Quelques défis en modélisation des systèmes hybrides [1]

Benoît Caillaud

Inria
Rennes, France

MSR'13
15 novembre 2013

[1] Collaboration avec Albert Benveniste, Timothy Bourke et Marc Pouzet.

Modélisation des systèmes hybrides

Programmation des systèmes discrets et modélisation de leur environnement physique dans un même langage

Une multitude d'outils

- Simulink/Stateflow, LabVIEW, Modelica, Ptolemy, ...

Physique = non-régularité

électronique de puissance

mécanique multicorps

modes glissants

Modélisation des systèmes hybrides

Programmation des systèmes discrets et modélisation de leur environnement physique dans un même langage

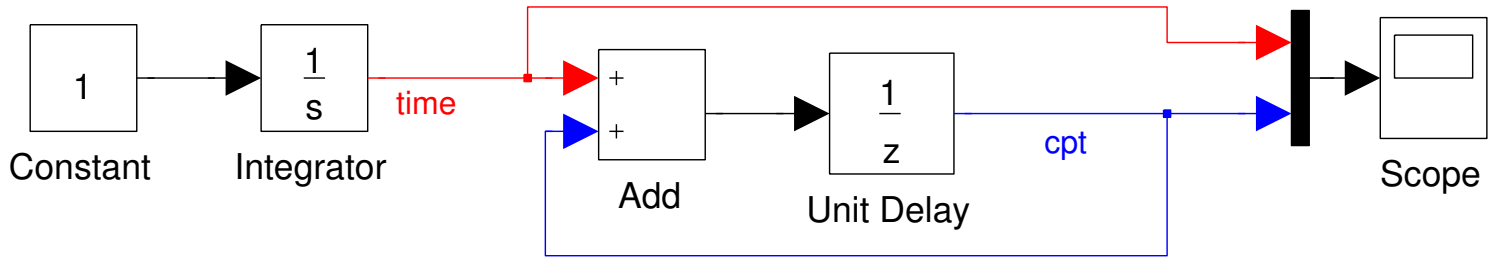
Focalisation sur la conception des langages pour une modélisation et une simulation fidèle

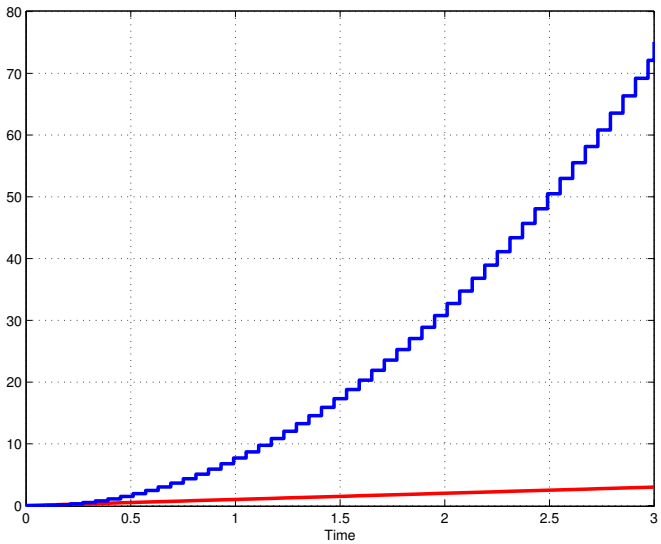
Notre approche

- Conception d'un langage/outil de modélisation hybride sur la base d'un langage réactif synchrone
- Recyclage de techniques de compilation issues des langages synchrones (SCADE / Lustre)
- Clarifier les principes et la sémantique des systèmes hybrides

Le cabinet des monstres. . .

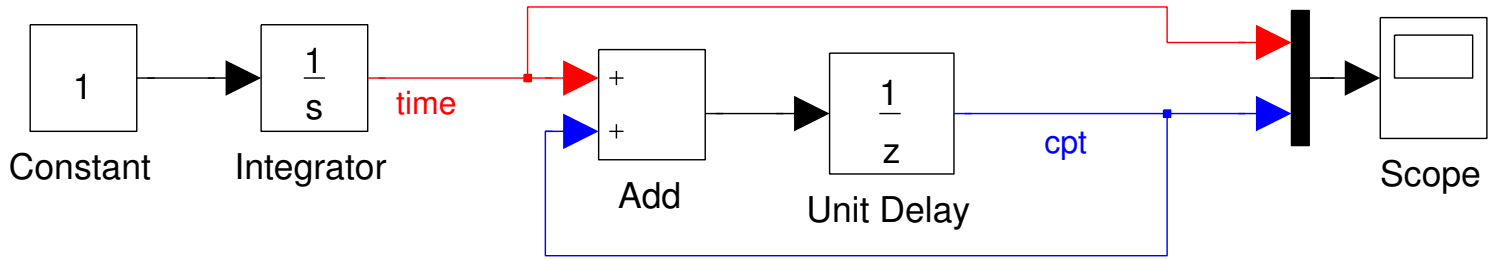
Problème de typage : mélange du continu et du discret

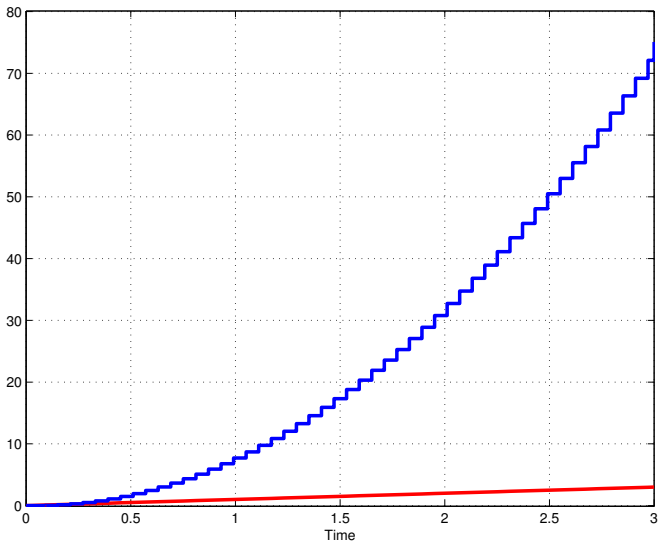




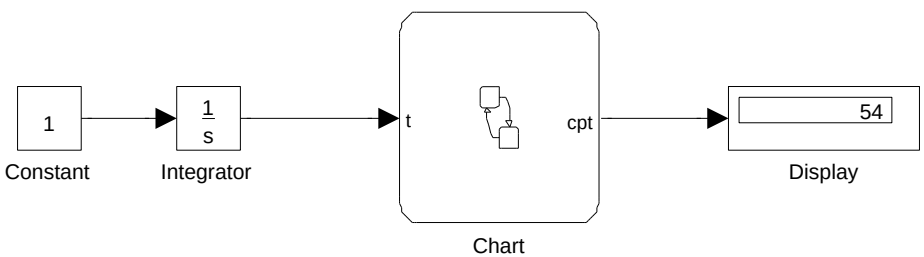
- Warning with 'Unit Delay' but not with 'Memory'.
- The shape of cpt depends on the steps chosen by the solver.
- Putting another component in parallel can change the result.

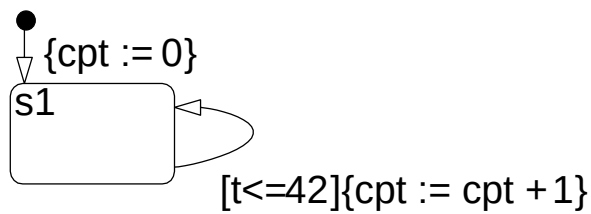
Problème de typage : mélange du continu et du discret



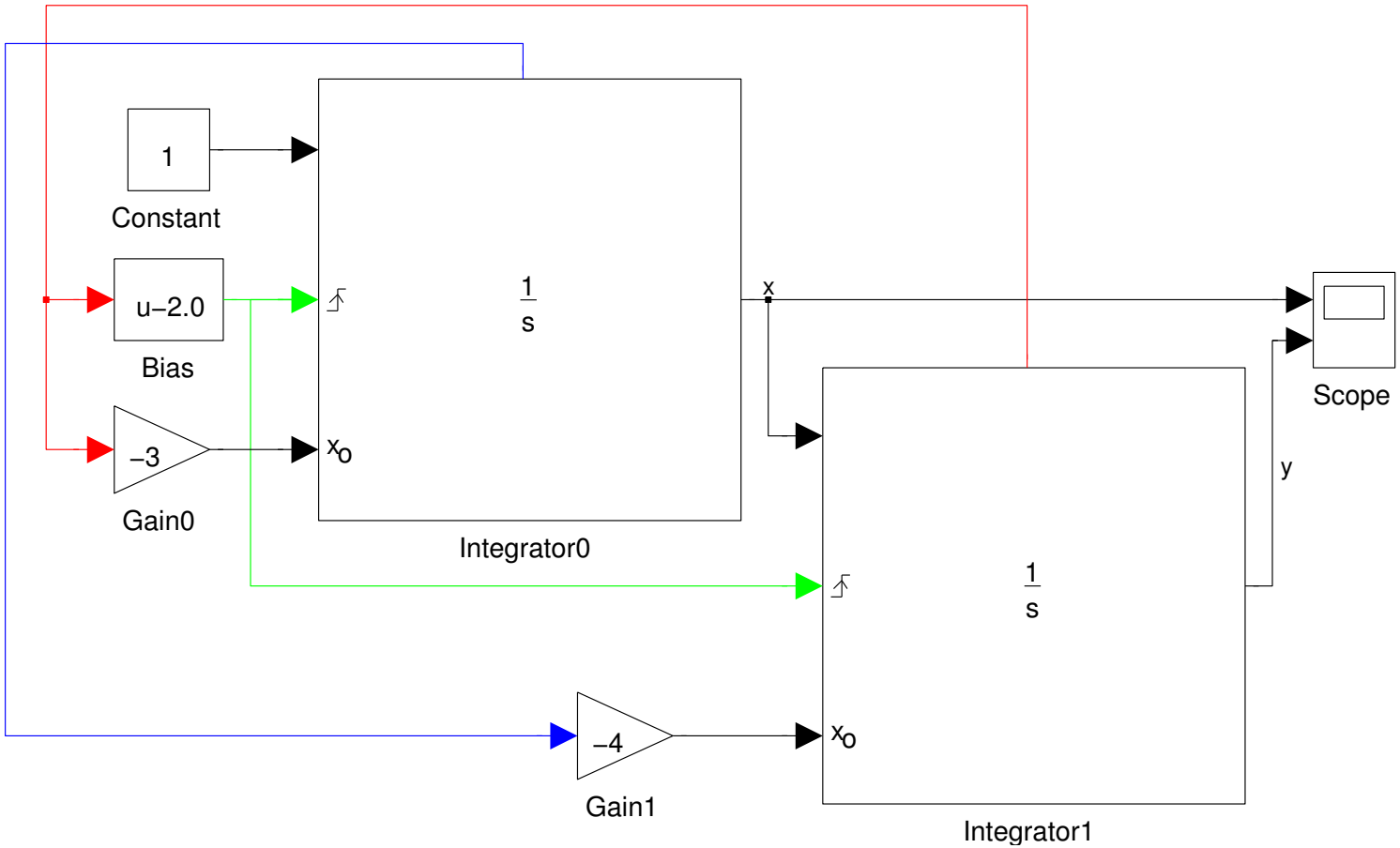


- Warning with 'Unit Delay' but not with 'Memory'.
- The shape of cpt depends on the steps chosen by the solver.
- Putting another component in parallel can change the result.
- Similar issues with Stateflow.





Causality issue: the Simulink state port



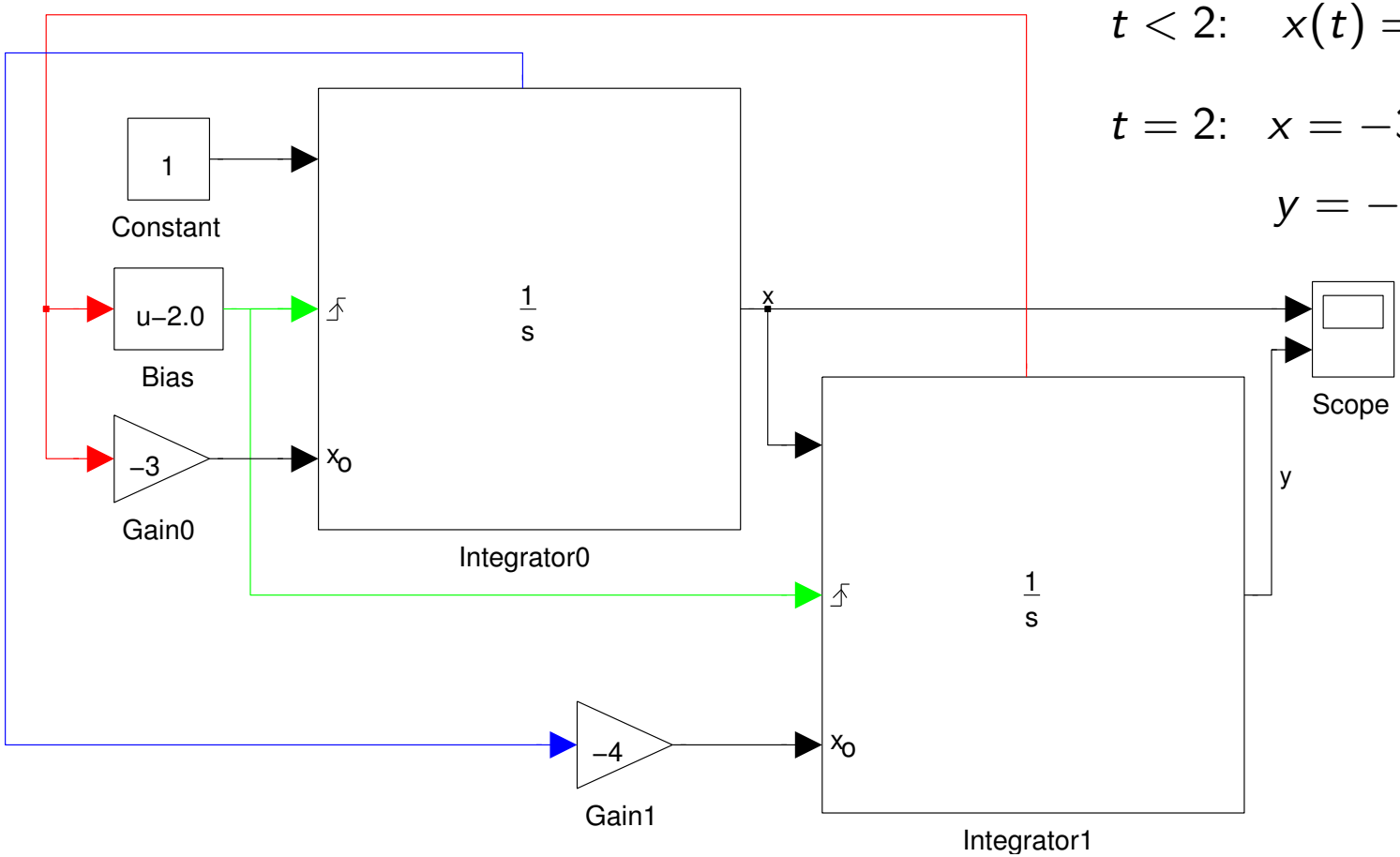
The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset.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

