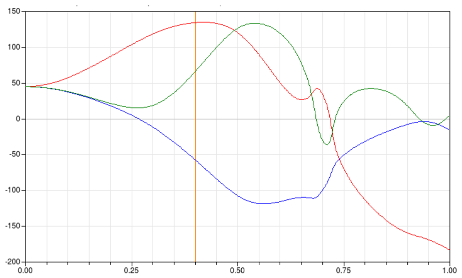
Time Domains in Hybrid Systems Modeling

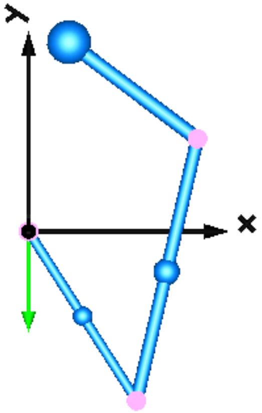
Albert Benveniste[1] Timothy Bourke[1] Benoît Caillaud[1]
Marc Pouzet[3]

1. INRIA
3. École normale supérieure

Brest, July 2016

From Dynamical to Hybrid Systems, informally





Dynamical system: smooth dynamics

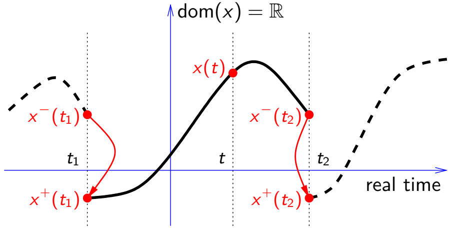
$$x : \mathbb{R} \to \mathbb{R}^n$$

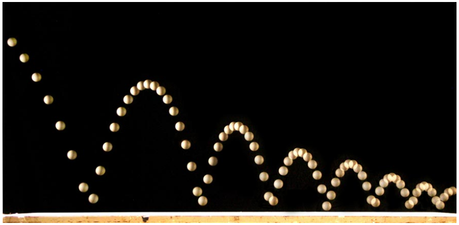
solution of the IVP

$$\begin{cases} f(\dot{x}, x, t) = 0 \\ x(t_0) = x_0 \end{cases}$$

Can we capture Hybrid Systems trajectories as $x : \mathbb{R} \to \mathbb{R}^n$?

From Dynamical to Hybrid Systems, informally



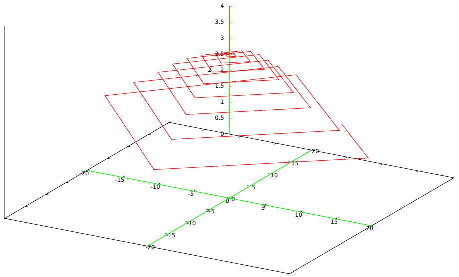


Simple Hybrid Systems: smooth dynamics almost all the time, except for state jumps $x^+ = g(x^-)$ at some discrete t.

$x : \mathbb{R} \to \mathbb{R}^n$ still works.

How general is this?

From Dynamical to Hybrid Systems, informally



$$\begin{cases} \dot{x} = -\mathrm{sgn}(x) + 2\mathrm{sgn}(y) \\ \dot{y} = -2\mathrm{sgn}(x) - \mathrm{sgn}(y) \\ \dot{z} = \mathrm{sgn}(x) + \mathrm{sgn}(y) \end{cases}$$

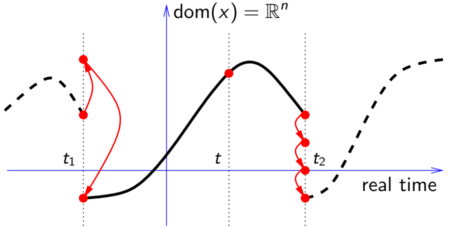
Non-Smooth Dynamical Systems: right-hand of differential equations is non-smooth.

- Filippov Differential Inclusions
- Complementarity Systems
- Siconos numerical library [Acary et al.]

$x : \mathbb{R} \to \mathbb{R}^n$ still works.

However...

From Dynamical to Hybrid Systems, informally



In general, Hybrid Systems trajectory may have:

- Instantaneous cascades of state jumps
- Chattering

Can not be captured as:

$$x : \mathbb{R} \to \mathbb{R}^n$$

Need a Time Domain "denser" than $\mathbb{R}$

Semantics of Hybrid Systems Modelers

Instrumental to design:

1. Static analyzers / model-checkers / theories for interactive provers
2. Compile-time analysis / simulation code generation
3. Numerical simulation environments (run-time)

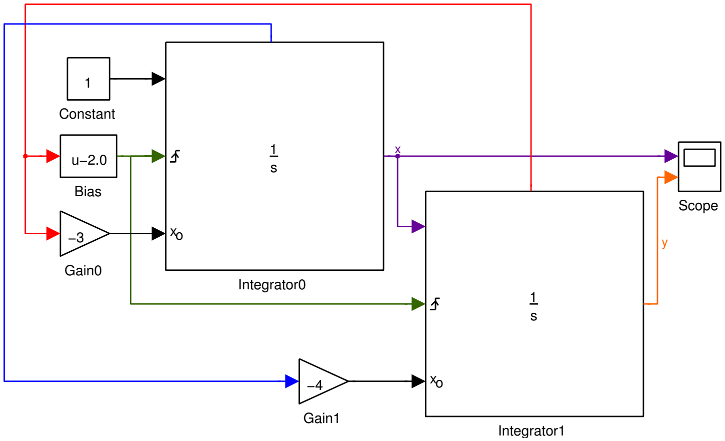
Need for a precise mathematical semantics

Focus of this talk:

- Comparison of Time Domains used to the define the semantics of hybrid systems modelers
- Emphasis on compile-time analysis / simulation code generation

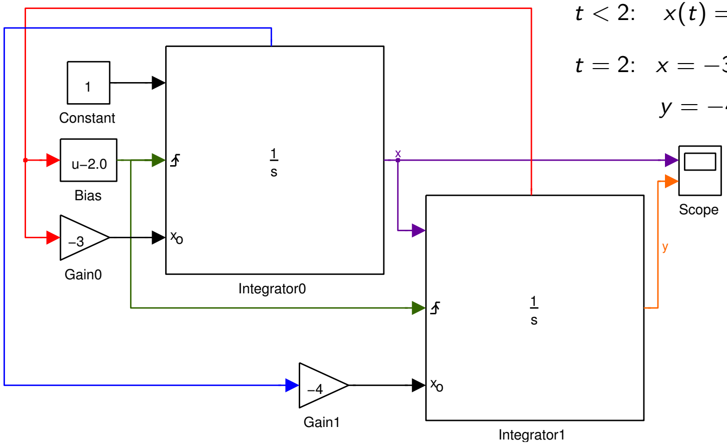
Our Cabinet of Curiosities. . .

