Techniques and Tools for Hybrid Systems Reachability Analysis

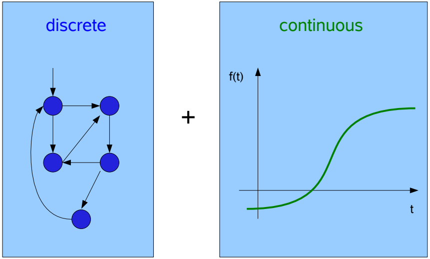
Stefan Schupp Johanna Nellen Erika Ábrahám

hybr i2d Theory of Hybrid Systems Informatik 2 | RWTH AACHEN UNIVERSITY

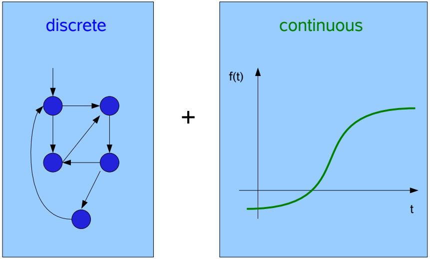
RiSE4CPS, Heidelberg, Germany
April 23, 2017

This work is part of the project
HyPro
which is funded by the German Research Council (DFG).

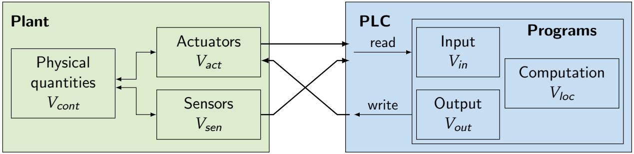
Hybrid systems in computer science



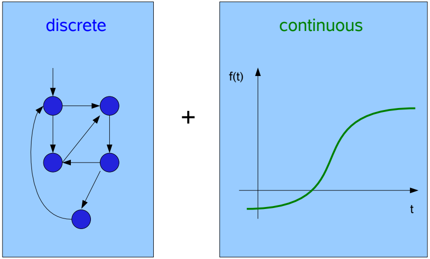
Hybrid systems in computer science



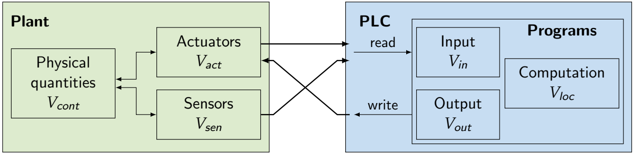
Example:
Physical systems controlled by programmable logic controllers (PLCs)



Hybrid systems in computer science

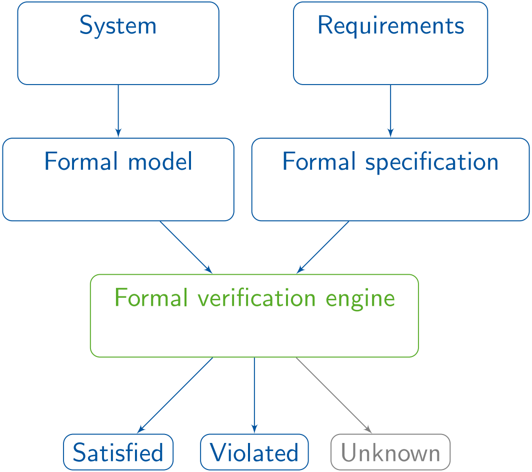


Example:
Physical systems controlled by programmable logic controllers (PLCs)

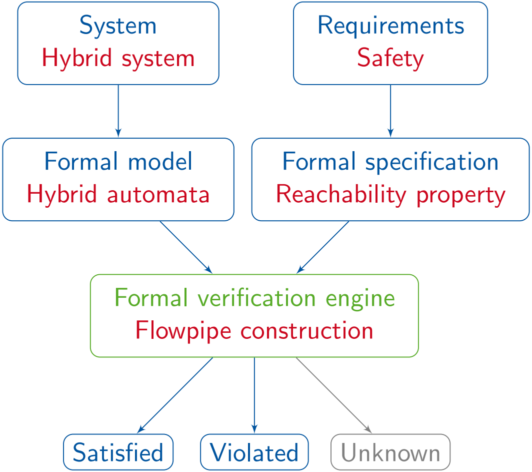


Such systems are typically complex and safety critical.

Formal verification



Formal verification



A hybrid automaton model of a steering controller

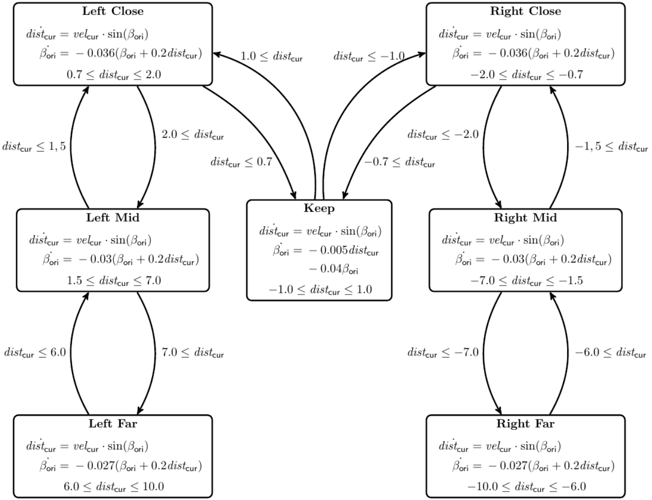


Figure 1: Steering Controller

Source: Möhlmann et al.: *Verifying a PI Controller using SoapBox and Stabhyli*. ARCH 2016

Some example requirements for the steering controller

- The distance of the current state $(dist_{cur}, \beta_{ori})$ to the optimal state $(0,0)$ is always bounded:

$$G(dist_{cur} \in [-10,10] \wedge \beta_{ori} \in [-5^\circ, 5^\circ])$$

- The car's orientation changes smoothly:

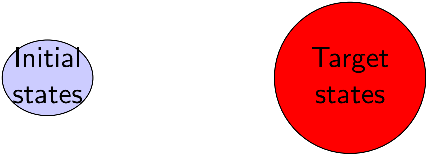
$$G(\dot{\beta}_{ori} \in [-0.3, 0.3])$$

Source: Möhlmann et al.: *Verifying a PI Controller using SoapBox and Stabhyli.* ARCH 2016

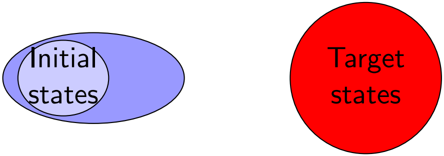
The reachability problem for hybrid systems

The reachability problem for hybrid systems poses the question, whether a given hybrid system can reach a certain set of target states from its initial states.

The reachability problem for hybrid systems poses the question, whether a given hybrid system can reach a certain set of target states from its initial states.

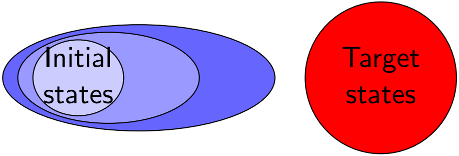


The reachability problem for hybrid systems poses the question, whether a given hybrid system can reach a certain set of target states from its initial states.



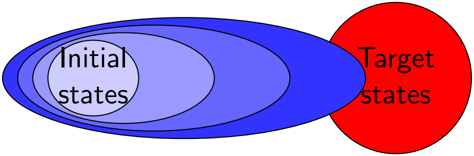
The reachability problem for hybrid systems

The reachability problem for hybrid systems poses the question, whether a given hybrid system can reach a certain set of target states from its initial states.

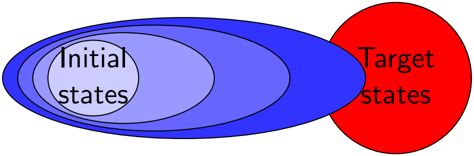


The reachability problem for hybrid systems

The reachability problem for hybrid systems poses the question, whether a given hybrid system can reach a certain set of target states from its initial states.



The reachability problem for hybrid systems poses the question, whether a given hybrid system can reach a certain set of target states from its initial states.



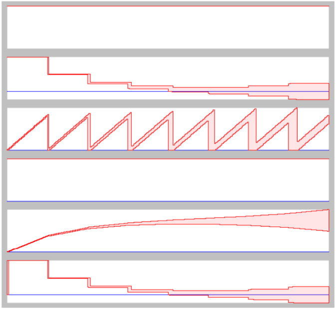
The reachability problem for hybrid systems is in general undecidable. Despite this fact, there are different (incomplete but practically useful) algorithms and tools for hybrid systems reachability analysis.

Impressive tool development in the last decade (incomplete list)

- HSolver [Ratschan et al., HSCC 2005]
- iSAT-ODE [Eggers et al., ATVA 2008]
- KeYmaera (X) [Platzer et al., IJCAR 2008]
- PowerDEVS [Bergero et al., Simulation 2011]
- SpaceEx [Frehse et al., CAV 2011]
- S-TaLiRo [Annapureddy et al., TACAS 2011]
- Ariadne [Collins et al., ADHS 2012]
- HySon [Bouissou et al., RSP 2012]
- Flow* [Chen et al., CAV 2013]
- HyCreate [Bak et al., HSCC 2013]
- HyEQ [Sanfelice et al., HSCC 2013]
- NLTOOLBOX [Testylier et al., ATVA 2013]
- SoapBox [Hagemann et al., ARCH 2014]
- Acumen [Taha et al., IoT 2015]
- C2E2 [Duggirala et al., TACAS 2015]
- Cora [Althoff et al., ARCH 2015]
- dReach [Kong et al, TACAS 2015]
- Isabelle/HOL [Immler, TACAS 2015]
- HyLAA [Bak et al., HSCC 2017]
- HyPro/HyDRA [Schupp et al., NFM 2017]

Verification techniques/tools for hybrid systems

(Rigorous/verified) simulation: Besides simulation for testing, rigorous/verified simulation can be used for (bounded) reachability analysis.
Some tools: Acumen, C2E2, HyEQ, HyLAA, HySon, S-TaLiRo, PowerDEVS

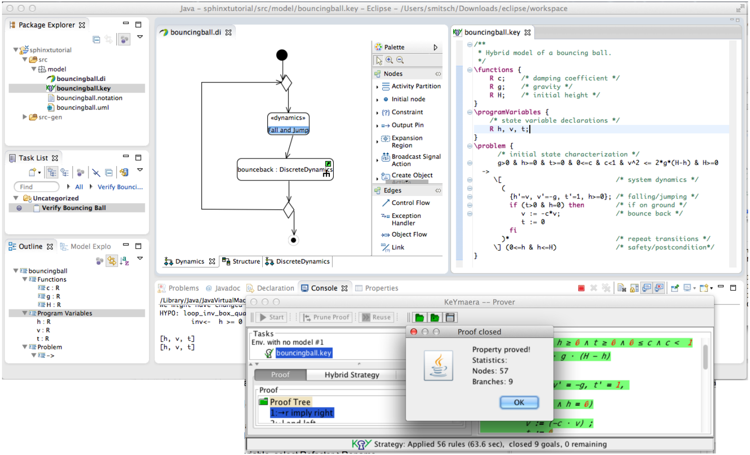


Source: http://www.acumen-language.org/

Verification techniques/tools for hybrid systems

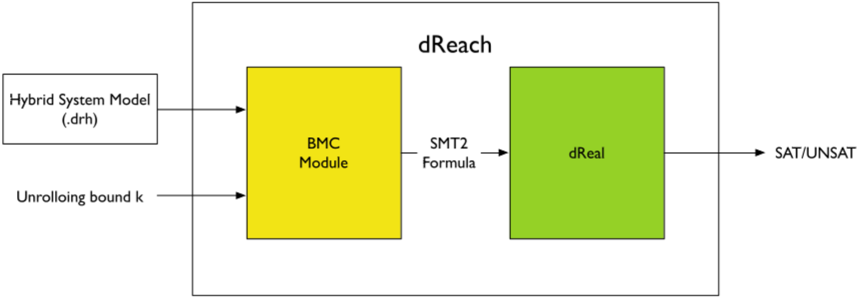
Deduction: Finding and showing invariants using theorem proving.

Some tools: Ariadne, Isabelle/HOL, KeYmaera



Source: http://symbolaris.com/info/keymaera.html

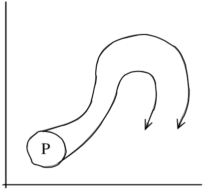
Bounded model checking / interval arithmetic: System executions and requirements are encoded by logical formulas; satisfiability checking tools (SMT solvers) are used for (bounded) reachability analysis.
Some tools: dReach, HSolver, iSAT-ODE



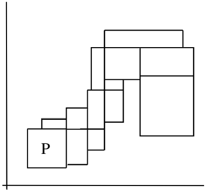
Source: http://dreal.github.io/dReach/

Over-approximating flowpipe construction: Iterative (bounded) forward reachability analysis based on some over-approximative symbolic state set representations.

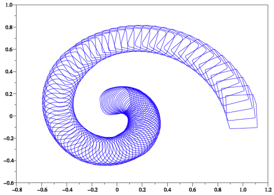
Some tools: Cora, Flow*, HyCreate, HyPro/HyDRA, NLTOOLBOX, SoapBox, SpaceEx

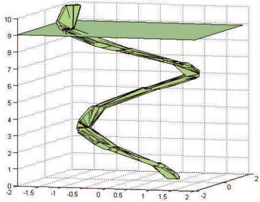


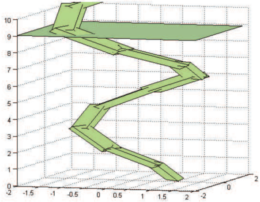
Exact behavior



Over-approximation







Source: [Bournez et al., HSCC 1999] [Stursberg et al., HSCC 2003]

Input: Hybrid automaton H, initial states X_0, target states T.
Output: (Bounded) reachability of T from X_0 in H.
Algorithm:

$Queue := \{X_0\}$;
$R := \{X_0\}$;
while $(Queue \neq \emptyset$ *and not break_cond*$)\{$
    Take a set P from *Queue*;
    Compute successor sets Reach(P);
    Add successor sets to *Queue* and R;
$\}$;
if $(\cup_{P \in R} P) \cap T = \emptyset)$ return "no" else return "yes";

Forward reachability analysis

Input: Hybrid automaton H, initial states X_0, target states T.
Output: (Bounded) reachability of T from X_0 in H.
Algorithm:

$Queue := \{X_0\}$;
$R := \{X_0\}$;
while ($Queue \neq \emptyset$ *and not break_cond*){
    Take a set P from *Queue*;
    Compute successor sets Reach(P);
    Add successor sets to *Queue* and R;
};
if $(\cup_{P \in R} P) \cap T = \emptyset)$ return "no" else return "yes";

Problems:

- How to represent state sets?
- How to compute set operations on them?
- How to compute Reach$(\cdot)$?

