# Why considering nonstandard semantics for hybrid systems and how to reconcile it with superdense time semantics

**Albert Benveniste**[1] Timothy Bourke[1] Benoît Caillaud[1]
Marc Pouzet[2]

1. INRIA
2. École normale supérieure

In Honor of Oded Maler, HSCC 2019

**Oded spent a few years with us for his postdoc. We were working on Signal.**

**His thesis had been quite mathematical. . . so we expected him to contribute to Signal with maths. . .**

**However, he was dreaming of worms, asking us whether Signal would help for this. We expressed doubts. . .**

**So he left and moved to Grenoble. . .**

**Not sure what his dreams are as of now**

**Context: mix of discrete and continuous time dynamics**

**Causality issues in Hybrid Systems Modeling Languages**

**Requirements on Semantic Models of Time**

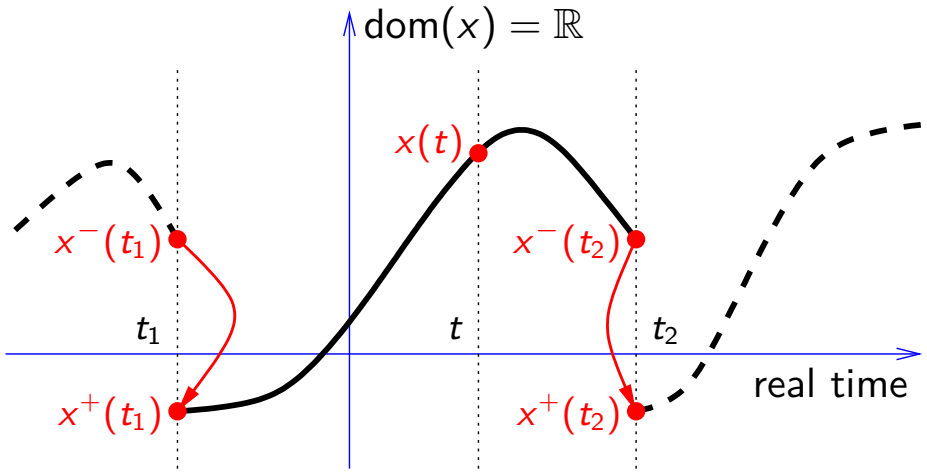
**Time Domains**

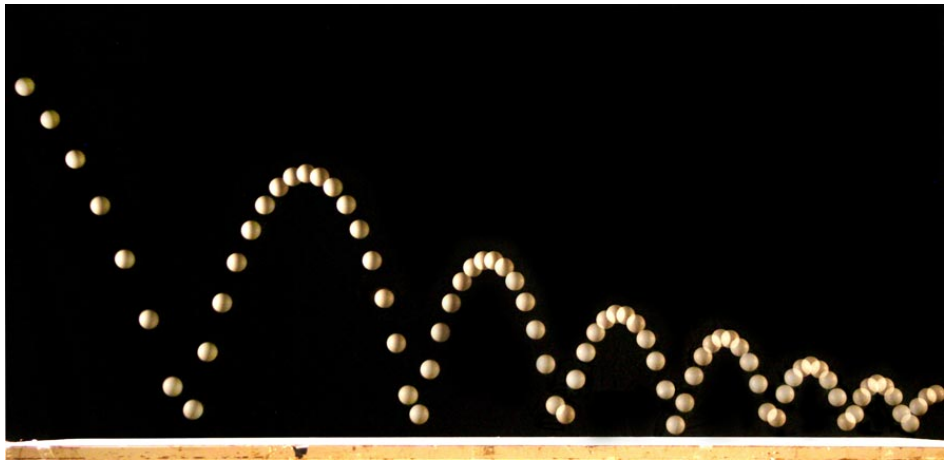
**Nonstandard Analysis in 2 slides**

**Nonstandard Semantics**

**Conclusion**

# From Dynamical to Hybrid Systems, informally



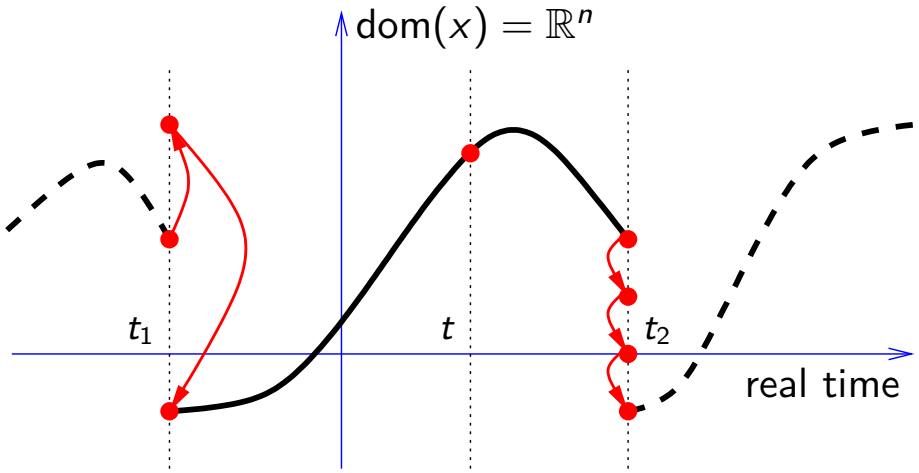



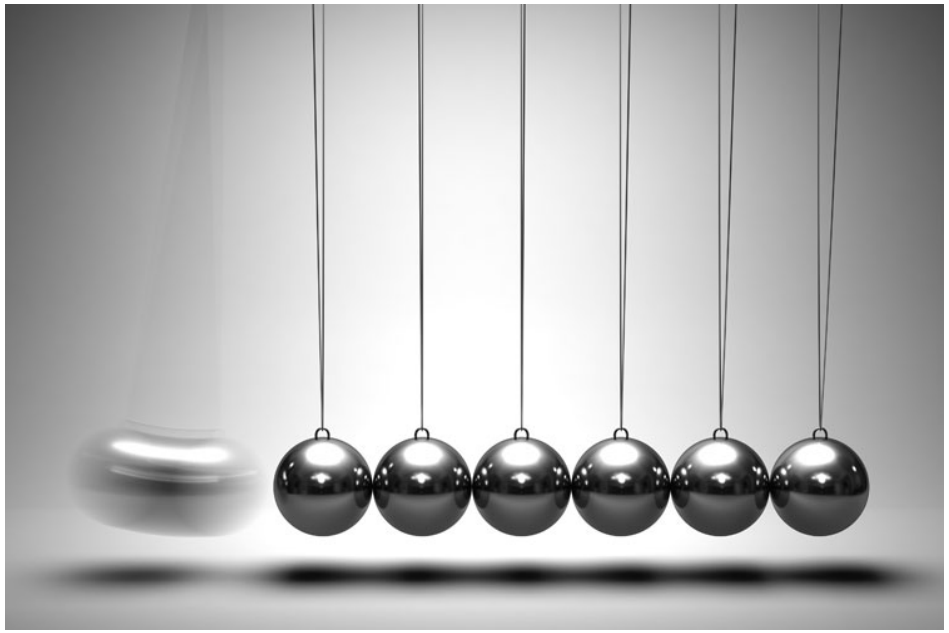
Simple Hybrid Systems:

- smooth dynamics almost all the time, except for state jumps at some discrete events, where $q^+ = g(q^-)$ holds.
- Time $= \mathbb{R}$ still works.

How general is this?

# From Dynamical to Hybrid Systems, informally



In general, Hybrid Systems trajectories may have:

- Instantaneous cascades of state jumps
- Chattering

This cannot be captured as Time $= \mathbb{R}$. We need a Time Domain "richer" than $\mathbb{R}$

# Semantics of Hybrid Systems Modelers

**A precise mathematical semantics is instrumental to design:**

- Static analyzers / model-checkers / theories for interactive provers
- Compile-time analyses / simulation / code generation
- Numerical simulation environments (run-time)

**Focus of this talk:**

- Which Time Domains for the semantics of hybrid systems modelers?

