δ-Reachability Analysis for Hybrid Systems

Soonho Kong
soonhok@cs.cmu.edu
Carnegie Mellon University
Discrete Control + Continuous Dynamics
Cardiac Cell Action Potential Model

Mode 1

flow 1:
\[
\frac{du}{dt} = \varepsilon - \frac{u}{\tau_{o1}}
\]
\[
\frac{dv}{dt} = \frac{1-v}{\tau_{v1}}
\]
\[
\frac{dw}{dt} = \frac{1-u}{\tau_{w}} - \frac{w}{\tau_{w1} + \frac{\tau_{w2} - \tau_{w1}}{1+e^{-2k_{u}(u-a_{u})}}} \frac{1}{\tau_{w1} + \frac{\tau_{w2} - \tau_{w1}}{1+e^{-2k_{u}(u-a_{u})}}}
\]
\[
\frac{ds}{dt} = \frac{1}{1+e^{-2k_{u}(u-a_{u})} - s} \frac{1}{\tau_{s1}}
\]

Mode 2

flow 2:
\[
\frac{du}{dt} = \varepsilon - \frac{u}{\tau_{o2}}
\]
\[
\frac{dv}{dt} = \frac{-v}{\tau_{v2}}
\]
\[
\frac{dw}{dt} = \frac{w_{m} - w}{\tau_{w1} + \frac{\tau_{w2} - \tau_{w1}}{1+e^{-2k_{u}(u-a_{u})}}}
\]
\[
\frac{ds}{dt} = \frac{1}{1+e^{-2k_{u}(u-a_{u})} - s} \frac{1}{\tau_{s1}}
\]

Mode 3

flow 3:
\[
\frac{du}{dt} = \varepsilon - \frac{1}{\tau_{o1}} + \frac{\tau_{o2} - \tau_{o1}}{1+e^{-2k_{u}(u-a_{u})}} \frac{w}{\tau_{s1}} + \frac{w \cdot s}{\tau_{sf}}
\]
\[
\frac{dv}{dt} = \frac{-v}{\tau_{v2}}
\]
\[
\frac{dw}{dt} = \frac{w}{\tau_{w}}
\]
\[
\frac{ds}{dt} = \frac{1}{1+e^{-2k_{u}(u-a_{u})} - s} \frac{1}{\tau_{s1}}
\]

Mode 4

flow 4:
\[
\frac{du}{dt} = \varepsilon + \frac{v(u - \theta_{v})(u_{u} - u)}{\tau_{fi}}
\]
\[
\frac{dv}{dt} = \frac{-v}{\tau_{v2}}
\]
\[
\frac{dw}{dt} = \frac{w}{\tau_{w}}
\]
\[
\frac{ds}{dt} = \frac{1}{1+e^{-2k_{u}(u-a_{u})} - s} \frac{1}{\tau_{s1}}
\]
Hybrid Systems

Discrete Control + Continuous Dynamics

Prostate Cancer Treatment Model

Mode 1 (on-treatment)

\[
\frac{dz}{dt} = \frac{-z_t}{\tau} + \mu_z
\]

\[\text{jump}_{1 \rightarrow 3} : \]
\[\nu = \frac{v(0)}{2}\]

\[x + y \leq r_0\]
\[\hat{\frac{dx}{dt}} + \frac{dy}{dt} < 0\]
\[v \cdot w \geq t_{\text{max}}\]

Mode 2 (off-treatment)

\[
\frac{dz}{dt} = \frac{z_0 - z_t}{\tau} + \mu_z
\]

Control (cancer therapy)

Mode 3 (dummy)

Plant (cancer progression)
Hybrid Systems

Quadcopter Control (CMU ECE)

Power Train Control (Toyota Research)

Autonomous Vehicle (CMU ECE)

Microfluidic Chip Design (Univ. of Waterloo)
Reachability Analysis of Hybrid Systems

Can we find a set of initial values/parameters for which a cardiac cell loses excitability?