Causality issue: the Simulink state port

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

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

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



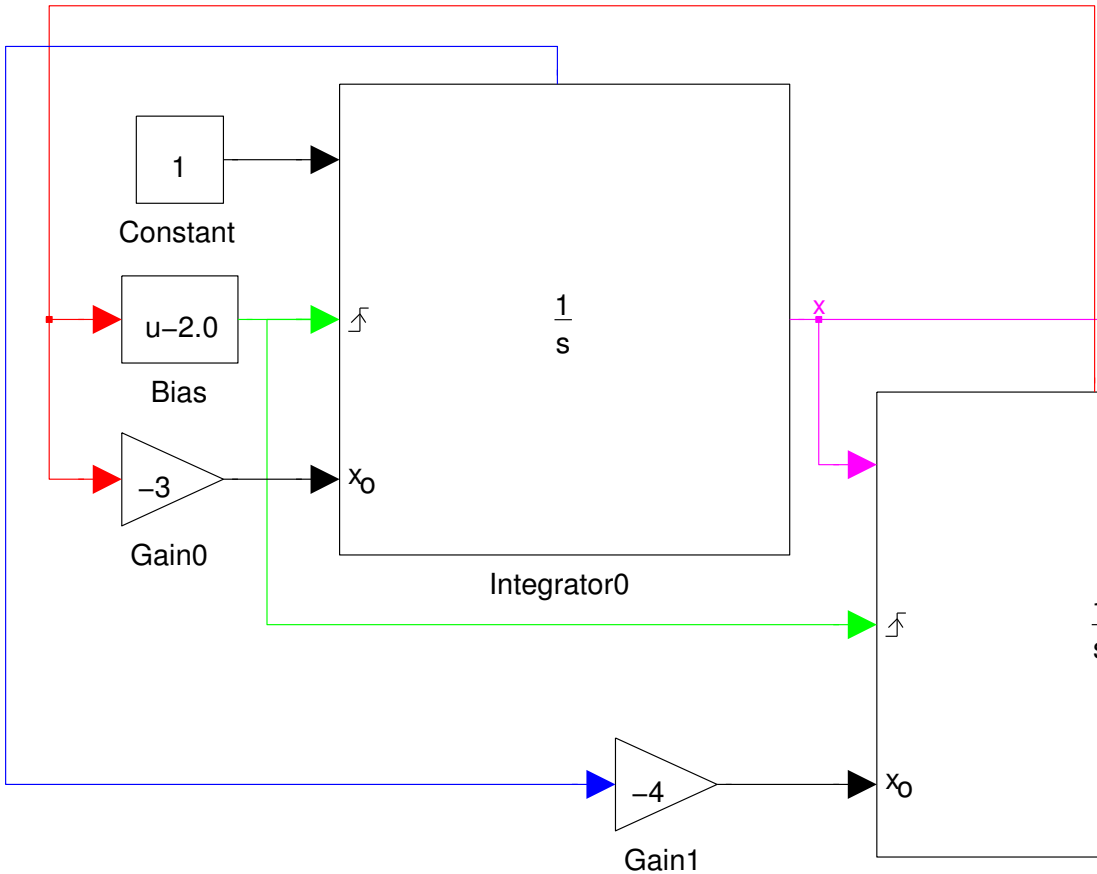
The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset.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

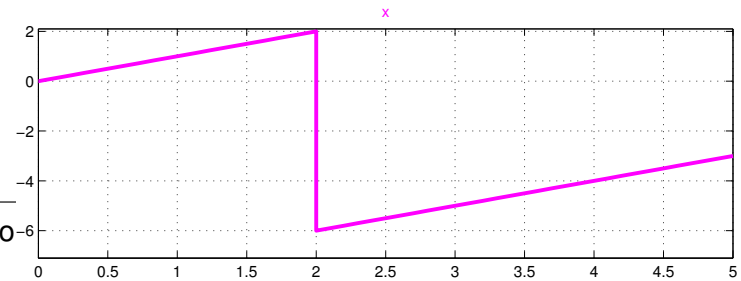
Causality issue: the Simulink state port

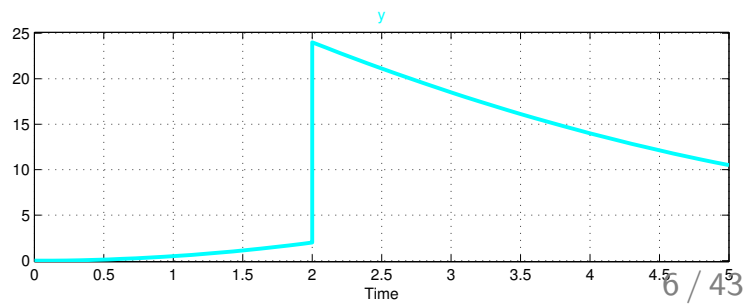
$$t < 2: \quad x(t) = t,\ y(t) = \frac{t^2}{2}$$

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

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





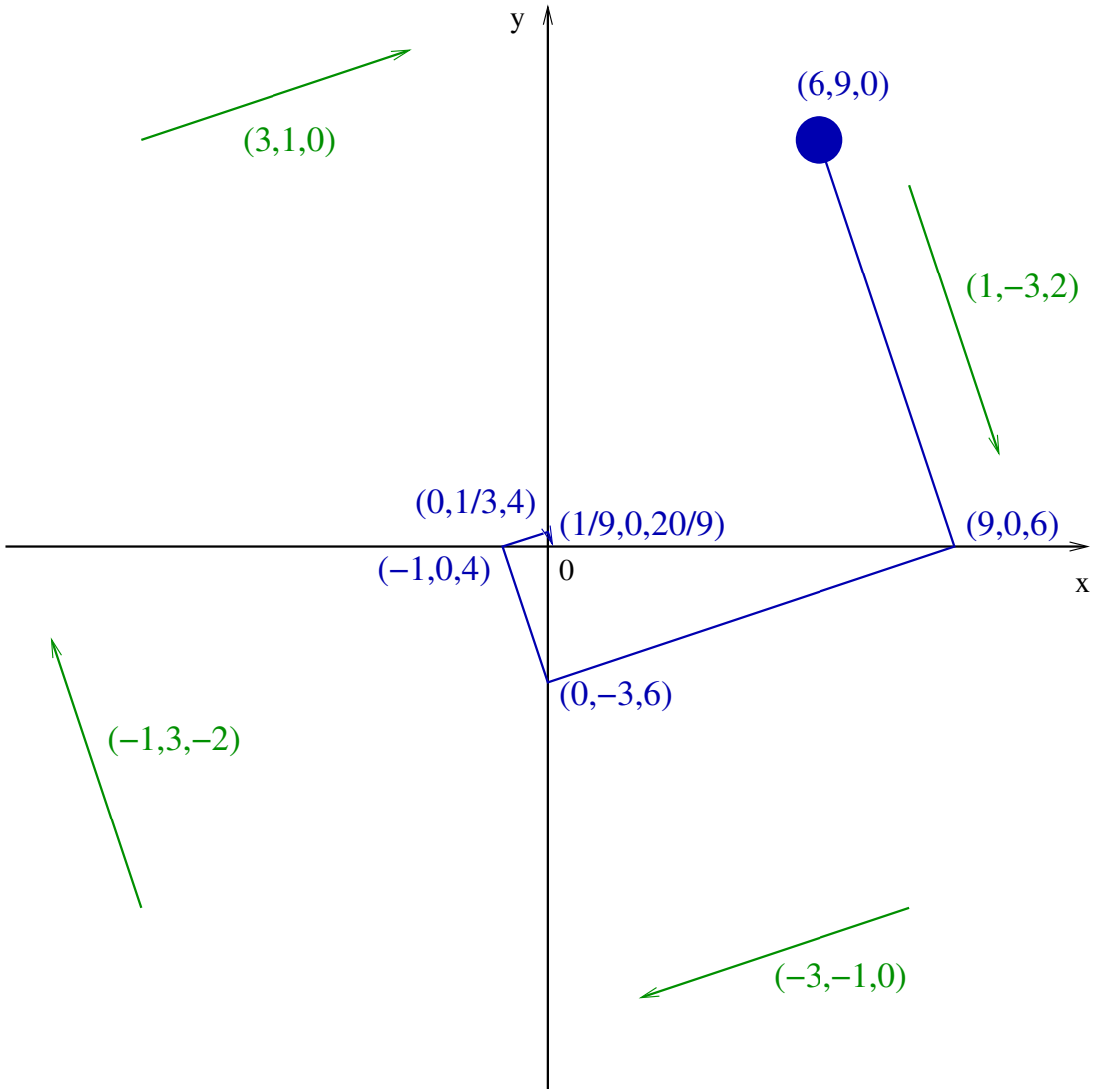


The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset.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

Modes glissants

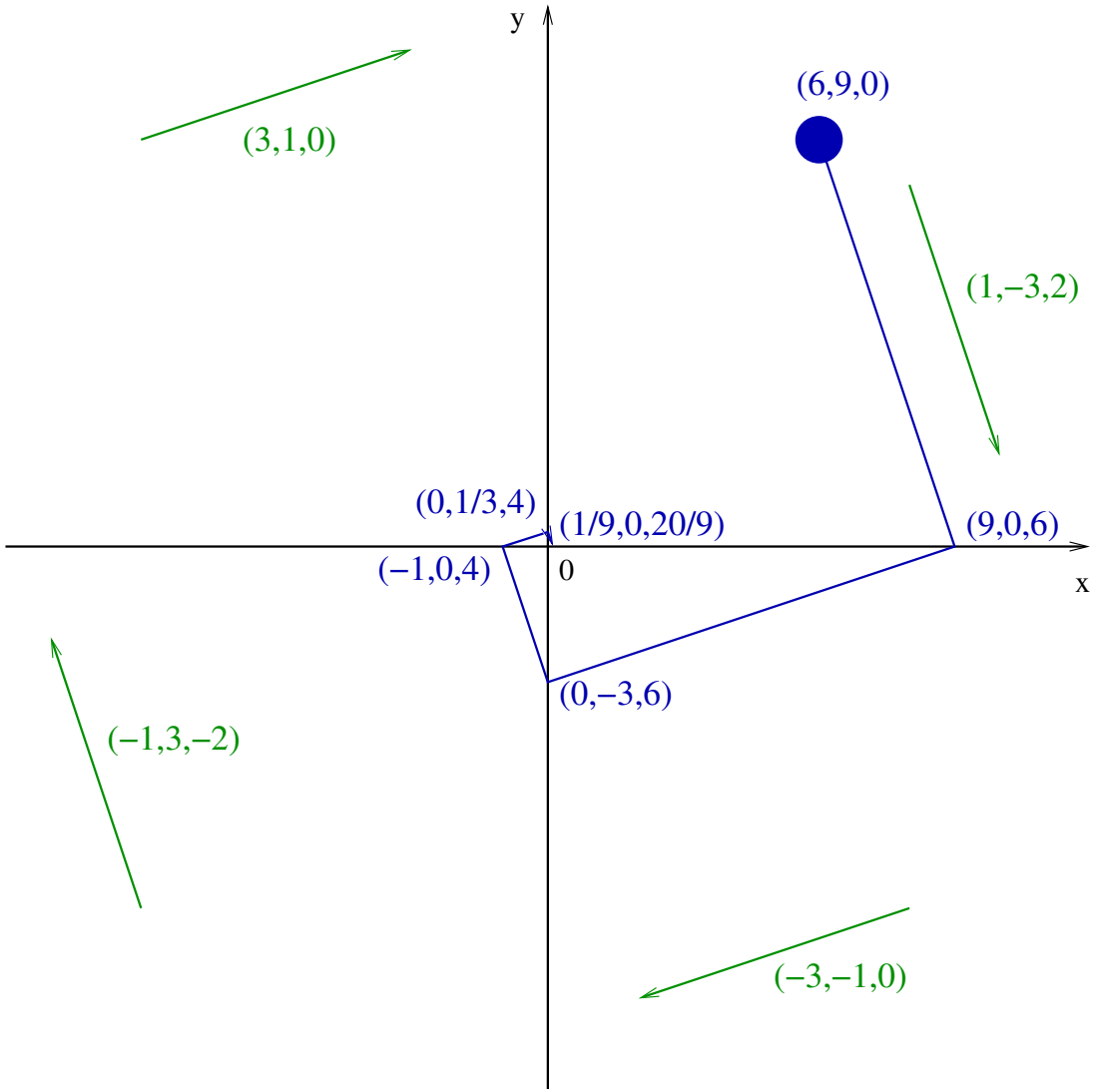
$$\begin{cases} \dot{x} &= -\text{sgn}(x) + 2\text{sgn}(y) \\ \dot{y} &= -2\text{sgn}(x) - \text{sgn}(y) \\ \dot{z} &= \text{sgn}(x) + \text{sgn}(y) \end{cases}$$

