×10^{-1}	719.1 s

*sparse matrix

Discrete-time setting

- Reachable states are only computed at **discrete time steps**
- Assumption: **Inputs** can only change at **discrete time steps**
- Comparison to state-of-the-art tool HYLAA
  - Uses **simulations**, exploiting superposition
- Same settings as before

Safety property verification in discrete time

Model	Dim	#Var	JuliaReach	Hylaa	Speedup
Motor	8	2	0.3 s	1.6 s	6.5
Building	48	1	0.5 s	2.5 s	4.7
PDE	84	84	22.2 s	3.5 s	0.2
Heat	200	1	4.2 s	13.8 s	3.3
Beam	348	1	7.0 s	169.1 s	24.2
MNA1	578	1	19.7 s	288.2 s	14.6
MNA5*	10913	2	435.7 s	3440.2 s	79.1

*sparse matrix

Safety property verification in discrete time

Model	Dim	#Var	JuliaReach	Hylaa	Speedup
Motor	8	2	0.3 s	1.6 s	6.5
Building	48	1	0.5 s	2.5 s	4.7
PDE	84	**84**	22.2 s	3.5 s	**0.2**
Heat	200	1	4.2 s	13.8 s	3.3
Beam	348	1	7.0 s	169.1 s	24.2
MNA1	578	1	19.7 s	288.2 s	14.6
MNA5*	10913	2	435.7 s	3440.2 s	79.1

*sparse matrix

Safety property verification in discrete time

Model	Dim	#Var	JuliaReach	Hylaa	Speedup
Motor	8	2	0.3 s	1.6 s	6.5
Building	48	1	0.5 s	2.5 s	4.7
PDE	84	84	22.2 s	3.5 s	0.2
Heat	200	1	4.2 s	13.8 s	3.3
Beam	348	1	7.0 s	169.1 s	24.2
MNA1	578	1	19.7 s	288.2 s	14.6
MNA5*	10913	2	435.7 s	3440.2 s	**79.1**

*sparse matrix

Conclusion

- **Generalized reachability algorithm** for LTI systems
- **Cartesian decomposition** approach
  - **Matrix operations in high dimensions**
  - **Set operations in low dimensions**
- **Outperforms** state-of-the-art tools SpaceEx and Hylaa
  - **Speed**: Over an order of magnitude faster
  - **Dimension**: Over an order of magnitude higher (SpaceEx)
- **Precision** sufficiently good in many cases