Cardiac Cell Action Potential

<table>
<thead>
<tr>
<th>Mode 1</th>
<th>Mode 2</th>
<th>Mode 3</th>
<th>Mode 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow 1 :</td>
<td>flow 2 :</td>
<td>flow 3 :</td>
<td>flow 4 :</td>
</tr>
<tr>
<td>[ \frac{du}{dt} = \frac{\varepsilon - u}{\tau_{o1}} ]</td>
<td>[ \frac{du}{dt} = \frac{\varepsilon - u}{\tau_{o2}} ]</td>
<td>[ \frac{du}{dt} = \frac{1}{\tau_{w1} + \tau_{w2} - \tau_{w1}} ]</td>
<td>[ \frac{du}{dt} = \frac{1}{\tau_{w1} + \tau_{w2} - \tau_{w1}} + \frac{w \cdot s}{\tau_{w1}} + \frac{1}{1 + e^{-2z_k(u-u_0)}} ]</td>
</tr>
<tr>
<td>[ \frac{dv}{dt} = \frac{1 - v}{\tau_{v1}} ]</td>
<td>[ \frac{dv}{dt} = -\frac{v}{\tau_{v2}} ]</td>
<td>[ \frac{dv}{dt} = -\frac{w}{\tau_{v2}} ]</td>
<td>[ \frac{dv}{dt} = \frac{v}{\tau_{v2}} ]</td>
</tr>
<tr>
<td>[ \frac{dw}{dt} = \frac{1}{\tau_{w1} + \tau_{w2} - \tau_{w1}} ]</td>
<td>[ \frac{dw}{dt} = \frac{w}{\tau_{w2} - \tau_{w1}} ]</td>
<td>[ \frac{dw}{dt} = \frac{w}{\tau_{w2} - \tau_{w1}} + \frac{1}{1 + e^{-2z_k(u-u_0)}} ]</td>
<td>[ \frac{dw}{dt} = \frac{-w}{\tau_{w2}} ]</td>
</tr>
<tr>
<td>[ \frac{ds}{dt} = \frac{1}{1 + e^{-2z_k(u-u_0)}} ]</td>
<td>[ \frac{ds}{dt} = -\frac{s}{\tau_{s1}} ]</td>
<td>[ \frac{ds}{dt} = -\frac{s}{\tau_{s1}} ]</td>
<td>[ \frac{ds}{dt} = \frac{1}{1 + e^{-2z_k(u-u_0)}} - \frac{s}{\tau_{s1}} ]</td>
</tr>
</tbody>
</table>

Minimal Resistor Model

- \( u \) state variable
- \( v \) state variable
- \( w \) state variable
- \( s \) state variable
- stimulus
Reachability Analysis of Hybrid Systems

Can we find a personalized treatment model which prevents the cancer recurrence in 5 years??

Prostate Cancer Treatment Model

Mode 1 (on-treatment)
\[
\frac{dx}{dt} = \frac{\tau \cdot z - x}{\tau} + \mu
\]

Mode 2 (off-treatment)
\[
\frac{dz}{dt} = \frac{z - z}{\tau} + \mu
\]

Mode 3 (dummy)
\[
\frac{dy}{dt} = \frac{y}{2}
\]

Control (cancer therapy)

Plant (cancer progression)

Hormone sensitive cancer cells (HSCs)
Castration resistant cancer cells (CRCs)

CAS
IAS

Tumor size
Time
Can we automate a non-trivial parking?
Can we automate a non-trivial parking?

\[
\frac{d}{dt} \tilde{x} = h(\tilde{x})
\]

\[
\frac{d}{dt} \tilde{x} = f(\tilde{x})
\]

\[
\frac{d}{dt} \tilde{x} = g(\tilde{x})
\]
Reachability Analysis of Hybrid Systems

Can a hybrid system run into an unsafe region of its state space?
Reachability Analysis of Hybrid Systems

Can a hybrid system run into an unsafe region of its state space?
Can a hybrid system run into an **unsafe** region of its state space?

