Reachability analysis in the Zélus language (WIP)

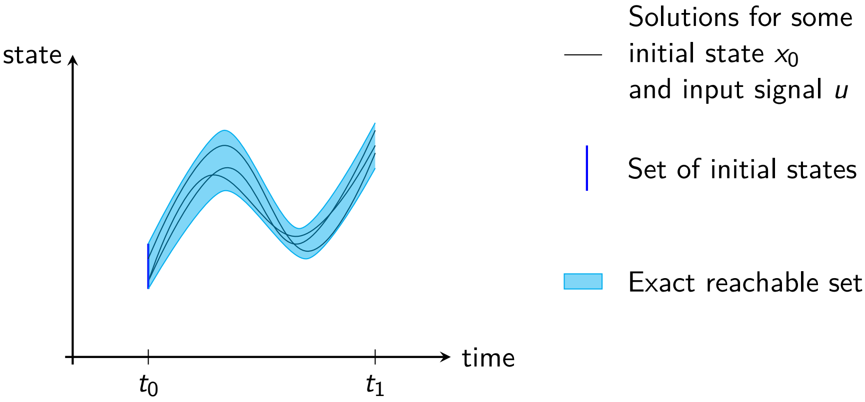
François Bidet[1], Marc Pouzet

LIX & PARKAS

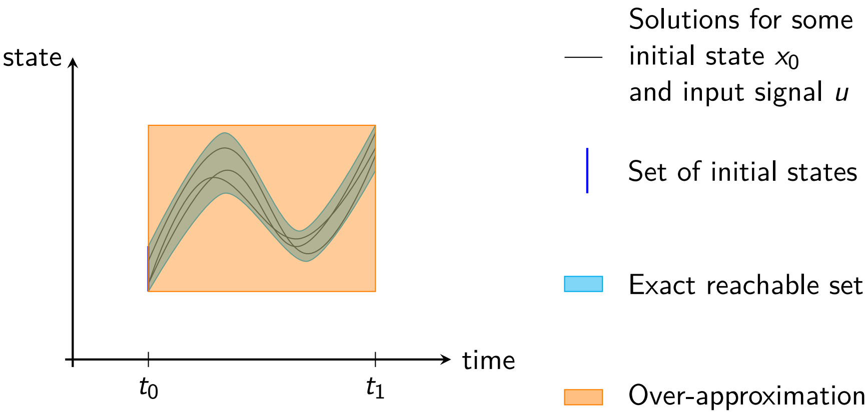
November 23, 2021

[1]supervised by Éric Goubault and Sylvie Putot

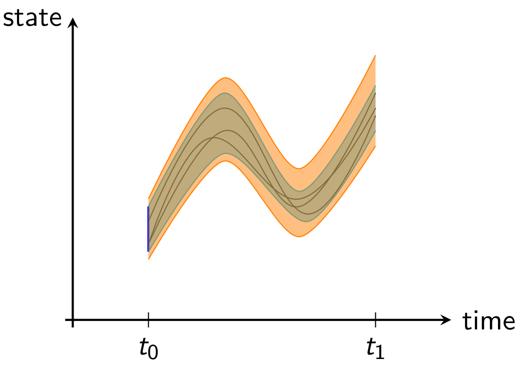
Over-approximation of the reachable set



Over-approximation of the reachable set



Over-approximation of the reachable set



Solutions for some initial state x_0 and input signal u

Set of initial states

Exact reachable set

Over-approximation

Reachability of hybrid systems

Goal

Given

- I the set of possible initial states: $x(0) \in I$
- U the range of the possible input signals: $u(t) \in U$
- $\varphi(x_0, u, t)$ the state at time t of a possible evolution whose initial state is $x(0) = x_0$ and the value of the input signal at time t is $u(t)$

we want to compute the set of reachable states over time

$$R(t) = \left\{ \varphi(x_0, u, t) \;\middle|\; x_0 \in I \land \forall \xi \in [0, t],\ u(\xi) \in U \right\}$$