Modes glissants



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

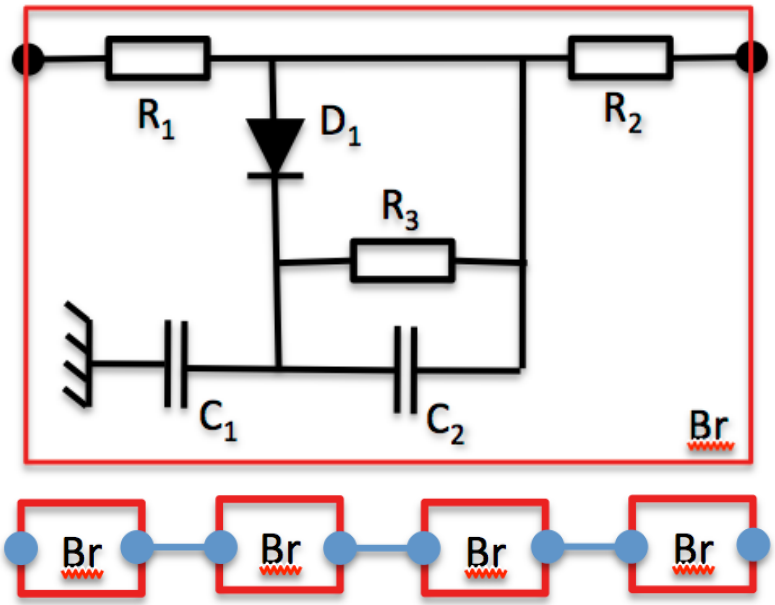
Modes glissants



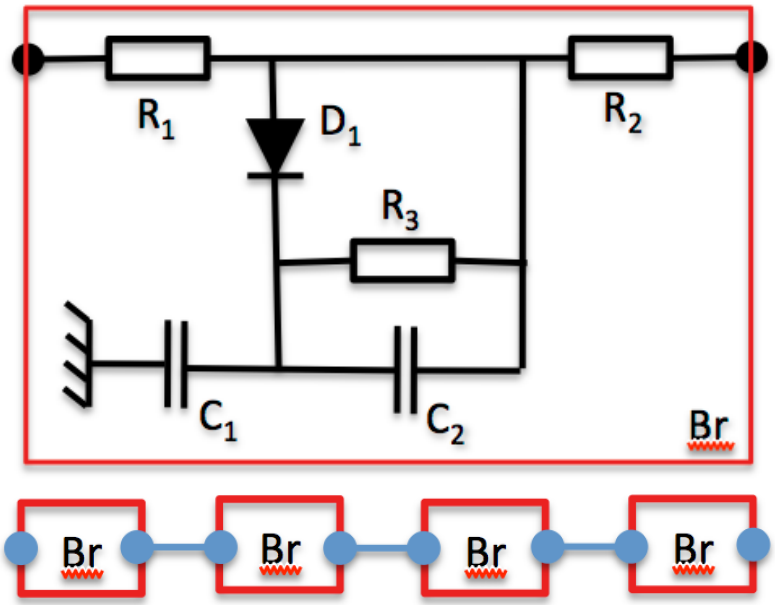
$$\begin{cases} \dot{x} &= -\mathrm{sgn}(x) + 2\mathrm{sgn}(y) \\ \dot{y} &= -2\mathrm{sgn}(x) - \mathrm{sgn}(y) \\ \dot{z} &= \mathrm{sgn}(x) + \mathrm{sgn}(y) \end{cases}$$

- Spirale “carrée”
- Attracteur : droite $x = y = 0$
- Zénon à $t = 7.5$
- Calculer z à $t = 10$?

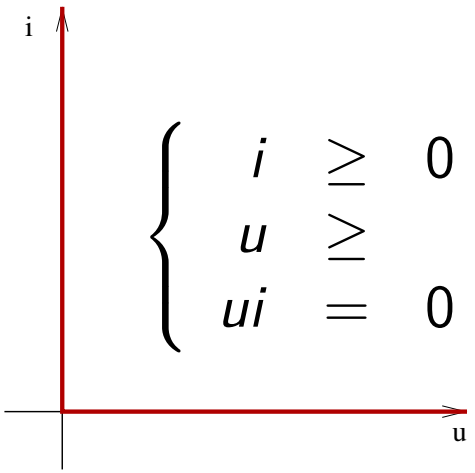
DAE + commutation de modes



DAE + commutation de modes

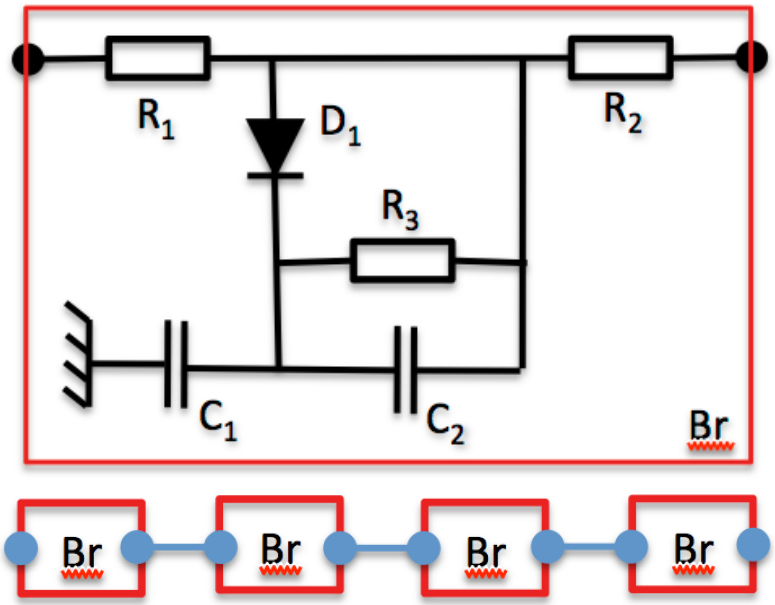


$$\begin{cases} N\mathbf{i} & = & \mathbf{0} \\ K\mathbf{u} & = & \mathbf{0} \\ \mathbf{u} - R\mathbf{i} & = & \mathbf{0} \\ C\dot{\mathbf{u}} - \mathbf{i} & = & \mathbf{0} \\ \forall i \quad 0 \leq \mathbf{i}_{D_i} & \perp & \mathbf{u}_{D_i} \geq 0 \end{cases}$$



$$\begin{cases} i & \geq & 0 \\ u & \geq & \\ ui & = & 0 \end{cases}$$

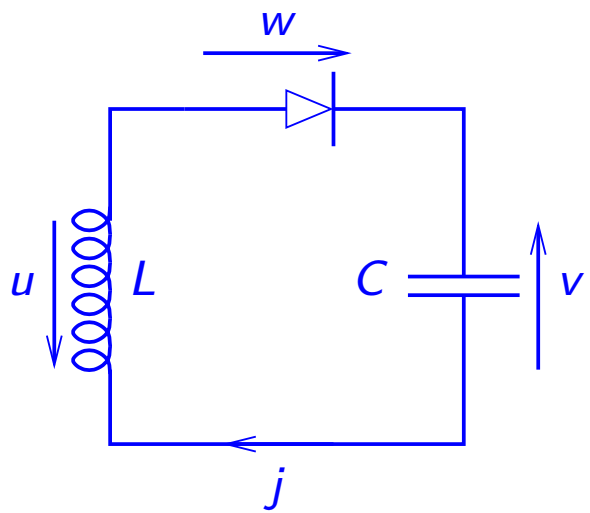
DAE + commutation de modes



$$\begin{cases} N\mathbf{i} = \mathbf{0} \\ K\mathbf{u} = \mathbf{0} \\ \mathbf{u} - R\mathbf{i} = \mathbf{0} \\ C\dot{\mathbf{u}} - \mathbf{i} = \mathbf{0} \\ \forall i \quad 0 \le \mathbf{i}_{D_i} \perp \mathbf{u}_{D_i} \ge 0 \end{cases}$$

- 2^n modes
- Ordonnancement dépend du mode
- Dymola : énumère les modes et explose
- Modularité $\Rightarrow$ DAE $\Rightarrow$ Modelica

Variation d'index : exemple LDC



$$\left[\begin{array}{lcl} 0 & = & j' - u/L \\ 0 & = & v' - j/C \\ 0 & = & u + v + w \\ 0 & \leq & j \\ 0 & \leq & w \\ 0 & = & jw \end{array}\right]$$

Variation d'index : exemple LDC

$$\underbrace{\begin{bmatrix} 0 &= j' - u/L \\ 0 &= v' - j/C \\ 0 &= u + v + w \\ 0 &\leq j \\ 0 &\leq w \\ 0 &= jw \end{bmatrix}}_{\text{Hybrid DAE system}}$$

$$\underbrace{\begin{bmatrix} 0 &= j' - u/L \\ 0 &= v' - j/C \\ 0 &= u + v + w \\ 0 &= w \end{bmatrix}}_{\text{mode } j \geq 0}$$

$$\Downarrow$$

$$\begin{bmatrix} 0 &= j' - u/L \\ 0 &= v' - j/C \\ 0 &= u + v \\ 0 &= w \end{bmatrix}$$

$$\Downarrow$$

$$\underbrace{\begin{bmatrix} j' &= -v/L \\ v' &= +j/C \\ u &= -v \\ w &= 0 \end{bmatrix}}_{\text{index 0}}$$

$$\underbrace{\begin{bmatrix} 0 &= j' - u/L \\ 0 &= v' \\ 0 &= u + v + w \\ 0 &= j \end{bmatrix}}_{\text{mode } w \geq 0}$$

$$\Downarrow$$

$$\begin{bmatrix} 0 &= j' - u/L \\ 0 &= v' \\ 0 &= u + v + w \\ 0 &= j \end{bmatrix}$$

$$\Downarrow$$

$$\underbrace{\begin{bmatrix} u &= Lj' \\ v' &= 0 \\ w &= -u - v \\ 0 &= j \\ j' &= 0 \end{bmatrix}}_{\text{index 1 (one differentiation)}}$$

Variation d'index : exemple LDC

$$\underbrace{\left[\begin{array}{lcl} 0 & = & j' - u/L \\ 0 & = & v' - j/C \\ 0 & = & u + v + w \\ 0 & \leq & j \\ 0 & \leq & w \\ 0 & = & jw \end{array}\right]}_{\text{Hybrid DAE system}}$$

$$\underbrace{\left[\begin{array}{lcl} 0 & = & j' - u/L \\ 0 & = & v' - j/C \\ 0 & = & u + v + w \\ 0 & = & w \end{array}\right]}_{\text{mode } j \geq 0}$$

$$\Downarrow$$

$$\underbrace{\left[\begin{array}{lcl} j' & = & -v/L \\ v' & = & +j/C \\ u & = & -v \\ w & = & 0 \end{array}\right]}_{\text{index 0}}$$

$$\underbrace{\left[\begin{array}{lcl} 0 & = & j' - u/L \\ 0 & = & v' - j/C \\ 0 & = & u + v + w \\ 0 & = & j \end{array}\right]}_{\text{mode } w \geq 0}$$

$$\Downarrow$$

$$\underbrace{\left[\begin{array}{lcl} u & = & Lj' \\ v' & = & 0 \\ w & = & -u - v \\ 0 & = & j \\ j' & = & 0 \end{array}\right]}_{\text{index 1 (one differentiation)}}$$

This hybrid DAE system has two modes with guards $[j \geq 0]$ and $[w \geq 0]$

- Indexes in each mode differ and causality analyses differ

$\Rightarrow$ Schedulings differ

Zélus

`zelus.di.ens.fr`

Reuse existing tools and techniques

Synchronous languages (SCADE/Lustre)

- Widely used for critical systems design and implementation
  - mathematically sound semantics
  - certified compilation (DO178C)
- Expressive language for both discrete controllers and mode changes
- Do not support modelling continuous dynamics!

Off-the-shelf ODEs numeric solvers

- Sundials CVODE (LLNL) among others, treated as black boxes
- Exploit existing techniques and (variable step) solvers

A conservative extension:

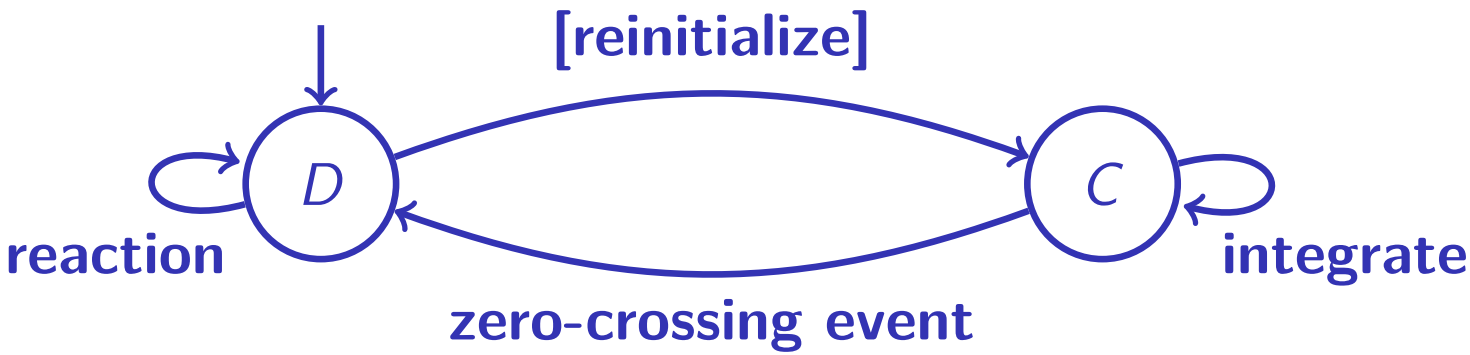
Any synchronous program must be compiled, optimized, and executed as per usual

Type systems to separate continuous from discrete

What is a discrete step?