Causality issue: the Simulink state port



The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset.
–Simulink Reference (2-685)

Causality issue: the Simulink state port



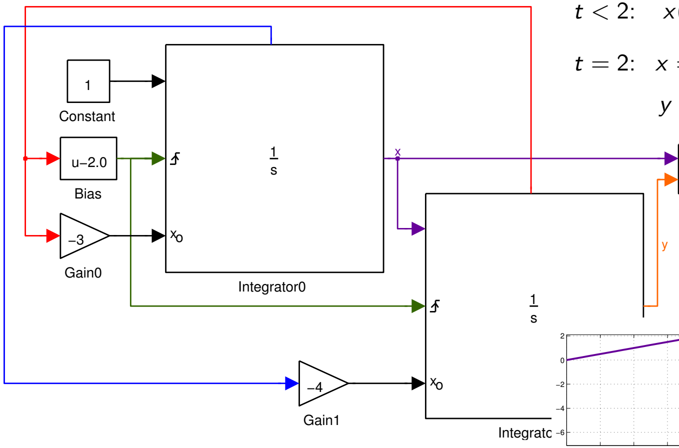
$$t < 2: \quad x(t) = t,\; y(t) = \frac{t^2}{2}$$

$$t = 2: \quad x = -3 \cdot \text{last}\, y = -6,$$

$$y = -4 \cdot \text{last}\, x = -8$$

The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset.
–Simulink Reference (2-685)

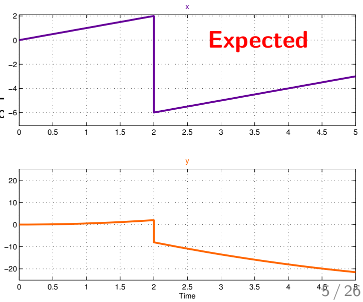
Causality issue: the Simulink state port



$$t < 2: \quad x(t) = t,\ y(t) = \frac{t^2}{2}$$

$$t = 2: \quad x = -3 \cdot \text{last } y = -6,$$

$$y = -4 \cdot \text{last } x = -8$$



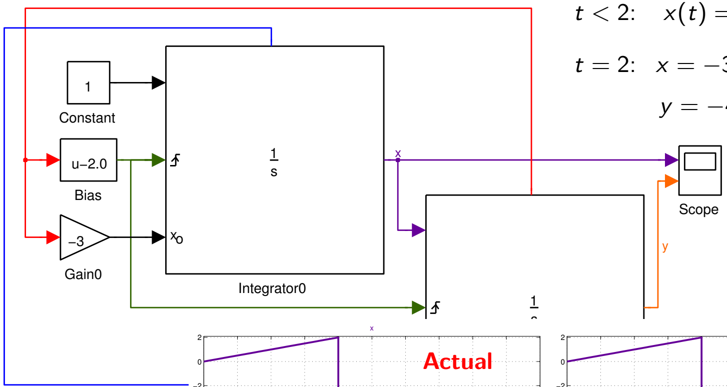
The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset.
–Simulink Reference (2-685)

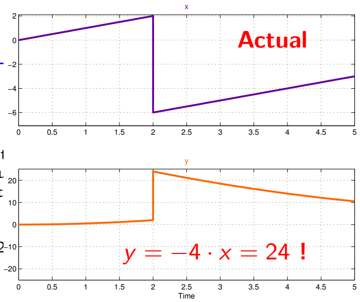
Causality issue: the Simulink state port

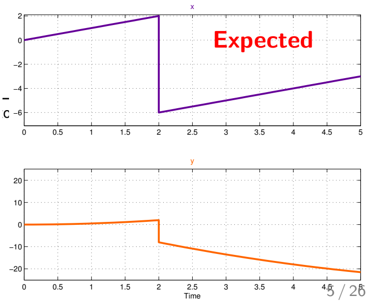
$$t < 2: \quad x(t) = t,\ y(t) = \frac{t^2}{2}$$

$$t = 2: \quad x = -3 \cdot \text{last } y = -6,$$

$$y = -4 \cdot \text{last } x = -8$$







The output of the stat
block's standard outpu
the block is reset in t
state port is the value
standard output if the
–Simulink Reference (2

Excerpt of C code produced by RTW (release R2009)

```
static void mdlOutputs(SimStruct * S, int_T tid)
{ _rtX = (ssGetContStates(S));
  ...
  _rtB = (_ssGetBlockIO(S));
  _rtB->B_0_0_0 = _rtX->Integrator1_CSTATE + _rtP->P_0;
  _rtB->B_0_1_0 = _rtP->P_1 * _rtX->Integrator1_CSTATE;
  if (ssIsMajorTimeStep (S))
    { ...
      if (zcEvent || ...)
        { (ssGetContStates (S))->Integrator0_CSTATE =
              _ssGetBlockIO (S))->B_0_1_0;
        }
      ...
  (_ssGetBlockIO (S))->B_0_2_0 =
    (ssGetContStates (S))->Integrator0_CSTATE;
    _rtB->B_0_3_0 = _rtP->P_2 * _rtX->Integrator0_CSTATE;
    if (ssIsMajorTimeStep (S))
    { ...
      if (zcEvent || ...)
       { (ssGetContStates (S))-> Integrator1_CSTATE =
              (ssGetBlockIO (S))->B_0_3_0;
       }
      ... } ... }
```

Before assignment: integrator state contains 'last' value

$x = -3 \cdot \text{last } y$

After assignment: integrator state contains the new value

$y = -4 \cdot x$

So, y is updated with the new value of x

There is a problem in the treatment of causality.