Alias

set of reachable states $\equiv$ reachable set

Reachability of hybrid systems

Goal

Given

- I the set of possible initial states: $x(0) \in I$
- ~~U the range of the possible input signals: $u(t) \in U$~~
- $\varphi(x_0, t)$ the state at time t of a possible evolution whose initial state is $x(0) = x_0$ ~~and the value of the input signal at time t is $u(t)$~~

we want to compute the set of reachable states over time

$$[R](t) \supset \left\{ \varphi(x_0, t) \;\middle|\; x_0 \in I \right\}$$

Alias

set of reachable states $\equiv$ reachable set

Representation of hybrid systems

Hybrid automaton [2]

$$
\begin{aligned}
X &= X_D \times X_C \\
U &= U_D \times U_C \\
Y &= Y_D \times Y_C \\
I &\subset X \\
f &: X \times U \to TX_C \\
E &\subset X \times U \times X \\
h &: X \times U \to Y
\end{aligned}
$$

[2] J. Lygeros, "Hierarchical, Hybrid Control of Large Scale Systems", 1996

Representation of hybrid systems

Hybrid automaton [2]

$$
\begin{aligned}
X &= X_D \times X_C \\
U &= U_D \times U_C \\
Y &= Y_D \times Y_C \\
I &\subset X \\
f &: X \times U \to TX_C \\
E &\subset X \times U \times X \\
h &: X \times U \to Y
\end{aligned}
$$

Problem

- Not executable
- Not user-friendly

[2] J. Lygeros, "Hierarchical, Hybrid Control of Large Scale Systems", 1996

Representation of hybrid systems: Zélus

```
let hybrid temperature(temp0,alpha1,temp1,alpha2,temp2) =
    temp where
    rec der temp = alpha1 *. (temp1 -. temp)
                 +. alpha2 *. (temp2 -. temp) init temp0

let hybrid thermostat(temp,target,hysteresis) = power where
    rec zup = up(temp -. target -. hysteresis)
    and zdown = up(target -. hysteresis -. temp)
    and init power = 0. (* off by default *)
    and present
    | zup -> do power = 0. done
    | zdown -> do power = 50. done

let hybrid room(t0) = (troom,theater,tresistor) where
    rec troom = temperature(t0,1e-1,theater,1e-3,0.)
    and theater = temperature(t0,1.,tresistor,1e-1,troom)
    and tresistor = thermostat(troom,20.,5e-1)
```

Representation in compiled Zélus

Abstraction of a compiled program

$$
\begin{array}{rcl}
\mathit{state} & & \\
\mathit{init} & : & \mathit{state} \\
\mathit{cont} & : & \mathit{state} \rightarrow X_C \\
\mathit{deriv} & : & \mathit{state} \rightarrow X_C \rightarrow TX_C \\
\mathit{trigger} & : & \mathit{state} \rightarrow X_C \rightarrow Z_{out} \\
\mathit{discrete} & : & \mathit{state} \rightarrow X_C \times Z_{in} \rightarrow \mathit{state}
\end{array}
$$

External solvers

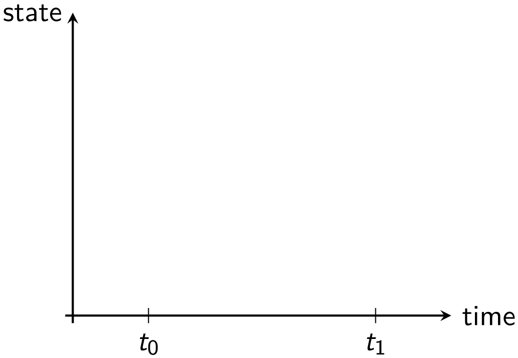
$$
\begin{array}{rcl}
\mathit{integrate} & : & X_C \times (X_C \rightarrow TX_C) \rightarrow \mathit{horizon} \times (\mathit{time} \rightarrow X_C) \\
\mathit{eventDectector} & : & (\mathit{time} \rightarrow Z_{out}) \times \mathit{horizon} \rightarrow \mathit{horizon} \times Z_{in}
\end{array}
$$

Our idea

Method

1. Replace the type `float` by an abstract type
2. Overload corresponding operators
3. Instantiate the abstract type by the needed one
4. Execute the computation