- Reject unreasonable parallel compositions
- Ensure by static typing that discrete changes occur on zero-crossings
- Signals are continuous during integration
- Statically detect causality loops, initialization issues

Simulation engine



$$\sigma' = d_\sigma(t, y) \qquad upz = g_\sigma(t, y) \qquad \dot{y} = f_\sigma(t, y)$$

Combinatorial and sequential functions

Time is logical as in `Lustre`. A signal is a sequence of values and nothing is said about the actual time to go from one instant to another.

```
let add (x,y) = x + y

let node min_max (x, y) = if x < y then x, y else y, x

let node after (n, t) = (c = n) where
  rec c = 0 → pre(min(tick, n))
  and tick = if t then c + 1 else c
```

When feed into the compiler, we get:

```
val add : int × int  --A--> int
val mix_max : 'a× 'a --D--> 'a× 'a
val after : int × int  --D--> bool
```

x, y, etc. are infinite sequences of values.

Examples

A simple heat controller with ODEs. Using an explicit Euler dicretization, these programs could have been written *almost as is* in `Scade 6` or `Lucid Synchrone`.

```
(* an hysteresis controller for a heater *)
let hybrid heater(active) = temp where
  rec der temp = if active then c -. k *. temp else -. k *. temp init temp0

let hybrid hysteresis_controller(temp) = active where
  rec automaton
      | Idle →   do active = false until (up(t_min -. temp)) then Active
      | Active → do active = true until (up(temp -. t_max)) then Idle

let hybrid main() = temp where
  rec active = hysteresis_controller(temp)
  and temp = heater(active)
```

The Bouncing ball

```
let hybrid bouncing(x0,y0,x'0,y'0) = (x,y) where
  der(x) = x' init x0
and
  der(x') = 0.0 init x'0
and
  der(y) = y' init y0
and
  der(y') = −. g init y'0 reset up(−. y) → −. 0.9 *. last y'
```

Its type signature is:

$$\text{float} \times \text{float} \times \text{float} \times \text{float} \xrightarrow{c} \text{float} \times \text{float}$$

- When $-.$ y crosses zero, re-initialize the speed y' with $-. 0.9 * \text{last y'}$.
- last y' stands for the previous value of y'.
- As y' is immediately reset, writing last y' is mandatory —otherwise, y' would instantaneously depend on itself.

Zero-crossings and Valued Signals

- $\mathtt{up}(e)$ tests the zero-crossing of expression e from strictly negative to strictly positive.
- Performed by the solver during integration.
- If $x = \mathtt{up}(e)$, all handlers using x are governed by the same zero-crossing.
- Handlers have priorities.

```
let hybrid f(x, y) = (v, z1, z2) where
  rec v = present z1 → 1 | z2 → 2 init 0
  and z1 = up(x)
  and z2 = up(y)
```

$$\mathtt{val\ f} : \mathtt{float} \times \mathtt{float} \xrightarrow{\mathtt{c}} \mathtt{float} \times \mathtt{zero} \times \mathtt{zero}$$

Emit a value only when a zero-crossing is detected

```
let hybrid f(x, y) = o where
  rec o = present up(x) → 42 | up(y) → 43

val f: float -C-> int signal
```

o is only present when either $up(x)$ or $up(y)$ and it carries an integer value.

Do pattern matching on valued signals

```
let hybrid default(x, y, x0) = o where
  rec o = present x(v) → v | y(w) → w init x0

val f: int signal -C-> int
```

returns the current value of x (when present), y (when present) and holds the previous value otherwise.

Two difficulties

Ensure that continuous and discrete time signals interfere correctly?

- Discrete time should stay logical and independent of the solver's step size.
- Otherwise, we get the same monsters as `Simulink`/`Stateflow` have.
- Provide a type system for that.

Ensure that fix-point exist and code can be scheduled?

- Algebraic loops must be statically detected.
- Usefull for code generation purposes.
- Introduce the operator last(x) as the "left limit" of a signal.
- An a type-system to ensure that programs can be statically scheduled.

Mixing discrete (logical) time and continuous time

Given:

```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Mixing discrete (logical) time and continuous time

Given:

```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Define:

```
let          wrong () = ()
  where rec
    der time = 1.0 init 0.0
    and y =  sum (time)
```

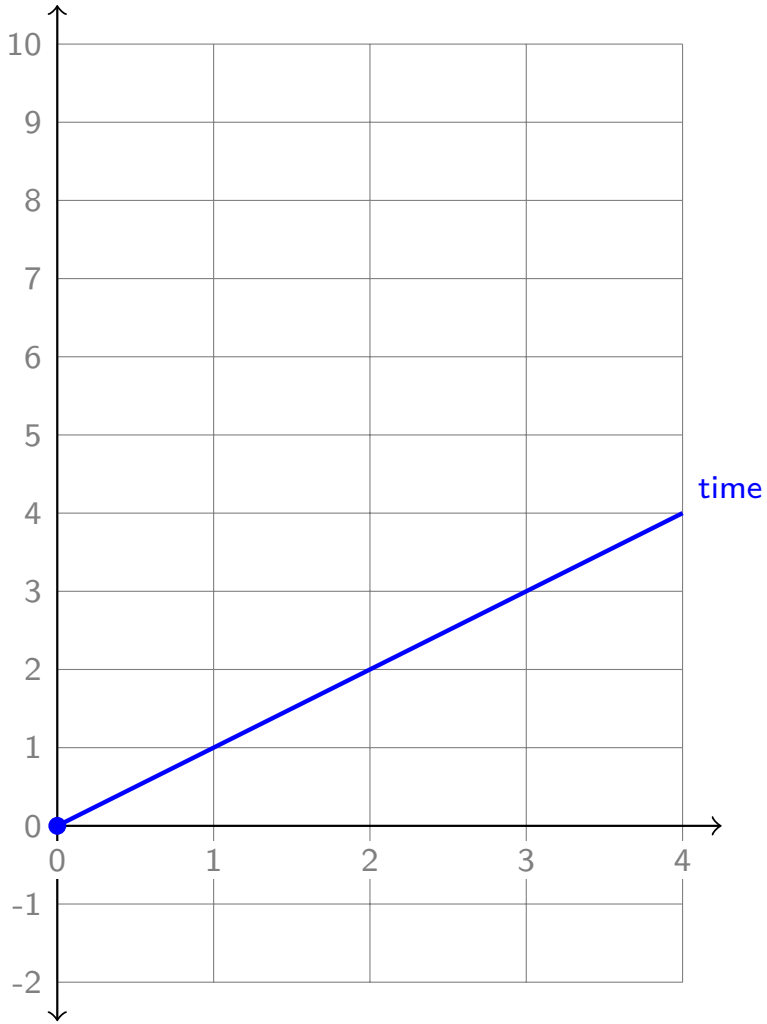
Mixing discrete (logical) time and continuous time

Given:

```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Define:

```
let          wrong () = ()
  where rec
    der time = 1.0 init 0.0
    and y =  sum (time)
```



Interpretation:

- Option 1: $\mathbb{N} \subseteq \mathbb{R}$
- Option 2: depends on solver
- Option 3: infinitesimal steps
- Option 4: type and reject

Mixing discrete (logical) time and continuous time

Given:

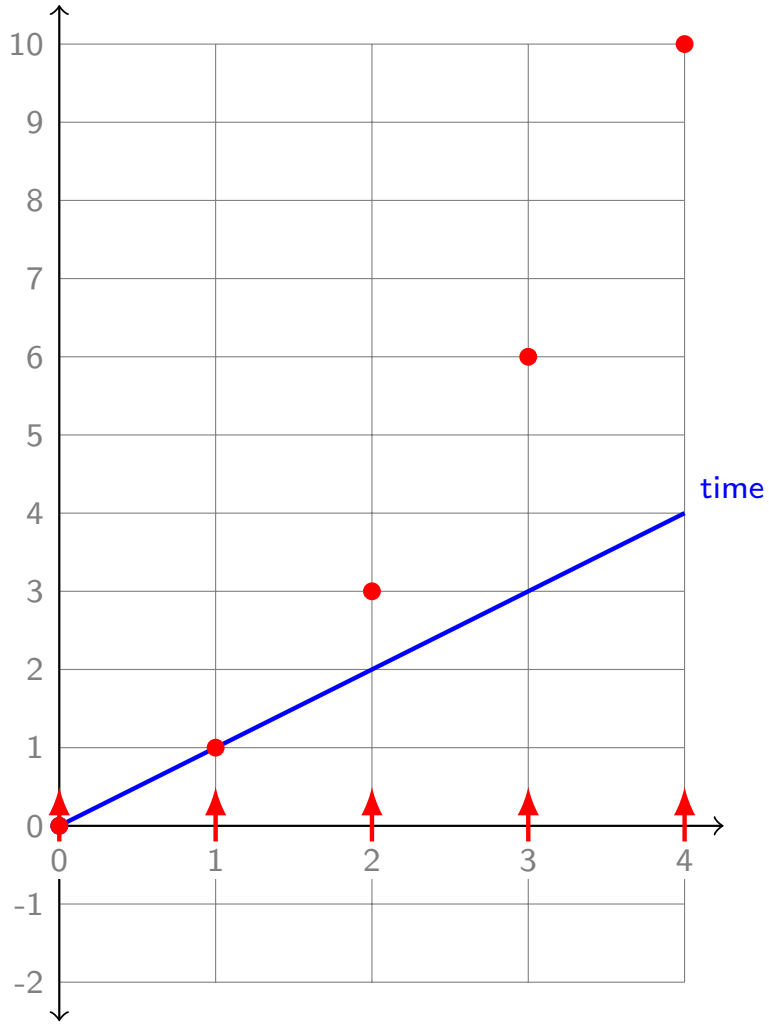
```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Define:

```
let          wrong () = ()
  where rec
    der time = 1.0 init 0.0
    and y =  sum (time)
```

Interpretation:

- Option 1: $\mathbb{N} \subseteq \mathbb{R}$
- Option 2: depends on solver
- Option 3: infinitesimal steps
- Option 4: type and reject



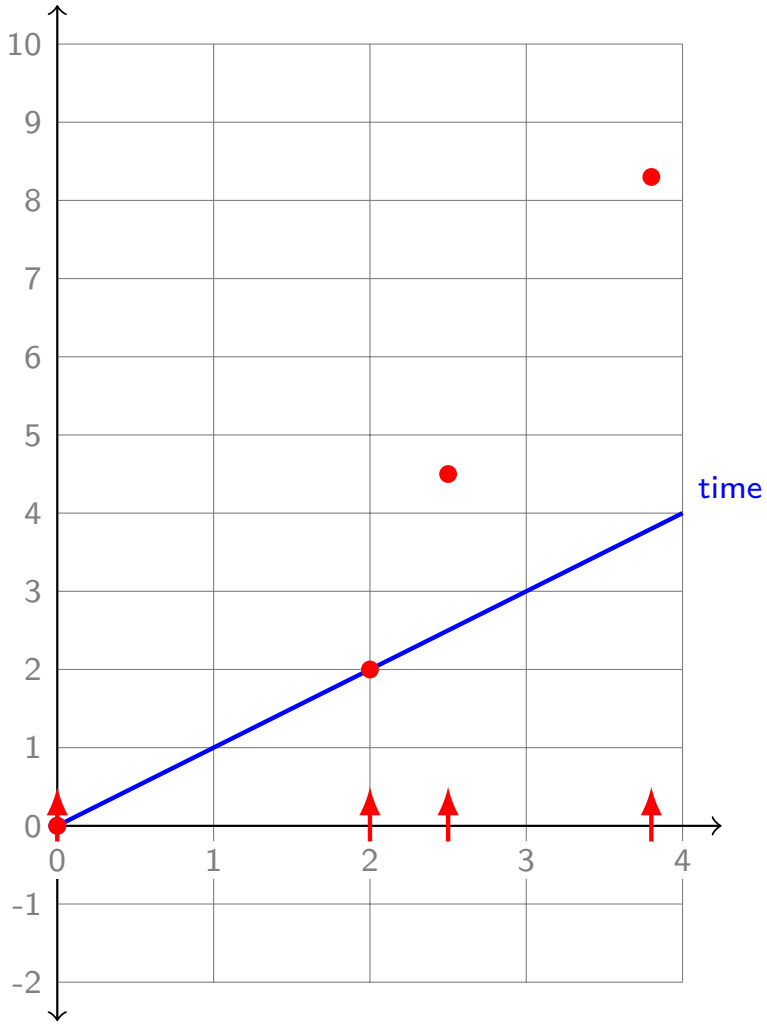
Mixing discrete (logical) time and continuous time

Given:

```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Define:

```
let          wrong () = ()
  where rec
    der time = 1.0 init 0.0
    and y =  sum (time)
```



Interpretation:

- Option 1: $\mathbb{N} \subseteq \mathbb{R}$
- Option 2: depends on solver
- Option 3: infinitesimal steps
- Option 4: type and reject

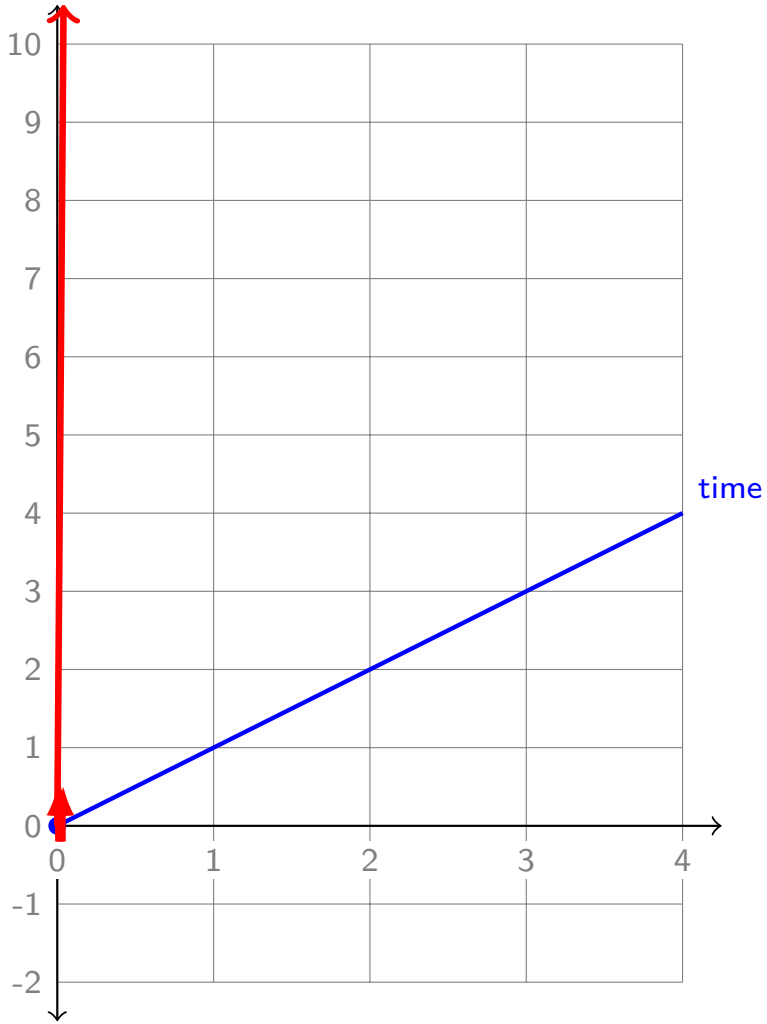
Mixing discrete (logical) time and continuous time

Given:

```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Define:

```
let          wrong () = ()
  where rec
    der time = 1.0 init 0.0
    and y =  sum (time)
```



Interpretation:

- Option 1: $\mathbb{N} \subseteq \mathbb{R}$
- Option 2: depends on solver
- Option 3: infinitesimal steps
- Option 4: type and reject

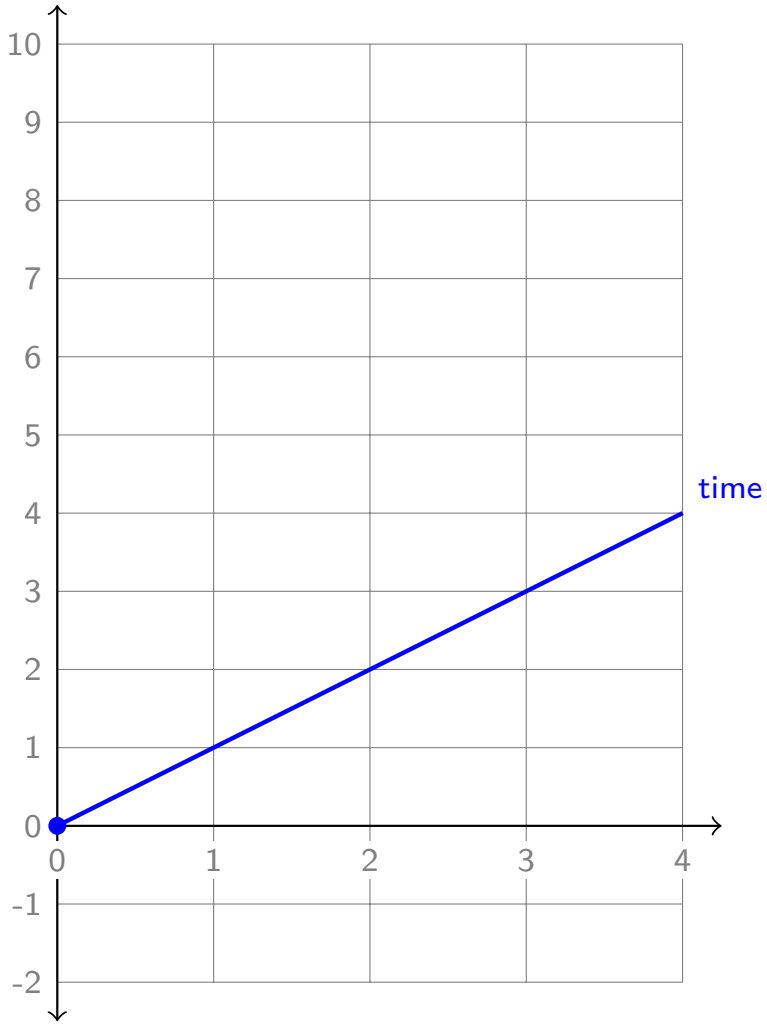
Mixing discrete (logical) time and continuous time

Given:

```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Define:

```
let          wrong () = ()
  where rec
    der time = 1.0 init 0.0
    and y = sum (time)
```



Interpretation:

- Option 1: $\mathbb{N} \subseteq \mathbb{R}$
- Option 2: depends on solver
- Option 3: infinitesimal steps
- Option 4: type and reject

Mixing discrete (logical) time and continuous time

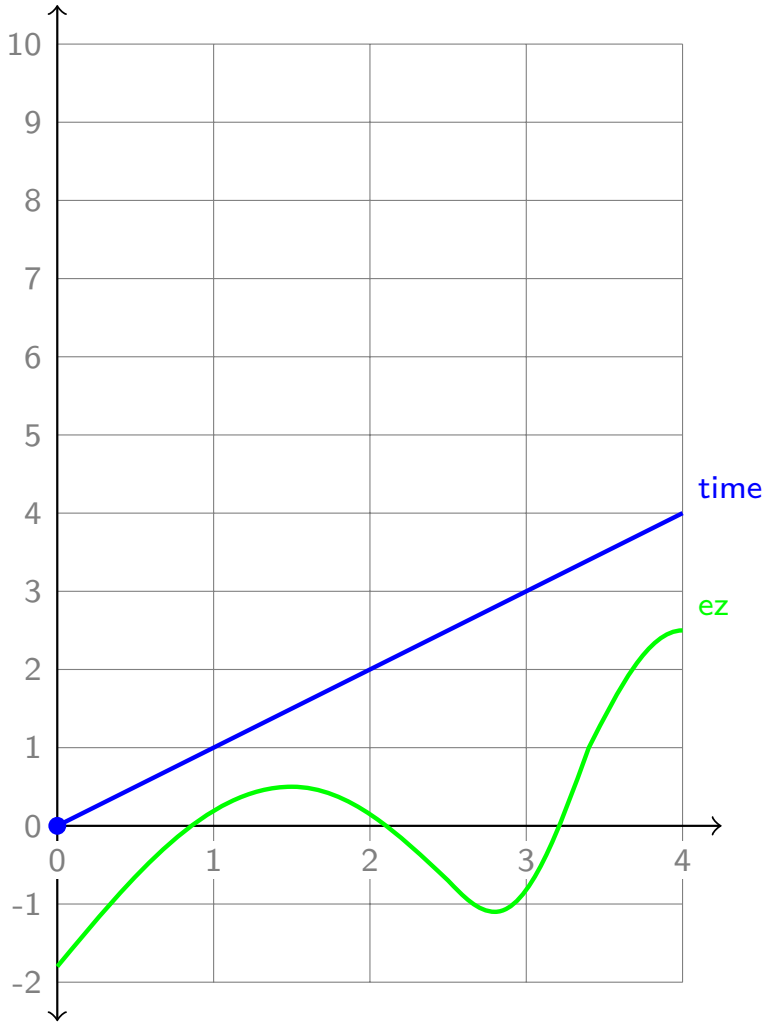
Given:

```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Define:

```
let hybrid correct () = ()
  where rec
    der time = 1.0 init 0.0
    and y = present up(ez) → sum (time)
                init 0.0
```

- node:
  function acting in discrete time
- hybrid:
  function acting in continuous time



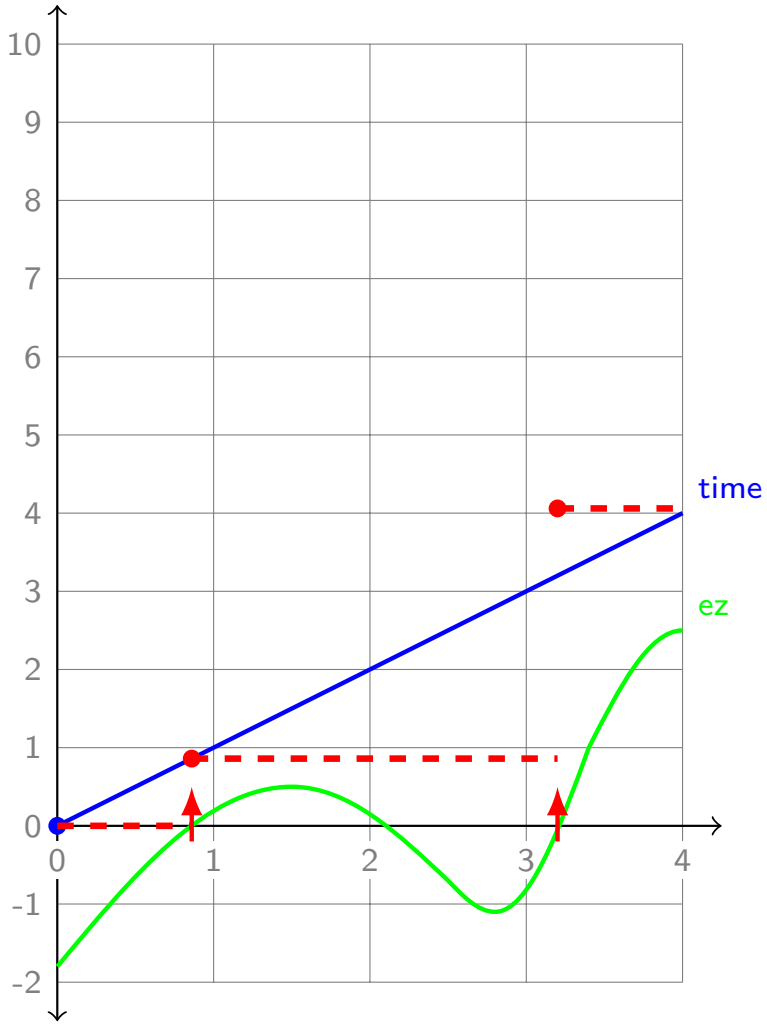
Mixing discrete (logical) time and continuous time

Given:

```
let node sum(x) = cpt where
  rec cpt = (0.0 fby cpt) +. x
```

Define:

```
let hybrid correct () = ()
  where rec
    der time = 1.0 init 0.0
    and y = present up(ez) → sum (time)
                init 0.0
```



- node:
  function acting in discrete time
- hybrid:
  function acting in continuous time

Explicitly relate simulation and logical time (using zero-crossings)

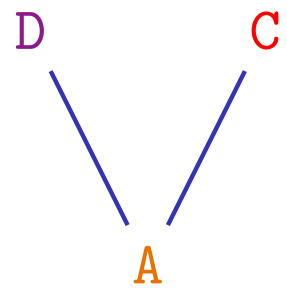
Try to minimize the effects of solver parameters and choices

Basic typing [LCTES'11]

A simple ML type system with effects.

The type language

$$
\begin{array}{lcl}
bt & ::= & \texttt{float} \mid \texttt{int} \mid \texttt{bool} \mid \texttt{zero} \\
t & ::= & bt \mid t \times t \mid \beta \\
\sigma & ::= & \forall \beta_1, ..., \beta_n . t \xrightarrow{k} t \\
k & ::= & \texttt{D} \mid \texttt{C} \mid \texttt{A}
\end{array}
$$



Initial conditions

$$
\begin{array}{lcl}
(+) & : & \texttt{int} \times \texttt{int} \xrightarrow{\texttt{A}} \texttt{int} \\
\texttt{if} & : & \forall \beta . \texttt{bool} \times \beta \times \beta \xrightarrow{\texttt{A}} \beta \\
(=) & : & \forall \beta . \beta \times \beta \xrightarrow{\texttt{D}} \texttt{bool} \\
\texttt{pre}(\cdot) & : & \forall \beta . \beta \xrightarrow{\texttt{D}} \beta \\
\cdot \texttt{fby} \cdot & : & \forall \beta . \beta \times \beta \xrightarrow{\texttt{D}} \beta \\
\texttt{up}(\cdot) & : & \texttt{float} \xrightarrow{\texttt{C}} \texttt{zero} \\
\cdot \texttt{on} \cdot & : & \texttt{zero} \times \texttt{bool} \xrightarrow{\texttt{A}} \texttt{zero}
\end{array}
$$

Causality issues (feedback loops)

Which programs should we accept?

- OK to reject (no solution).

  rec x = x +. 1.0

- OK as an algebraic constraint (e.g., Simulink and Modelica).

  rec x = 1.0 −. x

- But NOK if sequential code generation is targeted (algebraic loop).
- OK in constructive logic (Esterel)

  rec z1 = if c then z2 else y
  and z2 = if c then x else z1

- But it calls for an expensive boolean analysis.

Can we find a simple and uniform justification for a program mixing continuous-time and discrete-time signals to be causaly correct?