The standard bounded reachability problems for simple hybrid systems are **undecidable** [Alur et al, 1992].
Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are undecidable.
Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up

“I can’t find an algorithm, but neither can all these famous people.”
The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up

2. Don’t give Up
Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up

2. Don’t give Up

   A. Find a decidable fragment and solve it
Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up

2. Don’t give Up
   
   A. Find a decidable fragment and solve it

   B. Use approximation
Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are undecidable.

1. Give up

2. Don’t give Up

   A. Find a decidable fragment and solve it

   B. Use approximation
\( \delta \)-Reachability Analysis of Hybrid Systems

Given \( \delta \in \mathbb{Q}^+ \), \([H^\delta]\) and \([\text{unsafe}^\delta]\) over-approximate \([H]\) and \([\text{unsafe}]\)

\( \delta \)-reachability problem asks for one of the following answers:

- **Unreachable (Safe)**
- **\( \delta \)-reachable (\( \delta \)-Unsafe)**
\[ \delta\text{-Reachability Analysis of Hybrid Systems} \]

Given \( \delta \in \mathbb{Q}^+ \), \( \llbracket H \rrbracket \) and \( \llbracket \text{unsafe}^\delta \rrbracket \) over-approximate \( \llbracket H \rrbracket \) and \( \llbracket \text{unsafe} \rrbracket \)

\( \delta \)-reachability problem asks for one of the following answers:

- **Decidable** for a wide range of nonlinear hybrid systems
  - polynomials, log, exp, trigonometric functions, …
δ-Reachability Analysis of Hybrid Systems

Given $\delta \in \mathbb{Q}^+$, $\llbracket H \rrbracket$ and $\llbracket \text{unsafe}^{\delta} \rrbracket$ over-approximate $\llbracket H \rrbracket$ and $\llbracket \text{unsafe} \rrbracket$.

δ-reachability problem asks for one of the following answers:

- Decidable for a wide range of nonlinear hybrid systems
- Reasonable complexity bound (PSPACE-complete)
δ-Reachability Analysis of Hybrid Systems

1. “Unreachable” answers is sound.
2. Analysis is parameterized with $\delta$

If using a delta leads to a infeasible counterexample, you may try a smaller delta and possibly get rid of it.
3. Robustness

If your system is \textbf{δ-reachable} under a reasonably small \( \delta \), then a small error can lead your system to an \textbf{unsafe} state.
δ-Reachability Analysis of Hybrid Systems

“δ-reachability analysis checks robustness which implies safety.”
δ-Reachability Analysis of Hybrid Systems
A mode definition consists of mode id, mode invariant, flow, and jump. The mode id is a unique positive integer assigned to a mode. An invariant is a conjunction of logic formulae which must always hold in a mode. A flow describes the continuous dynamics of a mode by providing a set of ODEs. The first formula of jump is interpreted as a guard, a logic formula specifying a condition to make a transition. Note that this allows a transition but does not force it. The second argument of jump, n, denotes the target mode-id. The last one is reset, a logic formula connecting the old and new values for the transition.

**initial-condition** specifies the initial mode of a hybrid system and its initial configuration. **goal** shares the same syntactic structure of **initial-condition**.

```c
#define D 0.45
#define K 0.9
[0, 15] x;
[9.8] g;
[-18, 18] v;
[0, 3] time;

{ mode 1;
  invt: (v <= 0);
    (x >= 0);
  flow: \frac{d}{dt}[x] = v;
    \frac{d}{dt}[v] = -g - (D * v ^ 2);
  jump: (x = 0) ==> @2 (and (x' = x) (v' = - K * v)); }

{ mode 2;
  invt: (v >= 0);
    (x >= 0);
  flow: \frac{d}{dt}[x] = v;
    \frac{d}{dt}[v] = -g + (D * v ^ 2);
  jump: (v = 0) ==> @1 (and (x' = x) (v' = v)); }

init: @1 (and (x >= 5) (v = 0));
goal: @1 (and (x >= 0.45));
```

Fig. 3: An example of drh format: Inelastic bouncing ball with air resistance.
Logical Encoding of Reachability Problem

Can a system run into an \textit{unsafe} region after making $k$ steps?