**Context: mix of discrete and continuous time dynamics**

**Causality issues in Hybrid Systems Modeling Languages**

**Requirements on Semantic Models of Time**

**Time Domains**

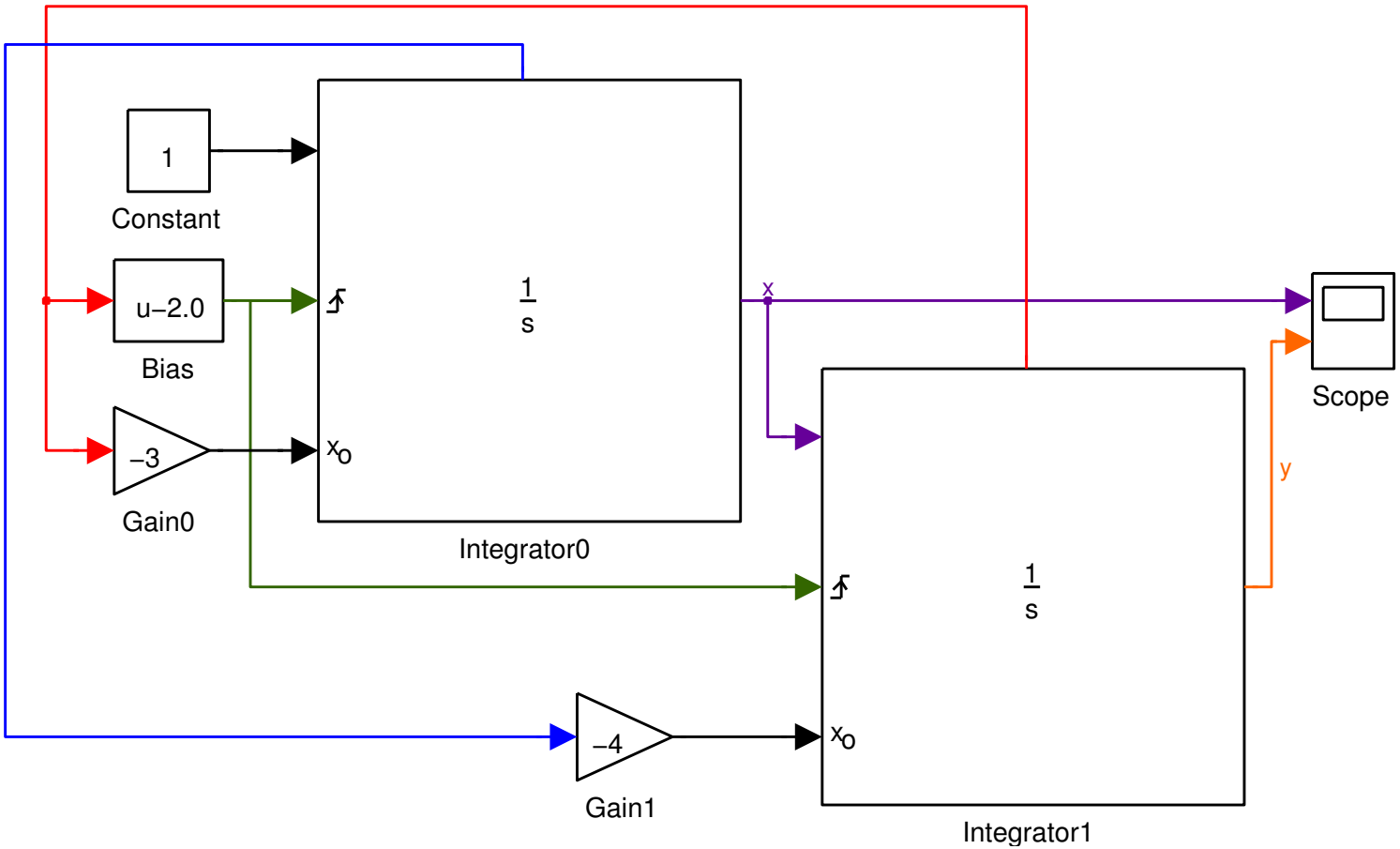
**Nonstandard Analysis in 2 slides**

**Nonstandard Semantics**

**Conclusion**

# A Cabinet of Curiosities . . .

# Wrong scheduling of the Simulink state port



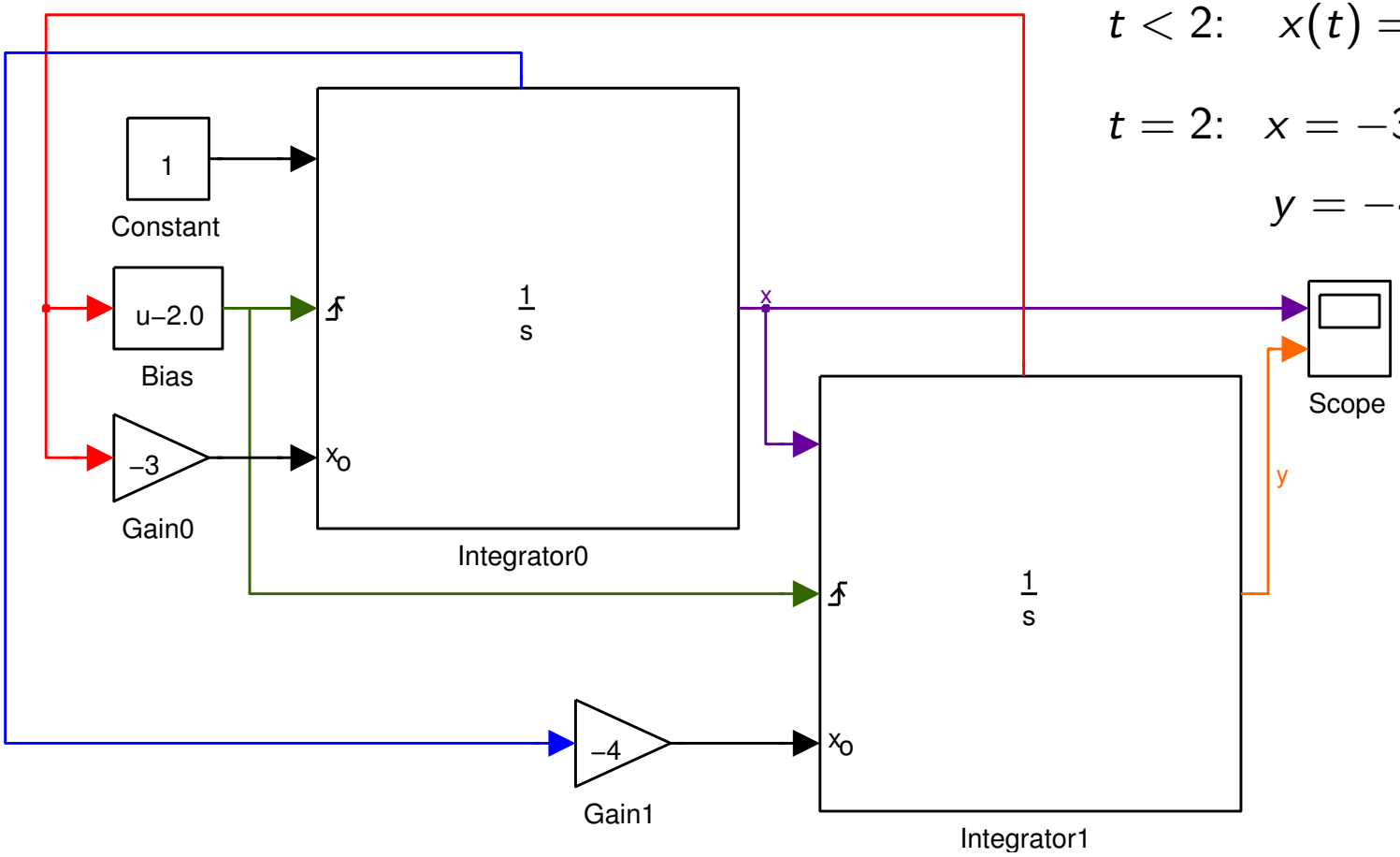

The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset—Simulink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

