Modeling and Analysis of Hybrid Systems

Erika Ábrahám

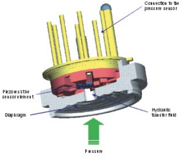
RWTH Aachen University, Germany

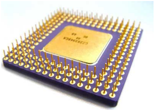
IFIP WG 2.2 meeting Lisbon, September 2013

Hybrid systems

Hybrid systems

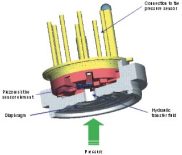
Discrete systems:

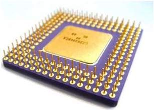




Hybrid systems

Discrete systems:





Combined with dynamic (continuous) behavior:

Contents

Contents

Modeling

Interdisciplinary research area:

Mathematics, Computer Science, Engineering Sciences

Modeling

Interdisciplinary research area:

Mathematics, Computer Science, Engineering Sciences

Modeling

Interdisciplinary research area:

Mathematics, Computer Science, Engineering Sciences

- Modeling formalism depends on the formal methods to be applied
- Engineering: Matlab/Simulink, hybrid SFCs
- Computer Science: Logics, hybrid automata

Modeling

Interdisciplinary research area:

Mathematics, Computer Science, Engineering Sciences

- Modeling formalism depends on the formal methods to be applied
- Engineering: Matlab/Simulink, hybrid SFCs
- Computer Science: Logics, hybrid automata

Modeling

Interdisciplinary research area:

Mathematics, Computer Science, Engineering Sciences

- Modeling formalism depends on the formal methods to be applied
- Engineering: Matlab/Simulink, hybrid SFCs
- Computer Science: Logics, hybrid automata

Time model can be discrete or continuous.

Modeling

Interdisciplinary research area:

Mathematics, Computer Science, Engineering Sciences

- Modeling formalism depends on the formal methods to be applied
- Engineering: Matlab/Simulink, hybrid SFCs
- Computer Science: Logics, hybrid automata

Time model can be discrete or continuous.

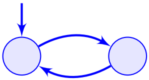
Modeling

Interdisciplinary research area:

Mathematics, Computer Science, Engineering Sciences

- Modeling formalism depends on the formal methods to be applied
- Engineering: Matlab/Simulink, hybrid SFCs
- Computer Science: Logics, hybrid automata

Time model can be discrete or continuous.



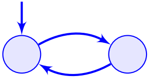
Modeling

Interdisciplinary research area:

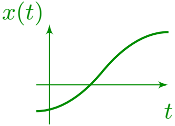
Mathematics, Computer Science, Engineering Sciences

- Modeling formalism depends on the formal methods to be applied
- Engineering: Matlab/Simulink, hybrid SFCs
- Computer Science: Logics, hybrid automata

Time model can be discrete or continuous.



$+$



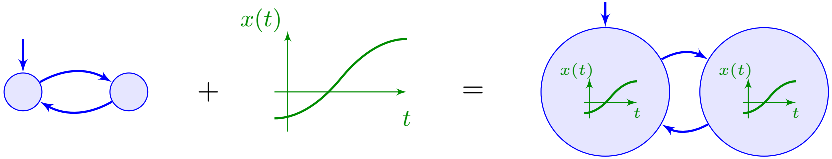
Modeling

Interdisciplinary research area:

Mathematics, Computer Science, Engineering Sciences

- Modeling formalism depends on the formal methods to be applied
- Engineering: Matlab/Simulink, hybrid SFCs
- Computer Science: Logics, hybrid automata

Time model can be discrete or continuous.



Example hybrid automaton: Thermostat

Example hybrid automaton: Thermostat

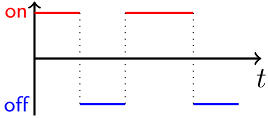
- Temperature x is regulated by a thermostat:

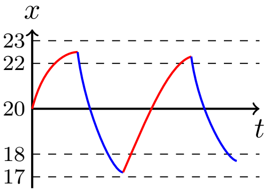
  $17° \leq x \leq 18° \rightsquigarrow$ "heater on" $\quad 22° \leq x \leq 23° \rightsquigarrow$ "heater off"

Example hybrid automaton: Thermostat

- Temperature x is regulated by a thermostat:

  $17° \leq x \leq 18° \rightsquigarrow$ "heater on" $\qquad 22° \leq x \leq 23° \rightsquigarrow$ "heater off"

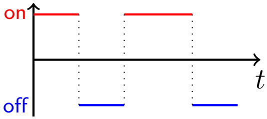


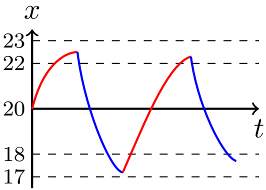


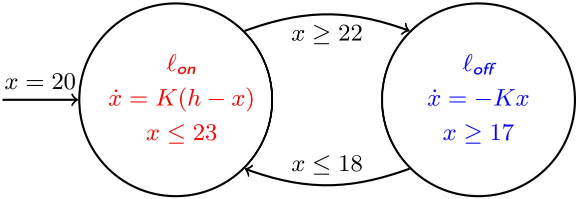
Example hybrid automaton: Thermostat

- Temperature x is regulated by a thermostat:

  $17° \leq x \leq 18° \rightsquigarrow$ "heater on" $\qquad 22° \leq x \leq 23° \rightsquigarrow$ "heater off"







Some interesting subclasses of hybrid automata

subclass	derivatives	conditions	bounded reachability	unbounded reachability
timed automata	$\dot{x} = 1$	$x \sim c$	decidable	decidable
initialized rectangular automata	$\dot{x} \in [c_1, c_2]$ reset by derivative change	$x \sim [c_1, c_2]$	decidable	decidable
linear hybrid automata I	$\dot{x} = c$	$x \sim g_{linear}(\vec{x})$	decidable	undecidable
linear hybrid automata II	$\dot{x} = f_{linear}(\vec{x})$	$x \sim g_{linear}(\vec{x})$	undecidable	undecidable
general hybrid automata	$\dot{x} = f(\vec{x})$	$x \sim g(\vec{x})$	undecidable	undecidable

[Henzinger et al., 1998]

Contents

What is the challenge?

- Hybrid systems are often safety-critical
- Methods for their reachability analysis are needed
- The reachability problem is undecidable for all but some simple subclasses

What is the challenge?

- Hybrid systems are often safety-critical
- Methods for their reachability analysis are needed
- The reachability problem is undecidable for all but some simple subclasses

- Analysis of discrete systems has a long tradition
- Hybrid systems bring new challenges:
  1. represent state sets
  2. methods to analyze the continuous behavior
  3. extensions to cover the discrete behavior

What is the challenge?