Most well-known state set representations

Geometric objects:

- boxes (hyper-rectangles) [Moore et al., 2009]
- oriented rectangular hulls [Stursberg et al., 2003]
- convex polyhedra [Ziegler, 1995] [Chen at el, 2011]
- template polyhedra [Sankaranarayanan et al., 2008]
- orthogonal polyhedra [Bournez et al., 1999]
- zonotopes [Girard, 2005])
- ellipsoids [Kurzhanski et al., 2000]

Other symbolic representations:

- support functions [Le Guernic et al., 2009]
- Taylor models [Berz and Makino, 1998, 2009] [Chen et al., 2012]

Set operations

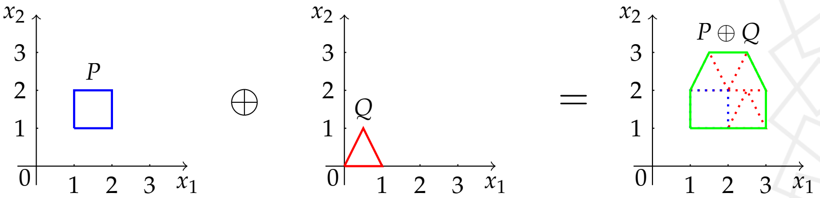
Some needed set operations:

intersection	projection
union	test for membership
linear transformation	test for emptiness
Minkowski sum	

Some needed set operations:

- intersection
- union
- linear transformation
- Minkowski sum
- projection
- test for membership
- test for emptiness

Reminder: Minkowski sum



$$P \oplus Q = \{p + q \mid p \in P \text{ and } q \in Q\}$$

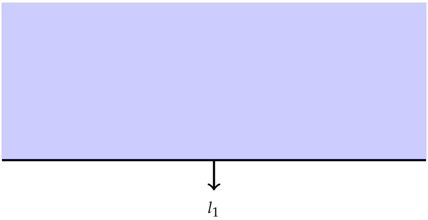
Example state set representation: Polytopes

Example state set representation: Polytopes

- Halfspace: set of points x satisfying $l \cdot x \leq z$

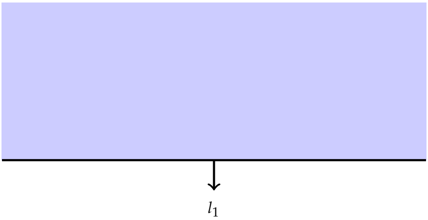
Example state set representation: Polytopes

- Halfspace: set of points x satisfying $l \cdot x \leq z$



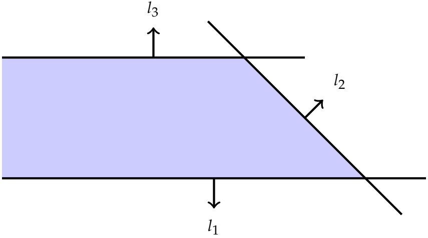
Example state set representation: Polytopes

- Halfspace: set of points x satisfying $l \cdot x \leq z$
- Polyhedron: an intersection of finitely many halfspaces

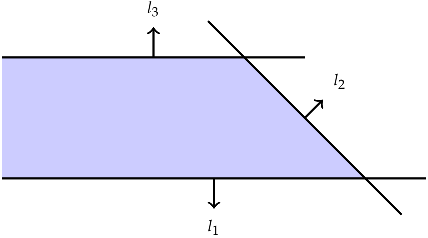


Example state set representation: Polytopes

- Halfspace: set of points x satisfying $l \cdot x \leq z$
- Polyhedron: an intersection of finitely many halfspaces

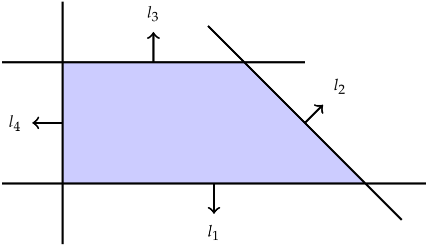


- Halfspace: set of points x satisfying $l \cdot x \leq z$
- Polyhedron: an intersection of finitely many halfspaces
- Polytope: a bounded polyhedron

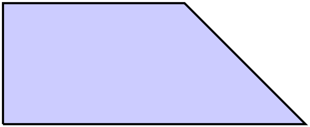


Example state set representation: Polytopes

- Halfspace: set of points x satisfying $l \cdot x \leq z$
- Polyhedron: an intersection of finitely many halfspaces
- Polytope: a bounded polyhedron

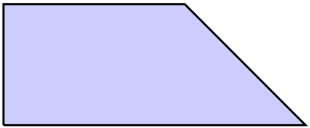


- Halfspace: set of points x satisfying $l \cdot x \leq z$
- Polyhedron: an intersection of finitely many halfspaces
- Polytope: a bounded polyhedron



Example state set representation: Polytopes

- Halfspace: set of points x satisfying $l \cdot x \leq z$
- Polyhedron: an intersection of finitely many halfspaces
- Polytope: a bounded polyhedron



representation	union	intersection	Minkowski sum
$\mathcal{V}$-representation by vertices	easy	hard	easy
$\mathcal{H}$-representation by facets	hard	easy	hard

Linear hybrid automata I and II

Linear hybrid automata I:

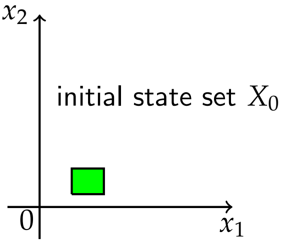
- derivatives: boxes
- conditions: convex linear sets
- resets: linear functions

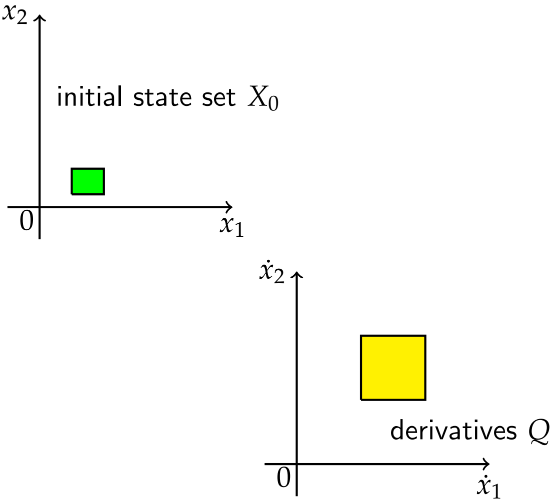
Linear hybrid automata II:

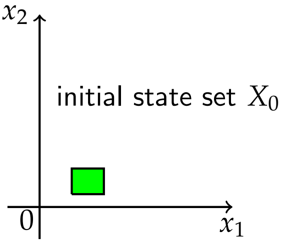
- derivatives: linear differential equations
- conditions: convex linear sets
- resets: linear functions

Linear hybrid automata I: Time evolution

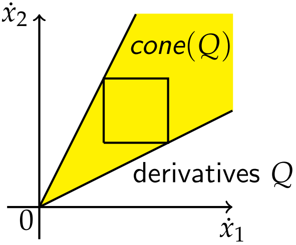
Linear hybrid automata I: Time evolution



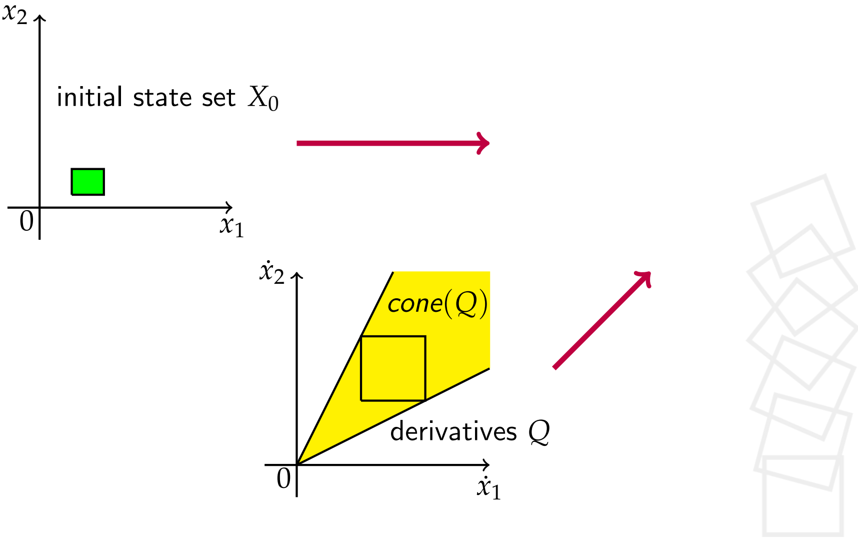

x_2
initial state set X_0
0
x_1
$\dot{x}_2$
derivatives Q
0
$\dot{x}_1$

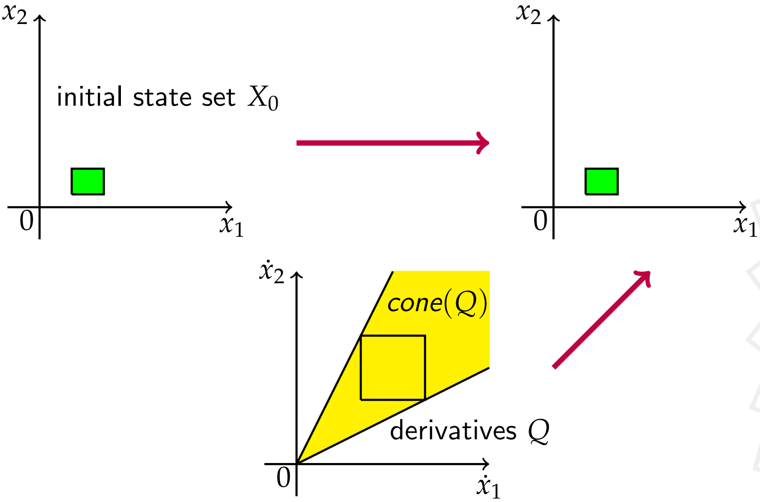


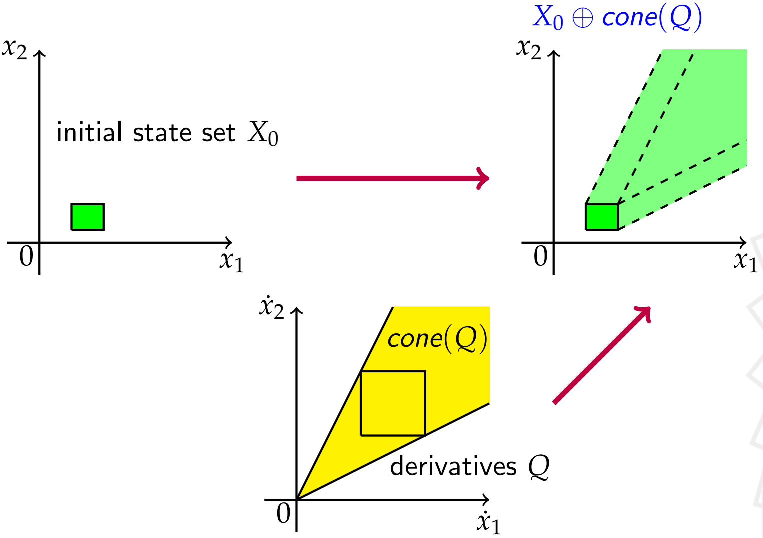
x_2
initial state set X_0
0
x_1

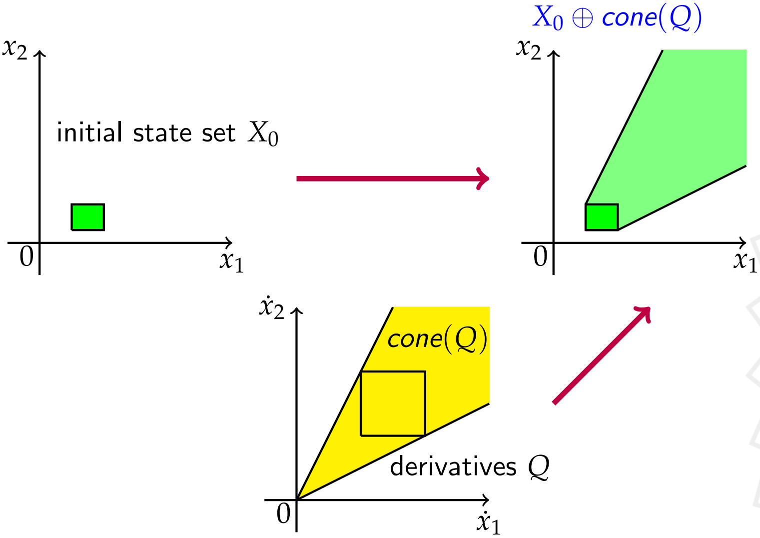


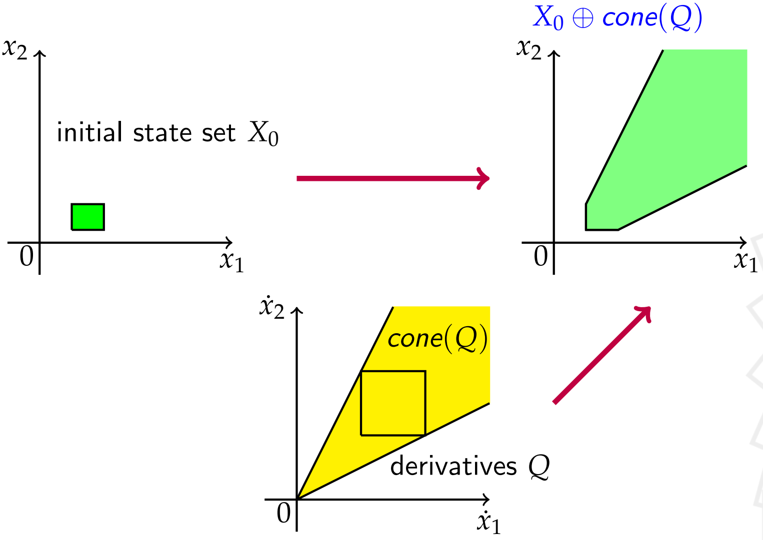
$\dot{x}_2$
cone(Q)
derivatives Q
0
$\dot{x}_1$


x_2
initial state set X_0
0
x_1
$\dot{x}_2$
$cone(Q)$
derivatives Q
0
$\dot{x}_1$

