Divide and Conquer: Variable Set Separation in Hybrid Systems Reachability Analysis

Stefan Schupp    Johanna Nellen    Erika Ábrahám

RWTH Aachen University, Germany

April 23, 2017

This work is part of the project HyPro which is funded by the German Research Council (DFG).
Hybrid systems (in computer science)

Typically large, complex, and safety-critical systems are addressed in the field of hybrid systems. A hybrid system combines discrete and continuous dynamics, as illustrated in the figure.

- **Discrete** systems are often modeled with graphs or automata, showing abrupt changes or transitions.
- **Continuous** systems are typically represented by differential equations, such as $f(t)$, showing smooth evolution over time $t$.

The combination of these two types of dynamics is crucial for modeling and analyzing real-world systems, such as control systems in engineering or network protocols in computer science.
Hybrid systems (in computer science)

Typically...

- large
- complex
- safety critical
Modeling language: Hybrid automata

Example: Thermostat

\[ \dot{x} = -x + 50 \]
\[ x \leq 23 \]

\[ \dot{x} = -x \]
\[ x \geq 17 \]

Operational semantics: \textit{flow} (time) and \textit{jump} (discrete) transitions
The reachability problem poses the question, whether in a given a hybrid automaton a certain set of target states can be reached from the initial states.
The **reachability problem** poses the question, whether in a given a hybrid automaton a certain set of target states can be reached from the initial states.
The reachability problem

The reachability problem poses the question, whether in a given a hybrid automaton a certain set of target states can be reached from the initial states.
The reachability problem poses the question, whether in a given a hybrid automaton a certain set of target states can be reached from the initial states.
The reachability problem poses the question, whether in a given a hybrid automaton a certain set of target states can be reached from the initial states.
The reachability problem poses the question, whether in a given a hybrid automaton a certain set of target states can be reached from the initial states.

The reachability problem for hybrid automata is in general undecidable.
Some tools for hybrid automata reachability analysis

- Ariadne [Collins et al., ADHS 2012]
- C2E2 [Duggirala et al., TACAS 2015]
- Cora [Althoff et al., ARCH 2015]
- dReach [Kong et al., TACAS 2015]
- HSolver [Ratschan et al., HSCC 2005]
- HyLAA [Bak et al., HSCC 2017]
- HySon [Bouissou et al., RSP 2012]
- Isabelle/HOL [Immler, TACAS 2015]
- iSAT-ODE [Eggers et al., ATVA 2008]
- KeYmaera (X) [Platzer et al., IJCAR 2008]
- NLTOOLBOX [Testylier et al., ATVA 2013]
- SoapBox [Hagemann et al., ARCH 2014]
- SpaceEx [Frehse et al., 2011]
- Flow* [Chen et al., CAV 2013]
- HyPro [Schupp et al., NFM 2017]
Forward reachability analysis

**Input:** Hybrid automaton $H$, initial states $\text{Init}$, target states $T$.

**Output:** Whether $T$ is reachable from $\text{Init}$ in $H$.

**Algorithm:**

\[
P := \text{Init}; \\
R := \text{Init}; \\
\text{while } (P \neq \emptyset) \{ \\
\quad P := \text{Reach}(P) \setminus R; \\
\quad R := R \cup P; \\
\}; \\
\text{if } (R \cap T = \emptyset) \text{ return "no" else return "yes";}
\]
Forward reachability analysis

Input: Hybrid automaton $H$, initial states $\text{Init}$, target states $T$.
Output: Whether $T$ is reachable from $\text{Init}$ in $H$.
Algorithm:

\[
\begin{align*}
P & := \text{Init}; \\
R & := \text{Init}; \\
\text{while } (P \neq \emptyset) \{ \\
&P := \text{Reach}(P) \setminus R; \\
&R := R \cup P; \\
\} \\
\text{if } (R \cap T = \emptyset) \text{ return } "\text{no}" \text{ else return } "\text{yes}";
\end{align*}
\]

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

Other symbolic representations:
- support functions [Le Guernic et al., 2009]
- Taylor models [Berz and Makino, 1998, 2009] [Chen et al., 2012]
Set operations

Some needed set operations:

- intersection
- union
- linear transformation
- Minkowski sum

projection

test for membership

test for emptiness
Set operations

Some needed set operations:

- intersection
- union
- linear transformation
- Minkowski sum

projection

test for membership

test for emptiness

Reminder: Minkowski sum

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

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

\[ V \text{-representation by vertices} \]

\[ H \text{-representation by facets} \]

easy
hard

Erika Ábrahám: Hybrid Systems Reachability Analysis
Example state set representation: Polytopes

- **Halfspace**: set of points satisfying $l \cdot x \leq z$
Example state set representation: Polytopes

- **Halfspace**: set of points satisfying $l \cdot x \leq z$
Example state set representation: Polytopes

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

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

- **Halfspace**: set of points 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 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 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 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 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:
- derivatives: constant
- conditions: convex linear sets.