- Hybrid systems are often safety-critical
- Methods for their reachability analysis are needed
- The reachability problem is undecidable for all but some simple subclasses

- Analysis of discrete systems has a long tradition
- Hybrid systems bring new challenges:
  1. represent state sets
  2. methods to analyze the continuous behavior
  3. extensions to cover the discrete behavior

- The problems are as expected:
  1. efficiency in computation time
  2. memory consumption
  3. precision

Some tools

- Uppaal [Behrmann et al., 2004]
- HyTech [Henzinger et al., 1997]
- PHAVer [Frehse, 2005]
- SpaceEx [Frehse et al., 2011]
- d/dt [Asarin et al., 2002]
- Ellipsoidal toolbox [Kurzhanski et al., 2006]
- MATISSE [Girard et al., 2007]
- Multi-Parametric Toolbox [Kvasnica et al., 2004]
- Flow* [Chen et al., 2012]

The two most popular techniques for reachability analysis

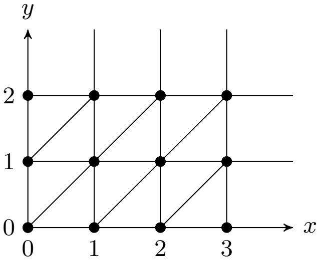
Given: hybrid automaton + set of unsafe states

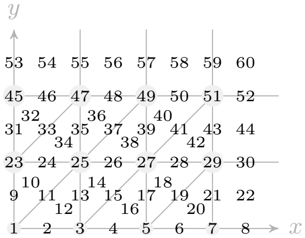
Abstraction	Iterative forward/backward search

Timed automata

[Baier and Katoen, Principles of Model Checking]

timed automaton	$\models_{TCTL}$	φ_{TCTL}
abstraction ↓	$\Updownarrow$	↓ transformation
Kripke structure	$\models_{CTL}$	φ_{CTL}





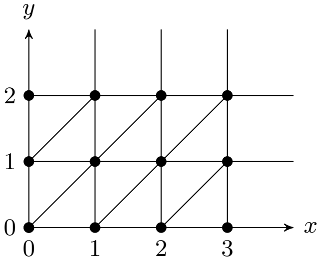
[Baier and Katoen, Principles of Model Checking]

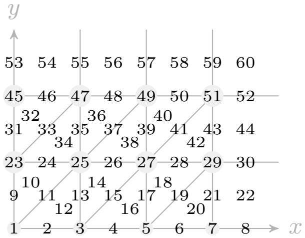
Timed automata

timed automaton $\models_{TCTL}$ φ_{TCTL}

abstraction $\downarrow$ $\quad \updownarrow \quad$ $\downarrow$ transformation

Kripke structure $\models_{CTL}$ φ_{CTL}





- More efficient approach: difference bound matrices

[Baier and Katoen, Principles of Model Checking]

Initialized rectangular automata

[Henzinger et al., 1998]

Initialized rectangular automata

Decidability proof by transformation:

$$\begin{array}{c}\text{Initialized rectangular automaton}\\ \downarrow \\ \text{Initialized singular automaton}\\ \downarrow \\ \text{Initialized stopwatch automaton}\\ \downarrow \\ \text{Timed automaton}\end{array}$$

[Henzinger et al., 1998]

Initialized rectangular automata

Decidability proof by transformation:

$$\begin{array}{c}\text{Initialized rectangular automaton}\\ \downarrow \\ \text{Initialized singular automaton}\\ \downarrow \\ \text{Initialized stopwatch automaton}\\ \downarrow \\ \text{Timed automaton}\end{array}$$

Not used in practice.

[Henzinger et al., 1998]

More general hybrid automata

More general hybrid automata

- Now it gets challenging... for non-constant derivatives even the bounded reachability problem is undecidable.
- Most methods over-approximate reachability.
- Therefore, only safety can be proven.

More general hybrid automata

- Now it gets challenging... for non-constant derivatives even the bounded reachability problem is undecidable.
- Most methods over-approximate reachability.
- Therefore, only safety can be proven.

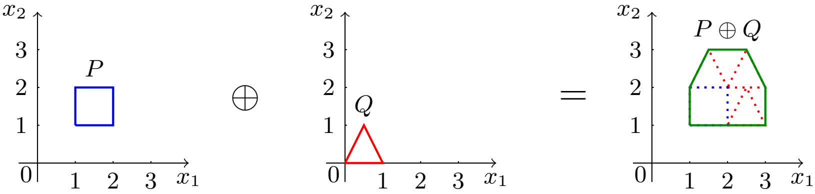
We need an over-approximative state set representation and operations on them like intersection, union, linear transformation and Minkowski sum.

More general hybrid automata

- Now it gets challenging... for non-constant derivatives even the bounded reachability problem is undecidable.
- Most methods over-approximate reachability.
- Therefore, only safety can be proven.

We need an over-approximative state set representation and operations on them like intersection, union, linear transformation and Minkowski sum.

Minkowski sum



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

Most well-known state set representations

Geometric objects:

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

Other symbolic representations:

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

The choice of the representation

The representation is crucial for

- the representation size,
- efficiency and
- accuracy.

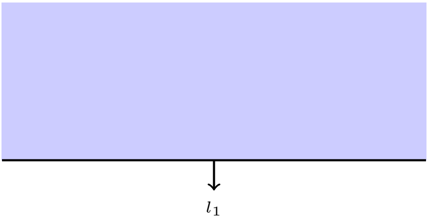
Example: Polytopes

Example: Polytopes

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

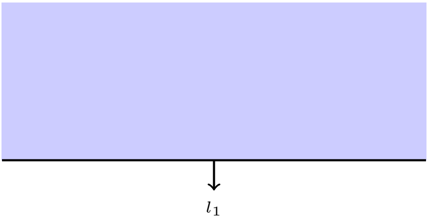
Example: Polytopes

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



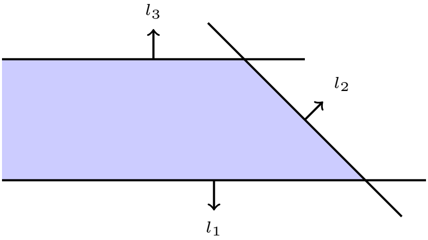
Example: Polytopes

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



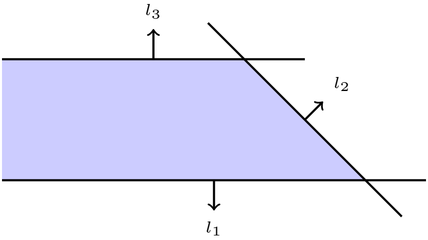
Example: Polytopes

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



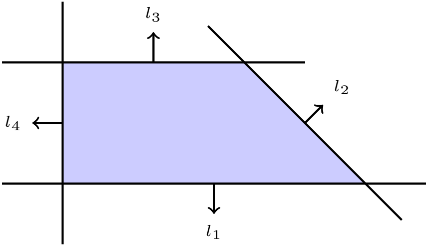
Example: Polytopes

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



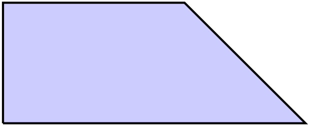
Example: Polytopes

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



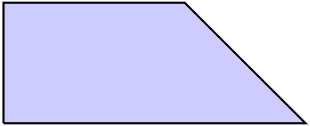
Example: Polytopes

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



Example: Polytopes

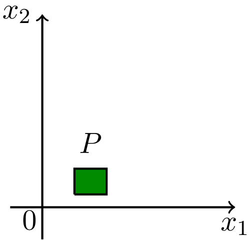
- Halfspace: set of points satisfying $l \cdot x \leq z$
- Polyhedron: an intersection of finitely many halfspaces
- Polytope: a bounded polyhedron