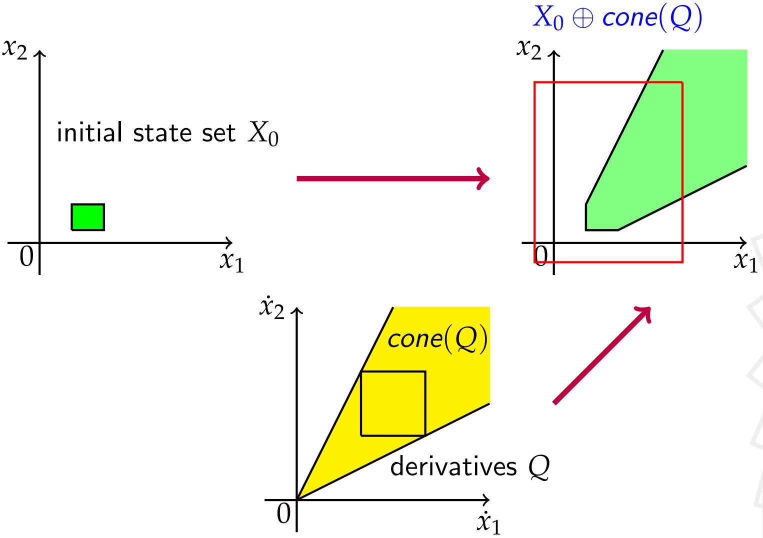

x_2
initial state set X_0
0
x_1
x_2
0
x_1
$\dot{x}_2$
cone(Q)
derivatives Q
0
$\dot{x}_1$


$X_0 \oplus cone(Q)$
x_2
x_1
0
initial state set X_0
$\dot{x}_2$
$\dot{x}_1$
$cone(Q)$
derivatives Q

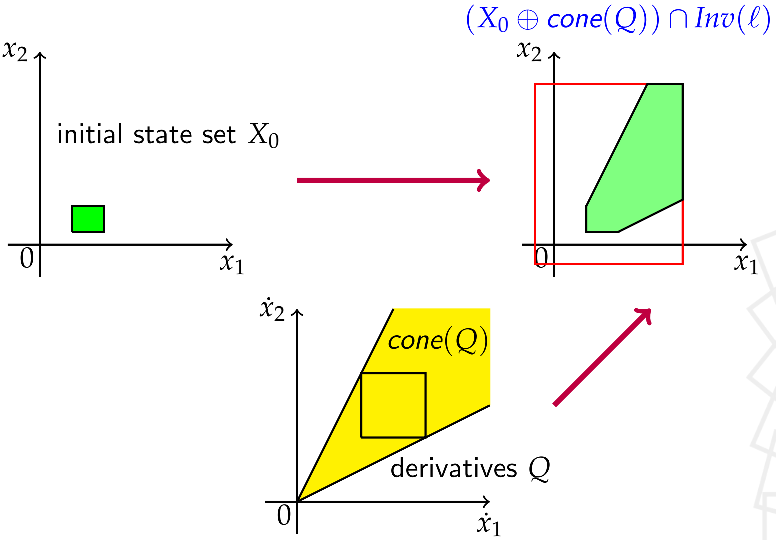

$X_0 \oplus cone(Q)$
x_2
x_1
0
initial state set X_0
x_2
x_1
0
$\dot{x}_2$
$\dot{x}_1$
0
$cone(Q)$
derivatives Q



$X_0 \oplus cone(Q)$
x_2
x_1
0
initial state set X_0
$\dot{x}_2$
$\dot{x}_1$
$cone(Q)$
derivatives Q

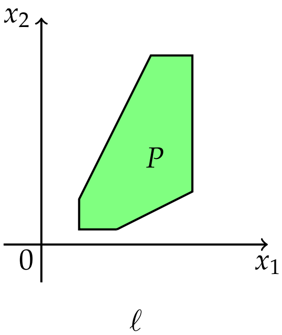


$X_0 \oplus cone(Q)$
x_2
initial state set X_0
0
x_1
x_2
0
x_1
$\dot{x}_2$
$cone(Q)$
derivatives Q
0
$\dot{x}_1$

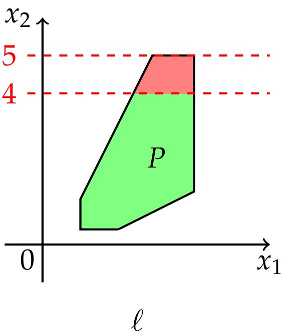

$(X_0 \oplus \mathit{cone}(Q)) \cap \mathit{Inv}(\ell)$
x_2
x_1
0
initial state set X_0
$\dot{x}_2$
$\dot{x}_1$
$\mathit{cone}(Q)$
derivatives Q

Linear hybrid automata I: Discrete steps (jumps)

Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$

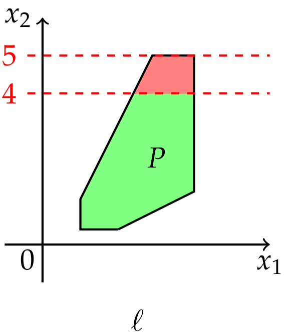


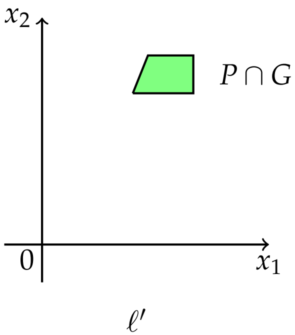
Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$



Linear hybrid automata I: Discrete steps (jumps)

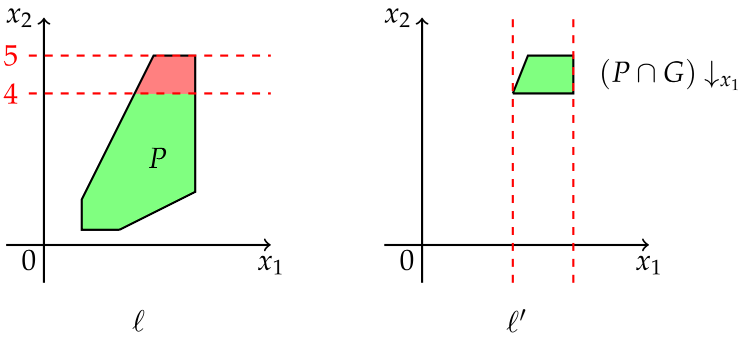
Example jump: $(\quad l,\quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]},\quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]},\quad l' \quad) \in Edge$



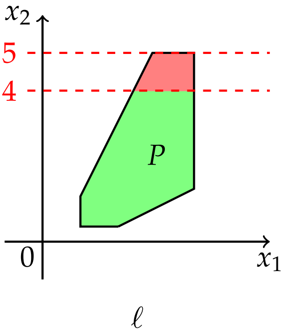


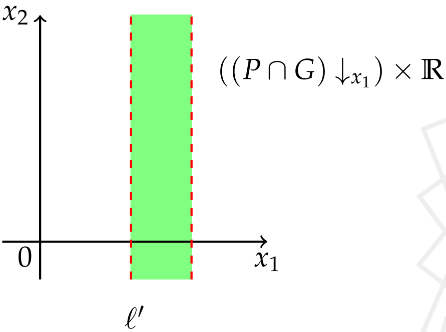
Linear hybrid automata I: Discrete steps (jumps)

Example jump: $\Big(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad \Big) \in Edge$



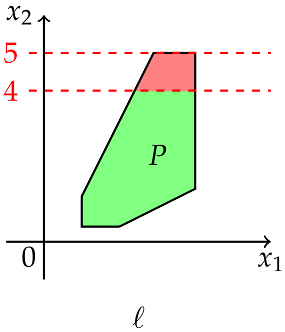
Example jump: $\left(\quad l,\quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R}\times[4,5]},\quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R}\times[2,4]},\quad l' \quad\right) \in Edge$

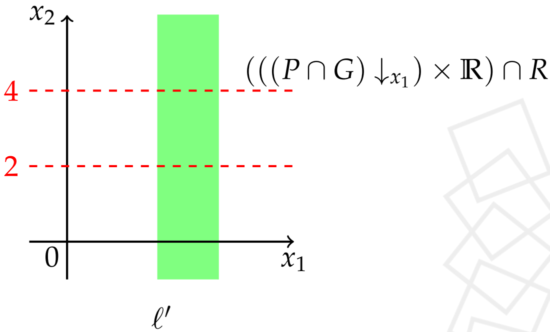




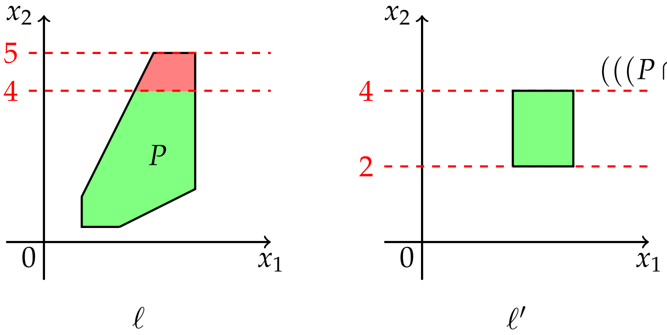
Linear hybrid automata I: Discrete steps (jumps)

Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad) \in Edge$



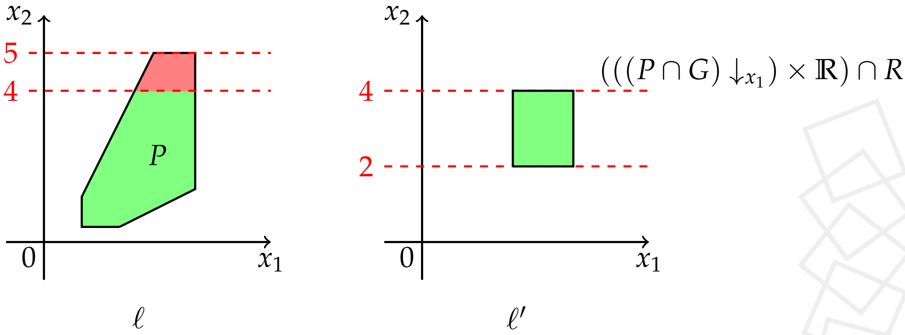


Example jump: $\left(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R} \times [4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R} \times [2,4]}, \quad l' \quad \right) \in Edge$



$$(((P \cap G) \downarrow_{x_1}) \times \mathbb{R}) \cap R$$

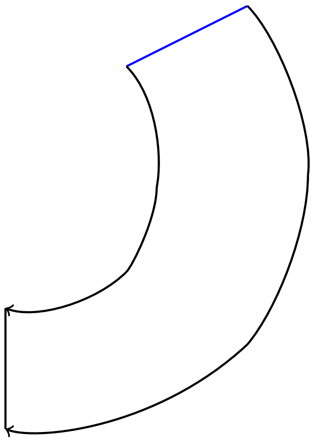
Example jump: $(\quad l, \quad \underbrace{4 \leq x_2 \leq 5}_{G \equiv \mathbb{R}\times[4,5]}, \quad \underbrace{x_2 :\in [2,4]}_{R \equiv \mathbb{R}\times[2,4]}, \quad l' \quad) \in Edge$



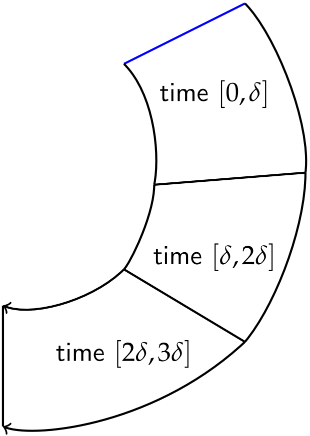
- Additionally: intersect the result with the invariant of l'.
- Computed via projection, Minkowski sum and intersection.

Linear hybrid automata II: Time evolution

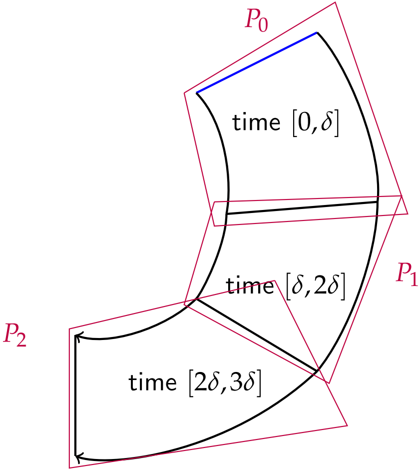
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$



- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$



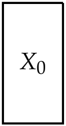
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i



- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:

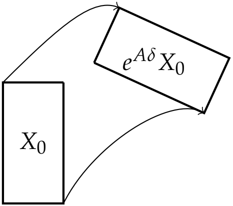
Linear hybrid automata II: Time evolution

- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:

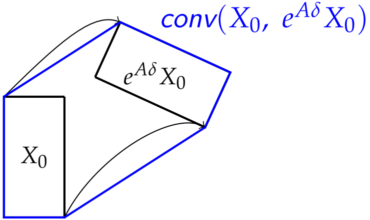


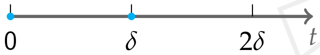
Linear hybrid automata II: Time evolution

- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$

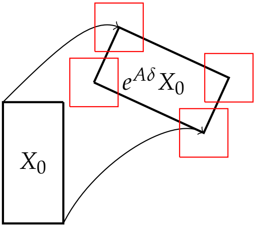


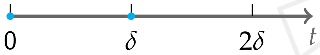
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



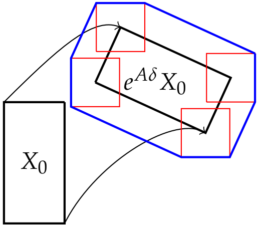


- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$

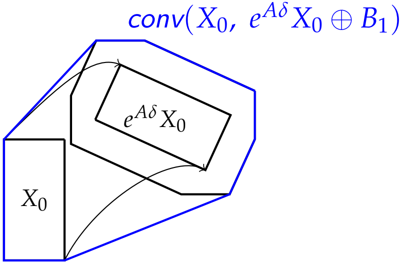


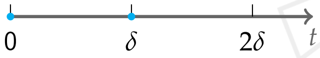


- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



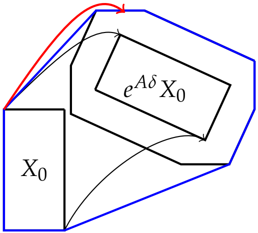
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$





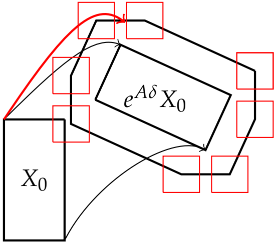
over-approximates flowpipe
for time $[0, \delta]$
under dynamics $\dot{x} = Ax$

- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



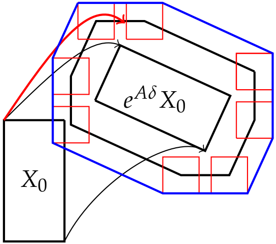
disturbance!

- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



disturbance!