$Init(\vec{x}_0)$
Logical Encoding of Reachability Problem

Can a system run into an unsafe region after making k steps?

\[ \text{Init}(\vec{x}_0) \land \text{flow}_{q_0}(\vec{x}_0, \vec{x}_0^t, t_0) \land \text{jump}_{q_0 \rightarrow q_1}(\vec{x}_0^t, \vec{x}_1) \]
Can a system run into an unsafe region after making k steps?

\[\text{Init}(\vec{x}_0) \land \text{flow}_{q_0}(\vec{x}_0, \vec{x}_0^t, t_0) \land \text{jump}_{q_0 \rightarrow q_1}(\vec{x}_0^t, \vec{x}_1) \land \text{flow}_{q_1}(\vec{x}_1, \vec{x}_1^t, t_1) \land \text{jump}_{q_1 \rightarrow q_2}(\vec{x}_1^t, \vec{x}_2)\]
Logical Encoding of Reachability Problem

Can a system run into an **unsafe** region after making \( k \) steps?

\[
\begin{align*}
Init(\vec{x}_0) & \land flow_{q_0}(\vec{x}_0, \vec{x}_0^t, t_0) \land jump_{q_0 \rightarrow q_1}(\vec{x}_0^t, \vec{x}_1) \land \\
& \quad flow_{q_1}(\vec{x}_1, \vec{x}_1^t, t_1) \land jump_{q_1 \rightarrow q_2}(\vec{x}_1^t, \vec{x}_2) \land \\
& \quad \ldots \land \\
& \quad flow_{q_k}(\vec{x}_k, \vec{x}_k^t, t_k)
\end{align*}
\]
Logical Encoding of Reachability Problem

Can a system run into an **unsafe** region after making k steps?

$$Init(\vec{x}_0) \land flow_{q_0}(\vec{x}_0, \vec{x}_0^t, t_0) \land jump_{q_0 \rightarrow q_1} (\vec{x}_0^t, \vec{x}_1) \land$$

$$flow_{q_1}(\vec{x}_1, \vec{x}_1^t, t_1) \land jump_{q_1 \rightarrow q_2} (\vec{x}_1^t, \vec{x}_2) \land$$

$$\ldots$$

$$flow_{q_k}(\vec{x}_k, \vec{x}_k^t, t_k) \land Unsafe(\vec{x}_k^t)$$
Can a system run into an unsafe region after making $k$ steps?

$$\exists \bar{x}_0, \bar{x}_1, \ldots, \bar{x}_k \exists t_0, t_1, \ldots, t_k
\begin{align*}
&\text{Init}(\bar{x}_0) \land \text{flow}_{q_0}(\bar{x}_0, \bar{x}_0^t, t_0) \land \text{jump}_{q_0 \rightarrow q_1}(\bar{x}_0^t, \bar{x}_1^t) \land \\
&\text{flow}_{q_1}(\bar{x}_1^t, \bar{x}_1^t, t_1) \land \text{jump}_{q_1 \rightarrow q_2}(\bar{x}_1^t, \bar{x}_2^t) \land \\
&\ldots \\
&\text{flow}_{q_k}(\bar{x}_k^t, \bar{x}_k^t, t_k) \land \text{Unsafe}(\bar{x}_k^t)
\end{align*}$$
δ-Reachability Analysis of Hybrid Systems

Hybrid System Model + Specification (drh format)

Encoder

Logic formula

dReal (δ-complete SMT solver)

δ-SAT

δ-reachable + Counterexample (Visualization)

UNSAT

Unreachable

Numerical Error

Unrolling Depth (k)
Decision Problem

Standard Form

\[ \phi := \exists^1 x \bigvee_{i} \bigwedge_{j} f_{i,j}(x) = 0 \]
δ-Decision Problem

δ-Weakening of ϕ

\[ \phi^\delta := \exists^I x \bigvee_{i} \bigwedge_{j} |f_{i,j}(x)| \leq \delta \]
Solving Logic Formula

DPLL<T> Framework

- **SAT solver** finds a satisfying **Boolean** assignment
- **Theory solver** checks whether the assignment is feasible under theory of **Real**
Solving Logic Formula