Example

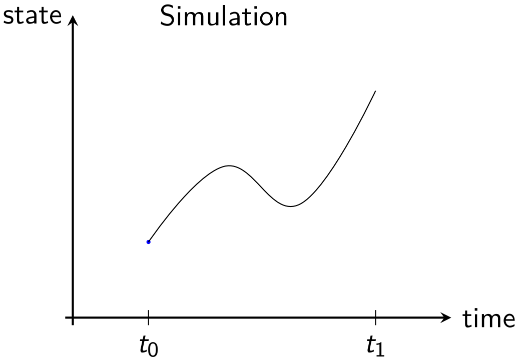


Our idea

Method

1. Replace the type `float` by an abstract type
2. Overload corresponding operators
3. Instantiate the abstract type by the needed one
4. Execute the computation

Example

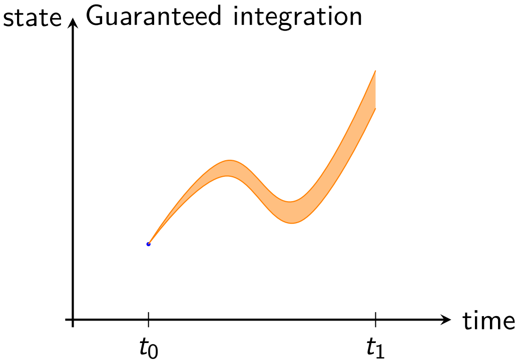


Our idea

Method

1. Replace the type `float` by an abstract type
2. Overload corresponding operators
3. Instantiate the abstract type by the needed one
4. Execute the computation

Example

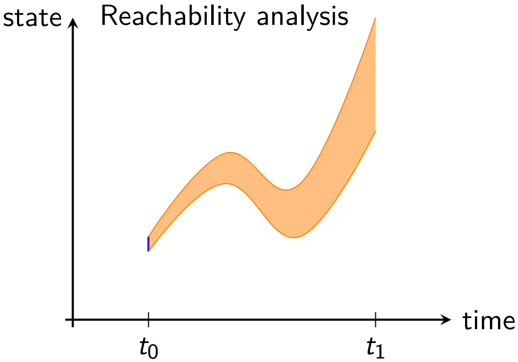


Our idea

Method

1. Replace the type `float` by an abstract type
2. Overload corresponding operators
3. Instantiate the abstract type by the needed one
4. Execute the computation

Example



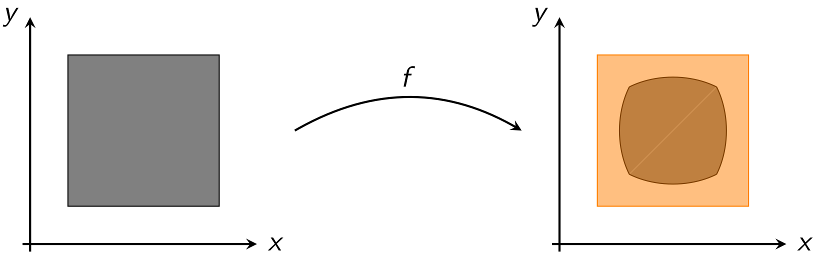
Straightforward over-approximation: arithmetics

Arithmetics

$$I_R = I_1 \otimes I_2 \quad \Longrightarrow \quad \forall (v_1, v_2) \in I_1 \times I_2,\ v_1 \oplus v_2 \in I_R$$

Example

$$f\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{x}{\sqrt{1+x^2+y^2}} \\ \frac{y}{\sqrt{1+x^2+y^2}} \end{pmatrix}$$



Over-approximation: switches

Switch

$$\textbf{if } \textit{cond} \textbf{ then } E_1 \textbf{ else } E_2$$

Condition with sets

$$\begin{aligned} [0,1] \leq [2,3] &\equiv \textbf{true} \\ [2,3] \leq [0,1] &\equiv \textbf{false} \\ [0,2] \leq [1,3] &\equiv \textbf{true} \cup \textbf{false} \quad ? \end{aligned}$$

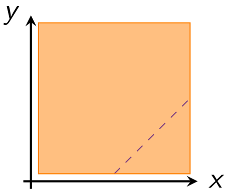
Over-approximation: switches

Condition as preimage mapping

cond	:	*set*	$\rightarrow$	*set*	$\times$	*set*	$\times$	*set*
		initial		true		false		undecidable