# Wrong scheduling of the Simulink state port

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

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

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



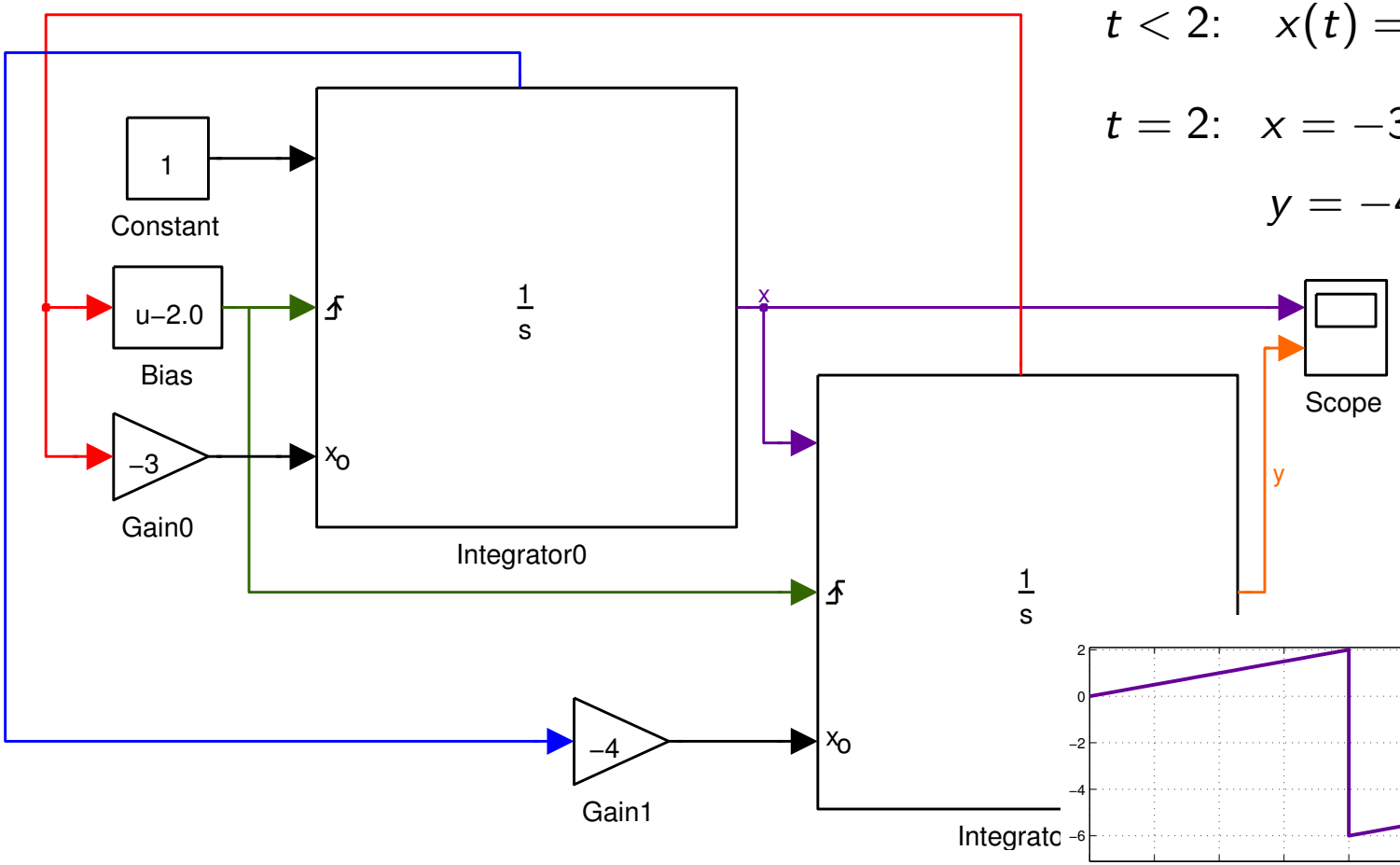

The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset—Simulink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

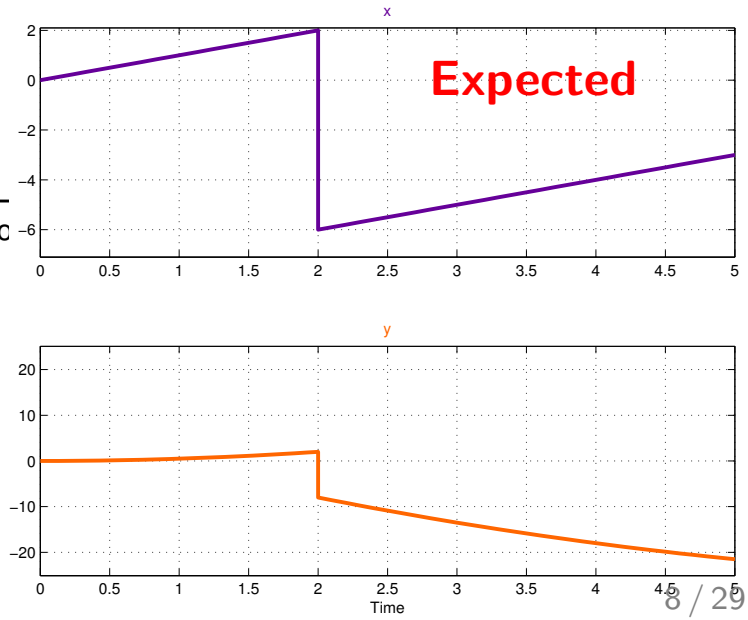
# Wrong scheduling of the Simulink state port

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

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

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



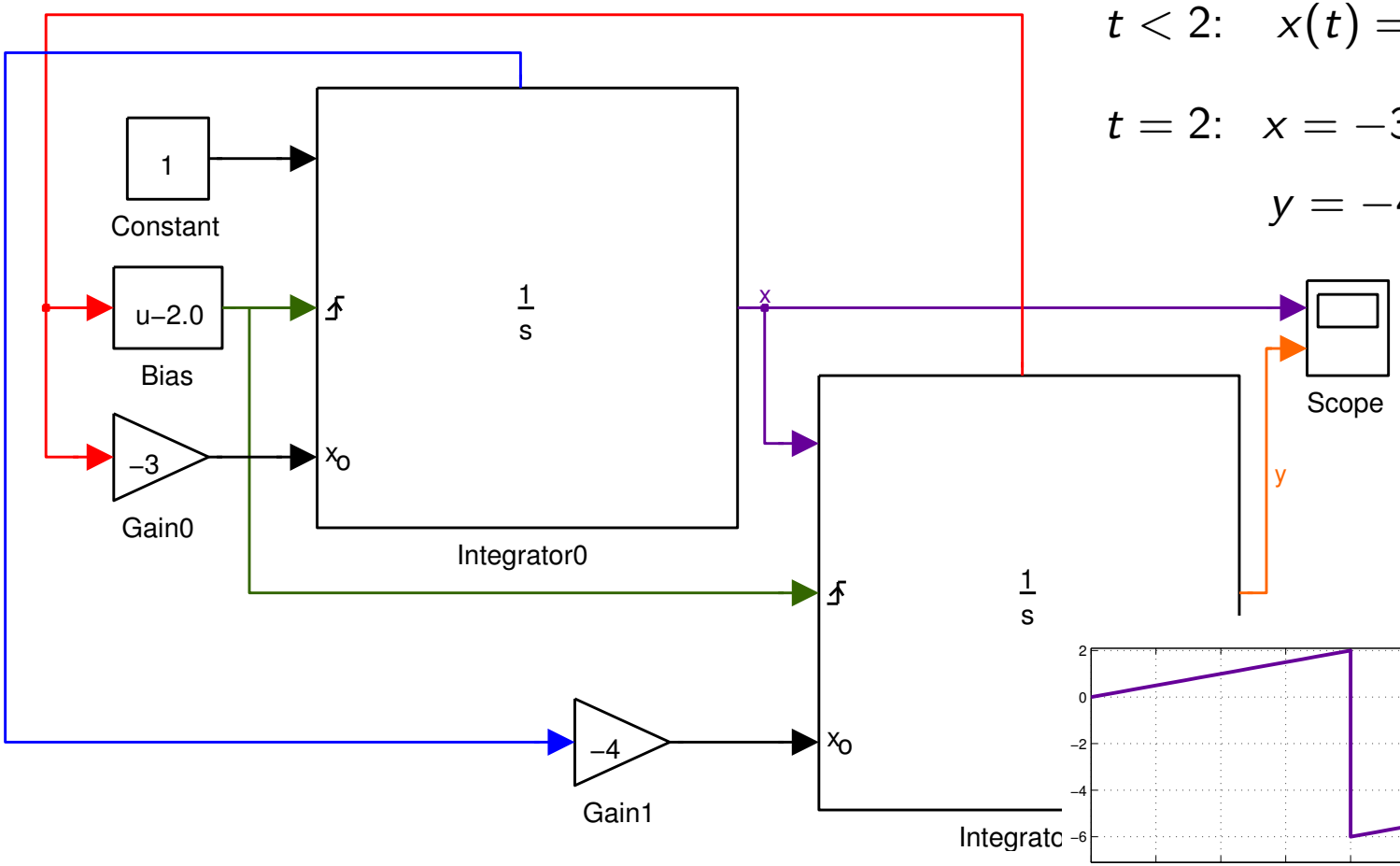

The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset—Simulink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

