Hybrid systems in computer science

Example:

Physical systems controlled by programmable logic controllers (PLCs)

Actuators

Sensors

Physical quantities

Plant

PLC

Programs

Input

Output

Such systems are typically complex and safety critical.
Hybrid systems in computer science

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

Plant
- Physical quantities $V_{cont}$
- Actuators $V_{act}$
- Sensors $V_{sen}$

PLC
- Input $V_{in}$
- Output $V_{out}$
- Computation $V_{loc}$

Erika Ábrahám: Hybrid Systems Reachability Analysis
Example:
Physical systems controlled by programmable logic controllers (PLCs)

Such systems are typically complex and safety critical.
Formal verification

System

Requirements

Formal model

Formal specification

Formal verification engine

Satisfied

Violated

Unknown
Formal verification

System
Hybrid system

Requirements
Safety

Formal model
Hybrid automata

Formal specification
Reachability property

Formal verification engine
Flowpipe construction

Satisfied
Violated
Unknown
A hybrid automaton model of a 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 \((\text{dist}_{\text{cur}}, \beta_{\text{ori}})\) to the optimal state \((0,0)\) is always bounded:

  \[
  G(\text{dist}_{\text{cur}} \in [-10,10] \land \beta_{\text{ori}} \in [-5^\circ,5^\circ])
  \]

- The car’s orientation changes smoothly:

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

Source: Möhlmann et al.: *Verifying a PI Controller using SoapBox and Stabhyli*. ARCH 2016
The reachability problem for hybrid systems 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 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.
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](http://symbolaris.com/info/keymaera.html)
Verification techniques/tools for hybrid systems

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/
Verification techniques/tools for hybrid systems

**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

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

\[
\text{Queue} := \{X_0\}; \\
R := \{X_0\}; \\
\text{while } (\text{Queue} \neq \emptyset \text{ and not break\_cond}) \{
\begin{align*}
&\text{Take a set } P \text{ from Queue}; \\
&\text{Compute successor sets } \text{Reach}(P); \\
&\text{Add successor sets to } \text{Queue} \text{ and } R;
\end{align*}
\]

\[
\text{if } (\bigcup_{P \in R} P \cap T = \emptyset) \text{ return } "\text{no}" \text{ else return } "\text{yes}";
\]
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:**

\[
\text{Queue} := \{ X_0 \}; \\
R := \{ X_0 \}; \\
\text{while } (\text{Queue} \neq \emptyset \text{ and not break cond}) \{ \\
\quad \text{Take a set } P \text{ from Queue}; \\
\quad \text{Compute successor sets } \text{Reach}(P); \\
\quad \text{Add successor sets to Queue and } R; \\
\}\; \\
\text{if } (\bigcup_{P \in R} P \cap T = \emptyset) \text{ return ”no” else return ”yes”};
\]

**Problems:**

- How to represent state sets?
- How to compute set operations on them?
- How to compute \( \text{Reach}(\cdot) \)?
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 et al, 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:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>intersection</td>
<td></td>
</tr>
<tr>
<td>union</td>
<td></td>
</tr>
<tr>
<td>linear transformation</td>
<td></td>
</tr>
<tr>
<td>Minkowski sum</td>
<td></td>
</tr>
<tr>
<td>projection</td>
<td></td>
</tr>
<tr>
<td>test for membership</td>
<td></td>
</tr>
<tr>
<td>test for emptiness</td>
<td></td>
</tr>
</tbody>
</table>

Reminder: Minkowski sum

\[ P \oplus Q = \{ p + q | p \in P \text{ and } q \in Q \} \]
Some needed set operations:

- intersection
- union
- linear transformation
- Minkowski sum

Reminder: Minkowski sum

\[ P \oplus Q = \{ p + q \mid p \in P \text{ and } q \in Q \} \]
Example state set representation: Polytopes

Halfspace: set of points $x$ satisfying $\mathbf{l} \cdot x \leq z$

Polyhedron: an intersection of finitely many halfspaces

Polytope: a bounded polyhedron

$\mathbf{l}$

$\mathbf{l}_1$

$\mathbf{l}_2$

$\mathbf{l}_3$

$\mathbf{l}_4$

representation

union

intersection

Minkowski sum

$V$-representation by vertices

easy

hard

easy

$H$-representation by facets

easy

hard

easy
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

![Diagram of a polyhedron with a halfspace plane and a normal vector \( l_1 \)]
Example state set representation: Polytopes