Linear hybrid automata II:
- derivatives: linear differential equations
- conditions: convex linear sets.
Linear hybrid automata I: Time evolution
\[ \dot{x}_1, \dot{x}_2 \]

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

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

Linear hybrid automata I: Time evolution
Linear hybrid automata I: Time evolution

\[
\begin{align*}
\dot{x}_1 & \in P \\
\dot{x}_2 & \in Q
\end{align*}
\]
Linear hybrid automata I: Time evolution

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

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

\[ P \]

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

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

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

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

\[ P \]

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

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

\[ Erika \ Ábrahám: Hybrid Systems Reachability Analysis \]

10 / 1
Linear hybrid automata I: Time evolution

\[ \begin{align*}
\dot{x}_1 &\in P \\
\dot{x}_2 &\in \text{cone}(Q)
\end{align*} \]

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

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

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

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

\[ P \]

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

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

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

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

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

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

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

\begin{align*}
(P \oplus \text{cone}(Q)) \cap \text{Inv}(\ell)
\end{align*}
Linear hybrid automata I: Discrete steps (jumps)

$\ell x_1 x_2$

Computed via projection and intersection.

Erika Ábrahám: Hybrid Systems Reachability Analysis
Linear hybrid automata I: Discrete steps (jumps)

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

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

Computed via projection and intersection.

Erika Ábrahám: Hybrid Systems Reachability Analysis
Computed via projection and intersection.
Linear hybrid automata II: Time evolution

Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$.

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

$$P_0 = cl(V_0, e^{A\delta}V_0 \oplus V_A \oplus V_B)$$

$$P_0 e^{A\delta} P_0 P_1 = e^{A\delta} P_0 \oplus V_B$$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$
Linear hybrid automata II: Time evolution

- Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by $P_i$
• Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$
• Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$.

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

\[ P_0 = \text{cl}(V_0, e^{A\delta}V_0) \]

\[ P_1 = e^{A\delta}P_0 \oplus V_B \]

نویستگی
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Linear hybrid automata II: Time evolution

- Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by $P_i$
- Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

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

$$P_0 = \text{cl}(V_0, e^{A\delta}V_0 \oplus \mathcal{V}_A \oplus \mathcal{V}_B)$$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
- Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by $P_i$
Linear hybrid automata II: Time evolution

- Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$
- Over-approximate flowpipe segment for time $[i\delta, (i + 1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$.

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

\[ P_i = \text{cl}(V_0, e^{A\delta}V_0 \oplus V_A \oplus V_B) \]
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

Over-approximate flowpipe segment for time $[i\delta, (i+1)\delta]$ by $P_i$
Assume initial set $V_0$ and flow $\dot{x} = Ax + Bu$

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

$$P_0 = \text{cl}(V_0, e^{A\delta}V_0 \oplus V_A)$$

$$P_1 = e^{A\delta}P_0 \oplus V_B$$
Zonotopes in reachability computation
Linear hybrid automata II: Discrete steps (jumps)

\[ P_0, P_1, P_2, P_3 \]
Linear hybrid automata II: Discrete steps (jumps)

\[ P_0, P_1, P_2, P_3 \]
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
Reachability analysis search tree
Reachability analysis search tree
Special application: PLC-controlled plants

Plant

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

PLC

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

Programs

- Read:
- Write:

Observation: Interleaving can be restricted!
Special application: PLC-controlled plants

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

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

- **Controller**

- **Plant**
Special application: PLC-controlled plants

Observation: Interleaving can be restricted!
Special application: PLC-controlled plants

Observation: Interleaving can be restricted!
Special application: PLC-controlled plants

- Still: high-dimensional models
Special application: PLC-controlled plants

- Still: high-dimensional models
- Relevant number of discrete variables
- Clocks for cycle synchronisation
Special application: PLC-controlled plants

- Still: high-dimensional models
- Relevant number of discrete variables
- Clocks for cycle synchronisation

Idea:
- Partition variable set $\leadsto$ sub-spaces
- Compute reachability in sub-spaces
- Synchronise on time

```
<table>
<thead>
<tr>
<th>Time interval</th>
<th>Computed reachability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[0,0]$</td>
<td></td>
</tr>
<tr>
<td>$[0,\delta]$</td>
<td></td>
</tr>
<tr>
<td>$[\delta,2\delta]$</td>
<td></td>
</tr>
<tr>
<td>$[2\delta,3\delta]$</td>
<td></td>
</tr>
</tbody>
</table>
```
Variable set partitioning

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

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

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

$\dot{x} = A_x \cdot x + B \cdot u$

$\dot{y} = A_y \cdot y + B \cdot u$

$\text{Inv} x \land \text{guard} x \land \text{reset} x$

$\text{Inv} y \land \text{guard} y \land \text{reset} y$
Variable set partitioning

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

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

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

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

\[ \text{guard} \]

\[ \text{guard}_x \wedge \text{guard}_y \]

\[ \text{reset} \]

\[ \text{reset}_x \wedge \text{reset}_y \]

\[ \text{Inv} \]

\[ \text{Inv}_x \wedge \text{Inv}_y \]
Variable set partitioning

Global space:

\[ V_0 \]

\[ V_1 = \text{cl}(V_0 \cup e^{A\delta}V_0 \oplus V_A \oplus V_B) \]

\[ V_2 = e^{A\delta}V_1 \oplus V_{B,U} \]

\[ V_3 = e^{A\delta}V_2 \oplus V_{B,U} \]