A Non-standard Semantics for Hybrid Modelers

We proposed in [CDC 2010, JCSS 2012] to build the semantics on Non-standard analysis.

der y = z init 4.0 and z = 10.0 − 0.1 ∗ y and k = y + 1.0

defines signals y, z and k, where for all $t \in \mathbb{R}^+$:

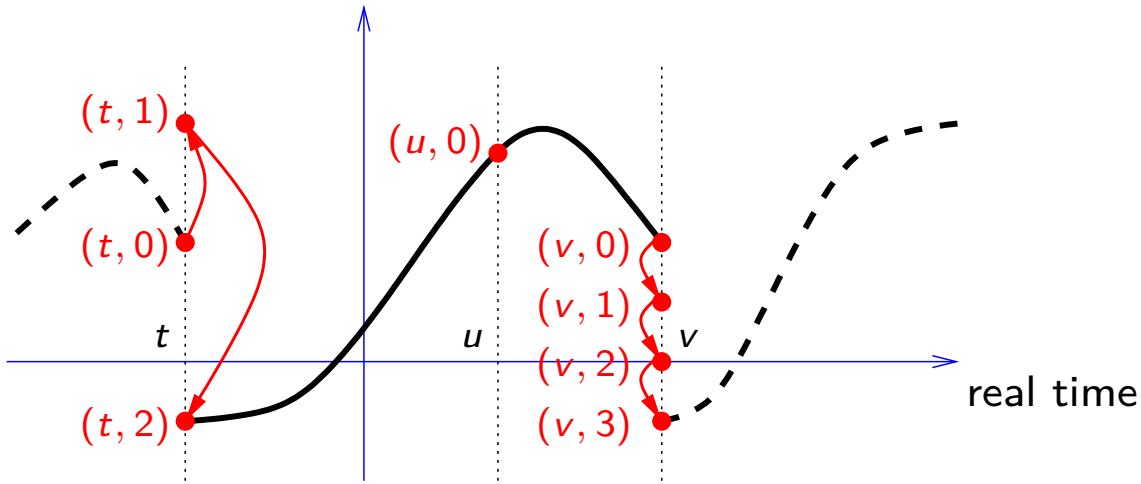
$$\frac{dy}{dt}(t) = z(t) \quad y(0) = 4.0 \quad z(t) = 10.0 - 0.1 \cdot y(t) \quad k(t) = y(t) + 1$$

Consider the value that y would have if computed by an ideal solver taking an infinitesimal step of duration ∂.
${}^\star y(n)$ stands for the values of y at instant $n\partial$, with $n \in {}^\star\mathbb{N}$ a non-standard integer.

$$\begin{aligned} {}^\star y(0) &= 4 & {}^\star z(n) &= 10 - 0.1 \cdot {}^\star y(n) \\ {}^\star y(n+1) &= {}^\star y(n) + {}^\star z(n) \cdot \partial & {}^\star k(n) &= {}^\star y(n) + 1 \end{aligned}$$

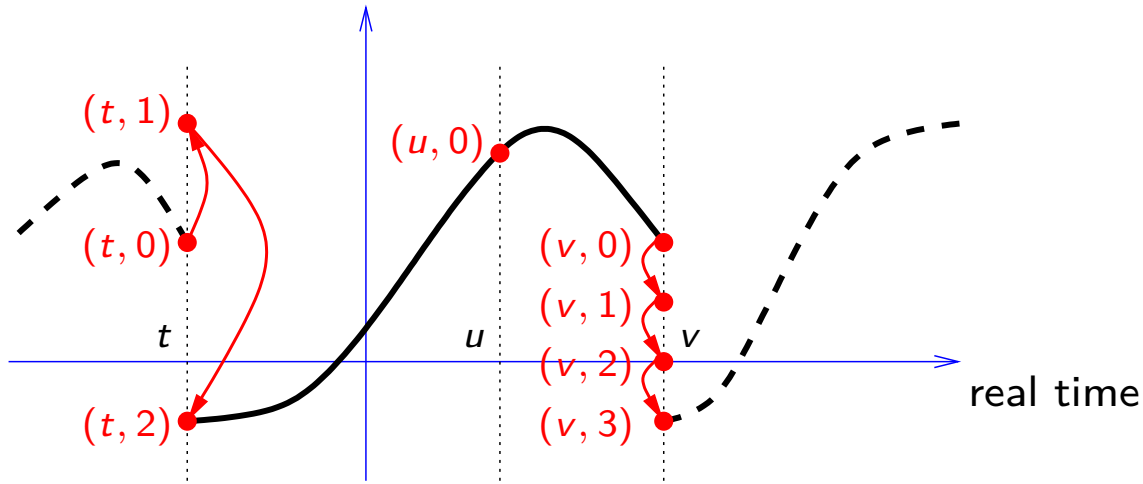
Non-standard time vs. Super-dense time

- Ed Lee & al. super-dense time modeling $\mathbb{R} \times \mathbb{N}$

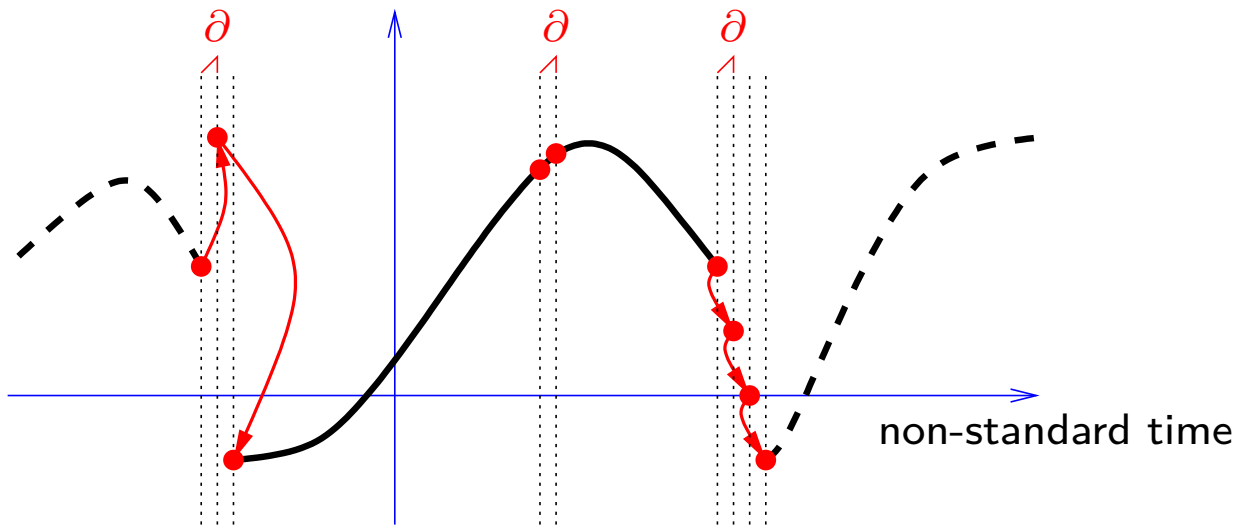


Non-standard time vs. Super-dense time

- Ed Lee & al. super-dense time modeling $\mathbb{R} \times \mathbb{N}$



- Benveniste & al. non-standard time modeling $\mathbb{T}_\partial = \{n\partial \mid n \in {}^\star\mathbb{N}\}$



ODEs with reset

Consider the sawtooth signal $y : \mathbb{R}^+ \mapsto \mathbb{R}^+$ such that:

$$\frac{dy}{dt}(t) = 1 \quad y(t) = 0 \text{ if } t \in \mathbb{N}$$

written (in Zélus):

der y = 1.0 init 0.0 reset up(y − 1.0) → 0.0

The ideal non-standard semantics is:

$$\begin{aligned}
{}^\star y(0) &= 0 & {}^\star y(n) &= \texttt{if}\ {}^\star z(n)\ \texttt{then}\ 0.0\ \texttt{else}\ {}^\star ly(n) \\
{}^\star ly(n) &= {}^\star y(n-1) + \partial & {}^\star c(n) &= ({}^\star y(n) - 1) \geq 0 \\
{}^\star z(0) &= \texttt{false} & {}^\star z(n) &= {}^\star c(n) \wedge \neg {}^\star c(n-1)
\end{aligned}$$

This set of equation is not causal: ${}^\star y(n)$ depends on itself.

Accessing the "left limit" of a signal

There are two ways to break this cycle:

- consider that the effect of the zero-crossing is delayed by one cycle, that is, the test is made on ${}^\star z(n-1)$ instead of on $z(n)$, or,
- distinguish the current value of ${}^\star y(n)$ from the value it would have had were there no reset, namely ${}^\star ly(n)$.

Testing a zero-crossing of ly (instead of y),

$$ {}^\star c(n) = ({}^\star ly(n) - 1) \geq 0, $$

gives a program that is causal since ${}^\star y(n)$ no longer depends instantaneously on itself.

```
der y = 1.0 init 0.0 reset up(last y − 1.0) → 0.0
```

Non standard semantics [JCSS'12]

Let ${}^\star\mathbb{R}$ and ${}^\star\mathbb{N}$ be the non-standard extensions of $\mathbb{R}$ and $\mathbb{N}$.
Let the global time base or *base clock* be the infinite set of instants:

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

$\mathbb{T}_\partial$ inherits its total order from ${}^\star\mathbb{N}$.
A signal is a partial function from $\mathbb{T}$ to a set of values.
The classical denotational semantics of synchronous languages can be replayed in this setting.

The operator last

- In non standard semantics, last x is the previous value of x
- When x is left continuous, last x is the left-limit.

A type-based causality analysis [submission to HSCC 2014]

Associate a type that express input/output dependences. E.g.,

```
let node plus(x, y) = x + 0 → pre y
```

We get: $f : \forall \alpha_1, \alpha_2 . \alpha_1 \times \alpha_2 \to \alpha_1$

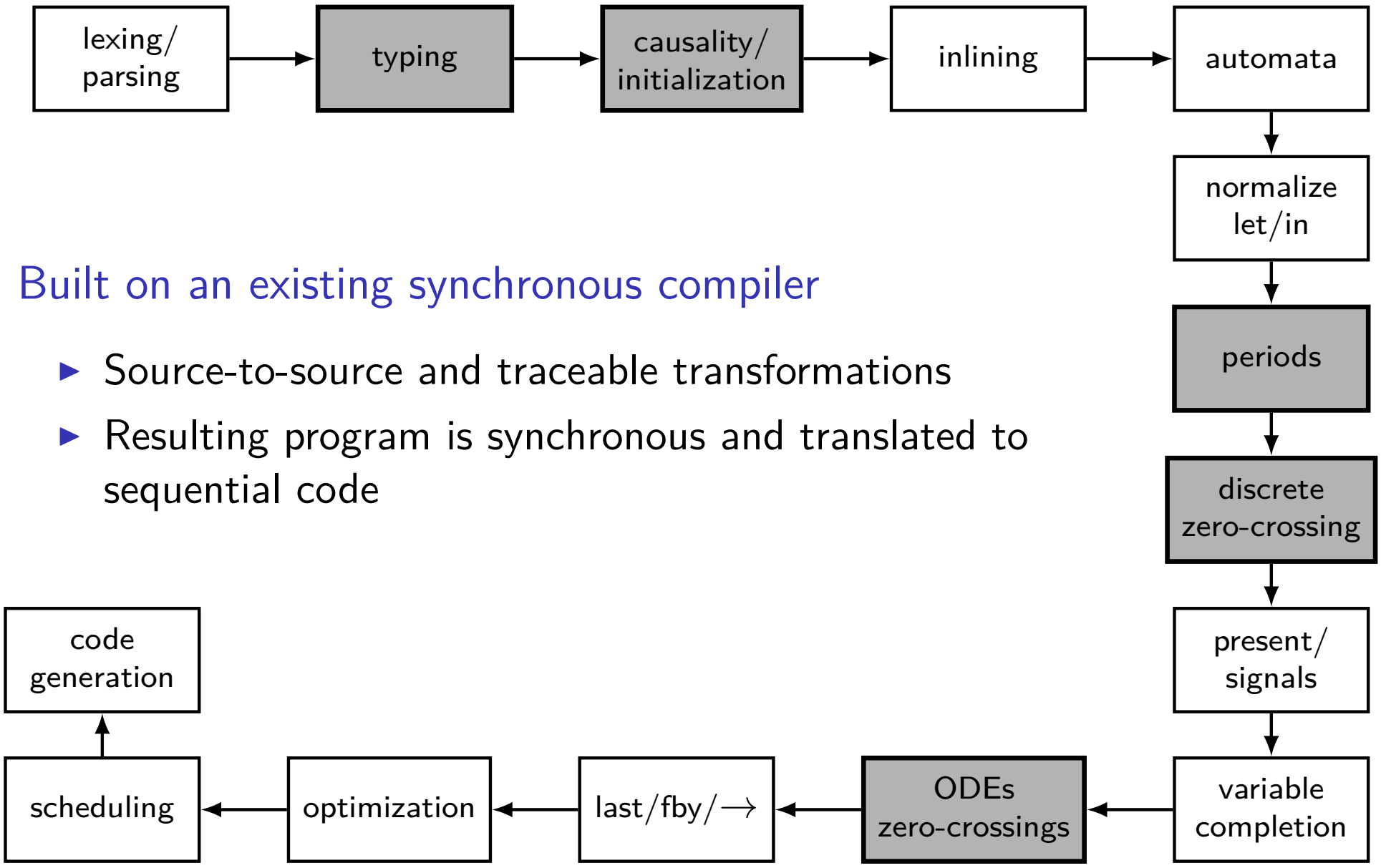
- pre(x) is a, discrete-time only, unit delay.
- der x breaks a loop: der temp = c −. temp init 20.0 is correct.
- last(x) is the left limit of a signal:
  - when x is a continuous-state variable (der x = ...), this is the `Simulink` state port.
  - writting last x in a discrete context always make sense.

The following is rejected; the next is accepted.

```
rec der y' = −. g init 0.0 reset up(−.y) → −0.9 ∗. y'
and der y = y' init y0
```

```
rec der y' = −. g init 0.0 reset up(−.y) → −0.9 ∗. last y'
and der y = y' init y0
```

Compiler architecture



Built on an existing synchronous compiler

- Source-to-source and traceable transformations
- Resulting program is synchronous and translated to sequential code

Comparison with existing tools

Simulink/Stateflow (Mathworks)

- Integrated treatment of automata *vs* two distinct languages
- More rigid separation of discrete and continuous behaviors

Modelica

- Do not handle DAEs
- Our proposal for automata has been integrated into version 3.3

Ptolemy (E.A. Lee et al., Berkeley)

- A unique computational model: synchronous
- Everything is compiled to sequential code (not interpreted)

Current work

Implementation/Optimization

- The current type system is very limited: if x and y are integers, $x = y$ is rejected in a hybrid node.
- Share states and zero-crossings, as much as possible. Simulink doesn't optimize this.

DAEs

- Only ODEs for the moment.
- DAEs raise several issues: no comprehensive theory for DAE Hybrid Systems and index reduction with multiple modes
- No clue for the moment of how to treat modes in a scalable manner.
- A promising approach: integrate NonSmooth Dynamical Systems techniques [Acary, Brogliato, ...] with complementary constraints and Filippov differential inclusions.

