Reachability Analysis of Hybrid Systems

Erika Ábrahám

RWTH Aachen University, Germany
Joint work with Xin Chen and Sriram Sankaranarayanan

Nancy, June 2014
Contents

1 Hybrid systems

2 Flowpipe-construction-based reachability analysis

3 Reachability analysis using Taylor models
1 Hybrid systems

2 Flowpipe-construction-based reachability analysis

3 Reachability analysis using Taylor models
Thermostat example

\[ \dot{x} = K(h - x) \quad \text{for} \quad x \leq 23 \]

\[ \dot{x} = -Kx \quad \text{for} \quad x \geq 17 \]
Some interesting subclasses of hybrid automata

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

[Henzinger et al., 1998]
Some tools

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

- **Goal:** over-approximate bounded reachability with time horizon $[0, T]$ and jump depth $k$
Goal: over-approximate bounded reachability with time horizon $[0, T]$ and jump depth $k$

We need a (possibly over-approximative) state set representation and operations on them like intersection, union, linear transformation and Minkowski sum.
Goal: over-approximate bounded reachability with time horizon $[0, T]$ and jump depth $k$

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

The representation is crucial for
- the representation size,
- efficiency and
- accuracy.
Iterative forward search

- **Goal**: over-approximate bounded reachability with time horizon \([0, T]\) and jump depth \(k\).

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

The representation is **crucial** for

- the *representation size*,
- *efficiency* and
- *accuracy*. 
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 et al., 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]
Example: Polytopes
Example: Polytopes
Example: Polytopes

<table>
<thead>
<tr>
<th>representation</th>
<th>union</th>
<th>intersection</th>
<th>Minkowski sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{V}$-representation by vertices</td>
<td>easy</td>
<td>hard</td>
<td>easy</td>
</tr>
<tr>
<td>$\mathcal{H}$-representation by facets</td>
<td>hard</td>
<td>easy</td>
<td>hard</td>
</tr>
</tbody>
</table>
Linear hybrid automata I: Time evolution
Linear hybrid automata I: Time evolution

\[
\dot{x}_1 \dot{x}_2 = 0
\]

\[P \cap (P \oplus \text{cone}(Q)) \cap \text{Inv} \ell\]

\[\text{cone}(Q)\]
Linear hybrid automata I: Time evolution

\[
\begin{align*}
\dot{x}_1 &= 0 \\
\dot{x}_2 &= 0 \\
\end{align*}
\]

\[P \subseteq (P \oplus \text{cone}(Q)) \cap \text{Inv}(\ell)\]
Linear hybrid automata I: Time evolution

\[
\dot{x}_1, \dot{x}_2 \in \text{cone}(Q) \cap \text{Inv}(\ell)
\]

\[
P \oplus \text{cone}(Q)
\]
Linear hybrid automata I: Time evolution

\[ \dot{x}_1, \dot{x}_2 \]

\[ \begin{bmatrix} P & \text{cone}(Q) \end{bmatrix} = P \oplus \text{cone}(Q) \cap \text{Inv}(\ell) \]
Linear hybrid automata I: Time evolution

\[ \dot{x}_1 \]

\[ \dot{x}_2 \]

\[ P \]

\[ x_1 \]

\[ x_2 \]

\[ \text{cone}(Q) \]

\[ \text{Inv}(\ell) \]

\[ x_1 \]

\[ x_2 \]
Linear hybrid automata I: Time evolution

\[ \dot{x}_1, \dot{x}_2 \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]

\[ P \oplus \text{cone}(Q) \]

\[ \text{cone}(Q) \]
Linear hybrid automata I: Time evolution

\[ \dot{x}_1 \]

\[ \dot{x}_2 \]

\[ P \]

\[ cone(Q) \]

\[ P \oplus cone(Q) \]

\[ (P \oplus cone(Q)) \cap Inv(\ell) \]
Linear hybrid automata I: Time evolution

\[ P \oplus \text{cone}(Q) \]

\[ x_2 \]

\[ x_1 \]

\[ 0 \]

\[ P \]

\[ x_2 \]

\[ x_1 \]

\[ 0 \]

\[ \dot{x}_2 \]

\[ \dot{x}_1 \]

\[ \text{cone}(Q) \]
Linear hybrid automata I: Time evolution

\[ P \oplus \text{cone}(Q) \]

\[ \dot{x}_1, \dot{x}_2 \]

\[ P \]

\[ \text{cone}(Q) \]
Linear hybrid automata I: Time evolution

\[
\begin{align*}
\dot{x}_1 &= 0 \\
\dot{x}_2 &= \text{cone}(Q) \\
(P \oplus \text{cone}(Q)) \cap \text{Inv}(\ell)
\end{align*}
\]
Computed via projection and Minkowski sum.
Linear hybrid automata I: Discrete steps (jumps)

Computed via projection and Minkowski sum.
Linear hybrid automata I: Discrete steps (jumps)

\[ \ell \times x_1 \times 2 \]

\[ \ell' \times x_1 \times 2 \]

Computed via projection and Minkowski sum.