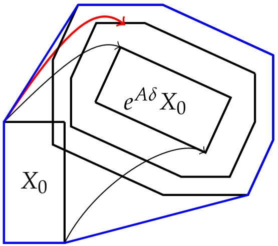
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



disturbance!

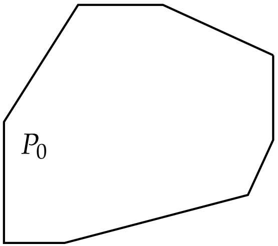
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$

$P_0 = conv(X_0,\ e^{A\delta}X_0 \oplus B_1 \oplus B_2)$

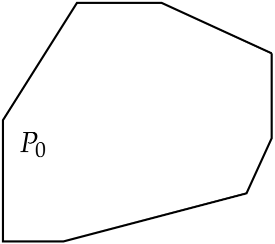


over-approximates flowpipe
for time $[0,\delta]$
under dynamics $\dot{x} = Ax + Bu$

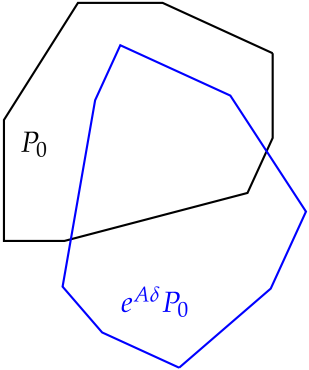
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$



- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:

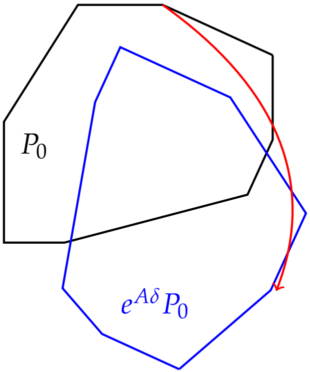


- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:



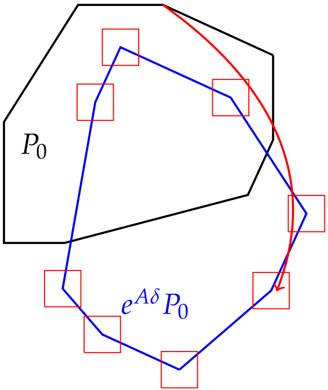
over-approximates flowpipe
for time $[\delta, 2\delta]$
under dynamics $\dot{x} = Ax$

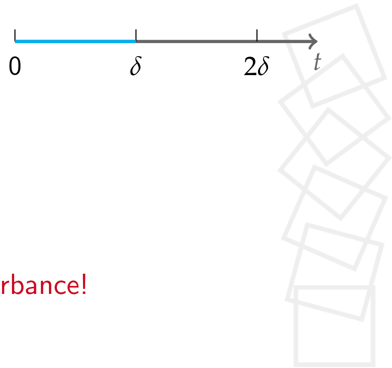
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:



disturbance!

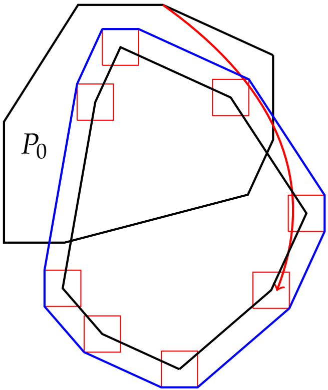
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:





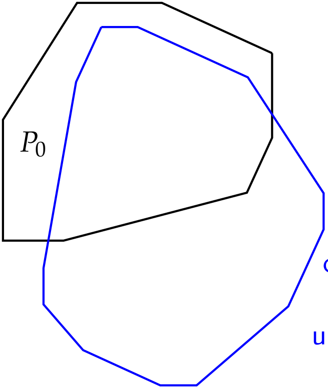
disturbance!

- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:



Linear hybrid automata II: Time evolution

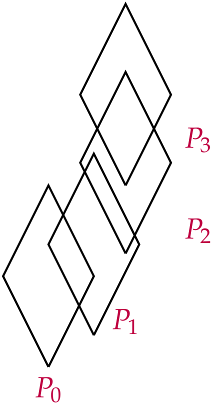
- Assume initial set X_0 and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by P_i
- The first flowpipe segment:
- Reminder matrix exponential: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
- The remaining ones:



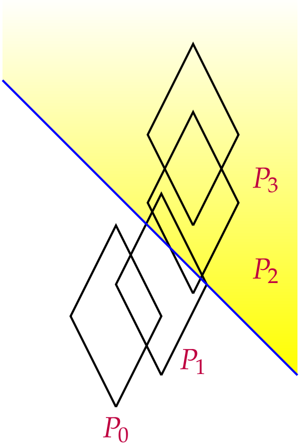
$P_1 = e^{A\delta} P_0 \oplus B_2$

over-approximates flowpipe
for time $[\delta, 2\delta]$
under dynamics $\dot{x} = Ax + Bu$

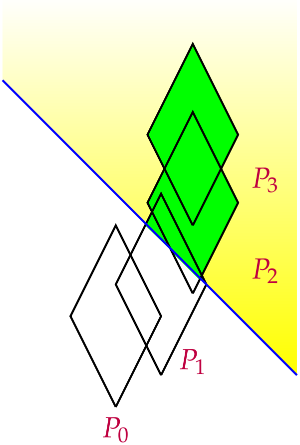
Linear hybrid automata II: Discrete steps (jumps)



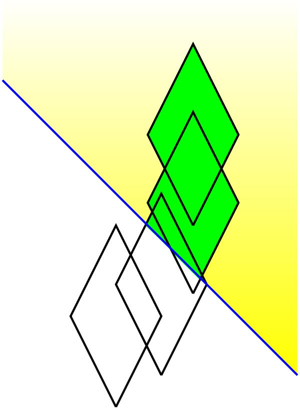
Linear hybrid automata II: Discrete steps (jumps)



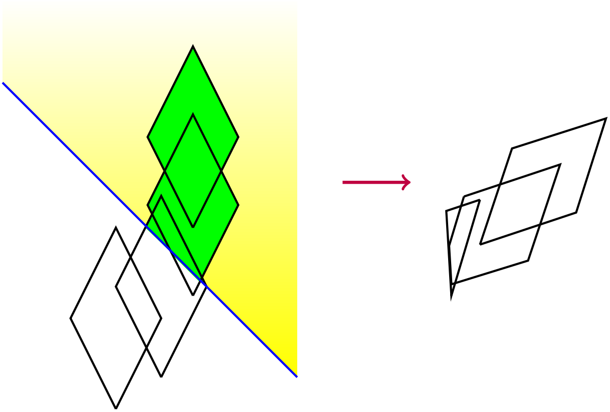
Linear hybrid automata II: Discrete steps (jumps)

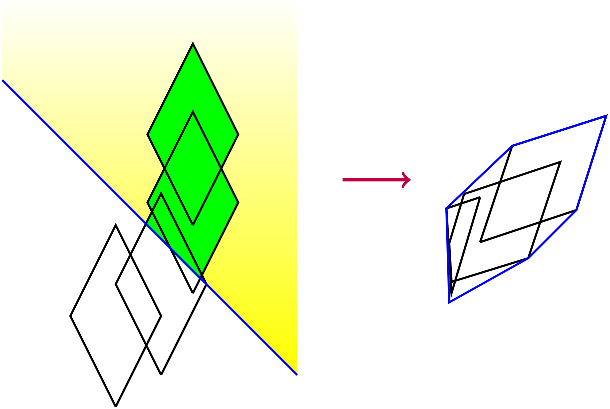


Linear hybrid automata II: Discrete steps (jumps)



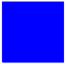
Linear hybrid automata II: Discrete steps (jumps)



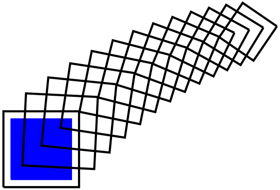


Linear hybrid automata II: The global picture

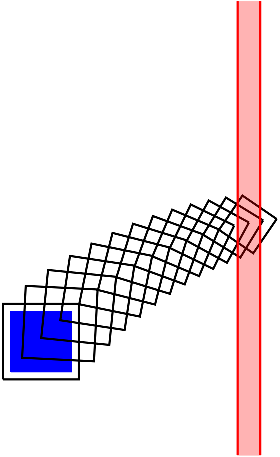
Linear hybrid automata II: The global picture



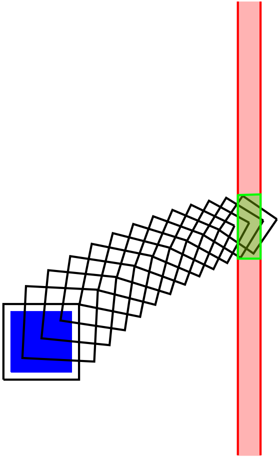
Linear hybrid automata II: The global picture



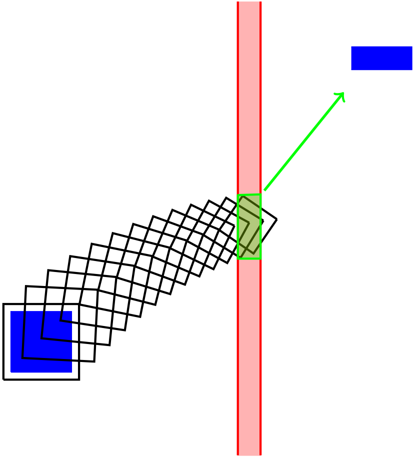
Linear hybrid automata II: The global picture



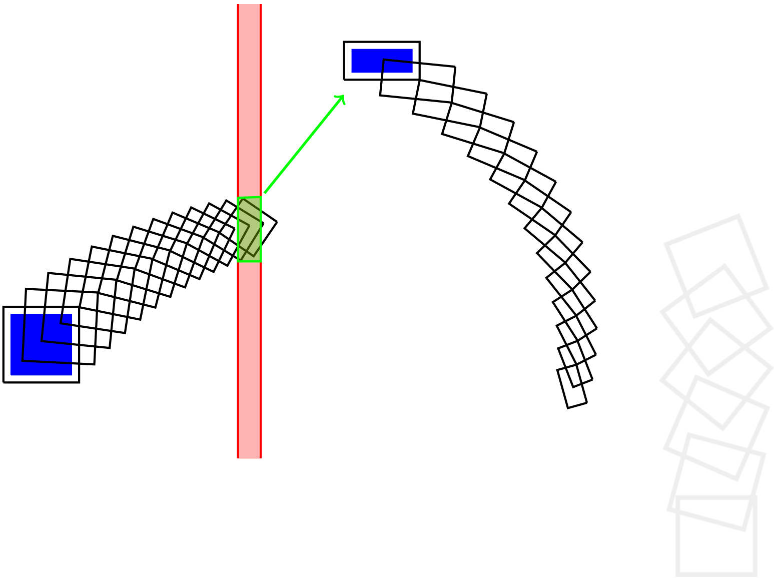
Linear hybrid automata II: The global picture



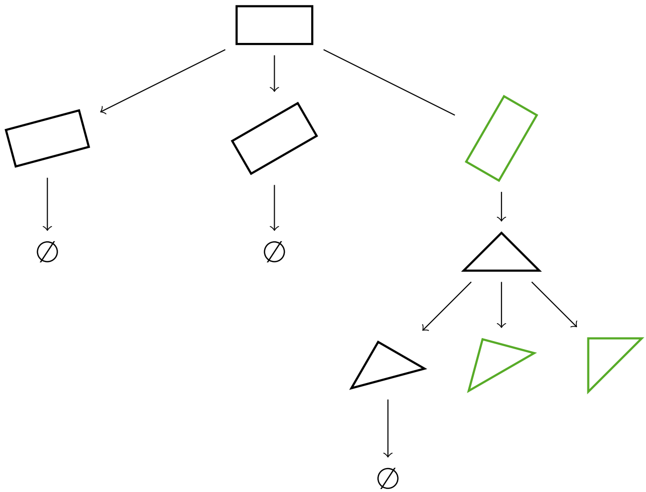
Linear hybrid automata II: The global picture



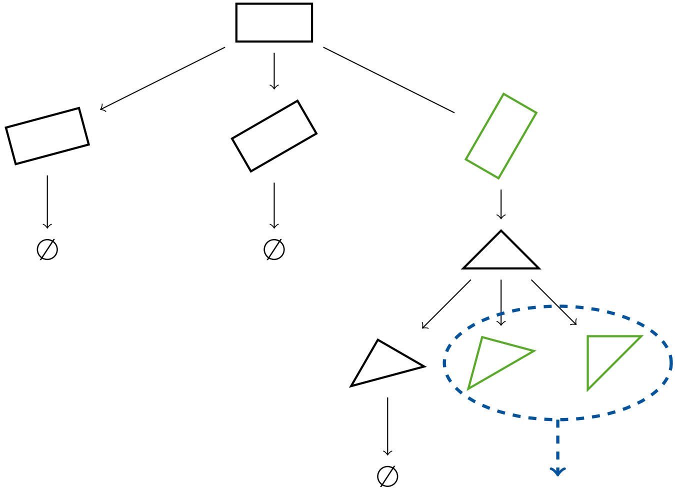
Linear hybrid automata II: The global picture



Reachability analysis search tree



Reachability analysis search tree



Flow* [Chen et al., CAV 2013]

A verification tool for cyber-physical systems

Available at https://flowstar.org/

- Taylor model-based approach
- non-linear dynamic
- adaptive refinement methods

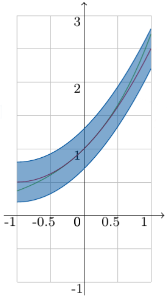


Image: Xin Chen

Has been used in a variety of verification tasks, e.g.