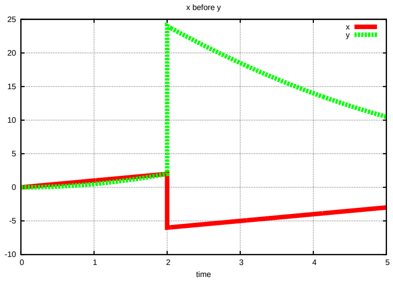
Causality: Modelica example

```
model scheduling
  Real x(start = 0);
  Real y(start = 0);
equation

  der(x) = 1;
  der(y) = x;

  when x >= 2 then
    reinit(x, −3 ∗ y)
  end when;
  when x >= 2 then
    reinit(y, −4 ∗ x);
  end when;

end scheduling;
```

OpenModelica 1.9.2beta1 (r24372)
Also in Dymola

Causality: Modelica example

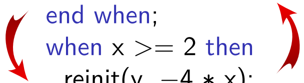
```
model scheduling
  Real x(start = 0);
  Real y(start = 0);
equation

  der(x) = 1;
  der(y) = x;

  when x >= 2 then
    reinit(x, −3 ∗ y)
  end when;
  when x >= 2 then
    reinit(y, −4 ∗ x);
  end when;

end scheduling;
```

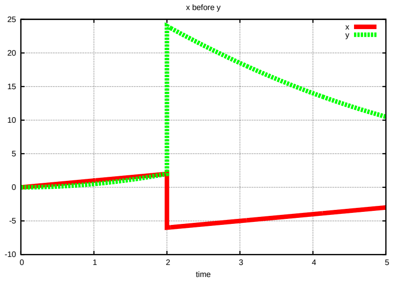


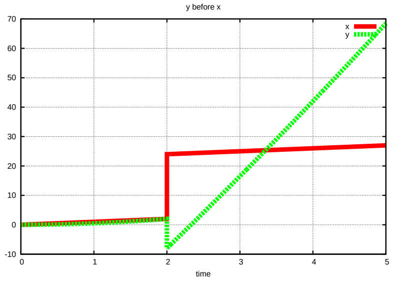
OpenModelica 1.9.2beta1 (r24372)
Also in Dymola

Causality: Modelica example

```
model scheduling
  Real x(start = 0);
  Real y(start = 0);
equation

  der(x) = 1;
  der(y) = x;

  when x >= 2 then
    reinit(x, −3 ∗ y)
  end when;
  when x >= 2 then
    reinit(y, −4 ∗ x);
  end when;

end scheduling;
```







OpenModelica 1.9.2beta1 (r24372)
Also in Dymola

Causality: Modelica example (cont.)

- A causal version (i.e., reinit(x, $-3 * \text{pre } y$) is scheduled properly.
  Normally, everything works correctly.
- But the non-causal program is accepted and the result is not well defined.
  What is the semantics of this program?
- It's not about forbidding algebraic loops, but the expressions here are clocked and not relational.
  Should the solver be left to resolve the non-determinism?
- Such problems are certainly not easy to solve, but
  the semantics of a model must not depend on its layout
- Studying causality is needed to understand the interactions between discrete and continuous-time behaviors.

Background: Synchronous Languages

Syntax of a simple synchronous language ($\approx$ Lustre)

$$
\begin{aligned}
d &::= \texttt{let}\, x = e \mid \texttt{let}\ f(p) = e\, \texttt{where}\, E \mid d; d \\
e &::= x \mid v \mid op(e) \mid e\, \texttt{fby}\, e \mid \texttt{pre}(e) \mid f(e) \mid (e, e) \\
p &::= (p, p) \mid x \\
E &::= () \mid E \;\texttt{and}\; E \mid x \;=\; e \mid \\
&\qquad\;\; \mid \texttt{if}\, e\, \texttt{then}\, E\, \texttt{else}\, E
\end{aligned}
$$

Examples

```
let min_max(x,y) = (a,b) where
  if x<y
  then a = x and b = y
  else a = y and b = x
```

```
let sum(x) = cpt where
  cpt = (0 fby pre(cpt)) + x
```

Background: Semantics of Synhronous Languages

Chronograms

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	1	2	3
y	$=$	3	6	5	1	1	9
$min\_max(x, y)$	$=$	$(2, 3)$	$(4, 6)$	$(2, 5)$	$(1, 1)$	$(1, 2)$	$(3, 9)$
$\texttt{pre}(x)$	$=$	nil	2	4	2	1	2
$x\,\texttt{fby}\,y$	$=$	2	6	5	1	1	9
$sum(x)$	$=$	2	6	8	9	11	14

Examples

```
let min_max(x,y) = (a,b) where
  if x<y
  then a = x and b = y
  else a = y and b = x
```

```
let sum(x) = cpt where
  cpt = (0 fby pre(cpt)) + x
```

Background: Synchronous Languages

Chronograms

time	=	0	1	2	3	4	5
x	=	2	4	2	1	2	3
y	=	3	6	5	1	1	9
$min\_max(x, y)$	=	$(2, 3)$	$(4, 6)$	$(2, 5)$	$(1, 1)$	$(1, 2)$	$(3, 9)$
$\texttt{pre}(x)$	=	*nil*	2	4	2	1	2
$x \,\texttt{fby}\, y$	=	2	6	5	1	1	9
$sum(x)$	=	2	6	8	9	11	14

Main features

- A signal is a sequence of values or stream
- A system is function from streams to streams.
- Operations apply pointwise to their arguments.
- All streams progress synchronously.