Lower than

$$(\leq)\begin{pmatrix}[0,2]\\ [1,3]\end{pmatrix} = \emptyset, \emptyset, \begin{pmatrix}[0,2]\\ [1,3]\end{pmatrix}$$



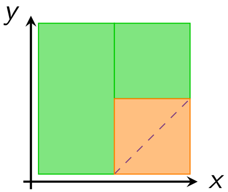
Over-approximation: switches

Condition as preimage mapping

$$\underset{}{\textit{cond}} : \underset{\text{initial}}{\textit{set}} \rightarrow \underset{\text{true}}{\textit{set}} \times \underset{\text{false}}{\textit{set}} \times \underset{\text{undecidable}}{\textit{set}}$$

Lower than

$$(\leq)\begin{pmatrix}[0,2]\\ [1,3]\end{pmatrix} = \begin{pmatrix}[0,1]\\ [1,3]\end{pmatrix} \cup \begin{pmatrix}[1,2]\\ [2,3]\end{pmatrix}, \emptyset, \begin{pmatrix}[1,2]\\ [1,2]\end{pmatrix}$$



Algorithm

1. create collection C containing *init*
2. while collection C is not empty:
   2.1 extract an element (state)
   2.2 integrate ODEs
   2.3 detect events
   2.4 compute next possible states (collection C')

   2.5 for all valid elements S in C', add S to C

Algorithm

Concrete simulation

1. create collection C containing *init* $\{(0, init)\}$
2. while collection C is not empty:
   2.1 extract an element (state) (t_0, s)
   2.2 integrate ODEs $\forall \delta \in [t_0, t_1],\ f(\delta) = x$
   2.3 detect events (t', z_{in})
   2.4 compute next possible states (collection C')
   $discrete(s, f(t'), z_{in}) = s'$
   2.5 for all valid elements S in C', add S to C
   (t', s') valid $\implies C := C \cup \{(t', s')\}$

Algorithm

Reachability analysis

1. create collection C containing *init* $\quad \{(0, [init])\}$
2. while collection C is not empty:
   2.1 extract an element (state) $\quad ([t_0], [s])$
   2.2 integrate ODEs $\quad \forall \delta \in [0, t_1],\ [f](\delta) \ni x([t_0] + \delta)$
   2.3 detect events $\quad \{([t'], z_{in}), \dots\}$
   2.4 compute next possible states (collection C')
   $$([t'], z_{in}) \mapsto [discrete]\big([s], [f]([t']), z_{in}\big) = \{[s'], \dots\}$$
   2.5 for all valid elements S in C', add S to C
   $$([t'], [s'])\ \text{valid} \implies C := C \cup \{([t'], [s'])\}$$

Reasoning on the semantics

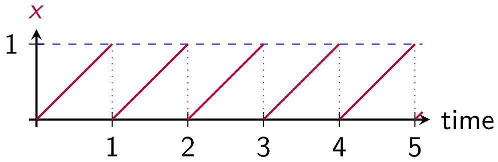
Small language

$$
\begin{array}{rcl}
E & ::= & x = e \mid E \textbf{ and } E \mid \textbf{der } x = e \\
 & & \mid \textbf{if } e \textbf{ then } E \textbf{ else } E \mid () \\
 & & \mid \textbf{init } x = e
\end{array}
$$

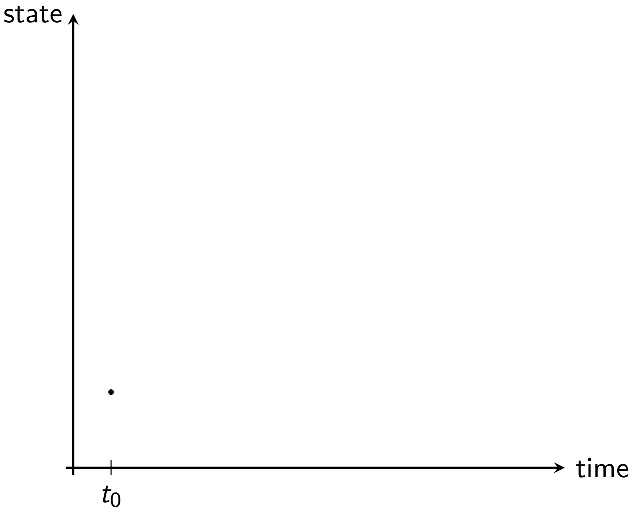
$$
\begin{array}{rcl}
e & ::= & \mathrm{op}(e, \ldots, e) \mid f(e, \ldots, e) \mid \textbf{last}\, x \\
 & & \mid v \mid x \mid \textbf{up}\, e
\end{array}
$$