DPLL\textless T\textgreater Framework

- **SAT solver** finds a satisfying **Boolean** assignment
- **Theory solver** checks whether the assignment is feasible under theory of **Real**
Solving Logic Formula

\[ l_1 \land l_2 \land \neg l_3 \land \neg l_4 \]

DPLL\(<T>\) Framework

- **SAT solver** finds a satisfying **Boolean** assignment
- **Theory solver** checks whether the assignment is feasible under theory of **Real**
Solving Logic Formula

DPLL<T> Framework

- **SAT solver** finds a satisfying **Boolean** assignment
- **Theory solver** checks whether the assignment is feasible under theory of **Real**
Solving Logic Formula

DPLL<T> Framework

- **SAT solver** finds a satisfying **Boolean** assignment
- **Theory solver** checks whether the assignment is feasible under theory of **Real**
Solving Logic Formula

**DPLL<T> Framework**

- **SAT solver** finds a satisfying **Boolean** assignment
- **Theory solver** checks whether the assignment is feasible under theory of **Real**
Main Algorithm of Theory Solver

Theory Solver

- Nonlinear Constraint Solver
- ODE Solver

List of Constraints

δ-SAT or UNSAT
Main Algorithm of Theory Solver: 
ICP (Interval Constraint Propagation)

$\text{Pruning}$

Monotone $B_1 \subseteq B_2 \implies f(B_1) \subseteq f(B_2)$

Reductive $f(B) \subseteq B$

Solution-Preserving $x \in B \land x \in Sol(f) \implies x \in f(B)$
Main Algorithm of Theory Solver: ICP (Interval Constraint Propagation)
Two Termination Conditions of ICP

δ-sat

Unsat
Example of Pruning Operations

\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \arctan(x)

\textbf{ANSWER:} \textcolor{red}{\textbf{UNSAT}}
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \arctan(x) \]
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \]

\[ x' = x \cap \sin^{-1}(y) \]
\[ = [0.5, 1.0] \cap \sin^{-1}([0.5, 1.0]) \]
\[ = [0.5, 1.0] \cap [0.524, 1.570] \]
\[ = [0.524, 1.0] \]
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \]

\[ y' = y \cap \sin(x) \]
\[ = [0.5, 1.0] \cap \sin([0.524, 1.0]) \]
\[ = [0.5, 1.0] \cap [0.5, 0.841] \]
\[ = [0.5, 0.841] \]
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \]

\[ x : [0.524, 1.0], y : [0.5, 0.841] \]
Example of Pruning Operations

$\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x)$

\[
x' = x \cap \tan^{-1}(y)
= [0.524, 1] \cap \tan^{-1}([0.5, 0.841])
= [0.524, 1] \cap [0.546, 1.117]
= [0.546, 1.0]
\]
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \]

\[
y' = y \cap \tan(x) \\
= [0.5, 0.841] \cap \tan([0.546, 1.0]) \\
= [0.5, 0.841] \cap [0.5, 1.0] \\
= [0.5, 0.785]
\]
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \text{atan}(x) \]

\[ x : [0.524, 1.0], \ y : [0.5, 0.785] \]
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \]
Example of Pruning Operations

\( \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \)
Example of Pruning Operations

\( \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \)
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \arctan(x) \]
Example of Pruning Operations

$\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x)$
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \tan(x) \]
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \]
Example of Pruning Operations

\[ \exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x) \]
Example of Pruning Operations

$\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \tan(x)$
Example of ICP

Graph for $\sin(x)$, $\arctan(x)$

ANSWER: SAT
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \text{atan}(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y \sin(x) \land y = \arctan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x)\]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x)$$
\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x)
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
\( \exists x \in [0.5, 2.0], \, y \in [0.0, 2.0] : \, y = \sin(x) \land y = \tan(x) \)
Example of ICP

$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \text{atan}(x)$
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x)$
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \text{atan}(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \atan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]
Example of ICP

\[ \exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \tan(x) \]

\[ \epsilon = 0.001 \]

**ANSWER: $\delta$-SAT**
Main Algorithm of ICP

Algorithm 1: Theory Solving in DPLL(ICP)

input: A conjunction of theory atoms, seen as constraints, 
c\(_1(x_1, \ldots, x_n), \ldots, c_m(x_1, \ldots, x_n)\), the initial interval bounds on all 
variables \(B^0 = I_1^0 \times \cdots \times I_n^0\), box stack \(S = \emptyset\), and precision \(\delta \in \mathbb{Q}^+\).

output: \(\delta\)-sat, or unsat with learned conflict clauses.