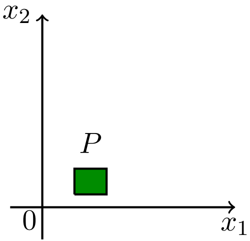
representation	union	intersection	Minkowski sum
$\mathcal{V}$-representation by vertices	easy	hard	easy
$\mathcal{H}$-representation by facets	hard	easy	hard

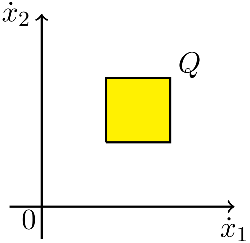
Linear hybrid automata I: Time evolution

Linear hybrid automata I: Time evolution

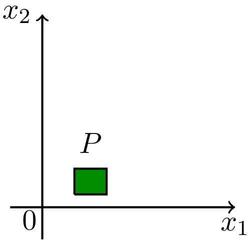


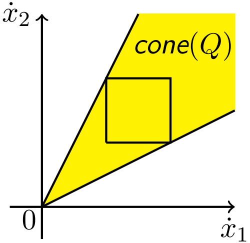
Linear hybrid automata I: Time evolution



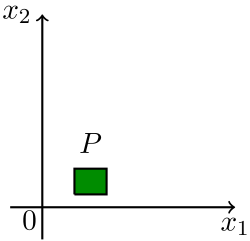


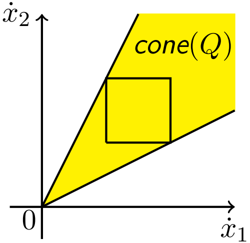
Linear hybrid automata I: Time evolution



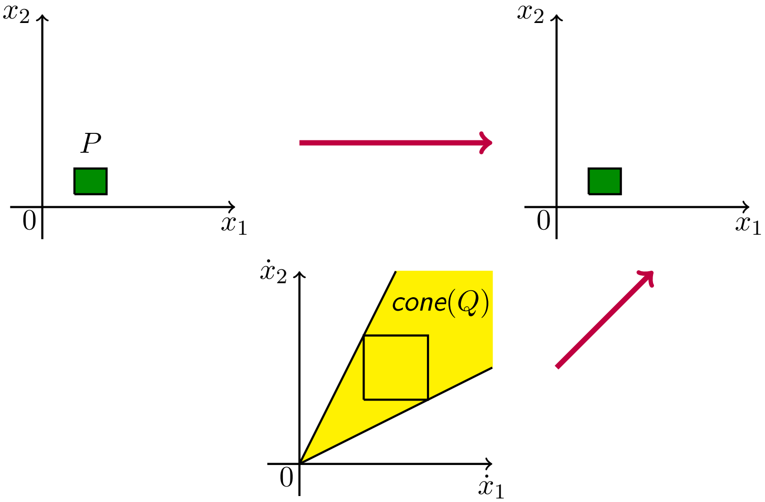


Linear hybrid automata I: Time evolution

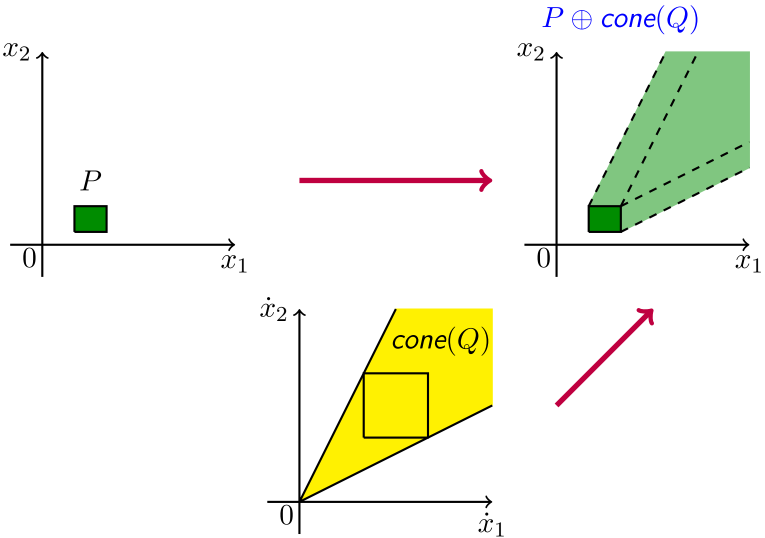




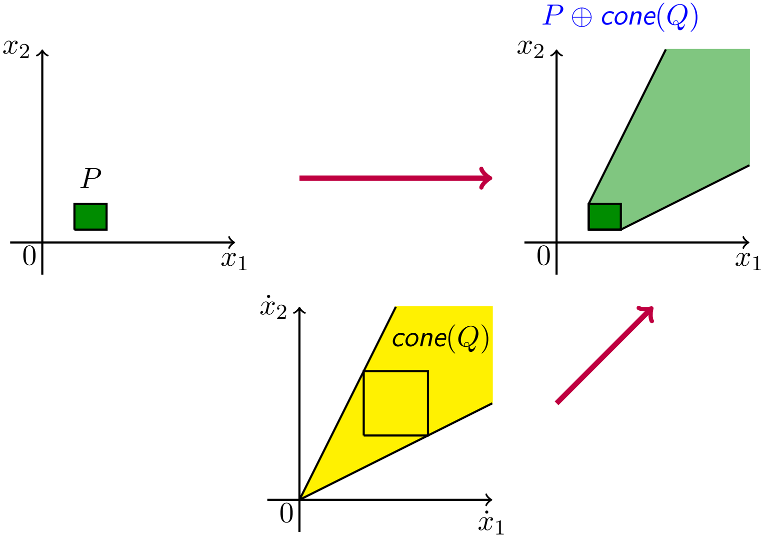
Linear hybrid automata I: Time evolution