Example

$$
\begin{array}{ll}
 & \textbf{if } \textbf{up}(x-1) \textbf{ then } x = 0 \textbf{ else } \textbf{der } x = 1 \\
\textbf{and} & \textbf{init } x = 0
\end{array}
$$

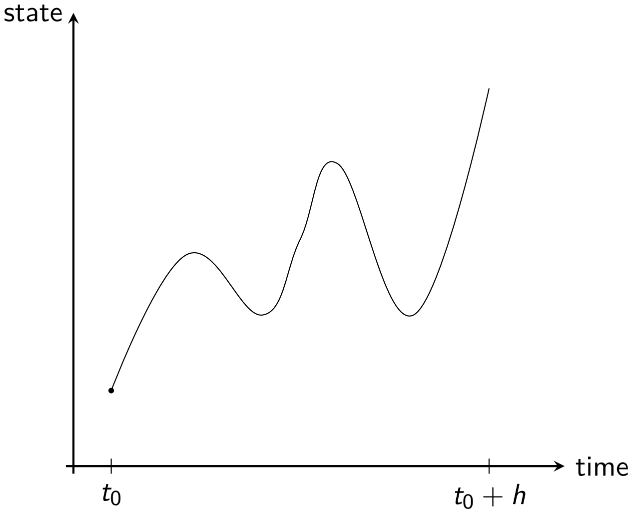


Intuition



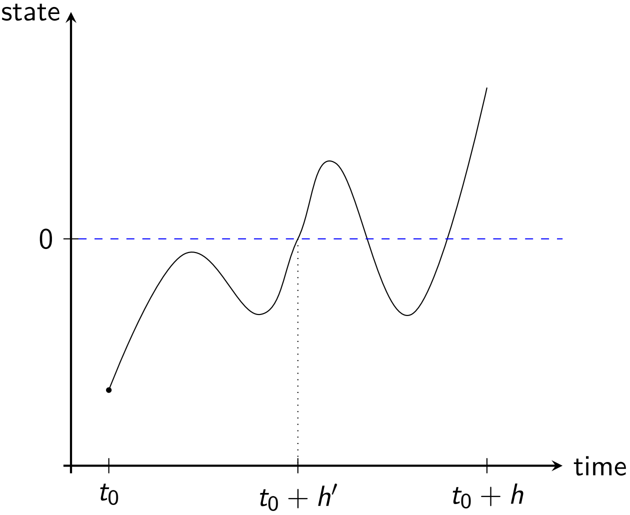
Intuition

Step: *der*



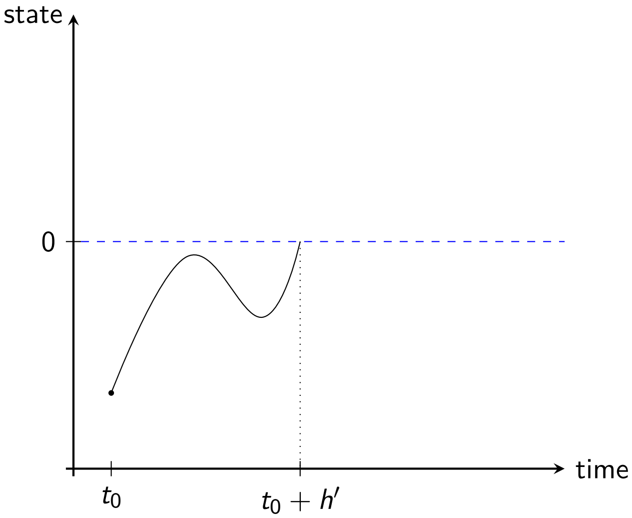
Intuition

Step: *event*



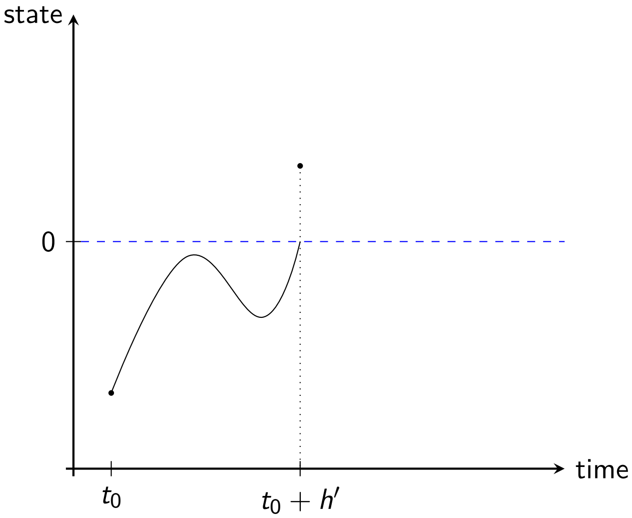
Intuition

Step: *event*



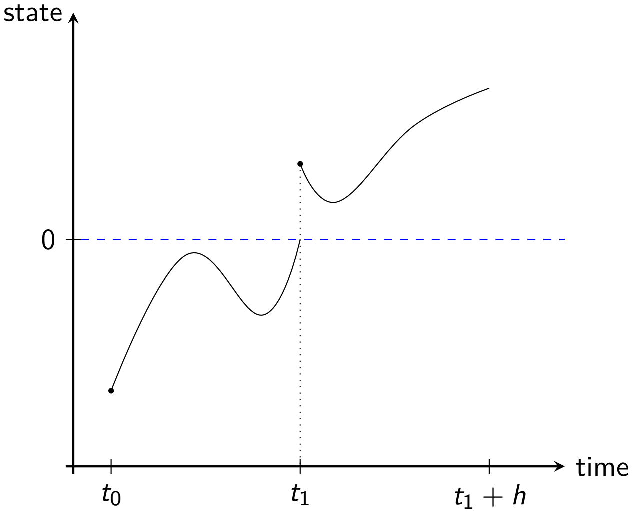
Intuition

Step: *disc*



Intuition

Step: *der*



Semantics

Integration (der)

$\forall n \in \mathbb{N}_{>0}$

$$
\begin{aligned}
{}^{h}[\![e]\!]^{\rho}_{T}(n, der) &\equiv [0,h] \to V \\
{}^{h}[\![E]\!]^{\rho}_{T}(n, der) &\equiv horizon \leq h,\ \ predicate
\end{aligned}
$$

$$
\begin{aligned}
&{}^{h}[\![\mathbf{up}\, x]\!]^{\rho}_{T}(n, der) := t \mapsto \mathbf{false} \\