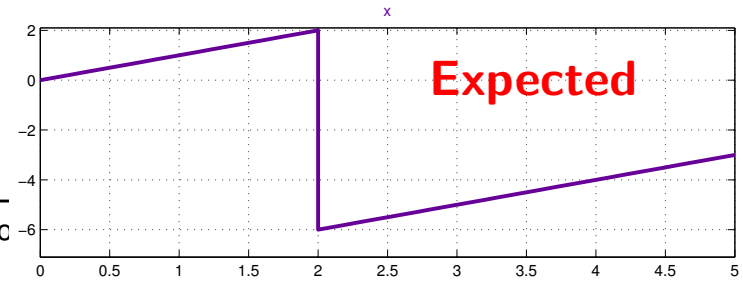
# Wrong scheduling of the Simulink state port

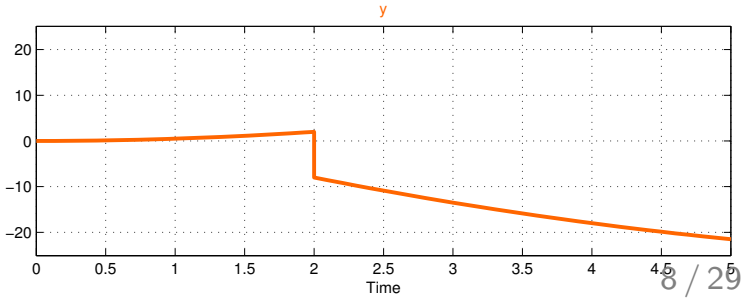
$$t < 2: \quad x(t) = t,\ y(t) = \frac{t^2}{2}$$

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

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



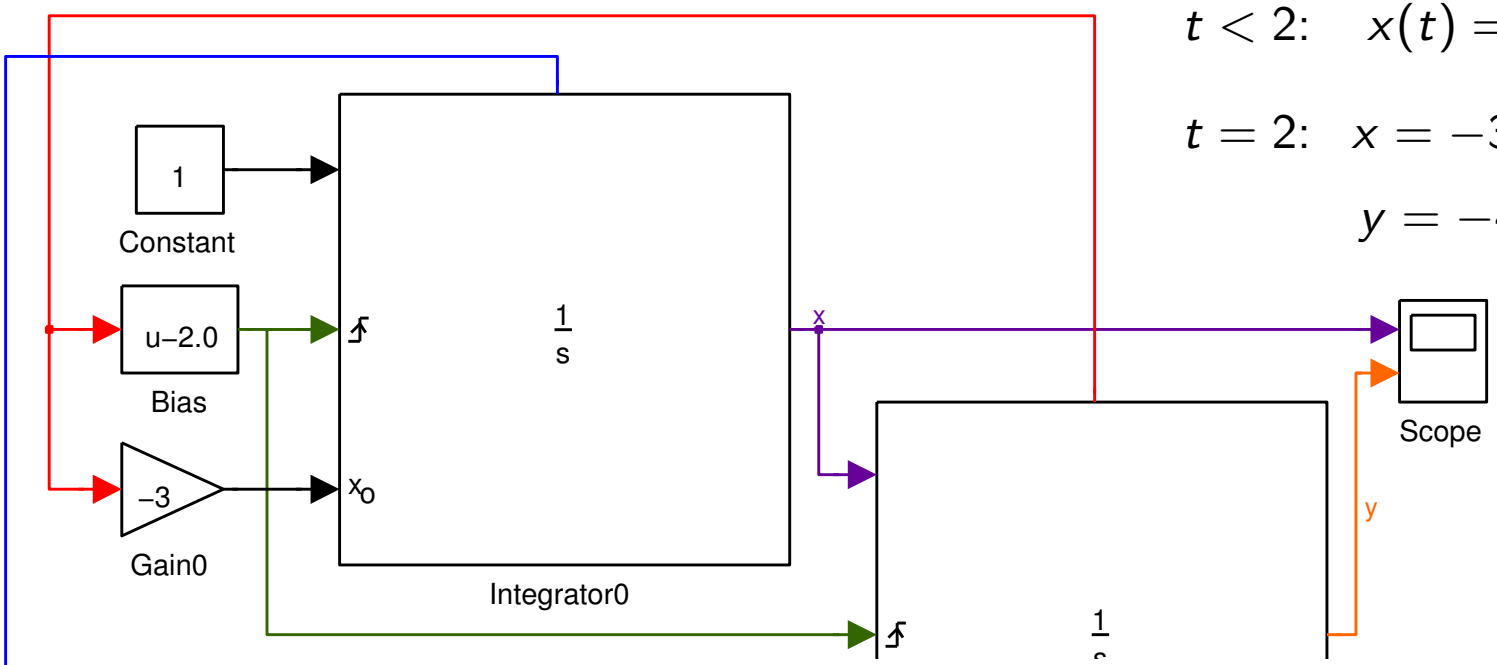

The output of the state port is the same as the output of the block's standard output port except for the following case. If the block is reset in the current time step, the output of the state port is the value that would have appeared at the block's standard output if the block had not been reset—Simulink Reference (2-685). **However, C code produced by RTW (release R2009) yields:**

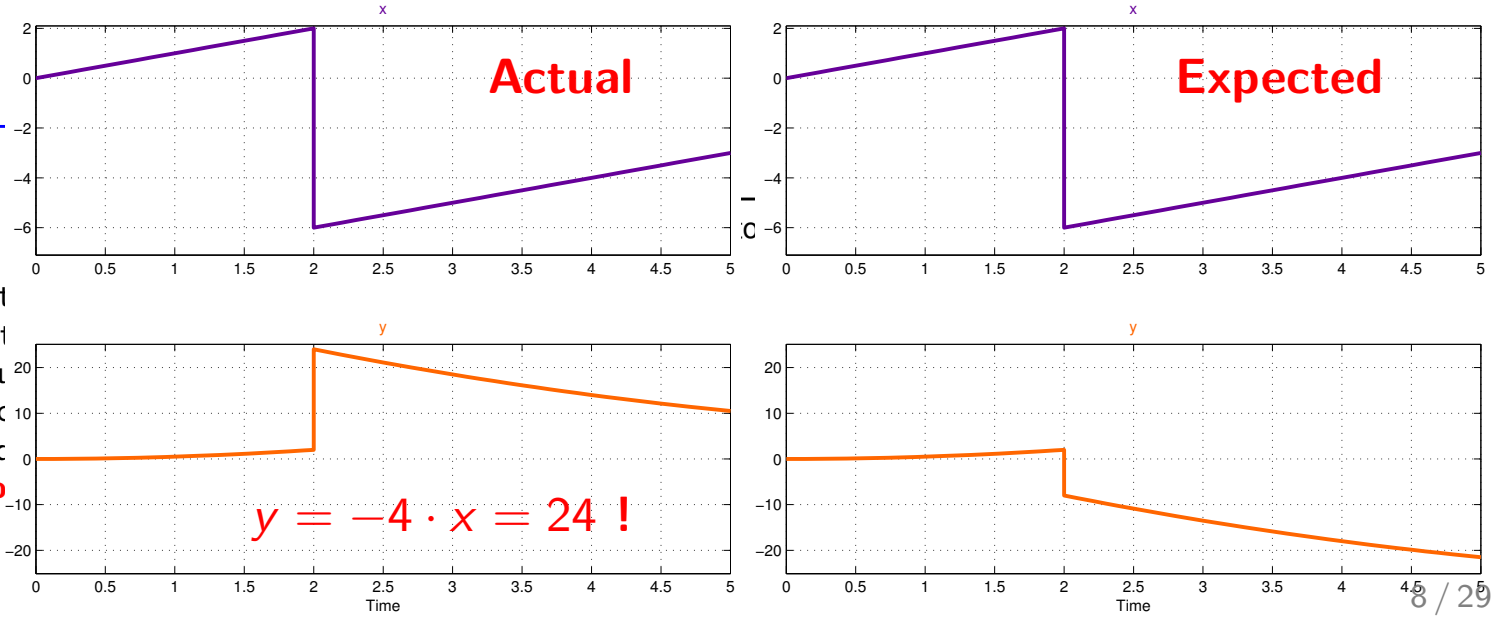
# Wrong scheduling of the Simulink state port

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

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

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




The output of the stat
block's standard outpu
block is reset in the cu
port is the value that wo
output if the block hac
685). **However, C co**
**yields:**