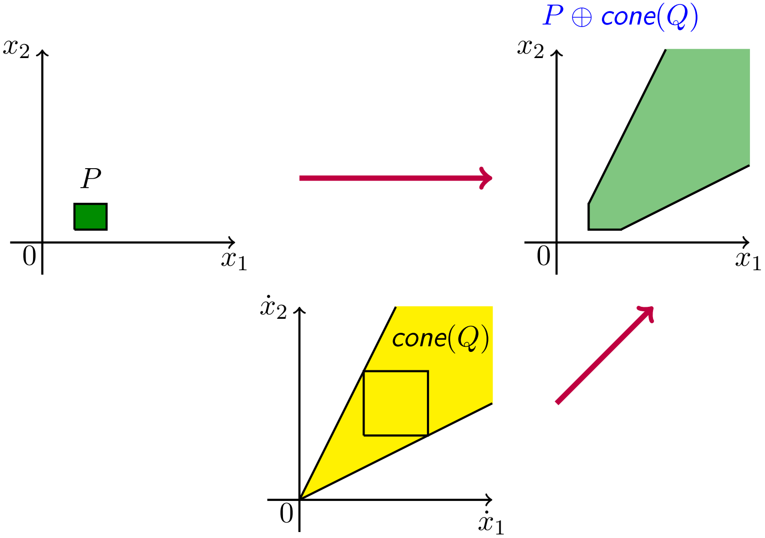


Linear hybrid automata I: Time evolution



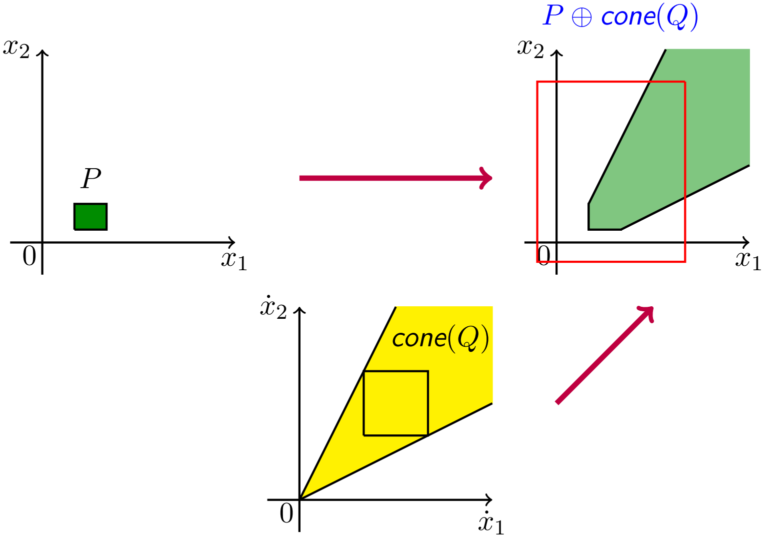
Linear hybrid automata I: Time evolution



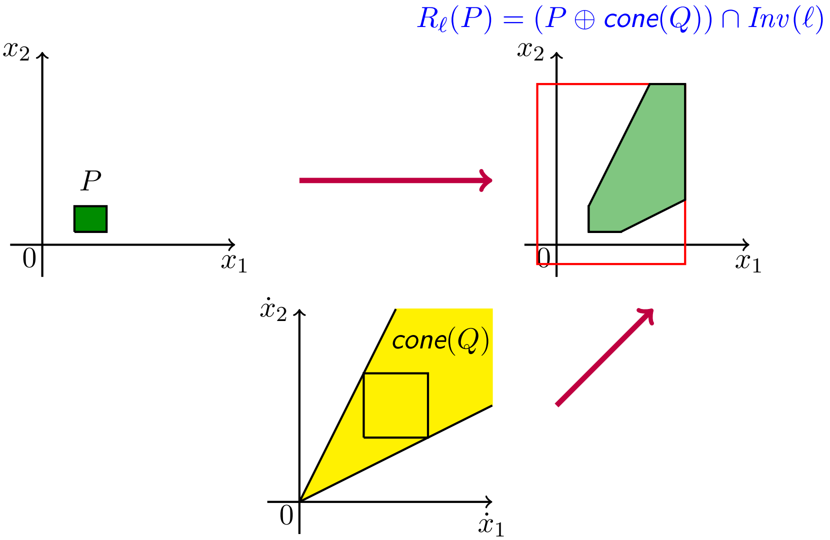


x_2
P
0
x_1
$P \oplus cone(Q)$
x_2
0
x_1
$\dot{x}_2$
$cone(Q)$
0
$\dot{x}_1$

Linear hybrid automata I: Time evolution



Linear hybrid automata I: Time evolution



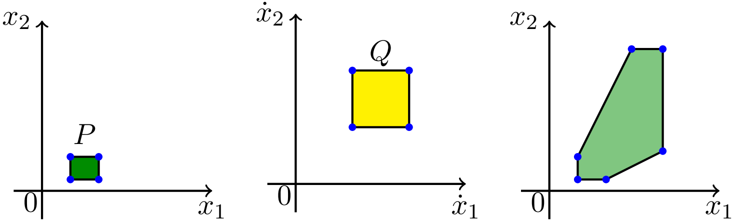
Linear hybrid automata I: Classical representation of $R_\ell(P)$

Linear hybrid automata I: Classical representation of $R_\ell(P)$

- Compute the vertices of $R_\ell(P) = (P \oplus \mathit{cone}(Q)) \cap Inv(\ell)$.

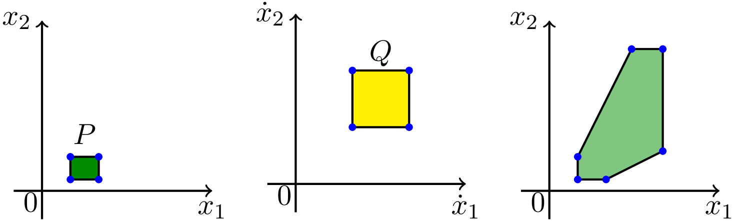
Linear hybrid automata I: Classical representation of $R_\ell(P)$

- Compute the vertices of $R_\ell(P) = (P \oplus \mathit{cone}(Q)) \cap \mathit{Inv}(\ell)$.
- Example:



Linear hybrid automata I: Classical representation of $R_\ell(P)$

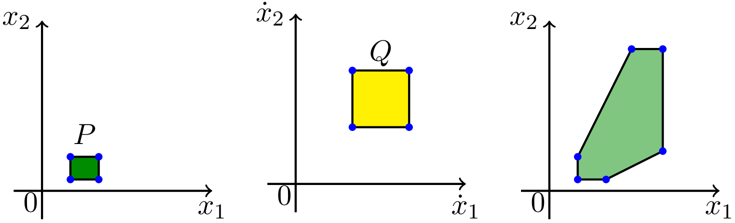
- Compute the vertices of $R_\ell(P) = (P \oplus \textit{cone}(Q)) \cap Inv(\ell)$.
- Example:



- Used by HyTech and PHAVer.

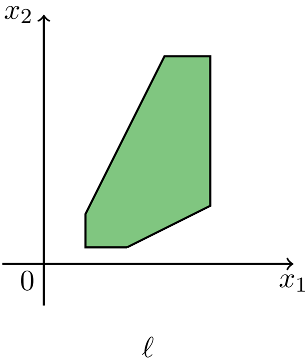
Linear hybrid automata I: Classical representation of $R_\ell(P)$

- Compute the vertices of $R_\ell(P) = (P \oplus \mathit{cone}(Q)) \cap \mathit{Inv}(\ell)$.
- Example:

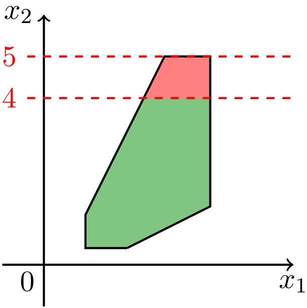


- Used by HyTech and PHAVer.
- Disadvantage: number of vertices might increase exponentially

Linear hybrid automata I: Discrete steps (jumps)