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

![Diagram of polyhedron formed by halfspaces](image)
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
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
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
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

<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 <strong>vertices</strong></td>
<td><strong>easy</strong></td>
<td><strong>hard</strong></td>
<td><strong>easy</strong></td>
</tr>
<tr>
<td>( \mathcal{H} )-representation by <strong>facets</strong></td>
<td><strong>hard</strong></td>
<td><strong>easy</strong></td>
<td><strong>hard</strong></td>
</tr>
</tbody>
</table>
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

\[
\begin{align*}
\dot{x}_1 &= 0 \\
\dot{x}_2 &= 0 \\
X_0 &= X_0 \oplus \text{cone}(Q) \\
\text{Inv}(\ell) &\cap (X_0 \oplus \text{cone}(Q))
\end{align*}
\]
Linear hybrid automata I: Time evolution

initial state set $X_0$

$x_1$ $x_2$

0
Linear hybrid automata I: Time evolution

initial state set \( X_0 \)

derivatives \( Q \)

\[
\begin{align*}
\dot{x}_1 &= 0 \\
\dot{x}_2 &= 0 \\
\dot{x}_1 &= \text{cone}(Q) \\
\dot{x}_2 &= \text{cone}(Q) \\
X_0 &= \text{cone}(Q) \cap \text{Inv}(\ell)
\end{align*}
\]
Linear hybrid automata I: Time evolution

initial state set $X_0$

$\dot{x}_1$

$\dot{x}_2$

derivatives $Q$

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

Initial state set $X_0$

Derivatives $Q$

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

initial state set $X_0$

derivatives $Q$

$X_0 \oplus \text{cone}(Q) \cap \text{Inv}(\ell)$
Linear hybrid automata I: Time evolution

\[ X_0 \oplus \text{cone}(Q) \]

Initial state set \( X_0 \)

Derivatives \( Q \)
Linear hybrid automata I: Time evolution

initial state set $X_0$

$X_0 \oplus \text{cone}(Q)$

derivatives $Q$
Linear hybrid automata I: Time evolution

Initial state set $X_0$

$X_0 \oplus \text{cone}(Q)$
Linear hybrid automata I: Time evolution

initial state set $X_0$

$\dot{x}_1$

$\dot{x}_2$

$X_0 \oplus \text{cone}(Q)$

$\dot{x}_1$

$\dot{x}_2$

$\text{cone}(Q)$

derivatives $Q$

$0$

$X_0 \oplus \text{cone}(Q)$
Linear hybrid automata I: Time evolution

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

initial state set \( X_0 \)

\[
(X_0 \oplus \text{cone}(Q)) \cap \text{Inv}(\ell)
\]
Linear hybrid automata I: Discrete steps (jumps)

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

Additionally: intersect the result with the invariant of \( l' \).

Computed via projection, Minkowski sum and intersection.
Example jump: \((l, 4 \leq x_2 \leq 5, \underbrace{G \equiv \mathbb{R} \times [4,5]}_{\mathbb{R} \equiv \mathbb{R} \times [2,4]}, x_2 : \in [2,4], l') \in \text{Edge}\)
Example jump: \( (l, \quad \begin{aligned} 4 \leq x_2 &\leq 5, \\ G &\equiv \mathbb{R} \times [4, 5] \end{aligned}, \quad \begin{aligned} x_2 :&\in [2, 4], \\ R &\equiv \mathbb{R} \times [2, 4] \end{aligned}, \quad l' \quad ) \in \text{Edge} \)
Example jump: \((l, 4 \leq x_2 \leq 5, \overbrace{G=\mathbb{R} \times [4,5]}^{\text{G}} \quad x_2 \in [2,4], \quad l' ) \in \text{Edge}\)

\[ (P \cap G) \downarrow_{x_1} \times \mathbb{R} \]
Example jump: \((l, 4 \leq x_2 \leq 5, G \equiv \mathbb{R} \times [4,5], x_2 : \in [2,4], l') \in Edge\)

Additionally: intersect the result with the invariant of \(l'\).