Systèmes dynamiques non-réguliers

Système dynamique :

$$\dot{q} = A.q + r$$

Perturbation non-régulière r, solution d'un (par exemple) problème de complémentarité linéaire (LCP) :

$$\begin{cases} r & = & B.x \\ y & = & N.q + M.x \\ 0 \leq x & \perp & y \geq 0 \end{cases}$$

Bibliothèque numérique Siconos [Acary]

Systèmes dynamiques non-réguliers

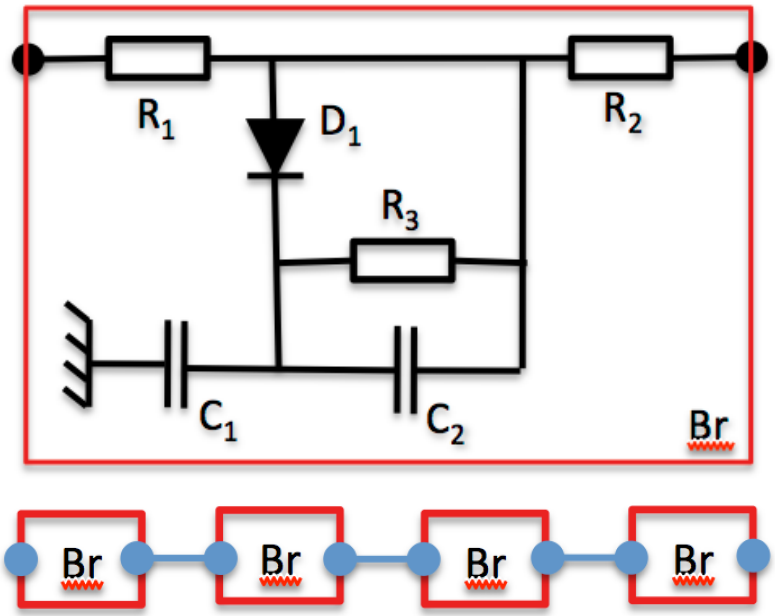
Système dynamique :

$$\dot{q} = A.q + r$$

Perturbation non-régulière r, solution d'un (par exemple) problème de complémentarité linéaire (LCP) :

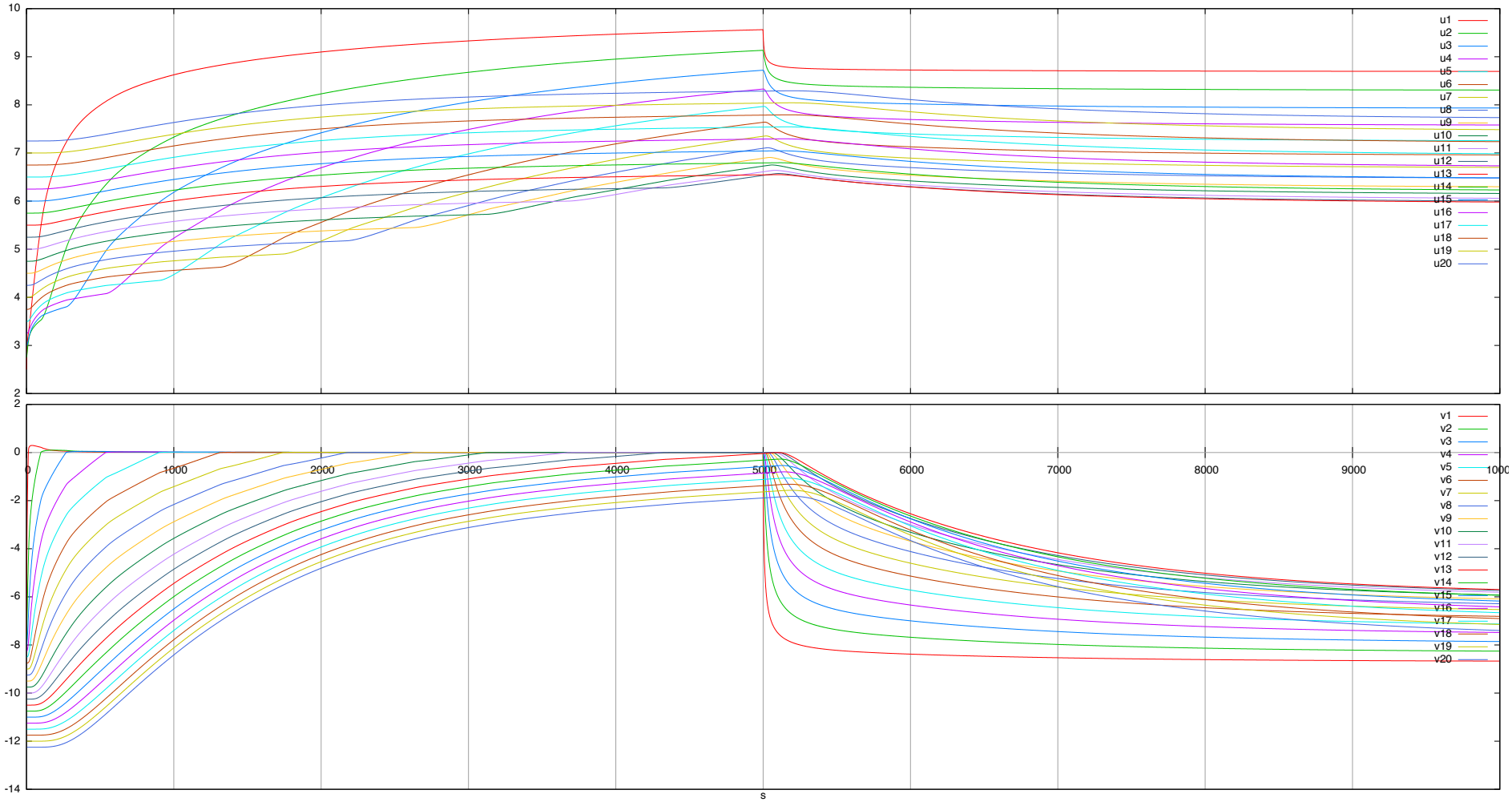
$$\begin{cases} r & = & B.x \\ y & = & N.q + M.x \\ 0 \leq x & \perp & y \geq 0 \end{cases}$$

Bibliothèque numérique Siconos [Acary]



$$\begin{cases} N\mathbf{i} & = & \mathbf{0} \\ K\mathbf{u} & = & \mathbf{0} \\ \mathbf{u} - R\mathbf{i} & = & \mathbf{0} \\ C\dot{\mathbf{u}} - \mathbf{i} & = & \mathbf{0} \\ \forall i \quad 0 \leq \mathbf{i}_{D_i} & \perp & \mathbf{u}_{D_i} \geq 0 \end{cases}$$

Systèmes dynamiques non-réguliers



Systèmes dynamiques non-réguliers

Des techniques similaires s'appliquent aux modes glissants [Acary 2010] :

Inclusion différentielle de Filippov :

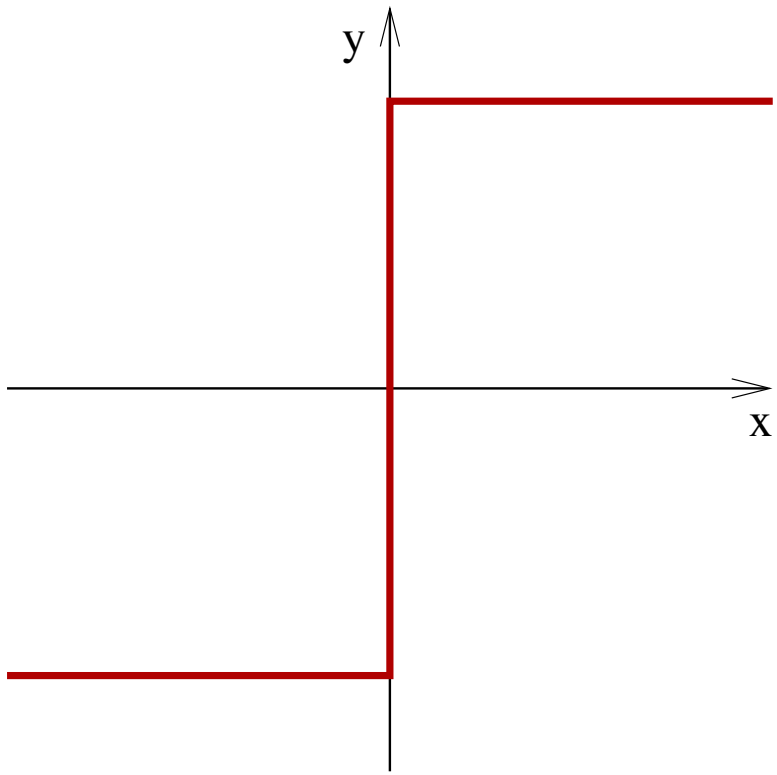
$$\dot{x} = A.x + B.\sigma$$

Avec :

$$\sigma \in \text{sgn}(C.x)$$

Réduction à un Mixed-LCP, permettant de moyenner σ sur un pas d'intégration :

$$\begin{cases} M.z + q = w - v \\ -1 \leq z \leq 1 \\ (1+z).w = 0 \\ (1-z).v = 0 \\ w, v \geq 0 \end{cases}$$



$$\sigma \in \text{sgn}(y)$$
$$\Longleftrightarrow$$
$$y \in N_{[-1,1]}(\sigma) =$$
$$\{y | \forall \alpha \in [-1,1], y(\sigma - \alpha) \geq 0\}$$

V. Acary and B. Brogliato.

Numerical methods for nonsmooth dynamical systems.
Springer, 2008.

Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet.

A Hybrid Synchronous Language with Hierarchical Automata: Static Typing and Translation to Synchronous Code.
In *ACM SIGPLAN/SIGBED Conference on Embedded Software (EMSOFT'11)*, Taipei, Taiwan, October 2011.

Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet.

Divide and recycle: types and compilation for a hybrid synchronous language.
In *ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES'11)*, Chicago, USA, April 2011.

Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet.

Non-Standard Semantics of Hybrid Systems Modelers.
Journal of Computer and System Sciences (JCSS), 78(3):877–910, May 2012.
Special issue in honor of Amir Pnueli.

Albert Benveniste, Benoit Caillaud, and Marc Pouzet.

The Fundamentals of Hybrid Systems Modelers.
In *49th IEEE International Conference on Decision and Control (CDC)*, Atlanta, Georgia, USA, December 15-17 2010.

Timothy Bourke and Marc Pouzet.

Zélus, a Synchronous Language with ODEs.
In *International Conference on Hybrid Systems: Computation and Control (HSCC 2013)*, Philadelphia, USA, April 8–11 2013. ACM.

Non-Standard Analysis

A bit of history

- Born in 1961 from Abraham Robinson, then developed by a small community of mathematicians.
- Proposed as a conservative enhancement of Zermelo-Fränkel set theory; some fancy axioms and principles; nice for the adicts
- Subject of controversies: what does it do for you that you cannot do using our brave analysis with $\forall\varepsilon\exists\eta\ldots$?
- 1988: a nice presentation of the topic by T. Lindstrom, kind of "*non-standard analysis for the axiom-averse*"
- 2006: used in Simon Bliudze PhD where he proposes the counterpart of a "Turing machine" for hybrid systems (supervised by D. Krob)

Why is non-standard analysis interesting for the computer scientist?

- it offers a step-based view of continuous and hybrid systems
- it is non-effective; still, it is amenable to symbolic executions and can thus be used for symbolic analyses at compile (and even run) time

Non-Standard Analysis

A bit of history

- Born in 1961 from Abraham Robinson, then developed by a small community of mathematicians.
- Proposed as a conservative enhancement of Zermelo-Fränkel set theory; some fancy axioms and principles; nice for the adicts
- Subject of controversies: what does it do for you that you cannot do using our brave analysis with $\forall\varepsilon\exists\eta\ldots$?
- 1988: a nice presentation of the topic by T. Lindstrom, kind of "*non-standard analysis for the axiom-averse*"
- 2006: used in Simon Bliudze PhD where he proposes the counterpart of a "Turing machine" for hybrid systems (supervised by D. Krob)

Why is non-standard analysis interesting for the computer scientist?

- it offers a step-based view of continuous and hybrid systems
- it is non-effective; still, it is amenable to symbolic executions and can thus be used for symbolic analyses at compile (and even run) time

Non-Standard Analysis

The aim

- to augment $\mathbb{R} \cup \{\pm\infty\}$ with elements that are *infinitely close* to x for each $x \in \mathbb{R}$, call ${}^\star\mathbb{R}$ the result;
- ${}^\star\mathbb{R}$ should obey the same algebra as $\mathbb{R}$: total order, $+, \times, \ldots$ any $f : \mathbb{R} \mapsto \mathbb{R}$ extends to ${}^\star f : {}^\star\mathbb{R} \mapsto {}^\star\mathbb{R}$, etc

Idea:

- mimic the construction of $\mathbb{R}$ from $\mathbb{Q}$ as Cauchy sequences; candidates for infinitesimals include:

$$\text{close to } 0 \quad : \quad \left\{\frac{1}{\sqrt{n}}\right\} > \left\{\frac{1}{n}\right\} > \left\{\frac{1}{n^2}\right\} > 0$$

$$\text{close to } +\infty \quad : \quad \{\sqrt{n}\} < \{n\} < \{n^2\}$$

Non-Standard Analysis

The aim

- to augment $\mathbb{R} \cup \{\pm\infty\}$ with elements that are *infinitely close* to x for each $x \in \mathbb{R}$, call ${}^\star\mathbb{R}$ the result;
- ${}^\star\mathbb{R}$ should obey the same algebra as $\mathbb{R}$: total order, $+, \times, \ldots$ any $f : \mathbb{R} \mapsto \mathbb{R}$ extends to ${}^\star f : {}^\star\mathbb{R} \mapsto {}^\star\mathbb{R}$, etc

Are we done? Not quite so:

- Sequences of reals $\{x_n\}$ generally do not converge
- Two sequences $\{x_n\}$ and $\{y_n\}$ converging to 0 may be s.t. $\{n \mid x_n > y_n\}$, $\{n \mid x_n < y_n\}$, and $\{n \mid x_n = y_n\}$ are all infinite sets

Non-Standard Analysis

The aim

- to augment $\mathbb{R} \cup \{\pm\infty\}$ with elements that are *infinitely close* to x for each $x \in \mathbb{R}$, call ${}^\star\mathbb{R}$ the result;
- ${}^\star\mathbb{R}$ should obey the same algebra as $\mathbb{R}$: total order, $+, \times, \ldots$ any $f : \mathbb{R} \mapsto \mathbb{R}$ extends to ${}^\star f : {}^\star\mathbb{R} \mapsto {}^\star\mathbb{R}$, etc

Are we done? Not quite so:

- Sequences of reals $\{x_n\}$ generally do not converge
- Two sequences $\{x_n\}$ and $\{y_n\}$ converging to 0 may be s.t. $\{n \mid x_n > y_n\}$, $\{n \mid x_n < y_n\}$, and $\{n \mid x_n = y_n\}$ are all infinite sets

Partition subsets of $\mathbb{N}$ into *neglectible*/*non-neglectible* ones, so that:

- finite or empty subsets are all neglectible
- neglectible sets are stable under finite unions
- for any subset P, either P or its complement is non-neglectible

Having such a decision mechanism relies on Zorn Lemma ($\approx$ axiom of choice) and is formalized as explained next.