&{}^{h}[\![x = e]\!]^{\rho}_{T}(n, der) := h, \\
&\qquad\qquad \forall t \in [0,h], \rho(x)(n, der)(t) = {}^{h}[\![e]\!]^{\rho}_{T}(n, der)(t) \\
&{}^{h}[\![\mathbf{der}\ \ x = e]\!]^{\rho}_{T}(n, der) := h', \forall t \in [0,h'],\ \ \rho(x)(n, der)(t) = f(t) \\
&\qquad \text{with } h', f = integrate(\rho(x)(n - 1_T, disc), {}^{h}[\![e]\!]^{\rho}_{T}(n, der)) \\
&{}^{h}[\![\mathbf{if}\ \ e\ \ \mathbf{then}\ \ E_1\ \ \mathbf{else}\ \ E_2]\!]^{\rho}_{T}(n, der) := \mathbf{if}\ \ {}^{h}[\![e]\!]^{\rho}_{T}(n - 1_T, disc) \\
&\qquad\qquad \mathbf{then}\ \ {}^{h}[\![E_1]\!]^{\rho}_{T\ \mathbf{on}\ e}(n, der) \\
&\qquad\qquad \mathbf{else}\ \ {}^{h}[\![E_2]\!]^{\rho}_{T\ \mathbf{on}\ \bar{e}}(n, der) \\
&{}^{h}[\![\mathbf{init}\ \ x = e]\!]^{\rho}_{T}(n, der) := h,\ \ \rho(x)(0_T, disc) = {}^{h}[\![e]\!]^{\rho}_{T}(0_T, disc)
\end{aligned}
$$