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

Example jump: \((l, \begin{cases} 
4 \leq x_2 \leq 5, \\
G \equiv \mathbb{R} \times [4, 5] 
\end{cases}, \begin{cases} 
x_2 \in [2, 4], \\
R \equiv \mathbb{R} \times [2, 4] 
\end{cases}, l') \in Edge\)

\(((P \cap G) \downarrow_{x_1}) \times \mathbb{R}) \cap R\)
Linear hybrid automata I: Discrete steps (jumps)

Example jump: \( (l, 4 \leq x_2 \leq 5, G = \mathbb{R} \times [4,5], x_2 : \in [2,4], l') \in \text{Edge} \)

Additionally: intersect the result with the invariant of \( l' \).

Computed via projection, Minkowski sum and intersection.
Assume initial set $X_0$ and flow $\dot{x} = Ax + Bu$.

Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by $P_{i+1}t_0 \delta X_0 e^{A\delta} X_0 \oplus B_1$ over-approximates flowpipe for time $[0, \delta]$ under dynamics $\dot{x} = Ax$.

$P_0 = conv(X_0, e^{A\delta} X_0 \oplus B_1 \oplus B_2)$ over-approximates flowpipe for time $[0, \delta]$ under dynamics $\dot{x} = Ax + Bu$.

$P_1 = e^{A\delta} P_0 \oplus B_2$ over-approximates flowpipe for time $[\delta, 2\delta]$ under dynamics $\dot{x} = Ax$. 
Linear hybrid automata II: Time evolution

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

Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by $P_i$ time $[0, \delta]$ 

$P_0$ over-approximates flowpipe for time $[0, \delta]$ under dynamics $\dot{x} = Ax + Bu$ 

$P_0 e^{A\delta} P_0$ over-approximates flowpipe for time $[\delta, 2\delta]$ under dynamics $\dot{x} = Ax + Bu$ 

$P_1$ over-approximates flowpipe for time $[\delta, 2\delta]$ under dynamics $\dot{x} = Ax + Bu$ 

$P_0 e^{A\delta} P_0 \oplus B_2$ over-approximates flowpipe for time $[2\delta, 3\delta]$ 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 $P_i$. 

\begin{align*}
P_0 &= \text{conv}(X_0, e^{A\delta}X_0 \oplus B_1 \\ &\quad \oplus B_2) \\
P_1 &= e^{A\delta}P_0 \oplus B_2 \\
P_2 &= \text{conv}(e^{A\delta}P_0 \oplus B_2, e^{2A\delta}P_0 \oplus B_2) \\
\end{align*}

\begin{align*}
\text{time } [0, \delta] \quad P_0 \\
\text{time } [\delta, 2\delta] \quad P_1 \\
\text{time } [2\delta, 3\delta] \quad P_2 \\
\end{align*}
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:

\[
\text{Reminder matrix exponential: }
\exp(X) = \sum_{k=0}^{\infty} X^k k!
\]

\[
\text{The remaining ones: }
t_0 = \delta
\]

\[
\exp(A\delta) \oplus B_1
\]

\[
\text{over-approximates flowpipe for time } [0, \delta]
\]

\[
\exp(A\delta) \oplus B_1 \oplus B_2
\]

\[
\text{over-approximates flowpipe for time } [\delta, 2\delta]
\]
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: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$

$\text{conv}(X_0, e^{A\delta}X_0)$
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: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$
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: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$

**conv($X_0$, $e^{A\delta}X_0 \oplus B_1$)**

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: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$

![Diagram of flowpipe segments and matrix exponential illustration]
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 $\dot{x} = Ax$

over-approximates flowpipe for time $[\delta, 2\delta]$ under dynamics $\dot{x} = Ax + Bu$
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: $e^X = \sum_{k=0}^{\infty} \frac{X^k}{k!}$

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

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:
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:
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:

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

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
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

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)
HyPro: State set representations

Boxes

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

Zonotopes

Support functions

Orthogonal polyhedra

Taylor models

Source: Xin Chen
HyPro/HyDRA: Structure

Hybrid automaton
Point
Halfspace

Hybrid structures

Box
HPolytope
VPolytope
PPL-Polytope
Zonotope
SupportFunction
Orthogonal polyhedra
Taylor model