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	$\perp$	$\perp$	$\perp$
y	$=$	nil	2	6	$\perp$	$\perp$	$\perp$
z	$=$	0	2	6	$\perp$	$\perp$	$\perp$
cpt	$=$	2	6	8	$\perp$	$\perp$	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \to (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	1	$\perp$	$\perp$
y	$=$	nil	2	6	8	$\perp$	$\perp$
z	$=$	0	2	6	$\perp$	$\perp$	$\perp$
cpt	$=$	2	6	8	$\perp$	$\perp$	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \rightarrow (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	=	0	1	2	3	4	5
x	=	2	4	2	1	$\perp$	$\perp$
y	=	nil	2	6	8	$\perp$	$\perp$
z	=	0	2	6	8	$\perp$	$\perp$
cpt	=	2	6	8	$\perp$	$\perp$	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \to (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

time	=	0	1	2	3	4	5
x	=	2	4	2	1	$\perp$	$\perp$
y	=	*nil*	2	6	8	$\perp$	$\perp$
z	=	0	2	6	8	$\perp$	$\perp$
cpt	=	2	6	8	9	$\perp$	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \rightarrow (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	1	2	$\perp$
y	$=$	nil	2	6	8	9	$\perp$
z	$=$	0	2	6	8	$\perp$	$\perp$
cpt	$=$	2	6	8	9	$\perp$	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \to (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	1	2	$\perp$
y	$=$	nil	2	6	8	9	$\perp$
z	$=$	0	2	6	8	9	$\perp$
cpt	$=$	2	6	8	9	$\perp$	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \to (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	1	2	$\perp$
y	$=$	nil	2	6	8	9	$\perp$
z	$=$	0	2	6	8	9	$\perp$
cpt	$=$	2	6	8	9	11	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \to (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	1	2	3
y	$=$	nil	2	6	8	9	11
z	$=$	0	2	6	8	9	$\perp$
cpt	$=$	2	6	8	9	11	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \to (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	1	2	3
y	$=$	nil	2	6	8	9	11
z	$=$	0	2	6	8	9	11
cpt	$=$	2	6	8	9	11	$\perp$

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\perp\}$: $\perp$ if undefined, $\perp < v \in V$
- $S(V) = \mathbb{N} \to (V \uplus \{\perp\})$

Background: Constructive Fixpoint Semantics

Define semantics as mutual least fixpoint of set of monotonous operators (one for each equation) [Berry 1999]

Step-by-step execution

$time$	$=$	0	1	2	3	4	5
x	$=$	2	4	2	1	2	3
y	$=$	nil	2	6	8	9	11
z	$=$	0	2	6	8	9	11
cpt	$=$	2	6	8	9	11	14

Program

```
let sum(x) = cpt where
    y = pre(cpt)
and z = 0 fby y
and cpt = z + x
```

Extended domains and streams

- $t \in \mathbb{N}$ dicrete time
- $v \in V \uplus \{\bot\}$: $\bot$ if undefined, $\bot < v \in V$
- $S(V) = \mathbb{N} \to (V \uplus \{\bot\})$

Requirements on Semantics

Recall, semantics to help designing:

1. Static analyzers / model-checkers / theories for interactive provers
2. Compile-time analysis / simulation code generation
3. Numerical simulation environments (run-time)

Therefore:

- Every well-typed program E should have a semantics $[\![E]\!]$
- The semantics should be structural, i.e., roughly speaking:

$$\begin{aligned} [\![E_1 \texttt{ and } E_2]\!] &= \{[\![E_1]\!]; [\![E_2]\!]\} \\ [\![\texttt{if } e \texttt{ then } E_1 \texttt{ else } E_2]\!] &= \texttt{if } [\![e]\!] \texttt{ then } [\![E_1]\!] \texttt{ else } [\![E_2]\!], \text{ etc.} \end{aligned}$$

- The alternative is informal "mytool" semantics

Requirements on Semantics

Recall, semantics to help designing:

1. Static analyzers / model-checkers / theories for interactive provers
2. Compile-time analysis / simulation code generation
3. Numerical simulation environments (run-time)

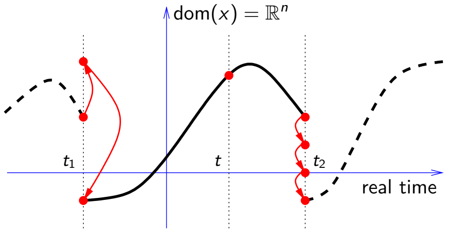
Therefore:

- Every well-typed program E should have a semantics $[\![E]\!]$
- The semantics should be structural, i.e., roughly speaking:

$$\begin{aligned} [\![E_1 \ \texttt{and}\ E_2]\!] &= \{[\![E_1]\!];[\![E_2]\!]\} \\ [\![\texttt{if}\ e\ \texttt{then}\ E_1\ \texttt{else}\ E_2]\!] &= \texttt{if}\ [\![e]\!]\ \texttt{then}\ [\![E_1]\!]\ \texttt{else}\ [\![E_2]\!],\ \text{etc.} \end{aligned}$$

- The alternative is informal "mytool" semantics

Time Domains

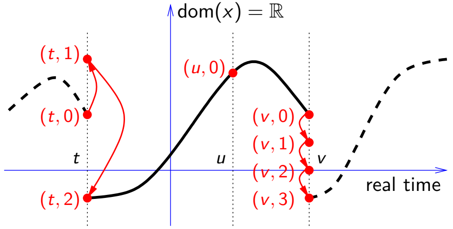


Phases of continuous dynamics interleaved with cascades of instantaneous state-jumps

However:

- Cascades may be complex or even unbounded
- The Time Domain should be such that time may progress during cascades of state-jumps

Time Domains



Superdense Model of Time:
$\mathbb{T} = \mathbb{R}_+ \times \mathbb{N}$
[Pnueli et al. 1992]
[Lee et al. 2005]

$\mathbb{T}$ is equipped with lexicographic order (as shown on the figure).

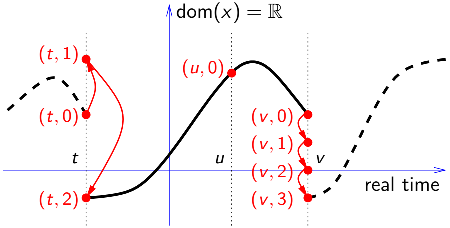
Two approaches for capturing signals with finite cascades of changes:

1. $x(t,n)$ defined for $0 \le n \le m_t$ and undefined for $n > m_t$ [the figure]
2. $x(t,n)$ defined for every n but $x(t,n) = x(t,m_t)$ for $n > m_t$ [Lee]

where m_t is the number of changes at time t.

In the figure: $m_t = 2, m_u = 0, m_v = 3$.