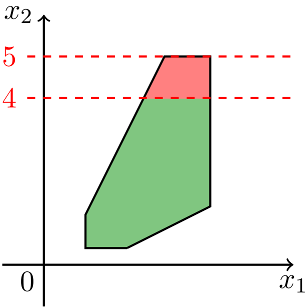


Linear hybrid automata I: Discrete steps (jumps)

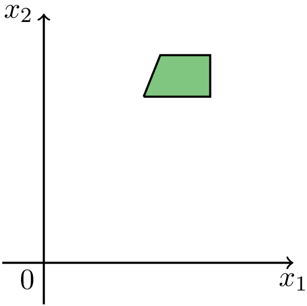


ℓ

Linear hybrid automata I: Discrete steps (jumps)

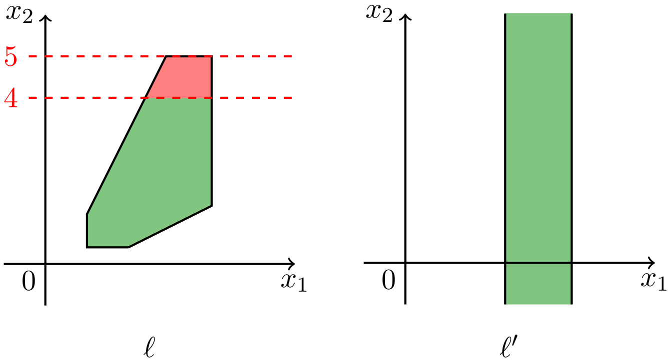


ℓ

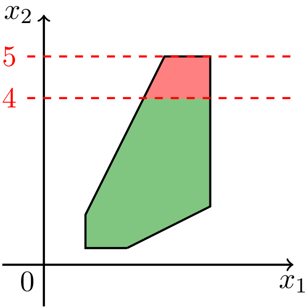


ℓ'

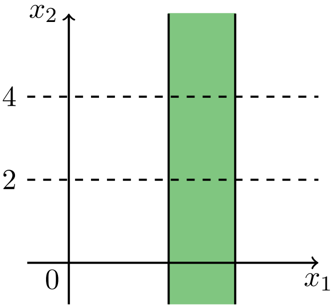
Linear hybrid automata I: Discrete steps (jumps)



Linear hybrid automata I: Discrete steps (jumps)

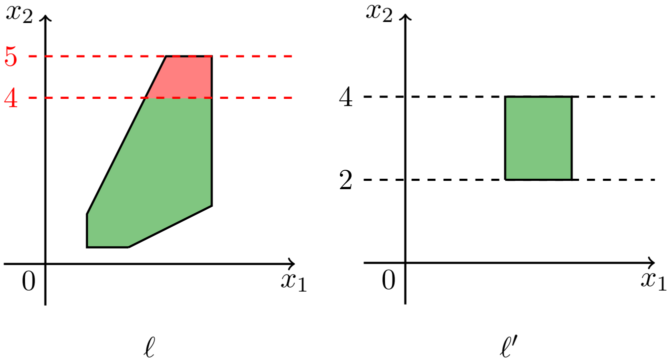


ℓ

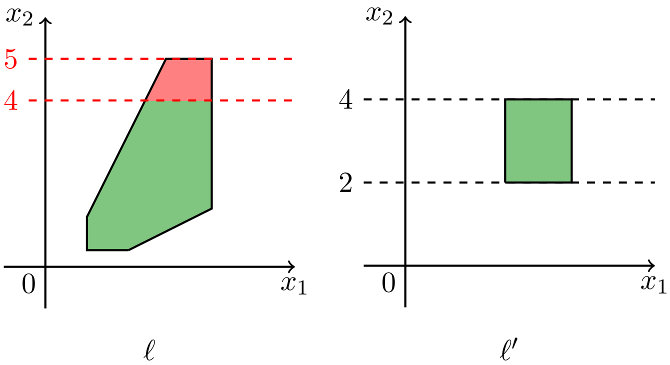


ℓ'

Linear hybrid automata I: Discrete steps (jumps)

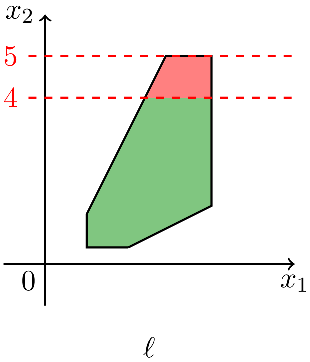


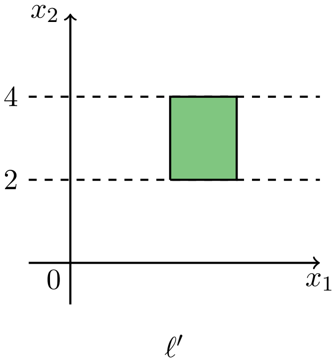
Linear hybrid automata I: Discrete steps (jumps)



- Computed via projection and Minkowski sum.

Linear hybrid automata I: Discrete steps (jumps)





- Computed via projection and Minkowski sum.
- Need to handle exponentially many vertices

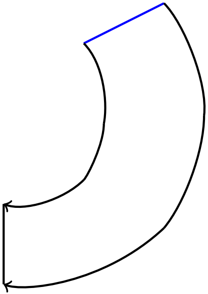
Linear hybrid automata I: Our contribution

Theorem

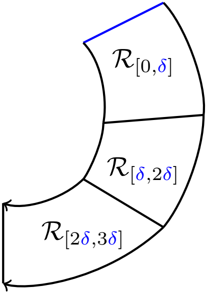
If the state space, dynamics, initial set and unsafe set are all polytopes then bounded reachability can be computed in polynomial time.

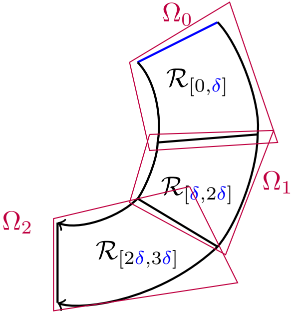
Linear hybrid automata II: Time evolution

Linear hybrid automata II: Time evolution



Linear hybrid automata II: Time evolution



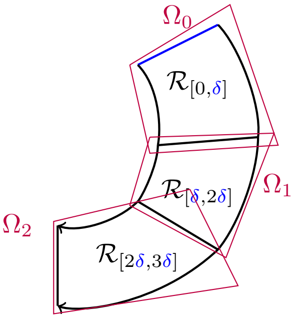

Ω_0
$\mathcal{R}_{[0,\delta]}$
$\mathcal{R}_{[\delta,2\delta]}$
Ω_1
Ω_2
$\mathcal{R}_{[2\delta,3\delta]}$

Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \qquad \subseteq \qquad \Omega_i$$



Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \qquad \subseteq \qquad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments

Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \qquad \subseteq \qquad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:

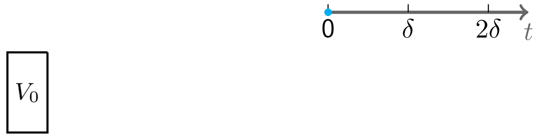
Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \quad \subseteq \quad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:



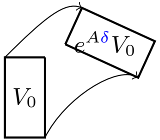
Linear hybrid automata II: Time evolution