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

Global space:

\[ V_0 \]

\[ V_1 = \text{cl}(V_0 \cup e^{A\delta}V_0 \oplus V_A \oplus V_B) \]

\[ V_2 = e^{A\delta}V_1 \oplus V_{B,U} \]

\[ V_3 = e^{A\delta}V_2 \oplus V_{B,U} \]

time [0,0]
time [0,\delta]
time [\delta,2\delta]
time [2\delta,3\delta]

Sub-space:

\[ V_{0,x} = V_0 \downarrow x \]

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

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

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

time [0,0]
time [0,\delta]
time [\delta,2\delta]
time [2\delta,3\delta]
Discrete variables

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

\[ 2.5 \leq y \leq 2.8 \]
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

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

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

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

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

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

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

Erika Abrahám: Hybrid Systems Reachability Analysis
Discrete variables

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

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

\[
\begin{align*}
\dot{x} &= 0 \\
\dot{y} &= 1 \\
2.5 \leq y &\leq 2.8
\end{align*}
\]
Discrete variables

\[
\begin{align*}
\dot{x} &= 0 \\
\dot{y} &= 1 \\
2.5 \leq y \leq 2.8
\end{align*}
\]
\[ \begin{align*}
\dot{x} &= 1 \\
\dot{y} &= 1
\end{align*} \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[
\begin{align*}
\dot{x} &= 1 \\
\dot{y} &= 1
\end{align*}
\]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]
\[ \dot{x} = 1 \]
\[ \dot{y} = 1 \]

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

1.5 \leq y \leq 1.8
Overapproximation by intersection

\[ 1.5 \leq y \leq 1.8 \]
Overapproximation by intersection

\[ 1.5 \leq y \leq 1.8 \]
Representation: Polytopes

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

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

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

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

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

$\dot{y} = 1$
Representation: Polytopes

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

\[ \frac{\Delta x}{\Delta y} = 1 \]

\[ \frac{\Delta y}{\Delta x} = 1 \]

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

$2.5 \leq y \leq 2.8$

$\dot{x} = 1$

$\dot{y} = 1$
Representation: Polytopes

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

Reachability computation process:

- **time** $[0,0]$: 
  - **disc**
  - **clock**
  - **rest**

- **time** $[0,\delta]$: 
  - $\cap\ Inv$
  - sub-space flowpipe segment
  - sub-space flowpipe segment

- **time** $[\delta,2\delta]$: 
  - sub-space flowpipe segment
  - sub-space flowpipe segment

- **time** $[2\delta,3\delta]$: 
  - $\emptyset$
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

HyPro clock + rest
HyPro original

Erika Ábrahám: Hybrid Systems Reachability Analysis
Results - leaking tank

HyPro clock + rest
HyPro original

Erika Ábrahám: Hybrid Systems Reachability Analysis
Results - two tanks

Erika Ábrahám: Hybrid Systems Reachability Analysis
<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>orig</td>
<td>clock</td>
<td>disc</td>
</tr>
<tr>
<td>box</td>
<td>agg</td>
<td>2.70</td>
<td>2.08</td>
<td>1.06</td>
</tr>
<tr>
<td>Leaking</td>
<td>box</td>
<td>none</td>
<td>2.62</td>
<td>2.09</td>
</tr>
<tr>
<td>tank</td>
<td>sf</td>
<td>agg</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td></td>
<td>sf</td>
<td>none</td>
<td>TO</td>
<td>1044.97</td>
</tr>
<tr>
<td>box</td>
<td>agg</td>
<td>4.39</td>
<td>2.60</td>
<td>0.97</td>
</tr>
<tr>
<td>Two</td>
<td>box</td>
<td>none</td>
<td>4.46</td>
<td>2.68</td>
</tr>
<tr>
<td>tanks</td>
<td>sf</td>
<td>agg</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td></td>
<td>sf</td>
<td>none</td>
<td>TO</td>
<td>TO</td>
</tr>
<tr>
<td>box</td>
<td>agg</td>
<td>0.07</td>
<td>0.09</td>
<td>0.06</td>
</tr>
<tr>
<td>Thermostat</td>
<td>box</td>
<td>none</td>
<td>0.11</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>sf</td>
<td>agg</td>
<td>35.87</td>
<td>22.69</td>
</tr>
<tr>
<td></td>
<td>sf</td>
<td>none</td>
<td>30.41</td>
<td>20.19</td>
</tr>
</tbody>
</table>
Conclusion and outlook

Conclusion:

- The HyPro library offers datatypes for the implementation of hybrid systems reachability analysis algorithms.
- Sufficient flexibility to deviate from standard methods.
- Example: Sub-space computations.

Outlook:

- CEGAR approach for iterative error reduction.
- Parallelisation.