Time Domains

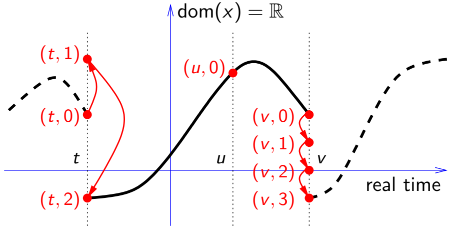


Superdense Model of Time:
$\mathbb{T} = \mathbb{R}_+ \times \mathbb{N}$
[Pnueli et al. 1992]
[Lee et al. 2005]

[Lee 2014]:

> *Such piecewise-continuous signals coexist nicely with standard ODE solvers. At the time of discontinuity or discrete event, the final value signal provides the initial boundary condition for the solver. [. . .]*

Time Domains

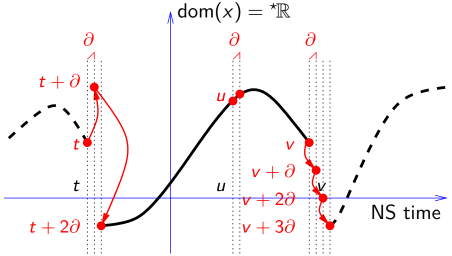


Superdense Model of Time

$\mathbb{T} = \mathbb{R}_+ \times \mathbb{N}$

[Pnueli et al. 1992]

[Lee et al. 2005]



Nonstandard Model of Time

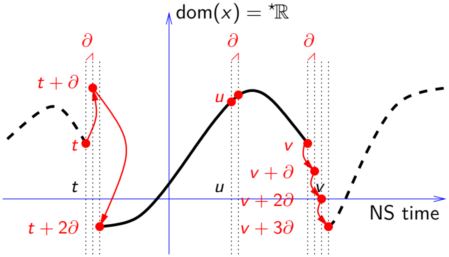
$\mathbb{T} = \{n\partial \mid n \in {}^\star\mathbb{N}\}$

[Benveniste et al. 2012]

Time Domains

Aim:

- getting rid of the burden of smoothness assumptions
- making hybrid systems discrete
- getting the semantics by reusing techniques from discrete systems



Nonstandard Model of Time

$\mathbb{T} = \{n\partial \mid n \in {}^\star\mathbb{N}\}$

[Benveniste et al. 2012]

A Brief History of Non-Standard Analysis

- Infinitesimal numbers have been considered by Archimède, Newton, Fermat, Leibniz, ... but lacked rigorous foundations.
- Non-standard Analysis was proposed by Abraham Robinson in 1961, then developed by a small community of mathematicians.
- Defined as a conservative enhancement of Zermelo-Fränkel set theory; some fancy axioms and principles; nice for the addicts
- Subject of controversies: what does it do for you that you cannot do using our brave analysis with $\forall\varepsilon\exists\eta\ldots$?
- Instrumental to solve a few problems: Algebraic topology, Stochastic processes (Brownian motion), ...
- 1988: a nice presentation of the topic by T. Lindstrom, kind of a "*paraphrase of the construction of reals*"
- 2006: used in Simon Bliudze PhD where he proposes the counterpart of a "Turing machine" for hybrid systems (supervised by D. Krob)
- 2010: used by Benveniste et al. to define the operational semantics of the Zélus hybrid systems modeling language.

Non-Standard Analysis in 1'

- ${}^{\star}\mathbb{R} \supset \mathbb{R} \cup \{-\infty, +\infty\}$
- Contains infinitely small numbers ($\exists \epsilon \in {}^{\star}\mathbb{R}, \forall x \in \mathbb{R}_{+}^{*}, 0 < \epsilon < x$) and infinitely large numbers $1/\epsilon$.
- Every non-standard number $x \in {}^{\star}\mathbb{R}$ admits a unique decomposition $x = st(x) + \epsilon$ into a real part $st(x) \in \mathbb{R}$ and an infinitesimal part ϵ
- Densification of the real line: for every $x \in \mathbb{R}$, there are infinitely many non-standard numbers y that are infinitely close to y: $y \sim x$
- $x \sim y$ iff $st(x - y) = 0$
- Infinite hierarchy of infinitesimals $\ldots < \epsilon^2 < \epsilon$ and infinites $1/\epsilon < 1/\epsilon^2 < \ldots$.
- Almost as usual: internalizing real operators and functions $x + y$, $f(x)$, ...
- Transfer principle: Given a first order formula ψ,

$$\vDash_{\mathbb{R}} \psi \iff \vDash_{{}^{\star}\mathbb{R}} \psi$$

A Toy Hybrid Systems Language

Syntax ≈ Zélus [Bourke et al. 2013] ≈ SCADE Hybrid [ANSYS 2015]

$$
\begin{array}{rcl}
d & ::= & \mathtt{let}\ x = e \mid \mathtt{let}\ f(p) = e\ \mathtt{where}\ E \mid d; d \\
e & ::= & x \mid v \mid op(e) \mid e\ \mathtt{fby}\ e \mid \mathtt{pre}(e) \mid f(e) \mid (e, e) \\
p & ::= & (p, p) \mid x \\
E & ::= & () \mid E\ \mathtt{and}\ E \mid x\ =\ e \mid \\
  &   & \mid \mathtt{init}\ x\ =\ e \mid \mathtt{der}\ x\ =\ e \mid \\
  &   & \mid \mathtt{if}\ e\ \mathtt{then}\ E\ \mathtt{else}\ E \\
  &   & \mid \mathtt{der}\ x\ =\ e \mid \\
  &   & \mid \mathtt{init}\ x\ =\ e \mid \mathtt{reinit}\ x\ =\ e \mid \\
  &   & \mid \mathtt{when}\ e\ \mathtt{do}\ E
\end{array}
$$

Zélus zelus.dmi.ens.fr

- Zélus $\approx$ Lucid Synchrone [Pouzet 2006] + ODEs
- Dataflow equations, hierarchical state machines [Emsoft 2011], arrays, higher-order (coming soon)
- Precise semantics , based on non-standard analysis [JCSS 2012]
- Type inference (rejects incorrect composition of discrete-/continuous-time dynamics) [LCTES 2011]
- Causality analysis [HSCC 2014]
- Similar initiative: SCADE Hybrid (ANSYS) [CC 2015], required changing $\sim$ 5% of the KCG compiler