1. \(S\).push\((B_0)\);
2. while \(S \neq \emptyset\) do
3.     \(B \leftarrow S\).pop();
4.     while \(\exists 1 \leq i \leq m, B \neq \text{Prune}(B, c_i)\) do
5.         //Pruning without branching, used as the assert() function.
6.             \(B \leftarrow \text{Prune}(B, c_i)\);
7.     end
8.     //The \(\epsilon\) below is computed from \(\delta\) and the Lipschitz constants of 
9.     functions beforehand.
10. if \(B \neq \emptyset\) then
11.     if \(\exists 1 \leq i \leq n, |I_i| \geq \epsilon\) then
12.         \(\{B_1, B_2\} \leftarrow \text{Branch}(B, i)\); //Splitting on the intervals
13.         \(S\).push(\(\{B_1, B_2\}\));
14.     else
15.         return \(\delta\)-sat; //Complete check() is successful.
16.     end
17. end
18. return unsat;
Pruning using ODEs (Forward)

How can we have smaller $X'_t$?
Pruning using ODEs
(Forward)
Pruning using ODEs (Forward)
Pruning using ODEs
(Forward)
Pruning using ODEs
(Backward)
Pruning using ODEs
(on Time)
Pruning using ODEs (on Time)
Pruning using ODEs
(on Time)
dReach: δ-Reachability Analysis of Hybrid Systems

- Open Source (GPL3), available at [https://dreal.github.io](https://dreal.github.io)
- Support polynomials, transcendental functions and nonlinear ODEs
- Formulas with 100+ ODEs have been solved.
Visualization of Counterexample

Numerical Error (δ)

Hybrid System Model + Specification (drh format)

Unrolling Depth (k)

BMC Encoder

SMT2 formula

dReach

(δ-complete SMT solver)

dReal

DPLL<\text{T}>

ICP Solver

Nonlinear Constraint Solver

ODE Solver

δ-SAT

δ-reachable + Counterexample (Visualization)

UnSAT

Unreachable
Visualization of Counterexample

3-mode Oscillator Model

Click and drag above to zoom / pan the data
Applications

* Cardiac Cells, Prostate Cancer (CMU, GIT, TU Vienna)

* Prostate Cancer (CMU, UPITT)

* Power-train Control (Toyota Research Lab)

* Microfluidic Chip Designs (Waterloo)

* Analog Circuits (City University London)

* Quadcopter Control, Autonomous Driving (CMU)

* FDA-accepted non-linear hybrid physiological model for diabetes, (UPENN)
Tools based on dReal/dReach

* ProbReach: Probabilistic reachability analysis of hybrid systems (Univ. of Newcastle)
* BioPSy: Parameter set synthesis on biological models (Univ. of Newcastle)
* SReach: Bounded model checker for stochastic hybrid systems (CMU)
* Osmosis: Semantic importance sampling for statistical model checking (CMU SEI)
* Sigma: Probabilistic programming language (MIT)
Conclusion

* δ-reachability analysis checks robustness of hybrid systems which implies safety.
* Decidable (PSPACE-Complete)
* It uses dReal, a δ-complete SMT solver, which supports nonlinear functions and nonlinear ODEs
* Based on DPLL⟨ICP⟩ framework
* Scalable with our experiments
* Open-source: available at http://dreal.github.io
Future Work

* Scalability
  * Parallelization
* Learning from failures during ICP
* Smart Backtracking in ICP (Non-chronological BackJumping)
* Expressivity
  * Support Exist-Forall formulas (for optimization problems)
Thank you

Any Questions?