GeometricObject
<Interface>

Optimizer
GLPK
SMT-RAT
Z3
SoPLEX

Parser
Reachability analysis
Converter

util
Logger
Plotter

algorithms

Erika Ábrahám: Hybrid Systems Reachability Analysis
HyPro: Linear optimization

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

**Obstacles:**
- Inexact linear optimization not suitable
- Exact linear optimization expensive

\[ \Rightarrow \text{combined application} \]
HyPro/HyDRA: Structure

Hybrid automaton

 algorithms
 Parser
 Reachability analysis

Box
HPolytope
VPolytope
PPL-Polytope
Zonotope
SupportFunction
Orthogonal polyhedra
Taylor model

Representations

Converter

Optimizer

GLPK
SMT-RAT
Z3
SoPlex

Reachability analysis

util

Erika Ábrahám: Hybrid Systems Reachability Analysis
HyDRA: Example

\[ x \in [0.5, 0.6] \]
\[ y \in [0.1, 0.2] \]

\[ \dot{x} = x + 4y \]
\[ \dot{y} = -4x + y \]
\[ x \geq 0 \]
\[ x \geq 0.25 \land x \leq 0.3 \]
\[ y := 0.9y + 0.3 \]
\[ x := x - 0.1 \]
$x \in [0.5, 0.6]$
$y \in [0.1, 0.2]$

\[ l_0 \]

\[ \dot{x} = x + 4y \]
\[ \dot{y} = -4x + y \]

linear transformation
HyDRA: Example

\[ x \in [0.5, 0.6] \]
\[ y \in [0.1, 0.2] \]

\[ l_0 \]
\[ \dot{x} = x + 4y \]
\[ \dot{y} = -4x + y \]
HyDRA: Example

\[ x \in [0.5, 0.6] \]
\[ y \in [0.1, 0.2] \]

\[ l_0 \]
\[ \dot{x} = x + 4y \]
\[ \dot{y} = -4x + y \]

Minkowski sum
$x \in [0.5, 0.6]$
$y \in [0.1, 0.2]$

\begin{align*}
 l_0 \\
 \dot{x} &= x + 4y \\
 \dot{y} &= -4x + y
\end{align*}
\[ x \in [0.5, 0.6] \]
\[ y \in [0.1, 0.2] \]

\[ l_0 \]
\[ \dot{x} = x + 4y \]
\[ \dot{y} = -4x + y \]
$x \in [0.5, 0.6]$

$y \in [0.1, 0.2]$

\[ l_0 \]

\[ \dot{x} = x + 4y \]

\[ \dot{y} = -4x + y \]

\[ x \geq 0 \]
$x \in [0.5, 0.6]$

$y \in [0.1, 0.2]$

\[ l_0 \]

\[ \dot{x} = x + 4y \]

\[ \dot{y} = -4x + y \]

\[ x \geq 0 \]

linear transformation

Erika Ábrahám: Hybrid Systems Reachability Analysis
\( x \in [0.5, 0.6] \)
\( y \in [0.1, 0.2] \)

\[
\begin{align*}
\dot{x} &= x + 4y \\
\dot{y} &= -4x + y \\
x &\geq 0
\end{align*}
\]
HyDRA: Example

\[ x \in [0.5, 0.6] \]
\[ y \in [0.1, 0.2] \]

\[ l_0 \]
\[ \dot{x} = x + 4y \]
\[ \dot{y} = -4x + y \]
\[ x \geq 0 \]

\[ x \geq 0.25 \land x \leq 0.3 \]
HyDRA: Example

\[ x \in [0.5, 0.6] \]
\[ y \in [0.1, 0.2] \]

\[
l_0
\]
\[
\dot{x} = x + 4y
\]
\[
\dot{y} = -4x + y
\]
\[ x \geq 0 \]

\[ x \geq 0.25 \land x \leq 0.3 \]
\[ y := 0.9y + 0.3 \]
\[ x := x - 0.1 \]

linear transformation
HyDRA: Example

\[ x \in [0.5, 0.6] \]
\[ y \in [0.1, 0.2] \]

\[ l_0 \]
\[ \dot{x} = x + 4y \]
\[ \dot{y} = -4x + y \]
\[ x \geq 0 \]

\[ x \geq 0.25 \land x \leq 0.3 \]
\[ y := 0.9y + 0.3 \]
\[ x := x - 0.1 \]

linear transformation
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
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**.