The Superdense Model of Time as a semantic domain

- $\mathbb{T} =_{\text{def}} \mathbb{R}_+ \times \mathbb{N}$; we identify $(t, 0) \in \mathbb{T}$ with $t \in \mathbb{R}_+$
- $x_{(t,n)}$ remains constant for $n \geq m^x_t$

equation	semantics
der $x = f(x, u)$; init $x = e$	$m^u_t = m^x_t = 0$ and $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$ and $x_0 = [\![e]\!]_0$ (1)
der $x = f(x, u)$; init $x = a$; when $x \geq 1$ do reinit $x = b$	$t_0 = 0$ and $t_{n+1} =$ $\inf\{s > t_n \mid \forall r \in (s - \varepsilon; s), x_r < 1 \wedge x_s \geq 1\}$ reset effective at $(t_n, 1)$, hence $m^x_{t_n} = 1$ $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$, for $t_n < t \leq t_{n+1}$, (2) $x_{t_0} = [\![a]\!]_{t_0}$, $x_{(t_n,1)} = [\![b]\!]_{(t_n,1)}$, $n \geq 1$ $\neq$ (1)
when z do reinit $x = b$	reuse of (2) not possible since $m^z_{t_n} \neq 0$

The Superdense Model of Time as a semantic domain

- $\mathbb{T} =_{\text{def}} \mathbb{R}_+ \times \mathbb{N}$; we identify $(t,0) \in \mathbb{T}$ with $t \in \mathbb{R}_+$
- $x_{(t,n)}$ remains constant for $n \geq m_t^x$

equation	semantics
$\text{der}\, x = f(x,u)$; $\text{init}\, x = e$	$m_t^u = m_t^x = 0$ and $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$ and $x_0 = [\![e]\!]_0$ (1)
$\text{der}\, x = f(x,u)$; $\text{init}\, x = a$; $\text{when}\, x \geq 1\, \text{do}$ $\text{reinit}\, x = b$	$t_0 = 0$ and $t_{n+1} =$ $\inf\{s > t_n \mid \forall r \in (s-\varepsilon; s), x_r < 1 \wedge x_s \geq 1\}$ reset effective at $(t_n, 1)$, hence $m_{t_n}^x = 1$ $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$, for $t_n < t \leq t_{n+1}$, (2) $x_{t_0} = [\![a]\!]_{t_0}$, $x_{(t_n,1)} = [\![b]\!]_{(t_n,1)}$, $n \geq 1$ $\neq$ (1)
$\text{when}\, z\, \text{do}$ $\text{reinit}\, x = b$	reuse of (2) not possible since $m_{t_n}^z \neq 0$

The Superdense Model of Time as a semantic domain

- $\mathbb{T} =_{\text{def}} \mathbb{R}_+ \times \mathbb{N}$; we identify $(t, 0) \in \mathbb{T}$ with $t \in \mathbb{R}_+$
- $x_{(t,n)}$ remains constant for $n \geq m^x_t$

equation	semantics
$\text{der}\, x = f(x, u)$; $\text{init}\, x = e$	$m^u_t = m^x_t = 0$ and $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$ and $x_0 = [\![e]\!]_0$ (1)
$\text{der}\, x = f(x, u)$; $\text{init}\, x = a$; $\text{when}\, x \geq 1\, \text{do}$ $\text{reinit}\, x = b$	$t_0 = 0$ and $t_{n+1} = \inf\{s > t_n \mid \forall r \in (s - \varepsilon; s), x_r < 1 \wedge x_s \geq 1\}$ reset effective at $(t_n, 1)$, hence $m^x_{t_n} = 1$ $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$, for $t_n < t \leq t_{n+1}$, (2) $x_{t_0} = [\![a]\!]_{t_0}$, $x_{(t_n,1)} = [\![b]\!]_{(t_n,1)}$, $n \geq 1$ $\neq$ (1)
$\text{when}\, z\, \text{do}$ $\text{reinit}\, x = b$	reuse of (2) not possible since $m^z_{t_n} \neq 0$

The Superdense Model of Time as a semantic domain

- $\mathbb{T} =_{\text{def}} \mathbb{R}_+ \times \mathbb{N}$; we identify $(t, 0) \in \mathbb{T}$ with $t \in \mathbb{R}_+$
- $x_{(t,n)}$ remains constant for $n \geq m^x_t$

equation	semantics
$\text{der}\, x = f(x, u);$ $\text{init}\, x = e$	$m^u_t = m^x_t = 0$ and $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$ and $x_0 = [\![e]\!]_0$ (1)
$\text{der}\, x = f(x, u);$ $\text{init}\, x = a;$ $\text{when}\, x \geq 1\, \text{do}$ $\text{reinit}\, x = b$	$t_0 = 0$ and $t_{n+1} =$ $\inf\{s > t_n \mid \forall r \in (s - \varepsilon; s), x_r < 1 \wedge x_s \geq 1\}$ reset effective at $(t_n, 1)$, hence $m^x_{t_n} = 1$ $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$, for $t_n < t \leq t_{n+1}$, (2) $x_{t_0} = [\![a]\!]_{t_0}$, $x_{(t_n,1)} = [\![b]\!]_{(t_n,1)}$, $n \geq 1$ $\neq$ (1)
$\text{when}\, z\, \text{do}$ $\text{reinit}\, x = b$	reuse of (2) not possible since $m^z_{t_n} \neq 0$

The Superdense Model of Time as a semantic domain

- $\mathbb{T} =_{\text{def}} \mathbb{R}_+ \times \mathbb{N}$; we identify $(t, 0) \in \mathbb{T}$ with $t \in \mathbb{R}_+$
- $x_{(t,n)}$ remains constant for $n \geq m^x_t$

equation	semantics
$\operatorname{der} x = f(x, u)$; $\operatorname{init} x = e$	$m^u_t = m^x_t = 0$ and $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$ and $x_0 = [\![e]\!]_0$ (1)
$\operatorname{der} x = f(x, u)$; $\operatorname{init} x = a$; when $x \geq 1$ do reinit $x = b$	$t_0 = 0$ and $t_{n+1} = \inf\{s > t_n \mid \forall r \in (s - \varepsilon; s), x_r < 1 \wedge x_s \geq 1\}$ reset effective at $(t_n, 1)$, hence $m^x_{t_n} = 1$ $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$, for $t_n < t \leq t_{n+1}$, (2) $x_{t_0} = [\![a]\!]_{t_0}$, $x_{(t_n,1)} = [\![b]\!]_{(t_n,1)}$, $n \geq 1$ $\neq$ (1)
when z do reinit $x = b$	reuse of (2) not possible since $m^z_{t_n} \neq 0$