Erika Ábrahám - Reachability Analysis of Hybrid Systems
Computed via projection and Minkowski sum.
Computed via projection and Minkowski sum.
Linear hybrid automata I: Discrete steps (jumps)

Computed via projection and Minkowski sum.
Computed via projection and Minkowski sum.
Assume \( \dot{x} = Ax + Bu \)

Compute \( X_0, X_1, \ldots \) such that \( R[\delta, (i+1)\delta] \subseteq X_i \)

The first flowpipe segment:

The remaining ones:

\( t_0 \delta \quad \delta \quad E_{(i+1)\delta} \quad V_0 = e^{-A\delta}V_0 \quad X_0 = e^{-A\delta}X_0 \quad X_1 \)
Assume $\dot{x} = Ax + Bu$
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$

![Diagram showing time evolution of flowpipes with intervals $R[0, \delta]$, $R[\delta, 2\delta]$, $R[2\delta, 3\delta]$]
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{i\delta,(i+1)\delta} \subseteq X_i$

The first flowpipe segment:
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$
- The first flowpipe segment:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$

The first flowpipe segment:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$

The first flowpipe segment:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$

The first flowpipe segment:

\[ e^{A\delta} V_0 \]

\[ V_0 \]
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$

The first flowpipe segment:
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute $X_0, X_1, \ldots$ such that $\mathcal{R}[i\delta, (i+1)\delta] \subseteq X_i$
- The first flowpipe segment:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $R_{[i\delta, (i+1)\delta]} \subseteq X_i$

The first flowpipe segment:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$

The first flowpipe segment:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$

The first flowpipe segment:
Assume $\dot{x} = Ax + Bu$

- Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$
- The first flowpipe segment:
- Assume \( \dot{x} = Ax + Bu \)
- Compute \( X_0, X_1, \ldots \) such that \( R_{[i\delta,(i+1)\delta]} \subseteq X_i \)
- The first flowpipe segment:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$

The first flowpipe segment:
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$
- The first flowpipe segment:
- The remaining ones:
Linear hybrid automata II: Time evolution

- Assume \( \dot{x} = Ax + Bu \)
- Compute \( X_0, X_1, \ldots \) such that \( \mathcal{R}_{i\delta,(i+1)\delta} \subseteq X_i \)
- The first flowpipe segment:
- The remaining ones:
Linear hybrid automata II: Time evolution

- Assume $\dot{x} = Ax + Bu$
- Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta, (i+1)\delta]} \subseteq X_i$
- The first flowpipe segment:
- The remaining ones:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $\mathcal{R}_{[i\delta,(i+1)\delta]} \subseteq X_i$

The first flowpipe segment:

The remaining ones:
Assume $\dot{x} = Ax + Bu$

Compute $X_0, X_1, \ldots$ such that $R_{i\delta,(i+1)\delta} \subseteq X_i$

The first flowpipe segment:

The remaining ones:
Assume \( \dot{x} = Ax + Bu \)

Compute \( X_0, X_1, \ldots \) such that \( R_{i\delta,(i+1)\delta} \subseteq X_i \)

The first flowpipe segment:

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: 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
Difficulties

- Standard techniques work for linear ODEs only
- Overestimation is heavily accumulated
- Flowpipe/guard and flowpipe/invariant intersections are hard to compute
Contents

1 Hybrid systems

2 Flowpipe-construction-based reachability analysis

3 Reachability analysis using Taylor models
Higher-order approximations

Polynomial $p$ is a $k$-order approximation of a smooth $f : D \to \mathbb{R} \in C^k$ iff $f(\vec{c}) = p(\vec{c})$ for the center point $\vec{c}$ of $D$ and for each $0 < m \leq k$:

$$\left. \frac{\partial^m f}{\partial \vec{x}^m} \right|_{\vec{x}=\vec{c}} = \left. \frac{\partial^m p}{\partial \vec{x}^m} \right|_{\vec{x}=\vec{c}}.$$
Several higher-order approximations of $f(x) = e^x$ with $x \in [-1, 1]$
Examples