# 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

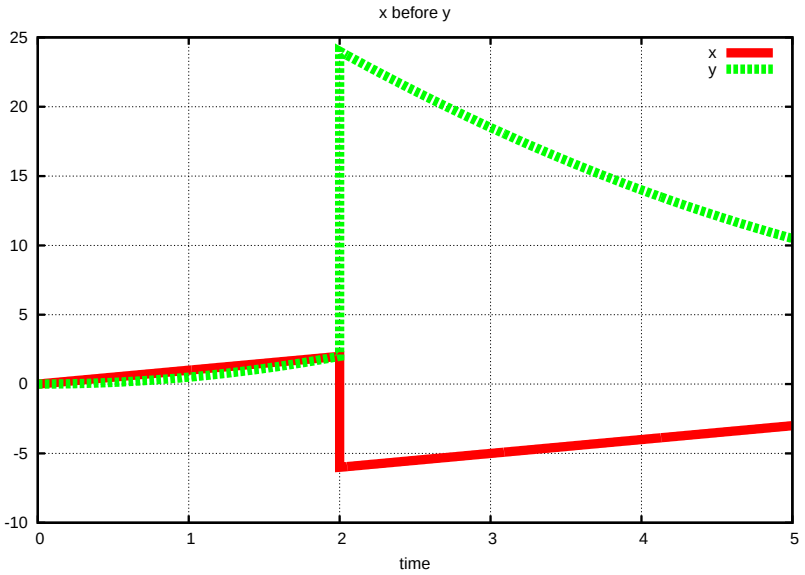
# Causality: Modelica example

```
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

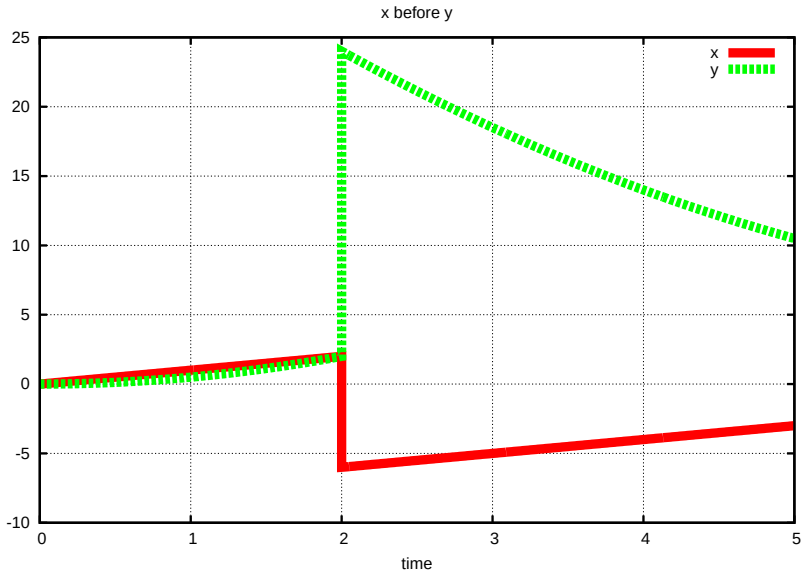
# Causality: Modelica example

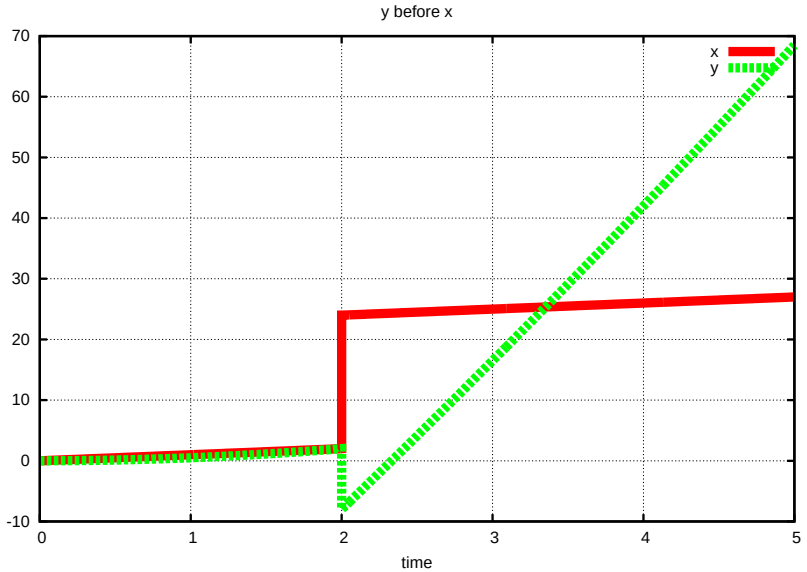
```
model scheduling
  Real x(start = 0);
  Real y(start = 0);
equation

  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'd)

- A causal version (i.e., reinit(x, $-3 * \text{pre } y$) is scheduled properly. Everything works as expected.
- 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 question is: what to do with them at events of mode changes? Not easy to solve, but anyway, the semantics of a model must not depend on its layout.
- Wrong scheduling of reset actions results from an incorrect causality analysis. Studying causality is thus needed to understand the interactions between discrete and continuous-time behaviors.

**Context: mix of discrete and continuous time dynamics**

**Causality issues in Hybrid Systems Modeling Languages**

**Requirements on Semantic Models of Time**

**Time Domains**

**Nonstandard Analysis in 2 slides**

**Nonstandard Semantics**

**Conclusion**

# A Toy Hybrid Systems Language

Syntax ≈ Zélus [HSCC 2013] ≈ SCADE Hybrid (ANSYS) [CC 2015]

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

Simple synchronous language ($\sim$ Lustre)
augmented with continuous time primitives

# Zélus ( zelus.di.ens.fr )


- Zélus ≈ Lucid Synchrone [Pouzet 2006] + ODEs
- Dataflow equations, hierarchical state machines [Emsoft 2011]
- Precise semantics [JCSS 2012]
- Discrete/Continuous type inference [LCTES 2011]: rejects spurious compositions of discrete- and continuous-time dynamics
- Causality analysis [HSCC 2014]
- Similar effort: SCADE Hybrid (ANSYS) [CC 2015] required changing only $\sim 5\%$ of the KCG compiler

# Requirements on Semantics

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:

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

- The alternative is informal "mytool" semantics

# Requirements on Semantics

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:

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

- The alternative is informal "mytool" semantics

**Context: mix of discrete and continuous time dynamics**

**Causality issues in Hybrid Systems Modeling Languages**

**Requirements on Semantic Models of Time**

**Time Domains**

**Nonstandard Analysis in 2 slides**

**Nonstandard Semantics**

**Conclusion**

# Proposed Models of Time for hybrid systems

## Superdense time:

- $\mathbb{T} = \mathbb{R}_{\geq 0} \times \mathbb{N}$, lexicographic order
- [Maler et al. 1991] [Lee et al. 2005] (for Ptolemy semantics)

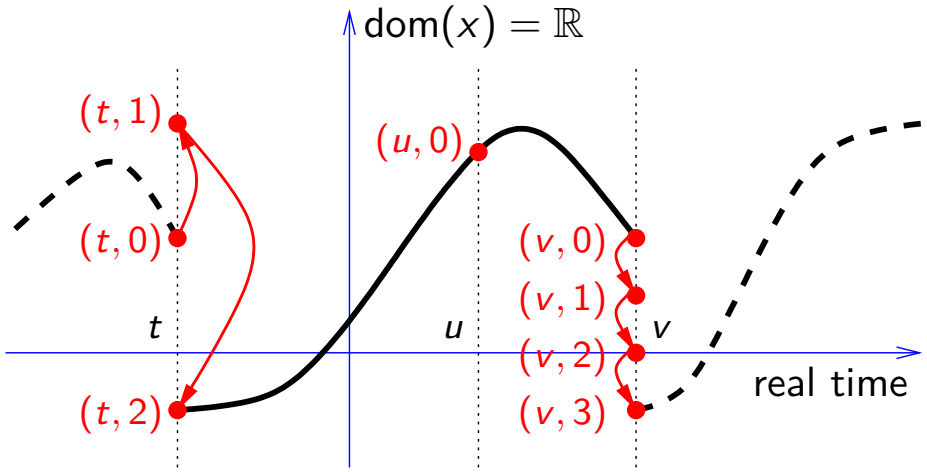
## Model of time used in Hybrid Automata (HA):

- $\mathbb{T} = \mathbb{N} \times \mathbb{R}_{\geq 0}$, lexicographic order
- [Henzinger 1996] [Frehse 2011] in formal verification
- Henzinger's HA only interact via common events