Semantics

Event detection (*event*)

$$^{h}[\![e]\!]^{\rho}_{T}(n, event) \equiv horizon, V$$
$$^{h}[\![E]\!]^{\rho}_{T}(n, event) \equiv horizon \leq h,\ \ predicate$$

$$^{h}[\![\mathbf{up}\, x]\!]^{\rho}_{T}(n, event) := h',\ \Big(\exists \varepsilon > 0,\ \forall t \in [h' - \varepsilon, h'[, \\ \rho(x)(n, der)(t) < 0 \wedge \rho(x)(n, der)(h') = 0\Big)$$
$$^{h}[\![x = e]\!]^{\rho}_{T}(n, event) := h',\ (h', \rho(x)(n, event)) = {}^{h}[\![e]\!]^{\rho}_{T}(n, event)$$
$$^{h}[\![\mathbf{der}\ \ x = e]\!]^{\rho}_{T}(n, event) := h, \mathbf{true}$$
$$^{h}[\![\mathbf{if}\ \ e\ \ \mathbf{then}\ \ E_1\ \ \mathbf{else}\ \ E_2]\!]^{\rho}_{T}(n, event) := \mathbf{if}\ \ {}^{h}[\![e]\!]^{\rho}_{T}(n - 1_T, disc) \\ \mathbf{then}\ \ {}^{h}[\![E_1]\!]^{\rho}_{T\ \mathbf{on}\ e}(n, event) \\ \mathbf{else}\ \ {}^{h}[\![E_2]\!]^{\rho}_{T\ \mathbf{on}\ \bar{e}}(n, event)$$
$$^{h}[\![\mathbf{init}\ \ x = e]\!]^{\rho}_{T}(n, event) := h,\ \ \mathbf{true}$$

Semantics

Discrete step (*disc*)

$$ {}^{h}[\![e]\!]^{\rho}_{T}(n, \mathit{disc}) \equiv V $$
$$ {}^{h}[\![E]\!]^{\rho}_{T}(n, \mathit{disc}) \equiv \mathit{predicate} $$

$$
\begin{aligned}
{}^{h}[\![\mathbf{up}\, x]\!]^{\rho}_{T}(n, \mathit{disc}) &:= (h, \mathbf{true}) = {}^{h}[\![\mathbf{up}\, x]\!]^{\rho}_{T}(n, \mathit{event}) \\
{}^{h}[\![x = e]\!]^{\rho}_{T}(n, \mathit{disc}) &:= \rho(x)(n, \mathit{disc}) = {}^{h}[\![e]\!]^{\rho}_{T}(n, \mathit{disc}) \\
{}^{h}[\![\mathbf{der}\ \ x = e]\!]^{\rho}_{T}(n, \mathit{disc}) &:= \mathbf{true} \\
{}^{h}[\![\mathbf{if}\ \ e\ \ \mathbf{then}\ \ E_1\ \ \mathbf{else}\ \ E_2]\!]^{\rho}_{T}(n, \mathit{disc}) &:= \mathbf{if}\ \ {}^{h}[\![e]\!]^{\rho}_{T}(n, \mathit{disc}) \\
&\qquad \mathbf{then}\ \ {}^{h}[\![E_1]\!]^{\rho}_{T\ \mathbf{on}\ e}(n, \mathit{disc}) \\
&\qquad \mathbf{else}\ \ {}^{h}[\![E_2]\!]^{\rho}_{T\ \mathbf{on}\ \bar{e}}(n, \mathit{disc}) \\
{}^{h}[\![\mathbf{init}\ \ x = e]\!]^{\rho}_{T}(n, \mathit{disc}) &:= \mathbf{true}
\end{aligned}
$$

Conclusion

Summary

- over-approximation replacing `float` by representation of sets
- same algorithm for concrete simulation and reachability analysis

Future work

- define proper semantics
- prove over-approximation of the semantics
- implement a prototype

Thank you for your attention