Non-Standard Analysis: the idea of Lindstrom

Pick $\mathcal{F}$ a free ultrafilter of $\mathbb{N}$:

- $\emptyset \notin \mathcal{F}$, $\mathcal{F}$ stable by intersection
- $P \in \mathcal{F}$ and $P \subseteq Q$ implies $Q \in \mathcal{F}$
- either P or $\mathbb{N} - P$ belongs to $\mathcal{F}$
- P finite implies $P \notin \mathcal{F}$

Existence of $\mathcal{F}$ follows from Zorn's lemma ($\Leftrightarrow$ axiom of choice)

Say that P is *neglectible* iff $P \notin \mathcal{F}$

Non-Standard Analysis: the idea of Lindstrom

Pick $\mathcal{F}$ a free ultrafilter of $\mathbb{N}$:

- $\emptyset \notin \mathcal{F}$, $\mathcal{F}$ stable by intersection
- $P \in \mathcal{F}$ and $P \subseteq Q$ implies $Q \in \mathcal{F}$
- either P or $\mathbb{N} - P$ belongs to $\mathcal{F}$
- P finite implies $P \notin \mathcal{F}$

Existence of $\mathcal{F}$ follows from Zorn's lemma ($\Leftrightarrow$ axiom of choice)

Say that P is *neglectible* iff $P \notin \mathcal{F}$

Non-Standard Analysis: the idea of Lindstrom

$(x_n), (x'_n) \in \mathbb{R}^{\mathbb{N}}$, define $(x_n) \approx (x'_n)$ iff set $\{n \mid x_n \neq x'_n\}$ is neglectible

$$^\star\mathbb{R} = \mathbb{R}^{\mathbb{N}}/\approx \text{ ; elements of } {}^\star\mathbb{R} \text{ are written } [x_n]$$

- For any two $(x_n), (y_n)$ exactly one among the sets $\{n \mid x_n > y_n\}$, $\{n \mid x_n < y_n\}$, $\{n \mid x_n = y_n\}$, is non-neglectible
  $\Longrightarrow$ any two sequences can always be compared modulo $\approx$

- By pointwise extension, a 1^{st}-order formula is true over $^\star\mathbb{R}$ iff it is true over $\mathbb{R}$: this is known as the *transfer principle*
  Ex: defining $+, -, \times \ldots$ by pointwise extension

- Say that

$$x = st([x_n]) \text{ if } x_n \to x \quad \text{modulo neglectible sets}$$

Non-Standard Analysis: the idea of Lindstrom

Theorem: [*standardisation*]

Any finite non-standard real $[x_n]$ possesses a unique standard part

Proof:

1. Pick
$$x = \sup\{u \in \mathbb{R} \mid [u] \leq [x_n]\}$$
where $[u]$ denotes the constant sequence equal to u.
2. Since $[x_n]$ is finite, x exists; remains to show that $[x_n] - x$ is infinitesimal.
3. If this is not true,
   - then there exists $y \in \mathbb{R}, y > 0$ such that $y < |x - [x_n]|$,
   - that is, either $x < [x_n] - [y]$ or $x > [x_n] + [y]$,
   - which both contradict the definition of x.
4. The uniqueness of x is clear, thus we can define $st([x_n]) = x$.

(Infinite non-standard reals have no standard part in $\mathbb{R}$.)

Integrals, ODE, and the Standardisation Principle

- internal functions and sets by pointwise extension:

$$\forall n, g_n : \mathbb{R} \mapsto \mathbb{R} \text{ yields } [g_n] : {}^\star\mathbb{R} \mapsto {}^\star\mathbb{R} \text{ by } [g_n]([x_n]) = [g_n(x_n)]$$

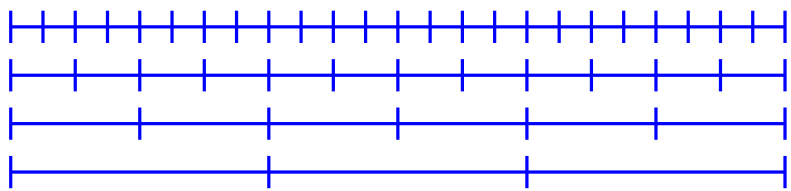
- Pick ∂ infinitesimal and $N \in {}^\star\mathbb{N}$ s.t. $(N-1)\partial < 1 \leq N\partial$, and consider the set

$$T = \{0, \partial, 2\partial, \ldots, (N-1)\partial, 1\}$$

By definition, if $\partial = [d_n]$, then $N = [N_n]$ with $N_n = \dfrac{1}{d_n}$ and $T = [T_n]$ with

$$T_n = \{0, d_n, 2d_n, \ldots, (N_n - 1)d_n, 1\}$$

picturing T: $T = [T_n]$



- For $f : [0,1] \mapsto \mathbb{R}$ a continuous function and ${}^\star f = [f, f, \ldots]$:

$$\left[\sum_{t \in T_n} \frac{1}{N_n} f(t_n)\right] \quad = \quad \sum_{t \in T} \frac{1}{N} {}^\star f(t)$$

Integrals, ODE, and the Standardisation Principle

- internal functions and sets by pointwise extension:

$$\forall n, g_n : \mathbb{R} \mapsto \mathbb{R} \text{ yields } [g_n] : {}^\star\mathbb{R} \mapsto {}^\star\mathbb{R} \text{ by } [g_n]([x_n]) = [g_n(x_n)]$$

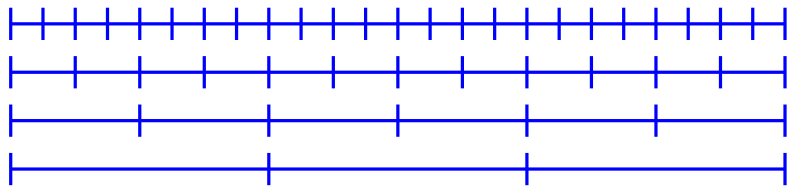
- Pick ∂ infinitesimal and $N \in {}^\star\mathbb{N}$ s.t. $(N-1)\partial < 1 \leq N\partial$, and consider the set

$$T = \{0, \partial, 2\partial, \ldots, (N-1)\partial, 1\}$$

By definition, if $\partial = [d_n]$, then $N = [N_n]$ with $N_n = \dfrac{1}{d_n}$ and $T = [T_n]$ with

$$T_n = \{0, d_n, 2d_n, \ldots, (N_n - 1)d_n, 1\}$$

picturing T: $T = [T_n]$



- For $f : [0,1] \mapsto \mathbb{R}$ a continuous function and ${}^\star f = [f, f, \ldots]$:

$$st\left(\left[\sum_{t \in T_n} \frac{1}{N_n} f(t_n)\right]\right) = st\left(\sum_{t \in T} \frac{1}{N} {}^\star f(t)\right) = \int_0^1 f(t)dt$$

Integrals, ODE, and the Standardisation Principle

Theorem: [*standardisation*] if $f : [0,1] \to \mathbb{R}$ is continuous, then

$$st\left(\left[\sum_{t\in T_n} \frac{1}{N_n} f(t_n)\right]\right) = st\left(\sum_{t\in T} \frac{1}{N}{}^{\star}f(t)\right) = \int_0^1 f(t)dt$$

Proof: If $f : \mathbb{R} \to \mathbb{R}$ is a standard function, we always have

$$\sum_{t\in T} \frac{1}{N}{}^{\star}f(t) = \left[\sum_{t\in T_n} \frac{1}{N_n} f(t_n)\right] \tag{1}$$

Now, f continuous implies $\sum_{t\in T_n} \frac{1}{N_n} f(t_n) \to \int_0^1 f(t)dt$, so, by definition of non-standard reals,

$$\int_0^1 f(t)dt = st\left(\sum_{t\in T} \frac{1}{N}{}^{\star}f(t)\right) \tag{2}$$

- If f is smooth so that its Riemann integral is well defined, any non-standard formulation of the integral of f has $\int_0^1 f(t)dt$ as its

Integrals, ODE, and the Standardisation Principle

Theorem: [*standardisation*] if $f : [0,1] \to \mathbb{R}$ is continuous, then

$$st\left(\left[\sum_{t\in T_n} \frac{1}{N_n} f(t_n)\right]\right) = st\left(\sum_{t\in T} \frac{1}{N}{}^\star f(t)\right) = \int_0^1 f(t)dt$$

Proof: If $f : \mathbb{R} \to \mathbb{R}$ is a standard function, we always have

$$\sum_{t\in T} \frac{1}{N}{}^\star f(t) = \left[\sum_{t\in T_n} \frac{1}{N_n} f(t_n)\right] \tag{1}$$

Now, f continuous implies $\sum_{t\in T_n} \frac{1}{N_n} f(t_n) \to \int_0^1 f(t)dt$, so, by definition of non-standard reals,

$$\int_0^1 f(t)dt = st\left(\sum_{t\in T} \frac{1}{N}{}^\star f(t)\right) \tag{2}$$

- If f is smooth so that its Riemann integral is well defined, any non-standard formulation of the integral of f has $\int_0^1 f(t)dt$ as its

Integrals, ODE, and the Standardisation Principle

Focus on ODEs. For every $0 < t \leq 1$:

$$\int_0^t f(u)du = st\left(\sum_{u\in T, u\leq t} \frac{1}{N}{}^\star f(t)\right) \quad \text{(Non-standard Riemann integral)}$$

Set $\partial = \frac{1}{N}$ and consider the ODE $\dot{x} = f(x,t), x_0$, in integral form

$$x(t) = x_0 + \int_0^t f(x(u),u)du \quad \text{(with the needed smoothness)} \tag{3}$$

$$\begin{aligned} x(t) &= st(x_0 + \sum_{k\,:\,0\leq k\partial\leq t} \frac{1}{N}{}^\star f({}^\star x(k\partial), k\partial)) \\ &= st\left({}^\star x(s_t)\right), \text{ for } s_t = \max\{t_k \mid t_k = k\partial \leq t\} \end{aligned} \tag{4}$$

where ${}^\star x$ is the non-standard semantics of the above ODE with time basis ∂:

$$\begin{cases} {}^\star x(t_k) = {}^\star x(t_{k-1}) + \partial \times f({}^\star x(t_{k-1}), t_{k-1}) \\ {}^\star x(t_0) = x_0 \end{cases} \tag{5}$$

Theorem: [*standardisation*]

Integrals, ODE, and the Standardisation Principle

Focus on ODEs. For every $0 < t \leq 1$:

$$\int_0^t f(u)du = st\left(\sum_{u\in T, u\leq t} \frac{1}{N}{}^\star f(t)\right) \quad \text{(Non-standard Riemann integral)}$$

Set $\partial = \frac{1}{N}$ and consider the ODE $\dot{x} = f(x,t), x_0$, in integral form

$$x(t) = x_0 + \int_0^t f(x(u),u)du \quad \text{(with the needed smoothness)} \tag{3}$$

$$\begin{aligned} x(t) &= st\left(x_0 + \sum_{k\,:\,0\leq k\partial\leq t} \frac{1}{N}{}^\star f({}^\star x(k\partial), k\partial)\right) \\ &= st\left({}^\star x(s_t)\right), \text{ for } s_t = \max\{t_k \mid t_k = k\partial \leq t\} \end{aligned} \tag{4}$$

where ${}^\star x$ is the non-standard semantics of the above ODE with time basis ∂:

$$\begin{cases} {}^\star x(t_k) = {}^\star x(t_{k-1}) + \partial \times f({}^\star x(t_{k-1}), t_{k-1}) \\ {}^\star x(t_0) = x_0 \end{cases} \tag{5}$$

Theorem: [*standardisation*]

Integrals, ODE, and the Standardisation Principle

Focus on ODEs. For every $0 < t \leq 1$:

$$\int_0^t f(u)du = st\left(\sum_{u\in T, u\leq t} \frac{1}{N}{}^\star f(t)\right) \quad \text{(Non-standard Riemann integral)}$$

Set $\partial = \frac{1}{N}$ and consider the ODE $\dot{x} = f(x,t), x_0$, in integral form

$$\begin{aligned} x(t) &= x_0 + \int_0^t f(x(u),u)du \quad \text{(with the needed smoothness)} \qquad (3)\\ x(t) &= st(x_0 + \sum_{k\,:\,0\leq k\partial\leq t} \frac{1}{N}{}^\star f({}^\star x(k\partial), k\partial)) \\ &= st\left({}^\star x(s_t)\right), \text{ for } s_t = \max\{t_k \mid t_k = k\partial \leq t\} \qquad (4)\end{aligned}$$

where ${}^\star x$ is the non-standard semantics of the above ODE with time basis ∂:

$$\begin{cases} {}^\star x(t_k) &= {}^\star x(t_{k-1}) + \partial \times f({}^\star x(t_{k-1}), t_{k-1}) \\ {}^\star x(t_0) &= x_0 \end{cases} \qquad (5)$$

Theorem: [*standardisation*]