- biological/medical systems (glucose control, spiking neurons, Lotka Volterra equations),
- circuits (oscillators, van der Pol circuit),
- mechanical systems (jet engine model)

HyPro/HyDRA [Schupp et al., NFM 2017]

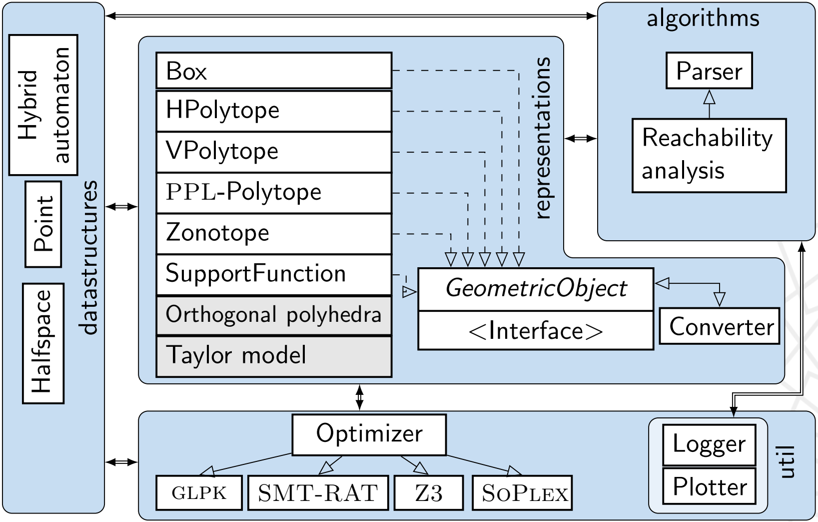
A free and open-source C++ library for state set representations for the reachability analysis of hybrid systems

Available at https://github.com/hypro/hypro

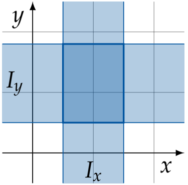
- state set representations
- conversion between different representations
- further datastructures and utility functions (hybrid automata, parser, logging, plotting)
- templated number type

Fast implementation of specialized reachability analysis methods

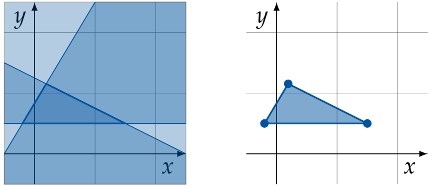
- Dimension reduction via sub-space computations [Schupp et al., QAPL 2017]
- CEGAR-like refinement loops and parallelisation (work in progress)


datastructures
Hybrid automaton
Point
Halfspace
representations
Box
HPolytope
VPolytope
PPL-Polytope
Zonotope
SupportFunction
Orthogonal polyhedra
Taylor model
GeometricObject
<Interface>
Converter
algorithms
Parser
Reachability analysis
Optimizer
GLPK
SMT-RAT
Z3
SoPlex
Logger
Plotter
util

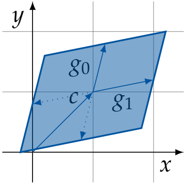
HyPro: State set representations



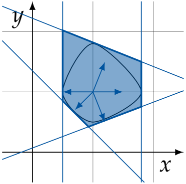
Boxes



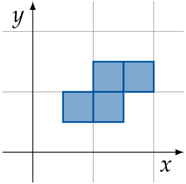
Convex polyhedra ($\mathcal{H}$, $\mathcal{V}$, PPL)



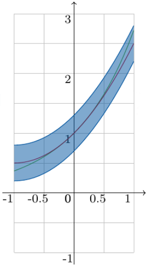
Zonotopes



Support functions

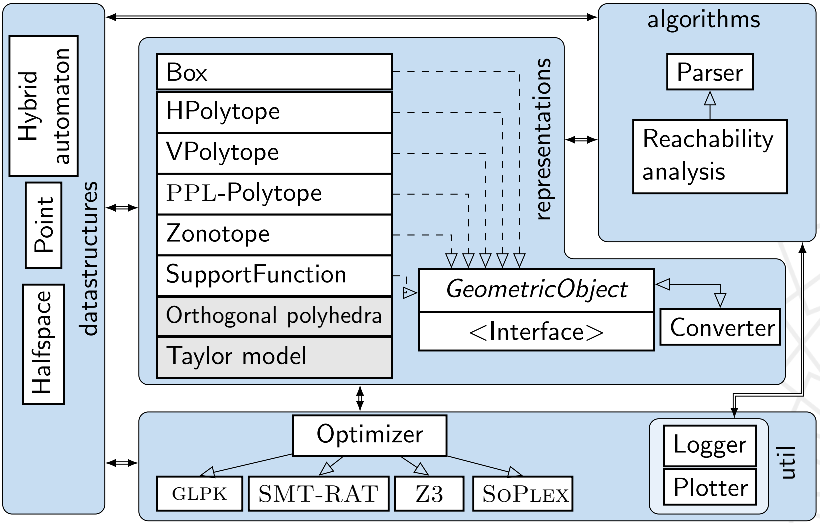


Orthogonal polyhedra



Taylor models

Source: Xin Chen

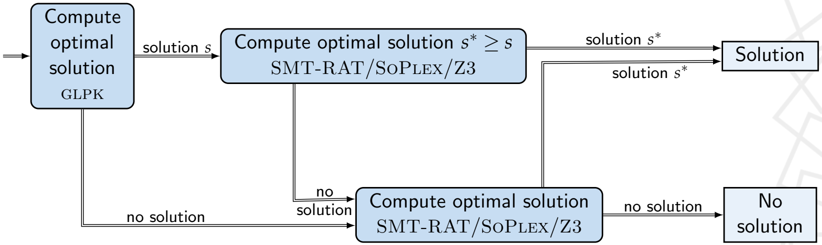

algorithms
Parser
Reachability analysis
datastructures
Hybrid automaton
Point
Halfspace
representations
Box
HPolytope
VPolytope
PPL-Polytope
Zonotope
SupportFunction
Orthogonal polyhedra
Taylor model
GeometricObject
<Interface>
Converter
Optimizer
GLPK
SMT-RAT
Z3
SoPlex
Logger
Plotter
util

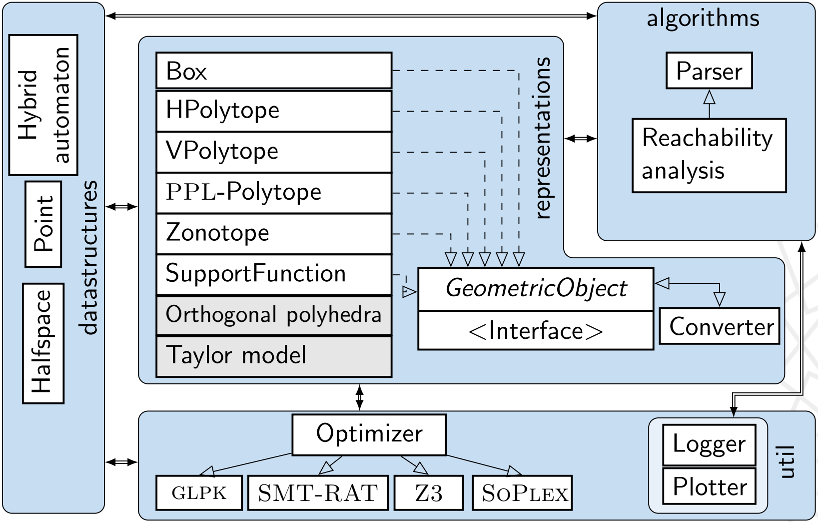
HyPro offers different number representations:
`cln::cl_RA`, `mpq_class`, `double`

Obstacles:

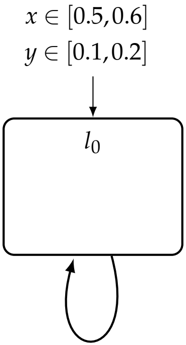
- inexact linear optimization not suitable
- exact linear optimization expensive

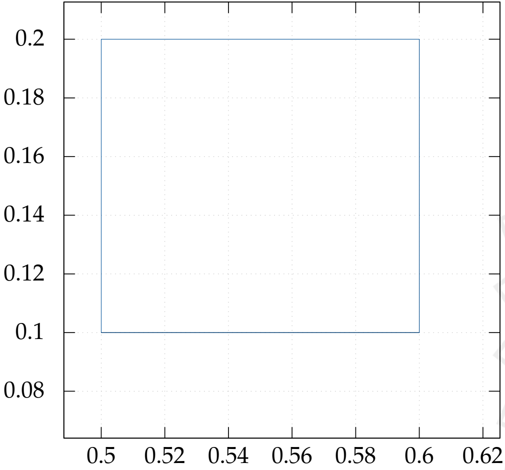
$\rightsquigarrow$ combined application



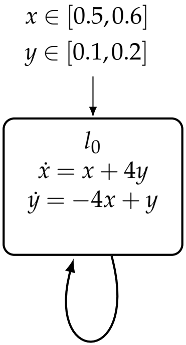

Hybrid automaton
Point
Halfspace
datastructures
Box
HPolytope
VPolytope
PPL-Polytope
Zonotope
SupportFunction
Orthogonal polyhedra
Taylor model
representations
GeometricObject
<Interface>
Converter
algorithms
Parser
Reachability analysis
Optimizer
GLPK
SMT-RAT
Z3
SoPlex
Logger
Plotter
util