The Superdense Model of Time as a semantic domain

- $\mathbb{T} =_{\text{def}} \mathbb{R}_+ \times \mathbb{N}$; we identify $(t, 0) \in \mathbb{T}$ with $t \in \mathbb{R}_+$
- $x_{(t,n)}$ remains constant for $n \geq m_t^x$

equation	semantics
$\text{der}\, x = f(x,u);$ $\text{init}\, x = e$	$m_t^u = m_t^x = 0$ and $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$ and $x_0 = [\![e]\!]_0$ (1)
$\text{der}\, x = f(x,u);$ $\text{init}\, x = a;$ $\text{when}\, x \geq 1\, \text{do}$ $\text{reinit}\, x = b$	$t_0 = 0$ and $t_{n+1} =$ $\inf\{s > t_n \mid \forall r \in (s - \varepsilon; s), x_r < 1 \wedge x_s \geq 1\}$ reset effective at $(t_n, 1)$, hence $m_{t_n}^x = 1$ $\dot{x}_t = [\![f]\!]_t(x_t, u_t)$, for $t_n < t \leq t_{n+1}$, (2) $x_{t_0} = [\![a]\!]_{t_0}$, $x_{(t_n,1)} = [\![b]\!]_{(t_n,1)}$, $n \geq 1$ $\neq$ (1)
$\text{when}\, z\, \text{do}$ $\text{reinit}\, x = b$	reuse of (2) not possible since $m_{t_n}^z \neq 0$

The Superdense Model of Time as a semantic domain

Such piecewise-continuous signals coexist nicely with standard ODE solvers. At the time of discontinuity or discrete event, the final value signal provides the initial boundary condition for the solver. [. . .]

Lessons:

- Superdense time semantics seems simple as long as you keep it informal
- Actually, it is hard to formalize
- In addition to the problems shown:
  - Smoothness assumptions are needed, and
  - Must be stated on the global system
  - Can not capture chattering (sliding modes).
- [Lee 2014]: getting rid of the above difficulties by moving to constructive semantics?

The Superdense Model of Time as a semantic domain

Moving to constructive semantics

- [Berry 1999] The *constructive semantics* gives a meaning to fixpoint problems specified via sets of equations
  - does not rely on arguments of numerical analysis (convergence of approximation schemes)
  - uses instead fixpoint theorems where the distance between signals is defined as the largest prefix of time in which the two signals coincide
  - constructive $\Rightarrow$ helps understanding causality issues
- No constructive semantics exists for continuous-time systems ($\mathbb{T} = \mathbb{R}_+$) [Matsikoudis and Lee 2014]
- [Lee 2014] invokes constructive semantics as given by the solver (which works by steps)
  - Non compositional, not structural
  - Depends on munerical convergence properties of discretization scheme

The Superdense Model of Time as a semantic domain

Moving to constructive semantics

- [Berry 1999] The *constructive semantics* gives a meaning to fixpoint problems specified via sets of equations
  - does not rely on arguments of numerical analysis (convergence of approximation schemes)
  - uses instead fixpoint theorems where the distance between signals is defined as the largest prefix of time in which the two signals coincide
  - constructive $\Rightarrow$ helps understanding causality issues
- No constructive semantics exists for continuous-time systems ($\mathbb{T} = \mathbb{R}_+$) [Matsikoudis and Lee 2014]
- [Lee 2014] invokes constructive semantics as given by the solver (which works by steps)
  - Non compositional, not structural
  - Depends on munerical convergence properties of discretization scheme

The Nonstandard Time Domain

$$
\begin{aligned}
{}^\star\mathbb{N}, {}^\star\mathbb{R} &=_{\text{def}} \text{non-standard extensions of } \mathbb{N}, \mathbb{R} \\
{}^\star\mathbb{R} \supseteq \mathbb{T} &=_{\text{def}} \{t_n = n\partial \mid n \in {}^\star\mathbb{N}\} \text{ where } \partial \text{ is an \textit{infinitesimal} time step} \\
{}^\bullet t &=_{\text{def}} \max\{s \mid s \in \mathbb{T}, s < t\} = t - \partial \\
t^\bullet &=_{\text{def}} \min\{s \mid s \in \mathbb{T}, s > t\} = t + \partial \\
\dot{x}_t &=_{\text{def}} \frac{x_{t^\bullet} - x_t}{\partial} \text{ (explicit scheme)} \quad \text{or} \quad \frac{x_t - x_{{}^\bullet t}}{\partial} \text{ (implicit scheme)}
\end{aligned}
$$

- with the non-standard interpretation, hybrid systems become "discrete time" and inherit a non-standard semantics
  - no more difficult than Lustre semantics
  - every syntactically correct program has a semantics
  - the non-standard semantics is structural and compositional
- does not depend on the particular choice for the time base ∂

Nonstandard Semantics

Set $\,^{\bullet}x_t = x_{^\bullet t}$, $x_t^{\bullet} = x_{t^\bullet}$, and $\dot{x}_t = \dfrac{x_t^{\bullet} - x_t}{\partial}$ in:

equation	semantics
der $x = e$; init $x = f$	$x_{t_0} = [\![f]\!]_{t_0}$ and $x_t^{\bullet} = x_t + \partial[\![e]\!]_t$ forall $t \in \mathbb{T}, t \geq t_0$
der $x = e$; init $x = a$; when $x \geq 1$ do reinit $x = b$	$z = {}^{\bullet}x_t{<}1 \wedge x_t{\geq}1$ $x_{t_0} = [\![a]\!]_{t_0}$ $x_t^{\bullet} =$ if z then $[\![b]\!]_{t^\bullet}$ else $x_t + \partial[\![e]\!]_t$, $t \geq t_0$