## Nonstandard time:

- $\mathbb{T} = \{n\partial \mid n \in {}^{\star}\mathbb{N}\}$, $\partial$ infinitesimal time step
- [Benveniste et al. 2012] for language semantics

# Superdense Model of Time




$\mathbb{T} = \mathbb{R}_+ \times \mathbb{N}$
lexicographic order
signal: $\mathbb{T} \to$ Real

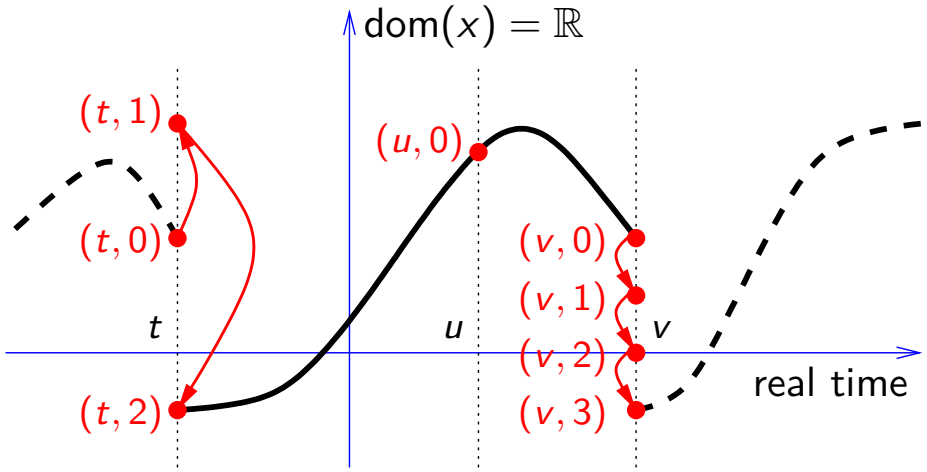
- Continuous time phases separated by (cascades of) events. Two approaches:

  1. $x(t, n)$ defined for $0 \leq n \leq m_t$ 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: $m_t = 2, m_u = 0, m_v = 3$.

# Superdense Model of Time

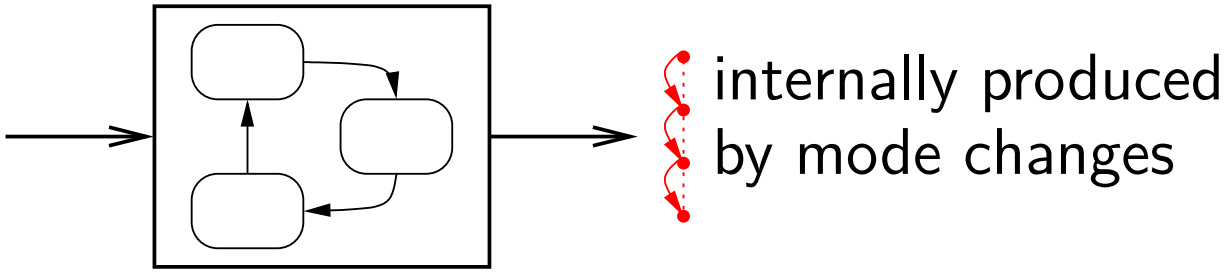



$\mathbb{T} = \mathbb{R}_+ \times \mathbb{N}$
lexicographic order
signal: $\mathbb{T} \to$ Real

From [Lee 2014, IEEE Access]:

> *[. . . ] 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. [. . . ]*

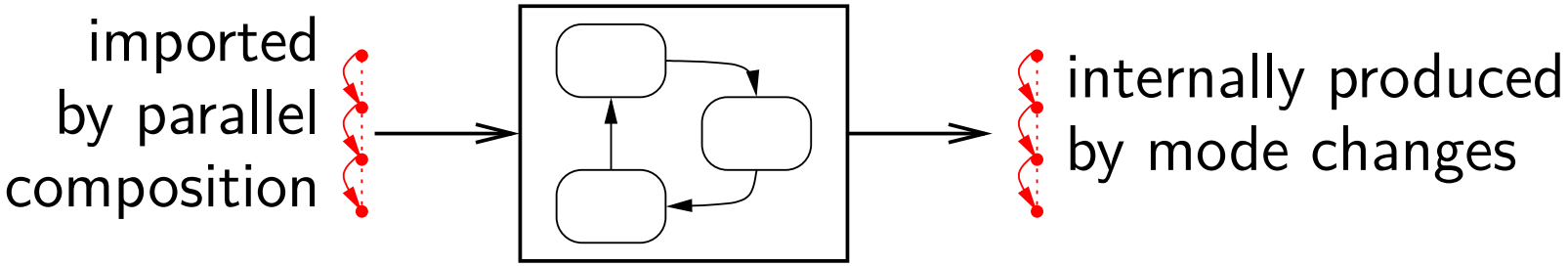
# Superdense Model of Time




This is no longer true when parallel composition is considered:

- in superdense time $(t, n)$, the $n$-component is used to capture internal mode changes of systems

# Superdense Model of Time

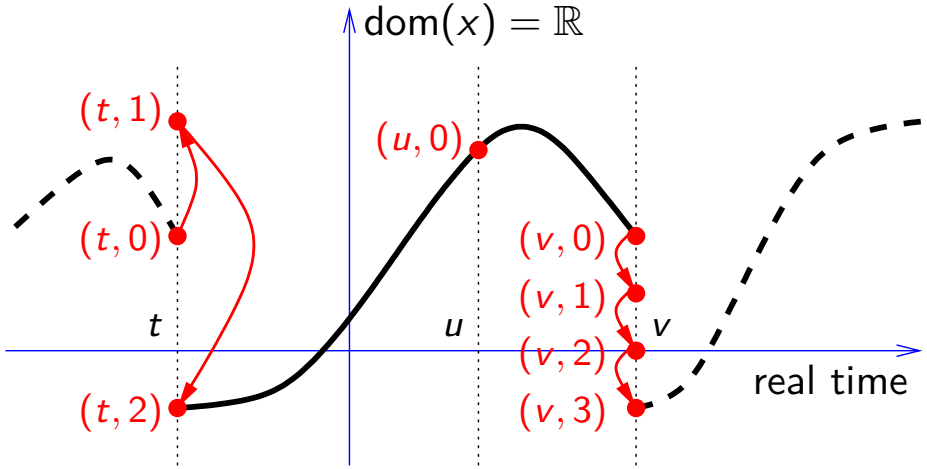



This is no longer true when parallel composition is considered:

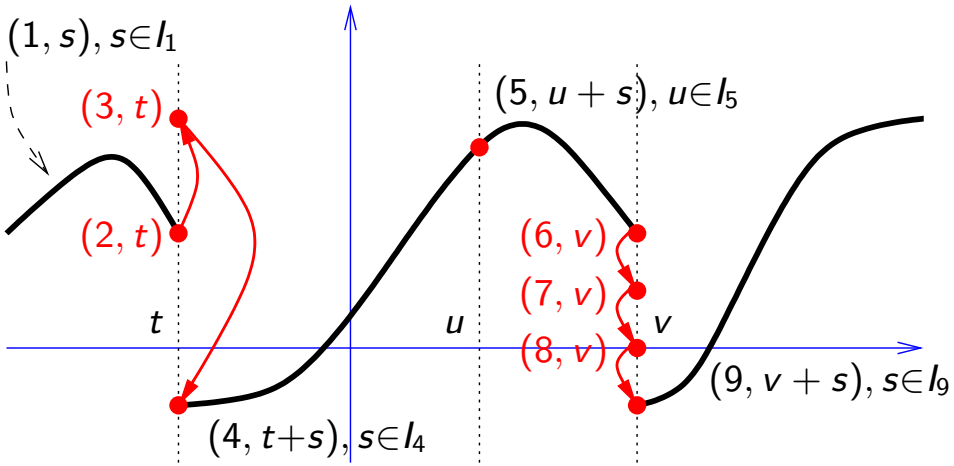
- in superdense time $(t, n)$, the $n$-component is used to capture internal mode changes of systems
- but can also be inherited from the (yet unknown) environment through inputs

Unless provision is given for the $\mathbb{N}$-component at any $t \in \mathbb{R}$ (makes it complicated), the superdense time semantics cannot be defined independently from the environment of a model: it is not compositional!

