Modeling and Analysis of Hybrid Systems
Reachability analysis using Taylor models

Prof. Dr. Erika Ábrahám

Informatik 2 - Theory of Hybrid Systems
RWTH Aachen University

SS 2015
Higher-order approximations

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

$$\frac{\partial^m f}{\partial x^m} \bigg|_{x=c} = \frac{\partial^m p}{\partial x^m} \bigg|_{x=c}.$$
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 $x \in [-1, 1]$

![Graph showing approximations of $f(x) = e^x$ with $x \in [-1, 1]$](image)
Examples

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

![Graph showing approximations of $f(x) = e^x$ with $p(x) = 1 + x + \frac{x^2}{2}$]
Taylor models

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

\[
\{x = p(x_0) + y \mid x_0 \in D \land y \in I\}
\]

- Taylor model arithmetic: Closed under many basic operations

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

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

0-order over-approximation

\[ p(x) = 1 \]

\[ p(x) + [-1.8, 1.8] \]
Higher-order over-approximations by Taylor models

\[ f(x) = e^x = 1 + x + \left[ -0.8, 0.8 \right] \]

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

\[ p(x) + [-0.3, 0.3] \]
Verified integration by Taylor models

Given:

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

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

Given:

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

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

1. Compute a \( k \)-order approximation \( p_k(x_0, t) \) over \( X_0 \times [0, \delta] \)
2. Determine remainder \( I_k \) such that \((p_k(x_0, t), I_k)\) over \( X_0 \times [0, \delta] \) over-approximates the flow pipe segment
3. Initial set for the next flowpipe segment: \((p_k(x_0, \delta), I_k)\) 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
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
1. Domain contraction

Intersection of a Taylor model \((p, I)\) over domain \(D \times T\) and a guard \(G\):

\[
x = p(x_0, t) + y \land y \in I \land x_0 \in D \land t \in T \land G(x)
\]

- Taylor model domain
- Guard predicate
1. Domain contraction

Intersection of a Taylor model \((p, I)\) over domain \(D \times T\) and a guard \(G\):

\[
x = p(x_0, t) + y \land y \in I \land x_0 \in D \land t \in T \land G(x)
\]

Domain contaction:

1. Split the domain in one dimension into two halves
2. Contract dimensions using interval constraint propagation
3. Drop empty (unsatisfiable) domain halves
1. Domain contraction: Example
1. Domain contraction: Example
2. Range over-approximation

1. Over-approximate \((p, I)\) by a set \(S\) which can be a
   - support function
   - Conservative over-approximation of a polynomial
   - zonotope
   - Zonotopes are order 1 Taylor models and vice versa

2. Over-approximate \(S \cap G\) by a Taylor model

[Sankaranarayanan et al., 2008], [Le Guernic et al., 2009]
3. Template method

**Goal:** Find a Taylor model \((p^*, 0)\) over domain

\[
D_u = [\ell_1, u_1] \times [\ell_2, u_2] \times \cdots \times [\ell_n, u_n]
\]

containing the intersection \((p, I) \cap G\)

[Sankaranarayanan et al., 2008], [Gulwani, 2008]
3. Template method

**Goal:** Find a Taylor model \((p^*, 0)\) over domain

\[ D_u = [\ell_1, u_1] \times [\ell_2, u_2] \times \cdots \times [\ell_n, u_n] \]

containing the intersection \((p, I) \cap G\)

**Tasks:**
- Fix \(p^*\)
- Compute proper parameters \(\ell_1, \ldots, \ell_n, u_1, \ldots, u_n\)

[Sankaranarayanan et al., 2008], [Gulwani, 2008]
3. Template method

Goal: Find a Taylor model \((p^*,0)\) over domain

\[
D_u = [\ell_1, u_1] \times [\ell_2, u_2] \times \cdots \times [\ell_n, u_n]
\]

containing the intersection \((p, I) \cap G\)

Tasks:
- Fix \(p^*\)
- Compute proper parameters \(\ell_1, \ldots, \ell_n, u_1, \ldots, u_n\)

\[
\forall x.((x = p(y) + z \land y \in D \land z \in I \land x \in G) \rightarrow \exists x_0.(x = p^*(x_0) \land x_0 \in D_u))
\]

[Sankaranarayanan et al., 2008], [Gulwani, 2008]
Example: 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

\[
g(t) = \begin{cases} 
\frac{t}{60} & t \leq 30 \\
\frac{120 - t}{180} & t \in [30, 120] \\
0 & t \geq 120
\end{cases}
\]

\[
i(t) = \begin{cases} 
1 + \frac{2G(t)}{9} & G(t) < 6 \\
\frac{50}{3} & G(t) \geq 6
\end{cases}
\]
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
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 [8] for more details).

The dynamics of a Brusselator are given by

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

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)

---

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) blasted 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* are available now, [link]

Publications