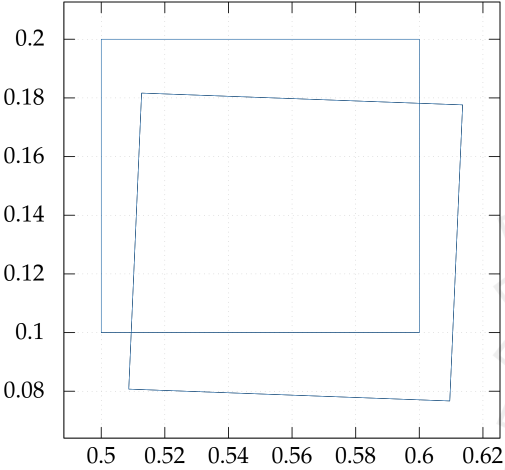
HyDRA: Example





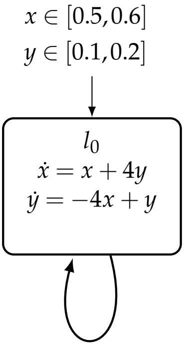
HyDRA: Example

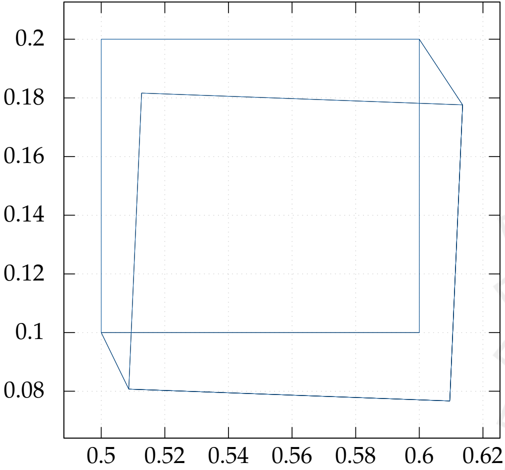




`linear transformation`

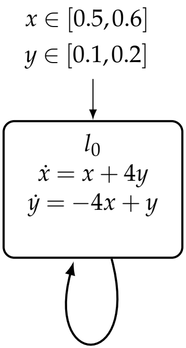
HyDRA: Example

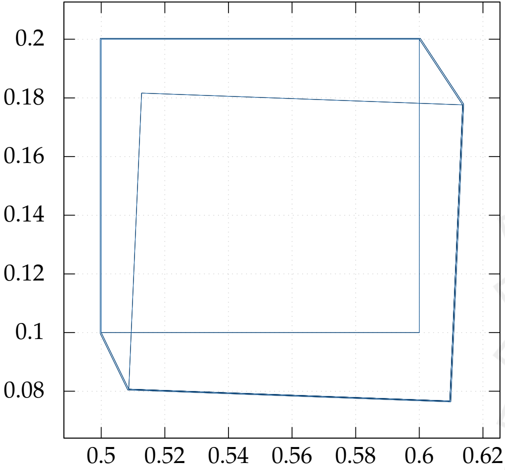




`union`

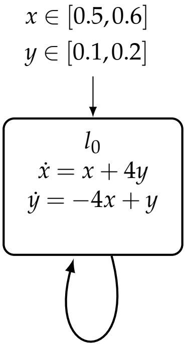
HyDRA: Example

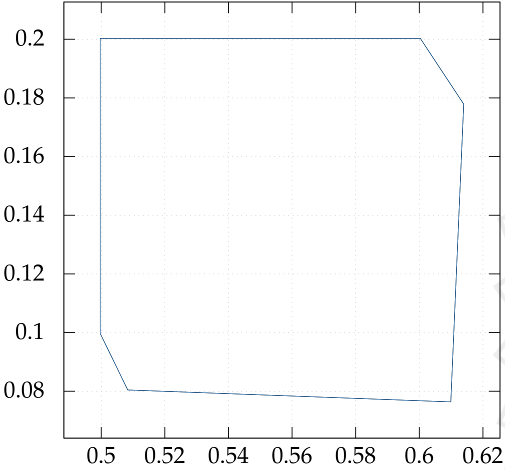




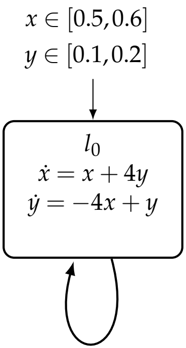
Minkowski sum

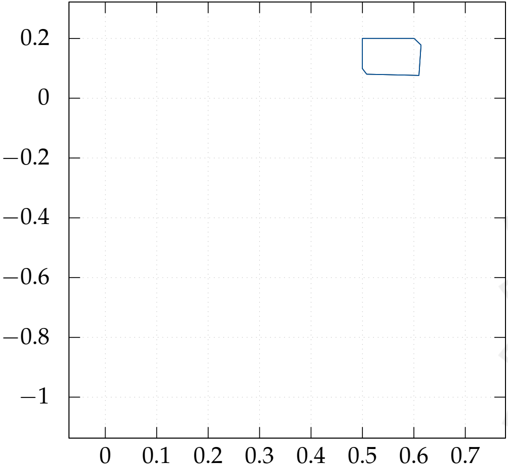
HyDRA: Example



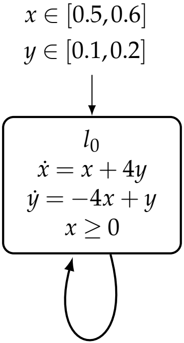


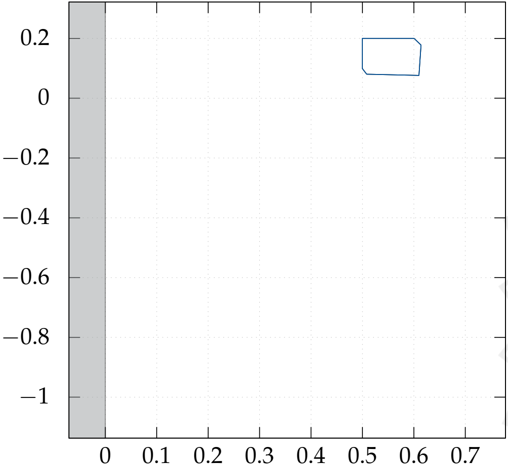
HyDRA: Example





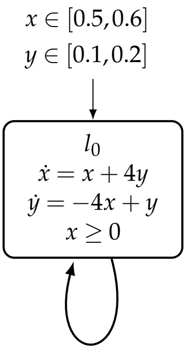
HyDRA: Example

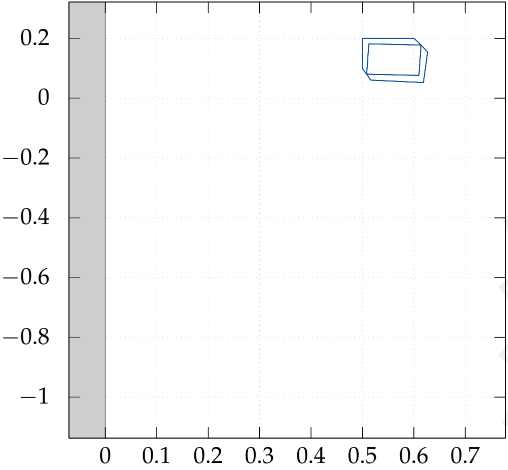




`intersection`

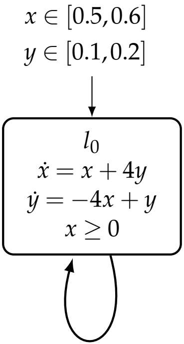
HyDRA: Example

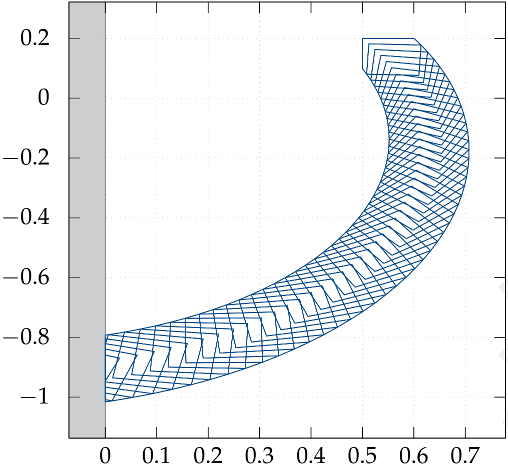




`linear transformation`

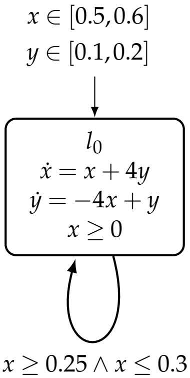
HyDRA: Example

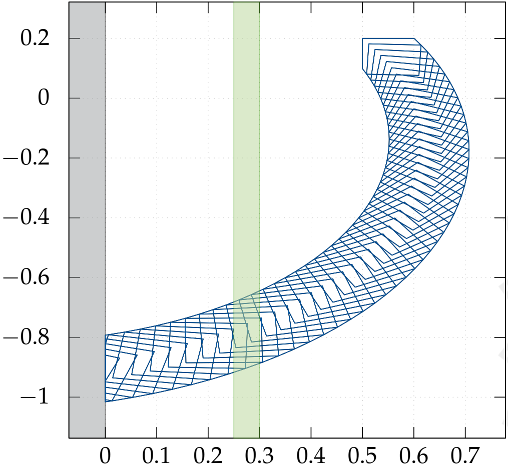




`linear transformation`

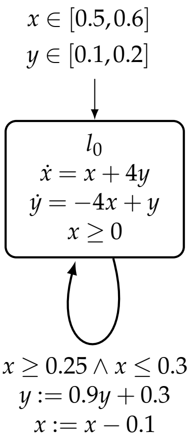
HyDRA: Example

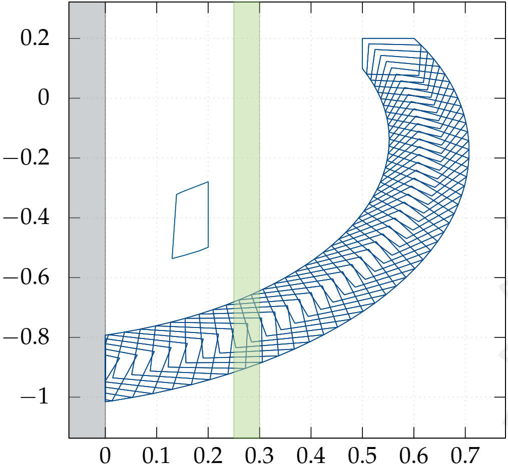




`intersection`

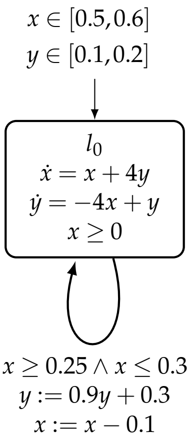
HyDRA: Example

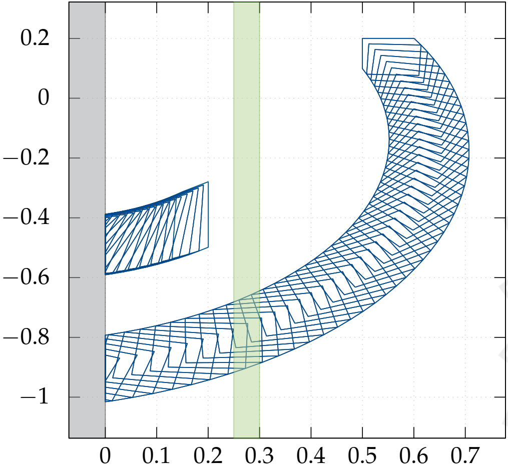




`linear transformation`

HyDRA: Example





`linear transformation`

Other HyPro applications

Two examples for HyPro applications:

- Sub-space computations
- CEGAR-based reachability analysis and parallelisation

HyPro application: Sub-space computations

- Motivation: PLC-controlled plants
- High-dimensional models

HyPro application: Sub-space computations

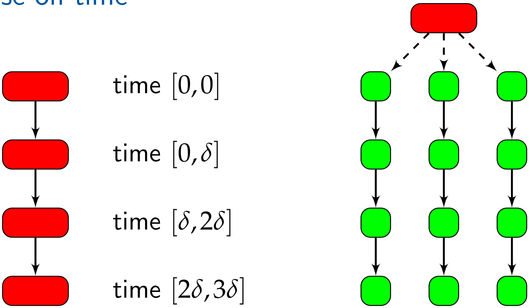
- Motivation: PLC-controlled plants
- High-dimensional models
- Relevant number of discrete variables
- Clocks for cycle synchronisation

HyPro application: Sub-space computations

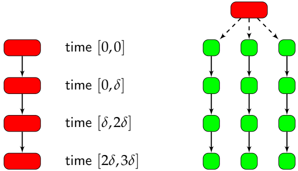
- Motivation: PLC-controlled plants
- High-dimensional models
- Relevant number of discrete variables
- Clocks for cycle synchronisation

Idea:

- Partition variable set $\rightsquigarrow$ sub-spaces
- Compute reachability in sub-spaces
- Synchronise on time

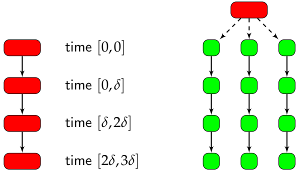


Variable set partitioning



What assures that we can compute locally in sub-spaces?

Variable set partitioning



What assures that we can compute locally in sub-spaces?

All variables x, y in different partitions should be syntactically independent.

$$(\dot{x},\dot{y}) = A \cdot (x,y)^T + B \cdot u$$

$$\dot{x} = A_x \cdot x^T + B_x \cdot u \;\wedge\; \dot{y} = A_y \cdot y^T + B_y \cdot u$$

$$Inv$$

$$Inv_x \;\wedge\; Inv_y$$

$$guard$$

$$guard_x \;\wedge\; guard_y$$

$$reset$$

$$reset_x \;\wedge\; reset_y$$

Global space:

X_0	time $[0,0]$
$\downarrow$	
$P_1 = \text{conv}(X_0 \cup e^{A\delta}X_0 \oplus \mathcal{V}_A \oplus \mathcal{V}_B)$	time $[0,\delta]$
$\downarrow$	
$P_2 = e^{A\delta}P_1 \oplus \mathcal{V}_{B,U}$	time $[\delta,2\delta]$
$\downarrow$	
$P_3 = e^{A\delta}P_2 \oplus \mathcal{V}_{B,U}$	time $[2\delta,3\delta]$

Variable set partitioning

Global space:

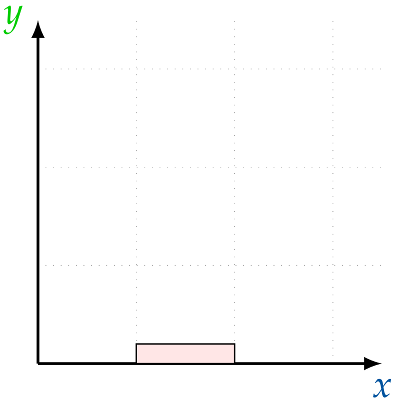
X_0	time $[0,0]$
$P_1 = \mathsf{conv}(X_0 \cup e^{A\delta}X_0 \oplus \mathcal{V}_A \oplus \mathcal{V}_B)$	time $[0,\delta]$
$P_2 = e^{A\delta}P_1 \oplus \mathcal{V}_{B,U}$	time $[\delta,2\delta]$
$P_3 = e^{A\delta}P_2 \oplus \mathcal{V}_{B,U}$	time $[2\delta,3\delta]$

Sub-space:

X_0

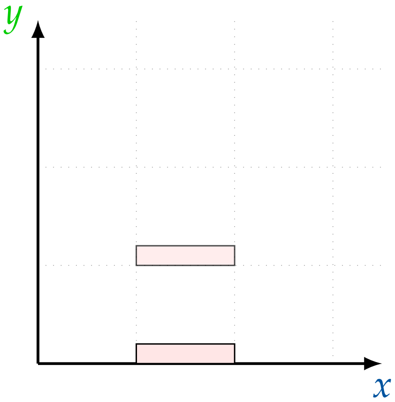
$X_{0,x} = X_0 \downarrow x$	$X_{0,y} = X_0 \downarrow y$	time $[0,0]$
$P_{1,x} = \mathsf{conv}(X_{0,x} \cup e^{A_x\delta}X_{0,x} \oplus \mathcal{V}_{A,x} \oplus \mathcal{V}_{B,x})$	$\cdots$	time $[0,\delta]$
$P_{2,x} = e^{A_x\delta}P_{1,x} \oplus \mathcal{V}_{B_x,u}$	$\cdots$	time $[\delta,2\delta]$
$P_{3,x} = e^{A_x\delta}P_{2,x} \oplus \mathcal{V}_{B_x,u}$	$\cdots$	time $[2\delta,3\delta]$