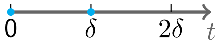
- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \qquad \subseteq \qquad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:





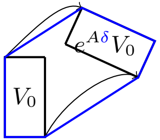
Linear hybrid automata II: Time evolution

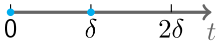
- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \qquad \subseteq \qquad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:





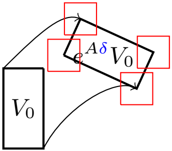
Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \quad \subseteq \quad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:

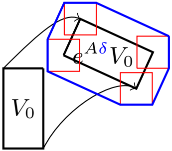


- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \quad \subseteq \quad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:

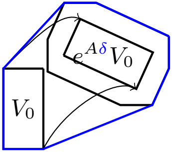


Linear hybrid automata II: Time evolution

- Scheme:

$$\begin{array}{ccc} \dot{x} = Ax + Bu & & \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V} \\ \mathcal{R}_{[i\delta,(i+1)\delta]} & \subseteq & \Omega_i \end{array}$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:



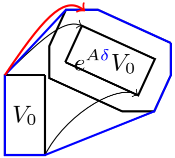
Linear hybrid automata II: Time evolution

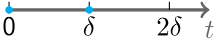
- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \quad \subseteq \quad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:





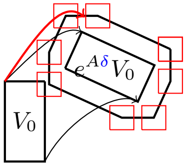
Linear hybrid automata II: Time evolution

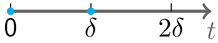
- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \quad \subseteq \quad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:





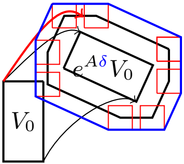
Linear hybrid automata II: Time evolution

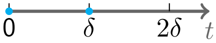
- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \qquad \subseteq \qquad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:





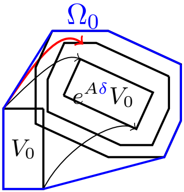
Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad \qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \quad \subseteq \quad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:



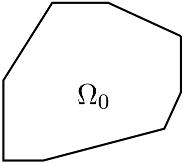
Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \qquad \subseteq \qquad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:

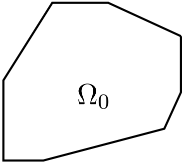


- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \quad \subseteq \quad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:
  2. The remaining ones:

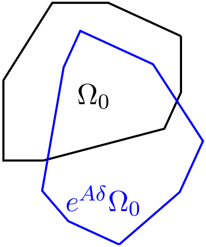


- Scheme:

$$\dot{x} = Ax + Bu \qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:
  2. The remaining ones:

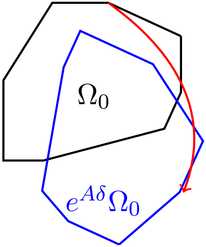


Linear hybrid automata II: Time evolution

- Scheme:

$$\begin{array}{ccc} \dot{x} = Ax + Bu & & \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V} \\ \mathcal{R}_{[i\delta,(i+1)\delta]} & \subseteq & \Omega_i \end{array}$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:
  2. The remaining ones:



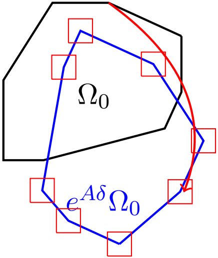
Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:
  2. The remaining ones:



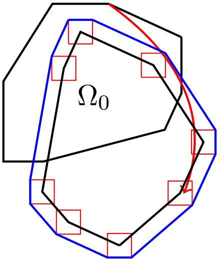
Linear hybrid automata II: Time evolution

- Scheme:

$$\dot{x} = Ax + Bu \qquad\qquad \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V}$$

$$\mathcal{R}_{[i\delta,(i+1)\delta]} \qquad \subseteq \qquad \Omega_i$$

- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:
  2. The remaining ones:

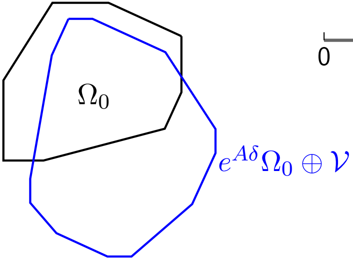


Linear hybrid automata II: Time evolution

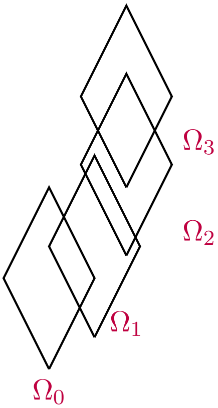
- Scheme:

$$\begin{array}{ccc} \dot{x} = Ax + Bu & & \Omega_{n+1} = e^{A\delta}\Omega_n \oplus \mathcal{V} \\ \mathcal{R}_{[i\delta,(i+1)\delta]} & \subseteq & \Omega_i \end{array}$$

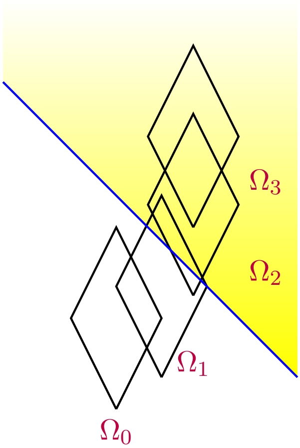
- Flowpipe over-approximation by a set of flowpipe segments
  1. The first flowpipe segment:
  2. The remaining ones:



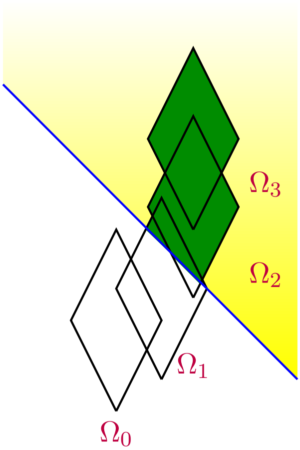
Linear hybrid automata II: Discrete steps (jumps)



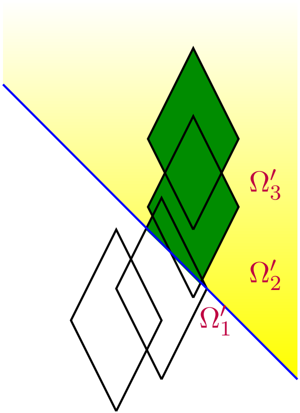
Linear hybrid automata II: Discrete steps (jumps)



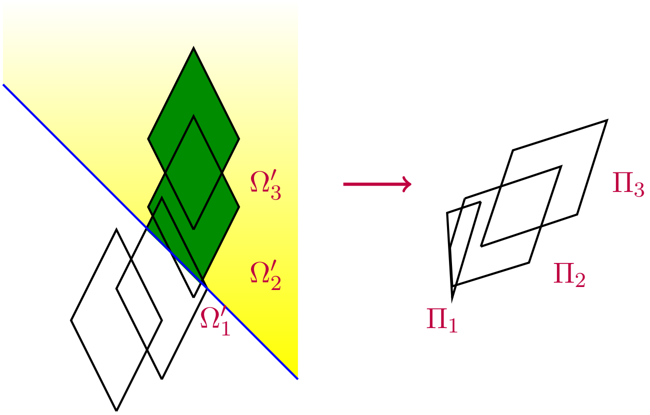
Linear hybrid automata II: Discrete steps (jumps)