- Just as for Lustre
- Since the non-standard semantics is step-based, constructive semantics exists [Benveniste et al. 2012]
  - Having $^*\mathbb{N}$ many steps instead of $\mathbb{N}$ many ones is not an issue
  - Of course, this semantics can not be used for simulation ($\neq$ programming languages)

Nonstandard Semantics

Set $\,^\bullet x_t = x_{^\bullet t}$, $x_t^\bullet = x_{t^\bullet}$, and $\dot{x}_t = \dfrac{x_t^\bullet - x_t}{\partial}$ in:

equation	semantics
der $x = e$; init $x = f$	$x_{t_0} = [\![f]\!]_{t_0}$ and $x_t^\bullet = x_t + \partial [\![e]\!]_t$ forall $t \in \mathbb{T}, t \geq t_0$
der $x = e$; init $x = a$; when $x \geq 1$ do reinit $x = b$	$z = {}^\bullet x_t {<} 1 \wedge x_t {\geq} 1$ $x_{t_0} = [\![a]\!]_{t_0}$ $x_t^\bullet =$ if z then $[\![b]\!]_{t^\bullet}$ else $x_t + \partial [\![e]\!]_t$, $t \geq t_0$

- Just as for Lustre
- Since the non-standard semantics is step-based, constructive semantics exists [Benveniste et al. 2012]
  - Having $^\star\mathbb{N}$ many steps instead of $\mathbb{N}$ many ones is not an issue
  - Of course, this semantics can not be used for simulation ($\neq$ programming languages)

There is no free lunch

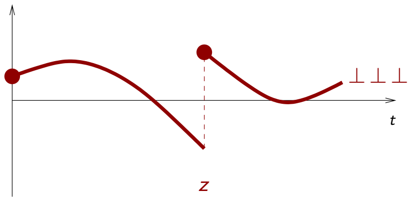
Theorem [Benveniste et al. 2014]

The nonstandard semantics of every causally-correct program is:

1. standardizable,
2. independent of ∂,
3. continuous

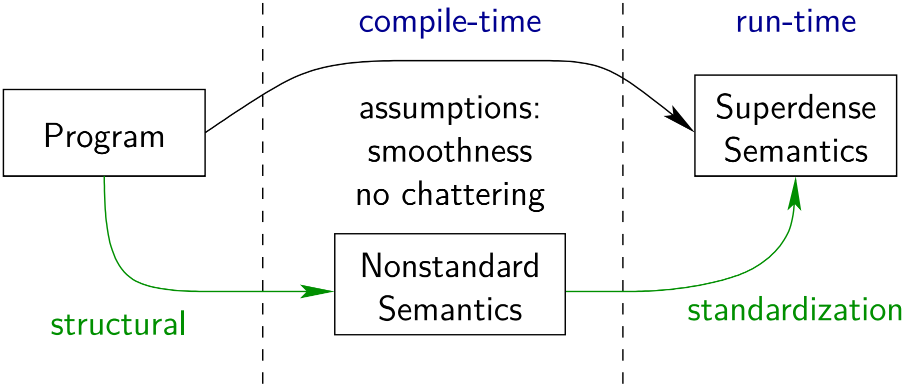
on every compact set of dates not containing:

1. an event, or
2. an undefined value ($\perp$)



- When defined, the superdense semantics coincides with the nonstandard semantics
- The nonstandard semantics is not effective (cannot be executed)

There is no free lunch



DAE Hybrid Systems: index theory & reduction

- With non-standard semantics, DAE become *dAE* (*difference* Algebraic Equations); define $x^\bullet =$ next x
- dAE may involve more equations than specified

$$\left\{\begin{array}{lcl} x^\bullet & = & f(x,u) \\ 0 & = & g(x) \end{array}\right. \quad \overset{\text{shifting}}{\Longrightarrow} \quad \left\{\begin{array}{lcl} x^\bullet & = & f(x,u) \\ 0 & = & g(x) \\ 0 & = & g(x^\bullet) \end{array}\right.$$

$$\overset{\text{substituting}}{\Longrightarrow} \quad \left\{\begin{array}{lcll} x^\bullet & = & f(x,u) & (1) \\ 0 & = & g(x) & (2) \\ 0 & = & g(f(x,u)) & (3) \end{array}\right.$$

Whence the constructive semantics ($\sim$ execution scheme):

1. Given x such that $g(x) = 0$
2. Use (3) to evaluate u (constraint solver needed)
3. Use (1) to evaluate $x^\bullet$, which satisfies $g(x^\bullet) = 0$, and repeat

DAE Hybrid Systems: index theory & reduction

- With non-standard semantics, DAE become *dAE* (*difference* Algebraic Equations); define $x^\bullet =$ next x
- dAE may involve more equations than specified

$$\left\{\begin{array}{lcl} x^\bullet & = & f(x,u) \\ 0 & = & g(x) \end{array}\right. \quad \overset{\text{shifting}}{\Longrightarrow} \quad \left\{\begin{array}{lcl} x^\bullet & = & f(x,u) \\ 0 & = & g(x) \\ 0 & = & g(x^\bullet) \end{array}\right.$$

$$\overset{\text{substituting}}{\Longrightarrow} \quad \left\{\begin{array}{lcll} x^\bullet & = & f(x,u) & (1) \\ 0 & = & g(x) & (2) \\ 0 & = & g(f(x,u)) & (3) \end{array}\right.$$

Thm: the diff. index of a DAE coincides with the index of the dAE obtained with the non-standard semantics

Cor: Defining the index of DAE Hybrid Systems as the index of its non-standard semantics yields a conservative extension of DAE and dAE indexes

Conclusion

- The superdense model of time is useful as a simulation semantics:
  - Even from this point of view it has limits
  - No support for nonsmooth dynamical systems simulation (with possible chattering)
- More is needed for supporting compilation:
  - Structural semantics
  - Getting rid of smoothness assumptions
- The nonstandard model of time is a good candidate:
  - Yields a structural semantics
  - No smoothness assumption
  - Coincides with superdense semantics, when defined
  - Supports the slicing of execution engine into
    - an event handler and
    - a ODE/DAE/nonsmooth solver