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

\[
\dot{x} = A_x \cdot x^T + B_x \cdot u \
\dot{y} = A_y \cdot y^T + B_y \cdot u
\]

guard

\[
guard_x \land guard_y
\]

reset

\[
reset_x \land reset_y
\]
Variable set partitioning

Global space:

\[ X_0 \]

\[ P_1 = \text{conv}(X_0 \cup e^{A\delta}X_0 \oplus V_A \oplus V_B) \]

\[ P_2 = e^{A\delta}P_1 \oplus V_B, U \]

\[ P_3 = e^{A\delta}P_2 \oplus V_B, U \]

- Time \([0,0] \)
- Time \([0,\delta] \)
- Time \([\delta,2\delta] \)
- Time \([2\delta,3\delta] \)
Variable set partitioning

Global space:

\[ P_1 = \text{conv}(X_0 \cup e^{A\delta}X_0 \oplus V_A \oplus V_B) \]

\[ P_2 = e^{A\delta}P_1 \oplus V_B, U \]

\[ P_3 = e^{A\delta}P_2 \oplus V_B, U \]

Sub-space:

\[ X_{0,x} = X_0 \downarrow x \]

\[ P_{1,x} = \text{conv}(X_{0,x} \cup e^{A_x\delta}X_{0,x} \oplus V_{A,x} \oplus V_{B,x}) \]

\[ P_{2,x} = e^{A_x\delta}P_{1,x} \oplus V_{B_x,u} \]

\[ P_{3,x} = e^{A_x\delta}P_{2,x} \oplus V_{B_x,u} \]
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

\[ \dot{x} = 0 \]
\[ \dot{y} = 1 \]

[Graph showing a shaded region on a 2D plane with axis labels x and y.]
Discrete variables

\[ \dot{x} = 0 \]
\[ \dot{y} = 1 \]
Discrete variables

\[
\begin{align*}
\dot{x} &= 0 \\
\dot{y} &= 1
\end{align*}
\]
Discrete variables

\[
\begin{align*}
\dot{x} &= 0 \\
\dot{y} &= 1
\end{align*}
\]
Discrete variables

\[
\begin{align*}
\dot{x} &= 0 \\
\dot{y} &= 1
\end{align*}
\]
Discrete variables

\[ \dot{x} = 0 \]
\[ \dot{y} = 1 \]
Discrete variables

\[ \dot{x} = 0 \]
\[ \dot{y} = 1 \]

\[ 2.5 \leq y \leq 2.8 \]
Discrete variables

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

2.5 \leq y \leq 2.8
\[
\begin{align*}
\dot{x} &= 1 \\
\dot{y} &= 1
\end{align*}
\]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[
\dot{x} = 1 \\
\dot{y} = 1
\]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[
\begin{align*}
\dot{x} &= 1 \\
\dot{y} &= 1
\end{align*}
\]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[
\begin{align*}
\dot{x} &= 1 \\
\dot{y} &= 1
\end{align*}
\]
\[ \dot{x} = 1 \quad \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ 2.5 \leq y \leq 2.8 \]
\[ x = 1 \]
\[ \dot{y} = 1 \]

\[ 2.5 \leq y \leq 2.8 \]
Representation: Polytopes

\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
Representation: Polytopes

\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\begin{align*}
\dot{x} &= 1 \\
\dot{y} &= 1
\end{align*}

Representation: Polytopes
Representation: Polytopes

\[ x = 1 \]
\[ y = 1 \]
\[
\dot{x} = 1 \\
\dot{y} = 1
\]

Representation: Polytopes
Representation: Polytopes

\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]

\[ 2.5 \leq y \leq 2.8 \]
$\dot{x} = 1$

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

$\dot{y} = 1$
\[
\begin{align*}
\dot{x} &= 1 \\
\dot{y} &= 1
\end{align*}
\]
Representation: Polytopes

\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
$\dot{x} = 1$
$\dot{y} = 1$
$2.5 \leq y \leq 2.8$
Representation: Polytopes

\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]

\[ 2.5 \leq y \leq 2.8 \]
Reachability computation

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

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.
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.
Results: Leaking tank

![Graph showing the comparison between HyPro clock + rest and HyPro original for a leaking tank.