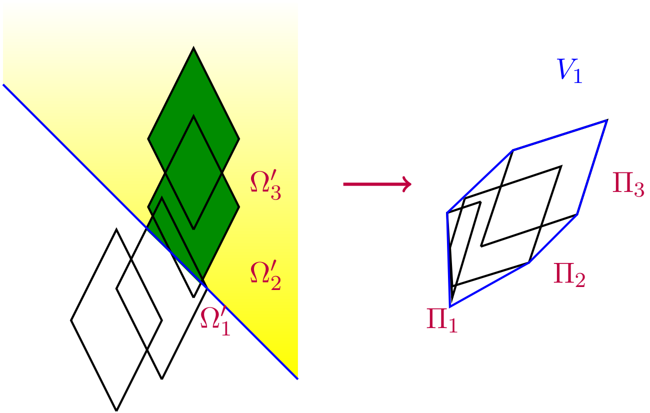
Linear hybrid automata II: Discrete steps (jumps)



Linear hybrid automata II: Discrete steps (jumps)



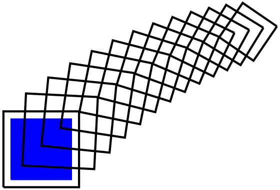
Linear hybrid automata II: Discrete steps (jumps)



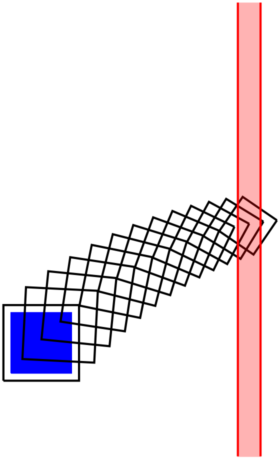
Linear hybrid automata II: The global picture

Linear hybrid automata II: The global picture

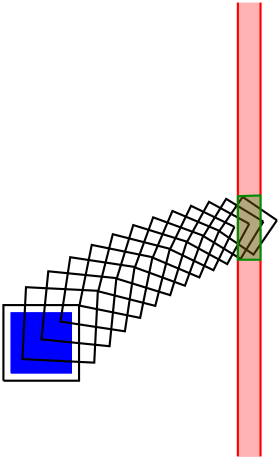
Linear hybrid automata II: The global picture



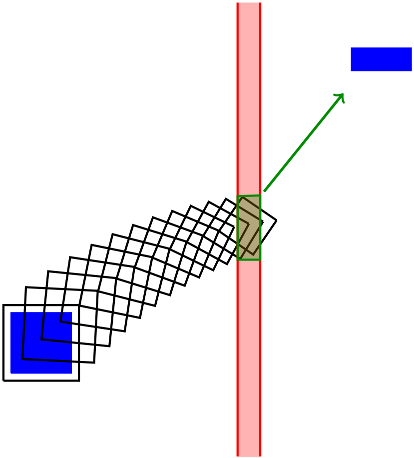
Linear hybrid automata II: The global picture



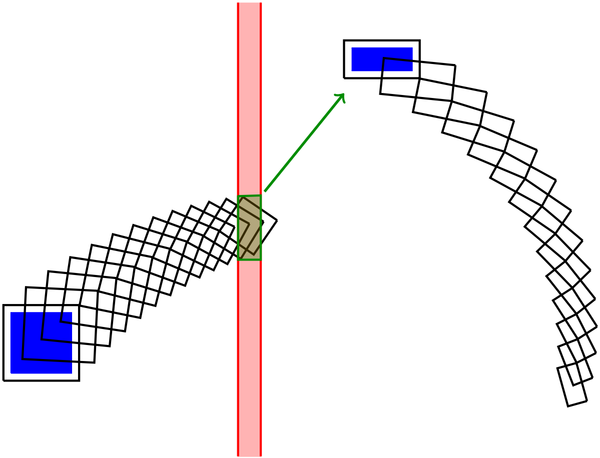
Linear hybrid automata II: The global picture



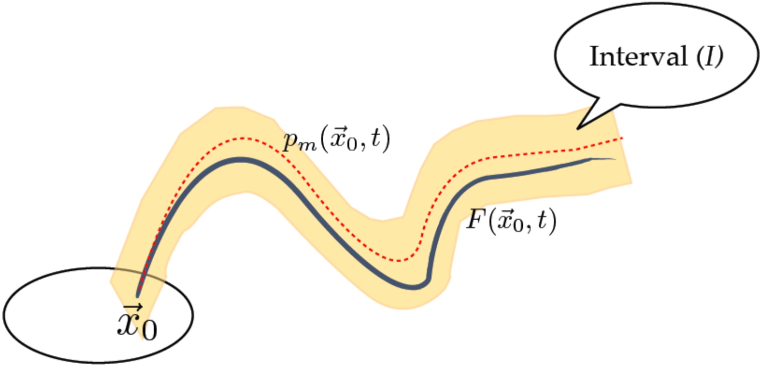
Linear hybrid automata II: The global picture



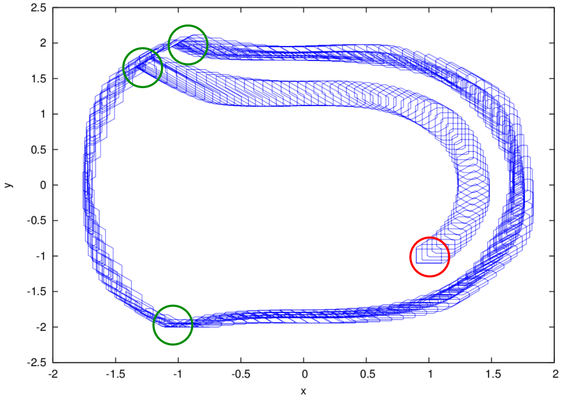
Linear hybrid automata II: The global picture



Our contribution: Taylor model representation of state sets



Example flowpipe computation using Taylor models



Our tool: Flow*

Current features:

- Deals with polynomial hybrid automata
- Uses adaptive orders and step sizes in Taylor model integration
- Includes several heuristics for flowpipe aggregation

Upcoming features:

- Time-varying uncertainties in ODEs
- Non-polynomial terms in ODEs, invariants and guards

Fetch the tool

http://systems.cs.colorado.edu/research/cyberphysical/taylormodels/

Flow*: Taylor Model-Based Analyzer for Hybrid Systems

What is Flow*?

Flow* is a tool which computes *Taylor model flowpipes* for a given continuous or hybrid systems. The current version of Flow* is able to handle hybrid systems with

- continuous dynamics defined by polynomial ordinary differential equations (ODEs),
- mode invariants and jump guards defined by conjunctions of polynomial constraints,
- jump resets defined by polynomial mappings.

What are flowpipes?

There are various definitions on *flowpipes*. Here, a flowpipe means an over-approximation of the reachable states in a time interval (or step).

Why Taylor models?