# Model of time used in Hybrid Automata



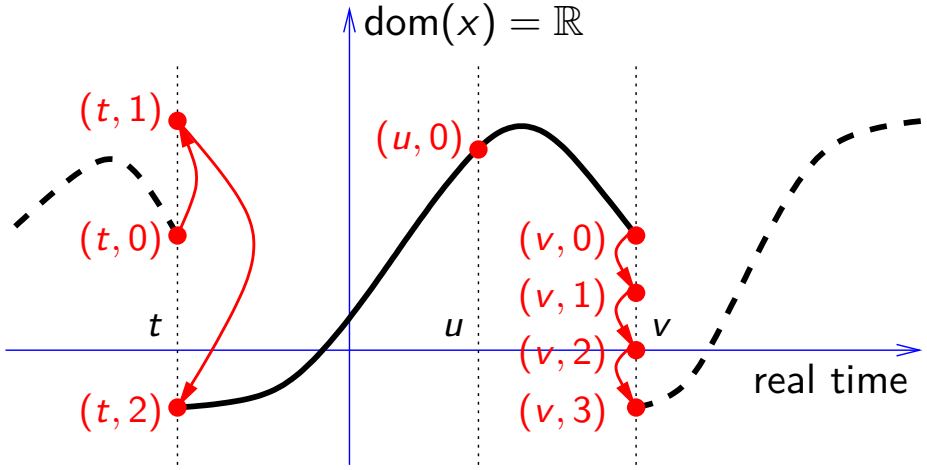

Superdense Model of Time
$\mathbb{T} = \mathbb{R}_{\geq 0} \times \mathbb{N}$
signal: $\mathbb{T} \rightarrow$Real



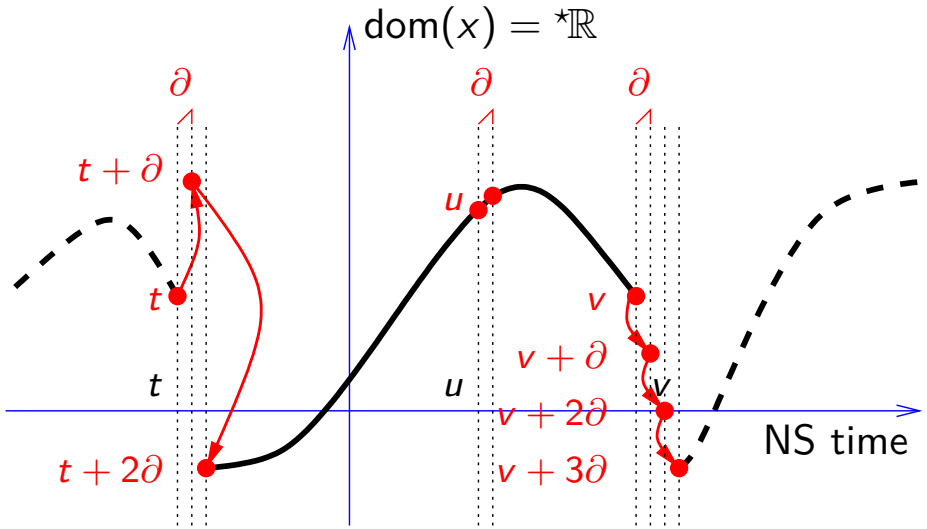

Time in Hybrid Automata
$\mathbb{T} = \mathbb{N} \times \mathbb{R}_{\geq 0}$
signal: $\mathbb{N} \rightarrow$Interval$\rightarrow$Real

Advantage: gives provision for stuttering invariance: a continuous interval can be naturally split into two adjacent subintervals $\implies$ compositional

# Nonstandard Model of Time




Superdense Model of Time
$\mathbb{T} = \mathbb{R}_{\geq 0} \times \mathbb{N}$
signal: $\mathbb{T} \rightarrow$Real




Nonstandard Model of Time
$\mathbb{T} = \{t = n\partial \mid n \in {}^\star\mathbb{N}\}$

$\partial$: infinitesimal time step

$\dot{x} = f(x, u)$ expanded as
$x_{t+\partial} = x_t + \partial \times f(x_t, u)$

# Nonstandard Model of Time

Key ideas:

- take discrete time with infinitesimal time step $\partial$

  $\Longrightarrow$ since time is discrete,
  we have yet another synchronous language
  whose semantics is compositional
  and causality analysis known

- ODE seen as its forward 1st Euler approximation

  $\Longrightarrow$ approximation error is infinitesimal: perfect semantics

There is no free lunch, however:

- our semantics is compositional
- can be used for compile time analysis and verifications
- but cannot be simulated (infinitesimals do not exist in practice)

  $\Longrightarrow$ additional work needed to generate executable semantics

# Nonstandard Model of Time

Key ideas:

- take discrete time with infinitesimal time step $\partial$

  $\Longrightarrow$ since time is discrete,
  we have yet another synchronous language
  whose semantics is compositional
  and causality analysis known

- ODE seen as its forward 1st Euler approximation

  $\Longrightarrow$ approximation error is infinitesimal: perfect semantics

There is no free lunch, however:

- our semantics is compositional
- can be used for compile time analysis and verifications
- but cannot be simulated (infinitesimals do not exist in practice)

  $\Longrightarrow$ additional work needed to generate executable semantics

Context: mix of discrete and continuous time dynamics

Causality issues in Hybrid Systems Modeling Languages

Requirements on Semantic Models of Time

Time Domains

Nonstandard Analysis in 2 slides

Nonstandard Semantics

Conclusion

# Nonstandard Analysis for the computer scientist

## History:

- Infinitesimal numbers were considered by Archimède, Newton, Fermat, Leibniz, ... but lacked rigorous foundations. **Nonstandard Analysis (NSA)** was proposed by A. Robinson in 1961 as a conservative enhancement of Zermelo-Fränkel set theory (nice for the addicts)
- Used to reformulate some problems in maths (Algebraic topology, Stochastic processes and Brownian motion,...). Controversial: what does it do that you cannot do using our brave analysis with $\forall\varepsilon\exists\eta\ldots$?

# Nonstandard Analysis for the computer scientist

## History:

- Infinitesimal numbers were considered by Archimède, Newton, Fermat, Leibniz, ... but lacked rigorous foundations. **Nonstandard Analysis (NSA)** was proposed by A. Robinson in 1961 as a conservative enhancement of Zermelo-Fränkel set theory (nice for the addicts)
- Used to reformulate some problems in maths (Algebraic topology, Stochastic processes and Brownian motion,...). Controversial: what does it do that you cannot do using our brave analysis with $\forall\varepsilon\exists\eta\ldots$?
- 1988: a nice presentation by T. Lindstrom using a construction similar to that of $\mathbb{R}$ from $\mathbb{Q}$: makes NSA usable for normally designed brains

# Nonstandard Analysis for the computer scientist

## History:

- Infinitesimal numbers were considered by Archimède, Newton, Fermat, Leibniz, ... but lacked rigorous foundations. **Nonstandard Analysis (NSA)** was proposed by A. Robinson in 1961 as a conservative enhancement of Zermelo-Fränkel set theory (nice for the addicts)
- Used to reformulate some problems in maths (Algebraic topology, Stochastic processes and Brownian motion,...). Controversial: what does it do that you cannot do using our brave analysis with $\forall\varepsilon\exists\eta\ldots$?
- 1988: a nice presentation by T. Lindstrom using a construction similar to that of $\mathbb{R}$ from $\mathbb{Q}$: makes NSA usable for normally designed brains
- NSA appears to be very useful for computer scientists;
  it allows to get rid of the burden of numerical analysis in dealing with continuous time dynamics when designing compilers
  [PhD Bliudze 2006, Benveniste et al. 2010]

# Nonstandard Analysis for the computer scientist

## Nonstandard Analysis in a tweet:

- The set of nonstandard real numbers (hyperreals) ${}^\star\mathbb{R} \supset \mathbb{R} \cup \{-\infty, +\infty\}$ contains infinitesimal numbers ($\varepsilon : \forall x \in \mathbb{R}_{>0}, 0 < \varepsilon < x$) and infinitely large numbers
- Every nonstandard number $x \in {}^\star\mathbb{R}$ admits a unique decomposition $x = st(x) + \varepsilon$ into a standard part $st(x) \in \mathbb{R}$ and an infinitesimal part $\varepsilon$. Say $x \sim y$ iff $st(x - y) = 0$
- Relations ($\leq$), operations ($x+y$), functions ($e^x$) extend to ${}^\star\mathbb{R}$. Transfer principle: $\forall \psi$ first order formula: $\models_{\mathbb{R}} \psi \iff \models_{{}^\star\mathbb{R}} \psi$