Several higher-order approximations of $f(x) = e^x$ with $x \in [-1, 1]$

1-order approximation
Several higher-order approximations of $f(x) = e^x$ with $x \in [-1, 1]$
Taylor models

- Taylor model: a pair \((p, I)\) over an interval domain \(D\)
- It defines the set

\[ \{ \vec{x} = p(\vec{x}_0) + \vec{y} \mid \vec{x}_0 \in D \land \vec{y} \in I \} \]

- Taylor model arithmetic: Closed under many basic operations

[Berz et al., 1999]
Higher-order over-approximations by Taylor models

\[ f(x) = e^x \]

\[ p(x) + [-1.8, 1.8] \]

0-order over-approximation
Higher-order over-approximations by Taylor models

\[ f(x) = e^x \]

\[ p(x) = 1 + x \]

\[ p(x) + [-0.8, 0.8] \]

1-order over-approximation
Higher-order over-approximations by Taylor models

\[ f(x) = e^x \]
\[ p(x) = 1 + x + \frac{x^2}{2} \]

2-order over-approximation
Verified integration by Taylor models

Given:

- ODE: \( \frac{d\vec{x}}{dt} = f(\vec{x}, t) \)
- Initial set: Taylor model \( X_0 \) over domain \( D_0 \)

[Berz et al., 1999]
Verified integration by Taylor models

Given:

- ODE: \( \frac{d\vec{x}}{dt} = f(\vec{x}, t) \)
- Initial set: Taylor model \( X_0 \) over domain \( D_0 \)

Integration step for time \([0, \delta]\):

[Berz et al., 1999]
Verified integration by Taylor models

Given:

- ODE: \( \frac{d\vec{x}}{dt} = f(\vec{x}, t) \)
- Initial set: Taylor model \( X_0 \) over domain \( D_0 \)

Integration step for time \([0, \delta]\):

1. Compute a \( k \)-order approximation \( p_k(\vec{x}_0, t) \) over \( X_0 \times [0, \delta] \)
2. Determine remainder \( I_k \) such that \((p_k(\vec{x}_0, t), I_k)\) over \( X_0 \times [0, \delta] \)
   over-approximates the flow pipe segment
3. Initial set for the next flowpipe segment: \((p_k(\vec{x}_0, \delta), I_k)\) over \( X_0 \)

[Berz et al., 1999]
Continuous systems:

Van-der-Pol oscillator  Brusselator  Rössler attractor
Continuous systems:

Van-der-Pol oscillator  Brusselator  Rössler attractor

How can we apply Taylor models to hybrid systems?
Taylor model flowpipe construction for hybrid systems

- Time evolution:
  - Verified integration using Taylor models
  - Compute flowpipe/invariant intersections

- For a discrete transition:
  - Compute flowpipe/guard intersections
  - Compute the image of the reset mapping
Over-approximate an intersection

We use three techniques in combination to over-approximate the intersection \((p, I) \cap G\):

1. Domain contraction
2. Range over-approximation
3. Template method
Example: Domain contraction

\[ D_0 \]

\[ \text{guard} \]

(a)

\[ D_0 \]

\[ \text{new TM} \]

(b)
Glycemic control in diabetic patients

$G$  plasma glucose concentration above the basal value $G_B$

$I$  plasma insulin concentration above the basal value $I_B$

$X$  insulin concentration in an interstitial chamber

\begin{align*}
\frac{dG}{dt} &= -p_1 G - X(G + G_B) + g(t) \\
\frac{dX}{dt} &= -p_2 X + p_3 I \\
\frac{dI}{dt} &= -n(I + I_b) + \frac{1}{V_i} i(t)
\end{align*}

$g(t)$  infusion of glucose into the bloodstream

$i(t)$  infusion of insulin into the bloodstream

\begin{align*}
g(t) &= \begin{cases} 
\frac{t}{60} & t \leq 30 \\
\frac{120-t}{180} & t \in [30, 120] \\
0 & t \geq 120
\end{cases} \\
i(t) &= \begin{cases} 
1 + \frac{2G(t)}{9} & G(t) < 6 \\
\frac{50}{3} & G(t) \geq 6
\end{cases}
\end{align*}
Glycemic Control in Diabetic Patients

Order of the Taylor models: 9    Time step: 0.02    Time horizon: [0,360]
Total time: 1804 s    Time of intersection: 443 s    Memory: 410 MB
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 of 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) blotted by as 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


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

\[
\begin{align*}
\dot{x} &= A + x^2 \cdot y - B \cdot x - x \\
\dot{y} &= B \cdot x - x^2 \cdot y
\end{align*}
\]

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)