- The x-axis represents time (t) in units.
- The y-axis represents the level of leakage (x) in units.
- Two lines are plotted:
  - A darker line labeled "HyPro clock + rest".
  - A lighter line labeled "HyPro original".

From the graph, it is evident that the "HyPro clock + rest" method consistently exhibits higher leakage levels compared to the "HyPro original" method throughout the observed time frame. This suggests that the "HyPro clock + rest" method may not be as effective in managing leakage as the original method."
Results: Leaking tank

HyPro clock + rest
HyPro original
Results: Two tanks
## Running times

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Rep.</th>
<th>Agg</th>
<th>HyPro</th>
<th>SPACEEx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>clock</td>
<td>disc</td>
</tr>
<tr>
<td>box agg</td>
<td>2.70</td>
<td>2.08</td>
<td>1.06</td>
<td>1.13</td>
</tr>
<tr>
<td>Leaking tank</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box none</td>
<td>2.62</td>
<td>2.09</td>
<td>1.06</td>
<td>1.13</td>
</tr>
<tr>
<td>sf agg</td>
<td>TO</td>
<td>TO</td>
<td>161.12</td>
<td>37.03</td>
</tr>
<tr>
<td>sf none</td>
<td>TO</td>
<td>TO</td>
<td>1044.97</td>
<td>19.49</td>
</tr>
<tr>
<td>box agg</td>
<td>4.39</td>
<td>2.60</td>
<td>0.97</td>
<td>1.15</td>
</tr>
<tr>
<td>Two tanks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box none</td>
<td>4.46</td>
<td>2.68</td>
<td>1.02</td>
<td>1.16</td>
</tr>
<tr>
<td>sf agg</td>
<td>TO</td>
<td>TO</td>
<td>900.11</td>
<td>329.80</td>
</tr>
<tr>
<td>sf none</td>
<td>TO</td>
<td>TO</td>
<td>35.04</td>
<td>14.64</td>
</tr>
<tr>
<td>box agg</td>
<td>0.07</td>
<td>0.09</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>Thermostat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box none</td>
<td>0.11</td>
<td>0.09</td>
<td>0.06</td>
<td>0.06</td>
</tr>
<tr>
<td>sf agg</td>
<td>35.87</td>
<td>22.69</td>
<td>1.17</td>
<td>0.29</td>
</tr>
<tr>
<td>sf none</td>
<td>30.41</td>
<td>20.19</td>
<td>1.18</td>
<td>0.30</td>
</tr>
</tbody>
</table>
Two examples for HyPro applications:

- Sub-space computations
- CEGAR-based reachability analysis and parallelisation
Other HyPro applications

Strategy:

S1: box, $\delta = 0.1$

S2: support f., $\delta = 0.01$

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:

```
          A
         / \
        /   \n       B     C
```
Other HyPro applications

Strategy:

S1: box, \( \delta = 0.1 \)

S2: support f., \( \delta = 0.01 \)

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:
Other HyPro applications

Strategy:

S1: box, \( \delta = 0.1 \)

S2: support f., \( \delta = 0.01 \)

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:

- A
  - B
    - E
    - F
  - C
    - D
Other HyPro applications

Strategy:

S1: box, $\delta = 0.1$

S2: support f., $\delta = 0.01$

S3: polytope, $\delta = 0.01$

Search tree:

A
 / \  
B   C
 / \   /
E   F   D
     \  /
      G

Extension: Parallelized search in different branches.
Other HyPro applications

Strategy:

S1: box, \( \delta = 0.1 \)

S2: support f., \( \delta = 0.01 \)

S3: polytope, \( \delta = 0.01 \)

Search tree:

A

B

E

G

C

F

D

Extension: Parallelized search in different branches.
Other HyPro applications

Strategy:

S1: box, $\delta = 0.1$

S2: support f., $\delta = 0.01$

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:

```
A
/|
B / \ C
/   \    
E   F   D
  \   /  
   G
```

Extension: Parallelized search in different branches.
Other HyPro applications

Strategy:

S1: box, \( \delta = 0.1 \)

S2: support f., \( \delta = 0.01 \)

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:

Extension: Parallelized search in different branches.
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