# Nonstandard Analysis for the computer scientist

## Nonstandard Analysis in a tweet:

- The set of nonstandard real numbers (hyperreals) ${}^\star\mathbb{R} \supset \mathbb{R} \cup \{-\infty, +\infty\}$ contains infinitesimal numbers ($\varepsilon : \forall x \in \mathbb{R}_{>0}, 0 < \varepsilon < x$) and infinitely large numbers
- Every nonstandard number $x \in {}^\star\mathbb{R}$ admits a unique decomposition $x = st(x) + \varepsilon$ into a standard part $st(x) \in \mathbb{R}$ and an infinitesimal part $\varepsilon$. Say $x \sim y$ iff $st(x - y) = 0$
- Relations ($\leq$), operations ($x+y$), functions ($e^x$) extend to ${}^\star\mathbb{R}$. Transfer principle: $\forall \psi$ first order formula: $\models_{\mathbb{R}} \psi \iff \models_{{}^\star\mathbb{R}} \psi$

## Our use of it:

$\mathbb{T} = \{n\partial \mid n \in {}^\star\mathbb{N}\}$ and $\partial$ infinitesimal. $\mathbb{T}$ is both dense and discrete

Context: mix of discrete and continuous time dynamics

Causality issues in Hybrid Systems Modeling Languages

Requirements on Semantic Models of Time

Time Domains

Nonstandard Analysis in 2 slides

Nonstandard Semantics

Conclusion

# Nonstandard Semantics

$t^\bullet = t + \partial$ , ${}^\bullet x_t = x_{\bullet t}$ , $x_t^\bullet = x_{t\bullet}$ in:

| equation | semantics |
|---|---|
| der $x = e$;<br>init $x = f$ | initialization: $x_0 = [\![f]\!]_0$<br>and then $x^\bullet = x + \partial[\![e]\!]$ |
| der $x = e$;<br>init $x = a$;<br>when $x \geq 1$ do reinit $x = g$ | crossing event: $z = [{}^\bullet x < 1] \wedge [x \geq 1]$<br>initialization: $x_0 = [\![a]\!]_0$<br>and then $x^\bullet =$ if $z$ then $[\![g^\bullet]\!]$ else $x + \partial[\![e]\!]$ |

- Just as for Lustre
- Nonstandard semantics step-based $\Longrightarrow$ 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 (in contrast to synchronous languages): Standardization needed to get final code

# Nonstandard Semantics

$t^\bullet = t + \partial$ , ${}^\bullet x_t = x_{\bullet t}$ , $x_t^\bullet = x_{t\bullet}$ in:

| equation | semantics |
|---|---|
| der $x = e$;<br>init $x = f$ | initialization: $x_0 = [\![f]\!]_0$<br>and then $x^\bullet = x + \partial[\![e]\!]$ |
| der $x = e$;<br>init $x = a$;<br>when $x \geq 1$ do reinit $x = g$ | crossing event: $z = [{}^\bullet x < 1] \wedge [x \geq 1]$<br>initialization: $x_0 = [\![a]\!]_0$<br>and then $x^\bullet =$ if $z$ then $[\![g^\bullet]\!]$ else $x + \partial[\![e]\!]$ |

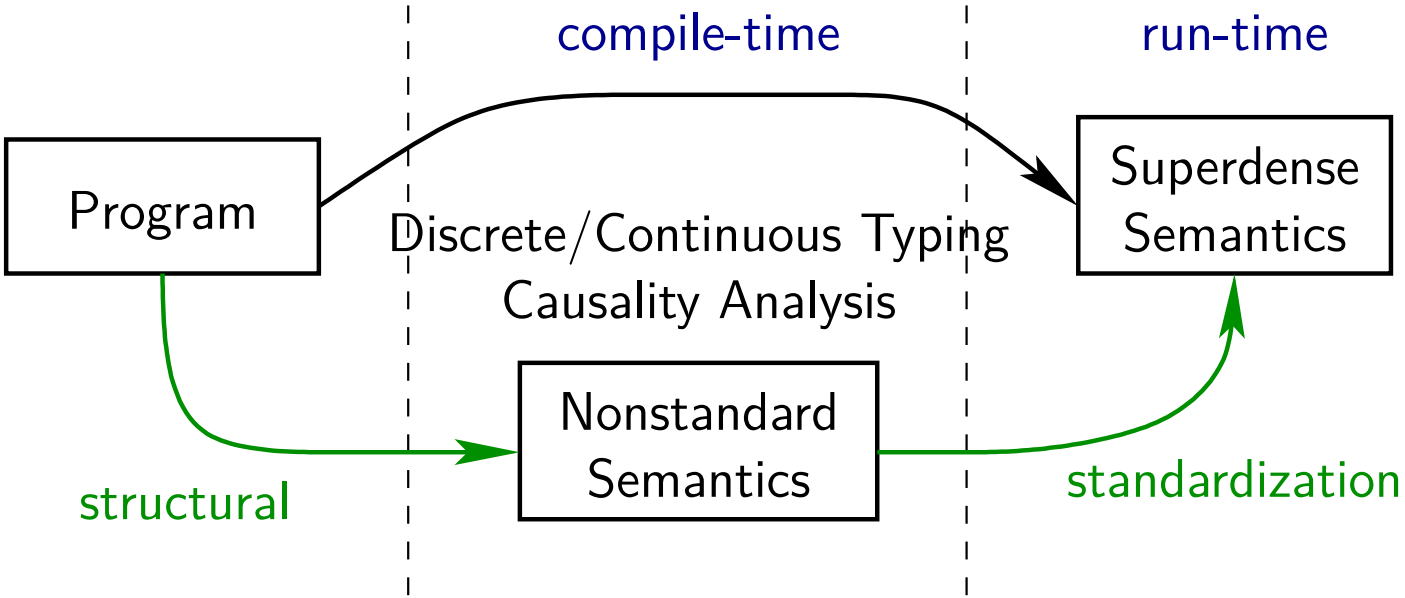
- Just as for Lustre
- Nonstandard semantics step-based $\Longrightarrow$ Constructive Semantics exists [Benveniste et al. 2012] (having ${}^\star\mathbb{N}$ many steps instead of $\mathbb{N}$ many ones is not an issue)
- Of course, this semantics can not be used for simulation (in contrast to synchronous languages): Standardization needed to get final code

# Standardization (Theorem [HSCC 2014])

1. The nonstandard semantics of every well-typed (wrt. cont/disc) and causally-correct program is standardizable;
2. Its standardization is independent of $\partial$ and
3. Continuous on every compact set of dates not containing:
   - an event, or
   - an undefined value ($\perp$)
4. This standardization coincides with the superdense time semantics when the latter is defined.

# Standardization (Theorem [HSCC 2014])

1. The nonstandard semantics of every well-typed (wrt. cont/disc) and causally-correct program is standardizable;
2. Its standardization is independent of $\partial$ and
3. Continuous on every compact set of dates not containing:
   - an event, or
   - an undefined value ($\perp$)
4. This standardization coincides with the superdense time semantics when the latter is defined.

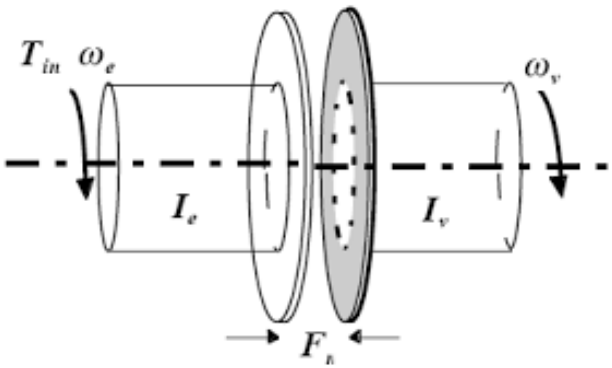


# Conclusion

- Superdense time: useful for simulation semantics of closed systems
- More is needed for supporting compilation:
  - Compositional semantics
  - Abstracting away from numerical issues
- Nonstandard time: good candidate
  - Compositional semantics; Abstracting away from numerical issues
  - Coincides with superdense semantics (when the latter is defined)
  - Ongoing work:
    - Chattering behavior
    - Multi-mode DAE systems
    - Impulsive systems

# Issues in multimode DAE systems

A simple clutch:




An unexpected issue:

- In this example, it is not even easy to guess what the restart value should be after the mode change
- The rotation speed after engagement depends on the individual rotation speeds before engagement
- One cannot expect the designer to specify this restart speed, it should be synthesized by the compiler

A new issue not considered so far...

**Would Oded fall in love with nonstandard semantics?**

**Well, I let you guess**