A Taylor model is the set defined by a polynomial (over an interval domain) bloated by an interval. The flow of a continuous system can be tightly enclosed by Taylor models. With proper interval-based techniques, we may construct Taylor model flowpipes for non-linear hybrid systems.

How to use Flow*?

A user manual can be found here.

Source code

The source code is released under the GNU General Public License (GPL). We are happy to release the code under a license that is more (or less) permissive upon request. source code

Some case studies on Flow* is available now. link

Publications

- Xin Chen, Erika Abraham and Sriram Sankaranarayanan. Flow*: An Analyzer for Non-Linear Hybrid Systems. Computer Aided Verification (CAV), 2013.
- Xin Chen, Erika Abraham and Sriram Sankaranarayanan. Taylor Model Flowpipe Construction for Non-linear Hybrid Systems. IEEE Real-Time Systems Symposium (RTSS), 2012.
- Yan Zhang, Xin Chen, Erika Abraham and Sriram Sankaranarayanan. Empirical Taylor Model Flowpipe Construction for Analog Circuits (Abstract). Frontiers of Analog Computation Workshop, 2013. (slides will be posted soon).

People

Constructing Flowpipes for Continuous and Hybrid Systems: Case-Studies.

Introduction

We present a set of benchmarks of continuous and hybrid systems as long as their running results on the tool Flow*. These studies are intended to benchmark the performance of Flow* tool and serve as a basis of comparison with other tools.

All these studies are run on the following computational platform.

CPU: Intel Core i7-860 Processor (2.80 GHz)
Memory: 4096 MB
System: Ubuntu 12.04 LTS

Continuous-Time Case Studies

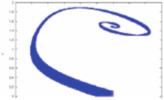
(A) Brusselator

The Brusselator system is a "chemical oscillator" (see here for more details).

The dynamics of a Brusselator are given by

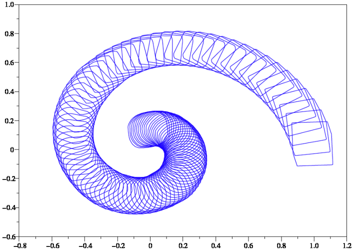
$$\begin{cases} \dot{x} &=& A + x^2 \cdot y - B \cdot x - x \\ \dot{y} &=& B \cdot x - x^2 \cdot y \end{cases}$$

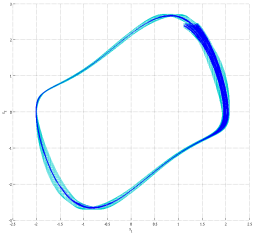
wherein A=1 and B=1.5 in our tests. We let Flow* compute the Taylor model flowpipes for the time horizon **[0,15]**. We first choose the initial set **x in [0.9,1]** and **y in [0,0.1]**, Flow* costs **7 seconds** to generate the flowpipes shown in the figure below. (model file)



Our contribution: Geometric library

- We develop an open-source C++ library supporting different geometric state set representations and operations on them.
- We use this library to implement novel reachability analysis algorithms using
  - convex polyhedra [EUROCAST'11],
  - rectangles [RP'11] or
  - combining geometric objects with Taylor models [RTSS'12, NSV'12, CAV'13].





Other methods: Bounded model checking

Bounded model checking: counterexample search using SMT-solvers

- Formalize safety by a LRA formula *Prop*
- Counterexamples of length k correspond to solutions of

$$\mathit{BMC}_k = \mathit{Init}(s_0) \wedge \mathit{Trans}(s_0, s_1) \wedge \ldots \wedge \mathit{Trans}(s_{k-1}, s_k) \wedge \neg \mathit{Prop}(s_k)$$

- Check BMC_k, $i = 0, 1, \ldots$, for satisfiability

[Biere et al, 2003], [Ábrahám et al., 2005]

Other methods: Bounded model checking

Bounded model checking: counterexample search using SMT-solvers

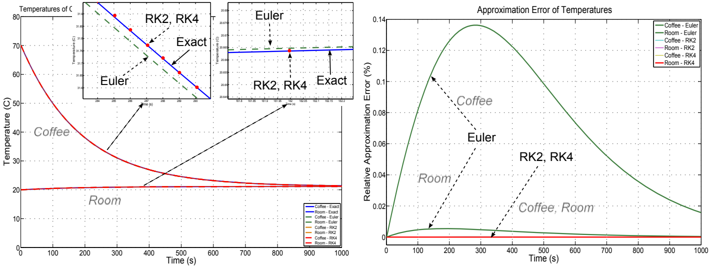
- Formalize safety by a LRA formula *Prop*
- Counterexamples of length k correspond to solutions of

$$\mathit{BMC}_k = \mathit{Init}(s_0) \wedge \mathit{Trans}(s_0, s_1) \wedge \ldots \wedge \mathit{Trans}(s_{k-1}, s_k) \wedge \neg \mathit{Prop}(s_k)$$

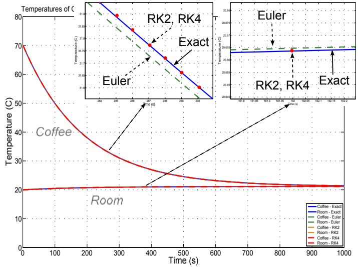
- Check BMC_k, $i = 0, 1, \ldots$, for satisfiability

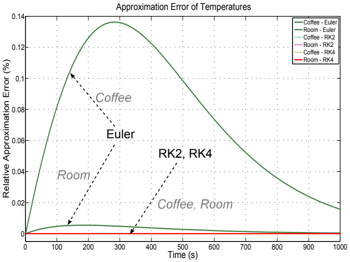
- Popular due to powerful solvers for LRA (HySat/iSAT, Yices/Z3,...)
- More general than reachability (path properties can be checked)
- Can be extended for verification
- Our contribution: SMT-RAT [Corzilius et al., 2012] library of theory solvers

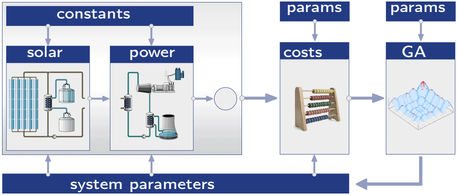
[Biere et al, 2003], [Ábrahám et al., 2005]

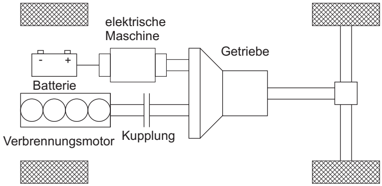

Temperatures of C
RK2, RK4
Exact
Euler
Euler
RK2, RK4
Exact
Coffee
Room
Temperature (C)
Time (s)
Coffee - Exact
Room - Exact
Coffee - Euler
Room - Euler
Coffee - RK2
Room - RK2
Coffee - RK4
Room - RK4
Approximation Error of Temperatures
Relative Approximation Error (%)
Coffee
Euler
Room
RK2, RK4
Coffee, Room
Time (s)

Other methods: Simulation-based approaches









Conclusion

- A lot happened in the last two decades
- There are several approaches and tools for hybrid automata with linear ODEs
- Some approaches are also available for non-linear ODEs
- There is a need for further development in terms of efficiency, scalability and expressivity