Discrete variables



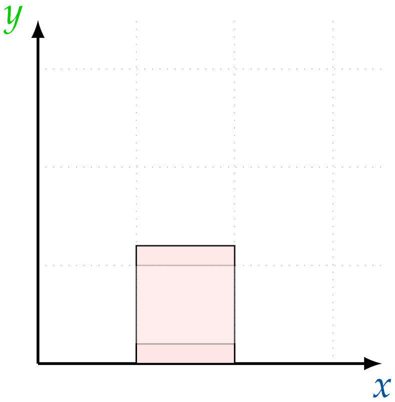
$$\dot{x} = 0$$
$$\dot{y} = 1$$

Discrete variables



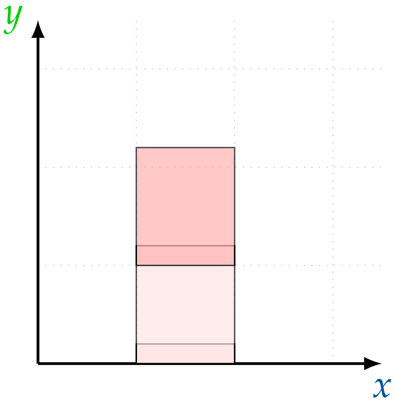
$\dot{x} = 0$
$\dot{y} = 1$

Discrete variables



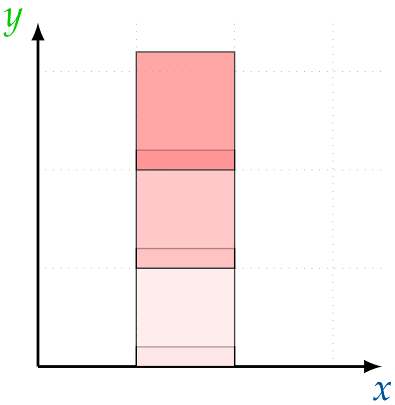
$$\dot{x} = 0$$
$$\dot{y} = 1$$

Discrete variables



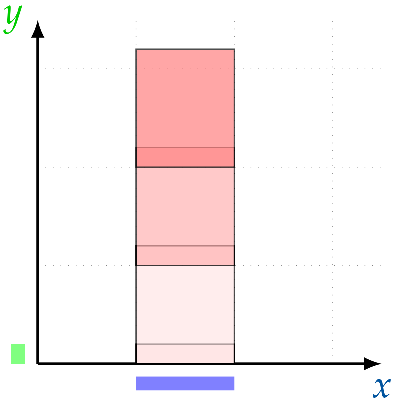
$$\dot{x} = 0$$
$$\dot{y} = 1$$

Discrete variables



$\dot{x} = 0$
$\dot{y} = 1$

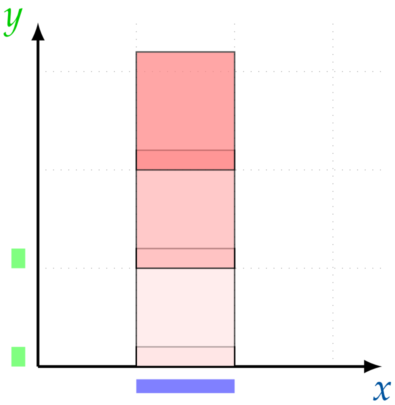
Discrete variables



$\dot{x} = 0$

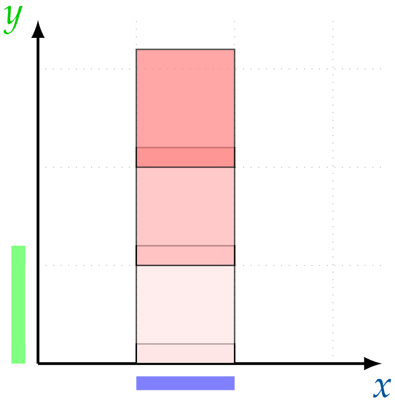
$\dot{y} = 1$

Discrete variables



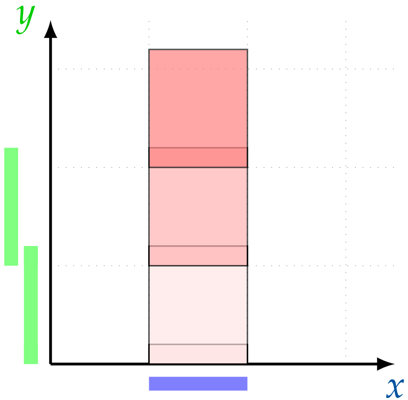
$\dot{x} = 0$

$\dot{y} = 1$



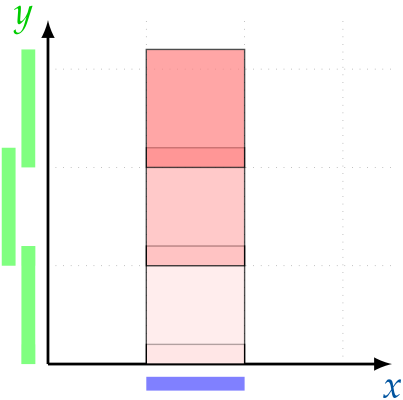
$$\dot{x} = 0$$
$$\dot{y} = 1$$

Discrete variables

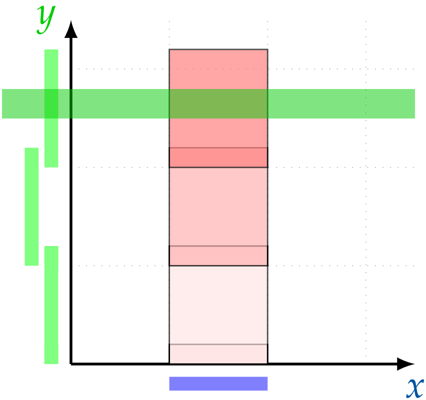


$$\dot{x} = 0$$
$$\dot{y} = 1$$

Discrete variables



$$\dot{x} = 0$$
$$\dot{y} = 1$$

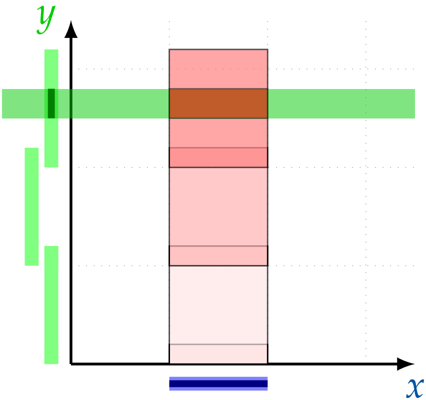


$2.5 \leq y \leq 2.8$

$\dot{x} = 0$

$\dot{y} = 1$

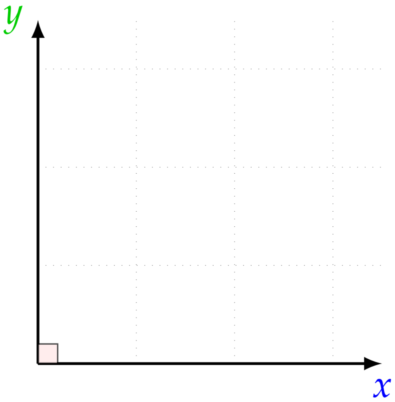
Discrete variables



$2.5 \leq y \leq 2.8$

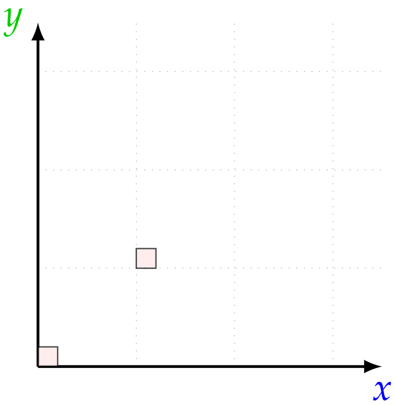
$\dot{x} = 0$
$\dot{y} = 1$

Representation: Boxes



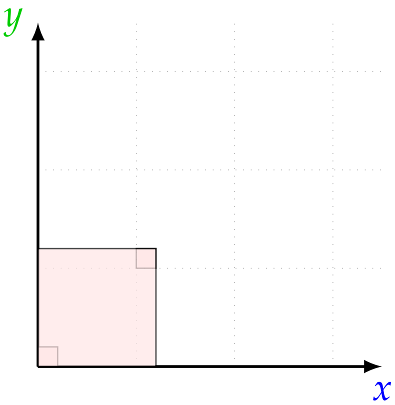
$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Boxes

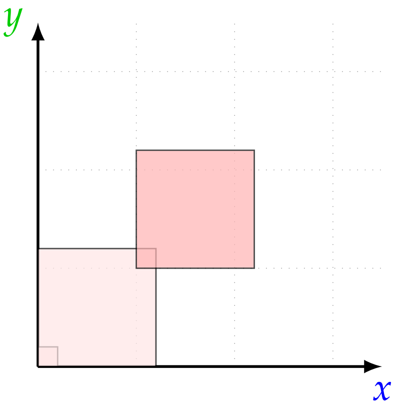


$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Boxes

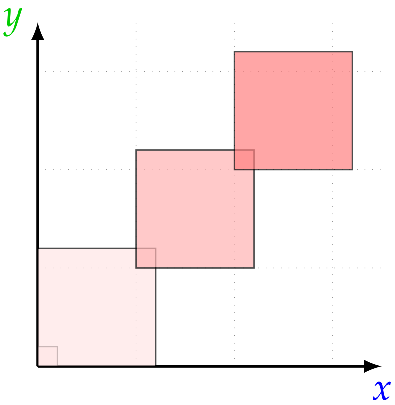


$$\dot{x} = 1$$
$$\dot{y} = 1$$



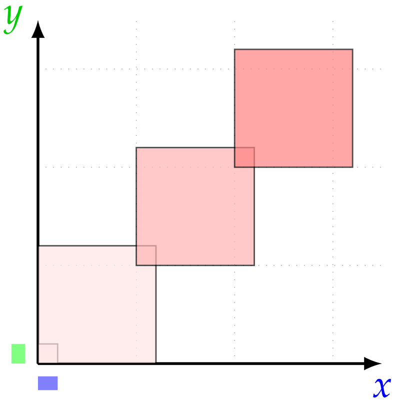
$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Boxes

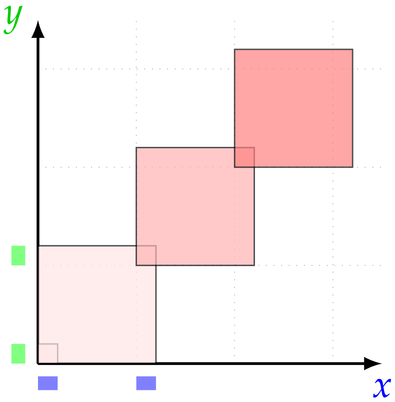


$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Boxes

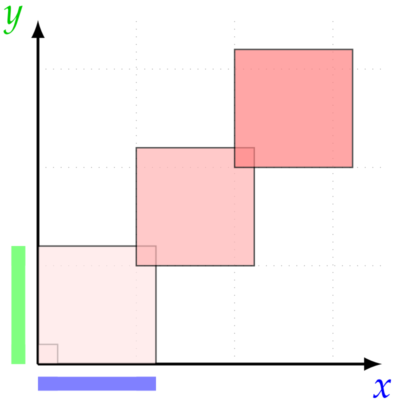


$$\dot{x} = 1$$
$$\dot{y} = 1$$



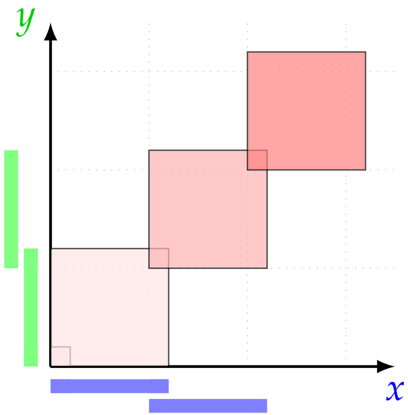
$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Boxes



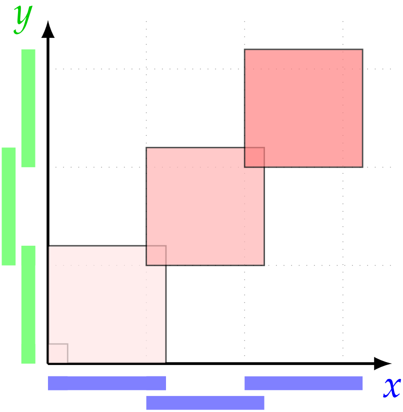
$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Boxes

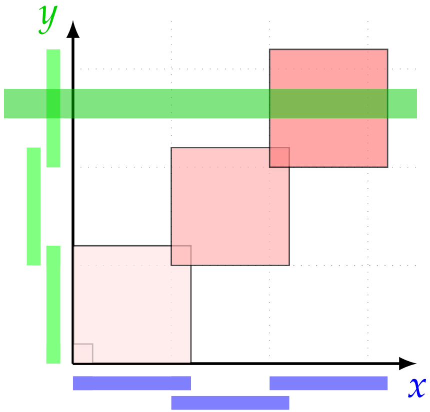


$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Boxes



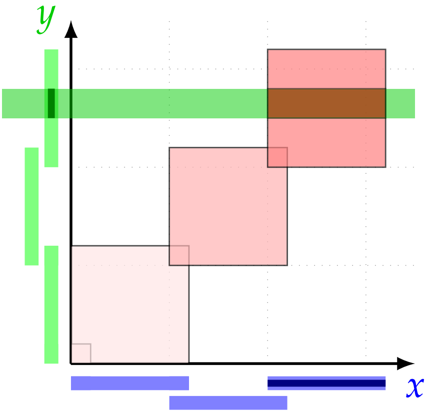
$$\dot{x} = 1$$
$$\dot{y} = 1$$



$2.5 \leq y \leq 2.8$

$\dot{x} = 1$
$\dot{y} = 1$

Representation: Boxes

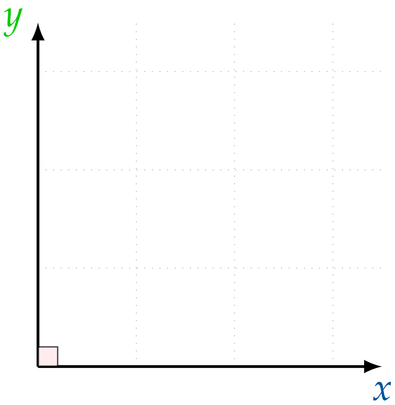


$2.5 \leq y \leq 2.8$

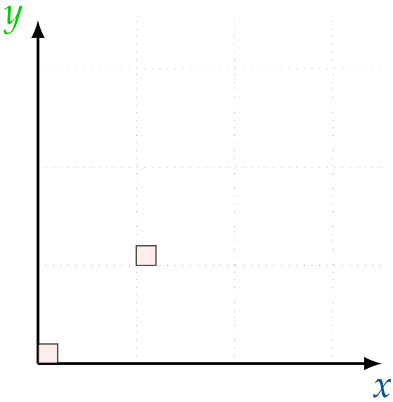
$\dot{x} = 1$

$\dot{y} = 1$

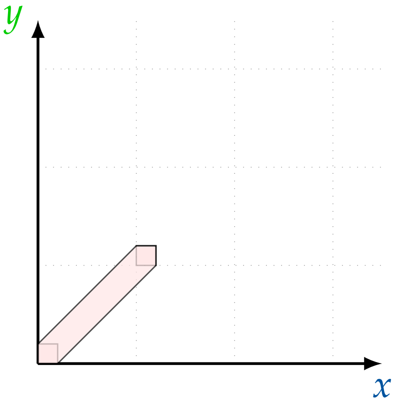
Representation: Polytopes



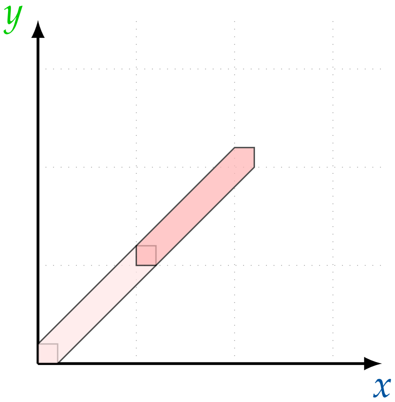
$$\dot{x} = 1$$
$$\dot{y} = 1$$



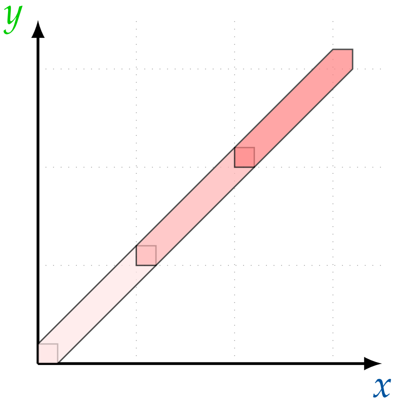
$\dot{x} = 1$
$\dot{y} = 1$



$$\dot{x} = 1$$
$$\dot{y} = 1$$

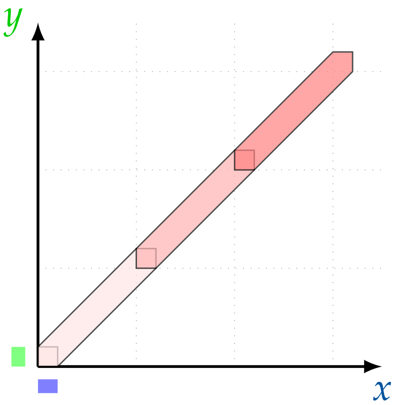


$$\dot{x} = 1$$
$$\dot{y} = 1$$

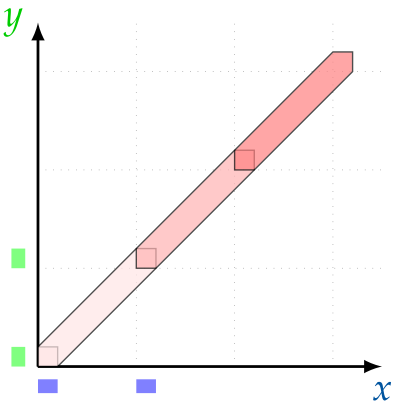


$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Polytopes

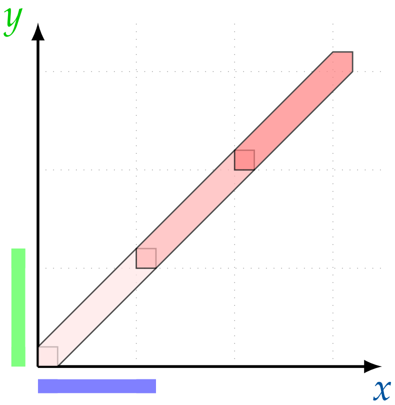


$$\dot{x} = 1$$
$$\dot{y} = 1$$



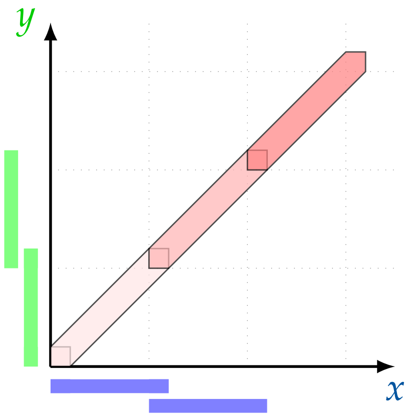
$$\dot{x} = 1$$
$$\dot{y} = 1$$

Representation: Polytopes

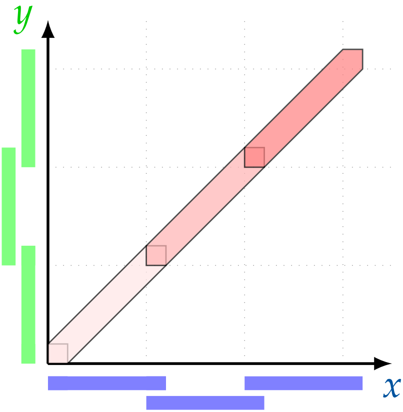


$$\dot{x} = 1$$
$$\dot{y} = 1$$

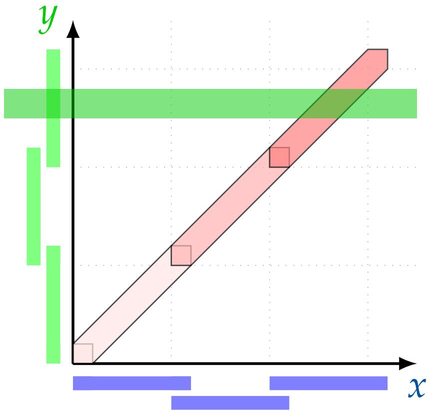
Representation: Polytopes



$$\dot{x} = 1$$
$$\dot{y} = 1$$



$$\dot{x} = 1$$
$$\dot{y} = 1$$

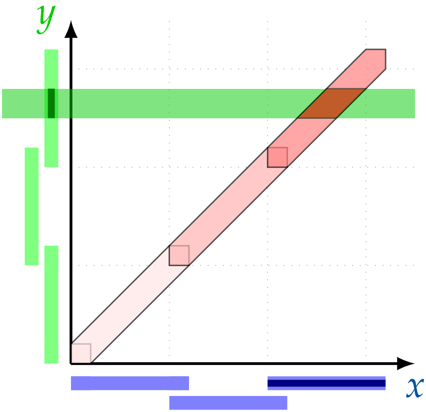


$2.5 \leq y \leq 2.8$

$\dot{x} = 1$

$\dot{y} = 1$

Representation: Polytopes

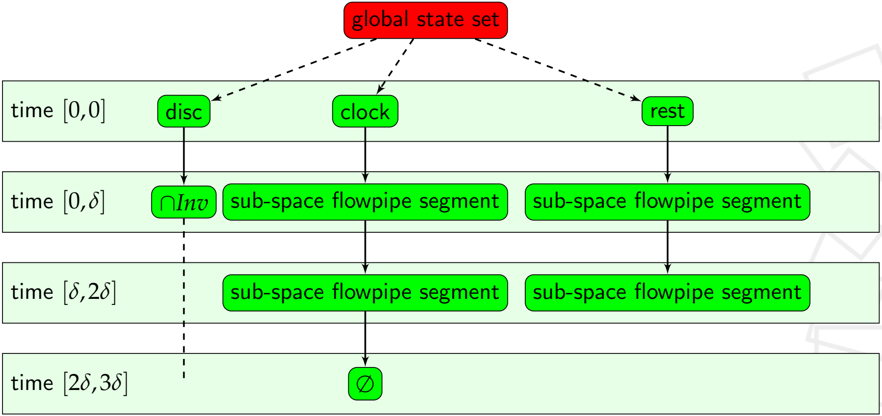


$2.5 \leq y \leq 2.8$

$\dot{x} = 1$

$\dot{y} = 1$

1. Partition variable set
2. Decompose initial state sets
3. Compute successors in sub-spaces
4. Discrete variables: no flowpipe, neglect disabled jumps for whole flowpipe



Some more aspects

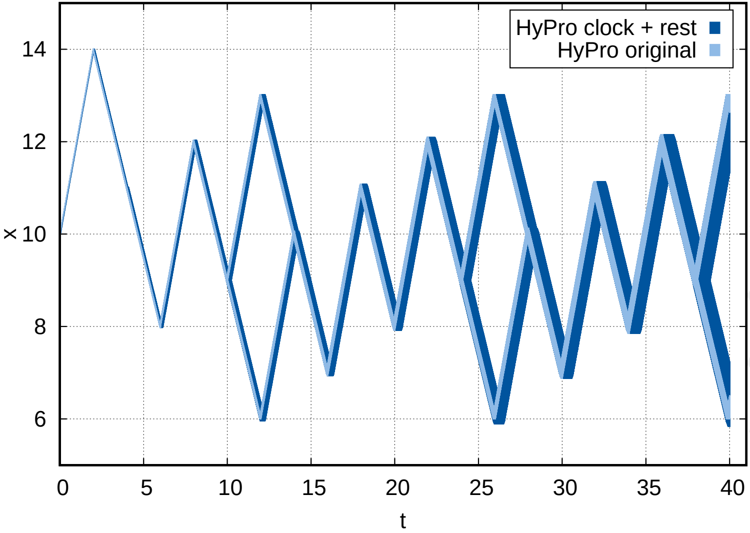
Some more aspects

- We can use different state set representations for different sub-spaces.
- We could even use different reachability analysis methods for different sub-spaces.

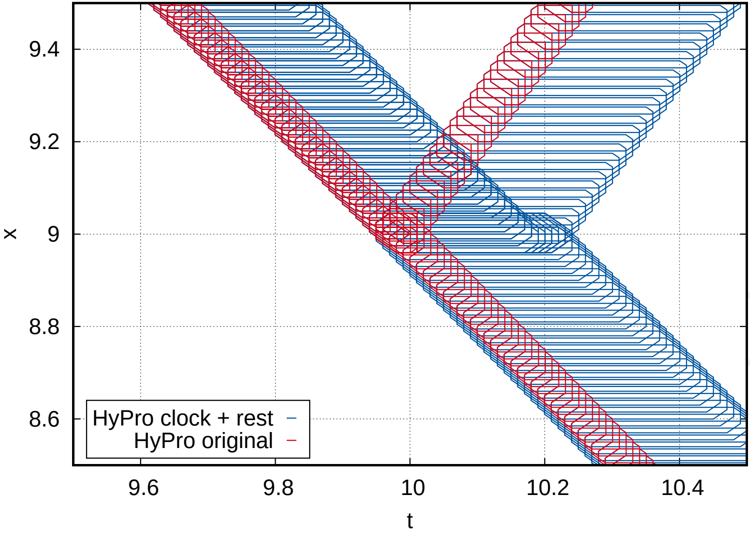
Some more aspects

- We can use different state set representations for different sub-spaces.
- We could even use different reachability analysis methods for different sub-spaces.

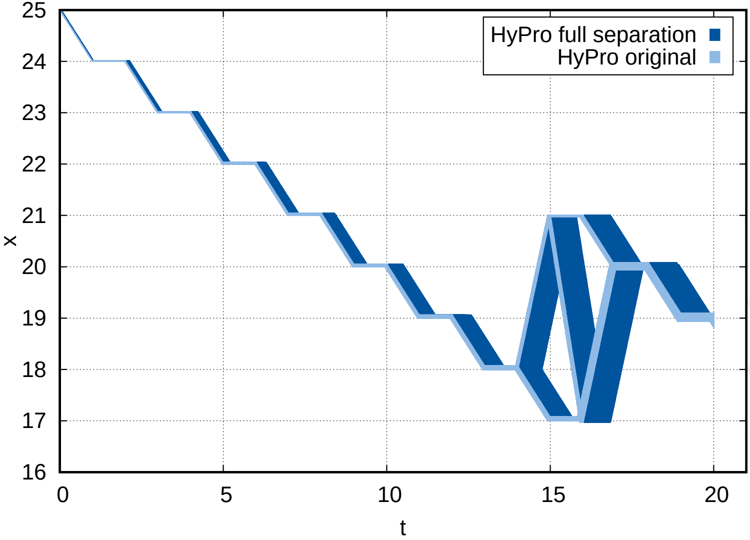
- In our implementation: 3 variable partitions
  - semantically independent discrete variables
  - semantically independent clocks
  - rest
- For discrete variables we use boxes, for the rest support functions.


HyPro clock + rest
HyPro original
x
t
14
12
10
8
6
0
5
10
15
20
25
30
35
40

Results: Leaking tank



Results: Two tanks



Running times

Bench-mark	Rep.	Agg	HyPro orig	HyPro clock	HyPro disc	HyPro disc & clock	SpaceEx orig
Leaking tank	box	agg	2.70	2.08	1.06	1.13	3.67
	box	none	2.62	2.09	1.06	1.13	3.82
	sf	agg	TO	TO	161.12	37.03	448.3
	sf	none	TO	1044.97	19.49	5.84	444.82
Two tanks	box	agg	4.39	2.60	0.97	1.15	5.49
	box	none	4.46	2.68	1.02	1.16	5.53
	sf	agg	TO	TO	900.11	329.80	TO
	sf	none	TO	TO	35.04	14.64	TO
Ther-mostat	box	agg	0.07	0.09	0.06	0.06	0.57
	box	none	0.11	0.09	0.06	0.06	0.57
	sf	agg	35.87	22.69	1.17	0.29	9.89
	sf	none	30.41	20.19	1.18	0.30	9.91

Other HyPro applications

Two examples for HyPro applications:

- Sub-space computations
- CEGAR-based reachability analysis and parallelisation

Other HyPro applications

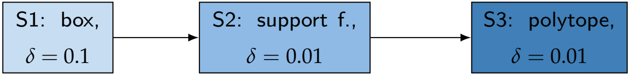
Strategy:

S1: box, $\delta = 0.1$ $\rightarrow$ S2: support f., $\delta = 0.01$ $\rightarrow$ S3: polytope, $\delta = 0.01$

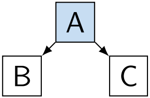
Search tree:

$\boxed{\text{A}}$

Strategy:



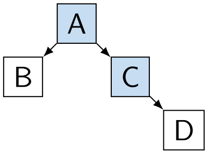
Search tree:



Strategy:

S1: box, $\delta = 0.1$ $\rightarrow$ S2: support f., $\delta = 0.01$ $\rightarrow$ S3: polytope, $\delta = 0.01$

Search tree:

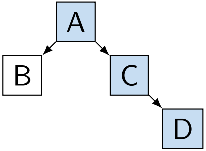


Other HyPro applications

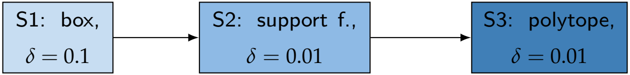
Strategy:

S1: box, $\delta = 0.1$ → S2: support f., $\delta = 0.01$ → S3: polytope, $\delta = 0.01$

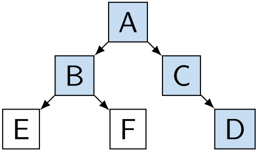
Search tree:



Strategy:

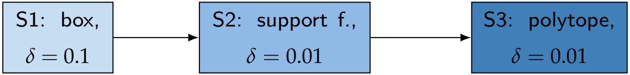


Search tree:

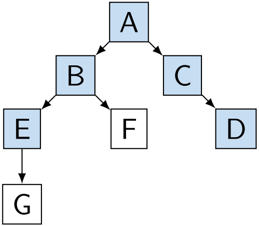


Other HyPro applications

Strategy:



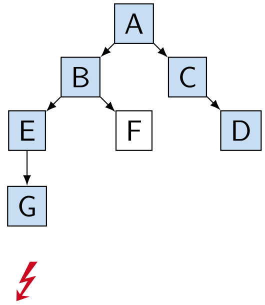
Search tree:



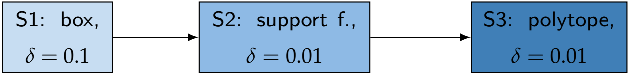
Strategy:

S1: box, $\delta = 0.1$ → S2: support f., $\delta = 0.01$ → S3: polytope, $\delta = 0.01$

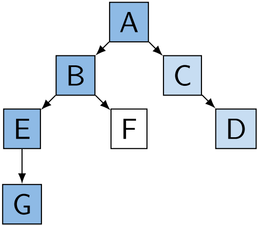
Search tree:



Strategy:



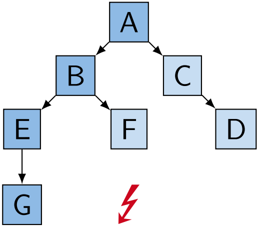
Search tree:



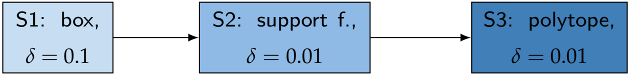
Strategy:

S1: box, $\delta = 0.1$ $\rightarrow$ S2: support f., $\delta = 0.01$ $\rightarrow$ S3: polytope, $\delta = 0.01$

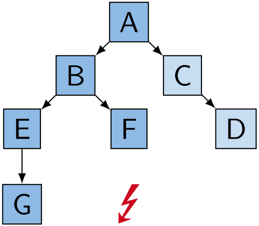
Search tree:



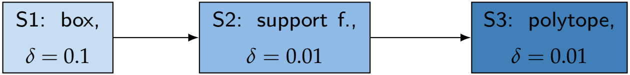
Strategy:



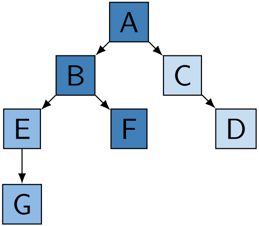
Search tree:



Strategy:

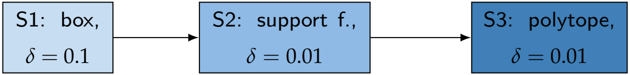


Search tree:

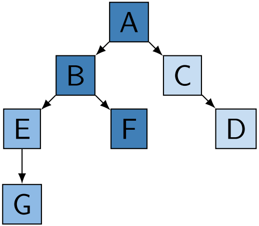


Other HyPro applications

Strategy:



Search tree:



Extension: Parallelized search in different branches.

Conclusion

- HyPro: open-source programming library
- State set representations for the implementation of hybrid systems reachability analysis algorithms
- Exact as well as inexact number representations
- Flexibility to deviate from standard methods
- Examples:
  - sub-space computations
  - CEGAR
  - parallelisation
- Available at https://github.com/hypro/hypro