The non-standard Semantics of Hybrid Systems\textsuperscript{a}

Marc Pouzet
UPMC/ENS/IUF
Marc.Pouzet@ens.fr

Course notes.

\textsuperscript{a}Notes from the two papers [3, 2]
From Simulink/Stateflow\textsuperscript{a} — data-flow programming:

• hybrid systems, mode switching
• input/output oriented
• ODEs (Ordinary Differential Equations):

\[ x'(t) = f(y, t, x) \text{ with } x(t_0) = x_0 \]

To Modelica\textsuperscript{b} — component based modeling:

• Modeling from first principles
• No inputs, no outputs, but constraints
• DAEs (Differential Algebraic Equations):

\[ x'(t) = f(y, t, x) \text{ with } g(x, y) = 0 \]

\textsuperscript{a}http://www.mathworks.fr/products/simulink/index.html
\textsuperscript{b}https://modelica.org/
Some invariants, though:

- dataflow (Simulink) or relational (Modelica) models $\neq$ hybrid automata
- different tools yield different executions for the same model
- problems in handling zero-crossings and resets

Various tools/languages deal with mixed signals:

- VHDL-AMS, VERILOG-AMS, SystemC-AMS for mixing digital synchronous circuits and analog circuits.
- Simulink/Stateflow, Modelica, Scicos, PtolemyII, etc.
Why do these difficulties remain?

A lot of work on the formal verification of hybrid automata (read survey [6]) but only a few on programming language aspects. In particular:

- At runtime, zero-crossings can be:
  - simultaneous by accident
  - semantically (mathematically) identical

- How should we handle simultaneous zero-crossings that do not commute?

- At compile time, how discrete computations and continuous evolutions can be segregated?

- Simulation is extremely sensitive to the choice of a discretization scheme (fixed step, adaptative, ...)

- Further difficulties that cannot be fixed at compile time:
  - ODE/DAEs may not possess solutions; (non-)zenoness
  - cascaded zero-Xings; possible non-determinism for DAE
Some programming language questions

Objective: define a language which arbitrarily mixes discrete-time and continuous-time signals. E.g., say SCADE/Lustre + ODEs.

Key issues:

- What is the definition of a discrete signal?
  → produced by a zero-crossing. This includes all clocks bound to time (e.g., periodic)

- What should be the semantics of the parallel/hierarchical composition?

- Causality/clock calculus analysis to reject deadlocks. Static scheduling for efficient execution.
The Zeno paradox

“It is impossible to traverse an infinite number of things in a finite time” (Achille and the Tortule, Aristotle)

Definition 1 (Zeno behavior) A system has a Zeno behaviour if it includes an infinite number of discrete steps in a finite amount of time.

Example: bouncing ball.

Consider a bouncing ball throw with initial position 0 with loss $= 1/2$ at each bounce.

$$\dot{y} = y' \text{ init 1}$$
$$\dot{y}' = -g \text{ init 0 reset 0.5 \times last}(y') \text{ every up}( -y)$$

- The position converges to 0 with an infinite number of bounces.
- Take $g = -1$ to simplify. The ball bounces at time: $t_i$ with $t_{i+1} = t_i + \frac{1}{2^i}$ and $\lim_{n \to \infty} (t_i) = \sum_{i=0}^{\infty} \frac{1}{2^i} = \frac{1}{1-\frac{1}{2}} = 2$. 

 MPRI Systèmes Synchrones, Marc Pouzet

Non determinism

As for synchronous languages, non determinism is unavoidable. E.g., it depends on the way inputs are sampled. Here we add non determinism due to the approximation of continuous trajectories (integration method).

**Avoid internal non determinism:** all programming construct must have a deterministic semantics.

Leave non determinism to the responsibility of the numerical solver in its way of approximating continuous trajectories.

**Question:** Could we define an ideal semantics for hybrid systems allowing to reason “as if” the whole system was running synchronous on a discrete global scale?

**Intuition:** take an infinitely precise base clock

\[ BaseClock(\partial) = \{ n\partial \mid n \in \mathbb{N}^* \} \]

where \( \partial \) stands for an infinitesimal. The system step at instant 0, \( \partial, 2\partial, 3\partial, ... \)
Non-standard analysis: the key idea

\[ \dot{y} = x \]

\[ \uparrow \]

\[ y_{t+\partial} - y_t = \partial x_t \]

where \( \partial \) is infinitesimal
An introduction to non-standard analysis

Some milestones.

- Seminal work by Abraham Robinson [8] in 1961, then developed by a small community of mathematicians.

- Proposed as a conservative enhancement of Zermelo-Frānkel set theory; some fancy axioms and principles; beautiful for the addicts.

- Subject of controversies: what does it do for you that you cannot do using our good old analysis with $\forall \varepsilon \exists \eta \ldots$?

- 1988: a nice presentation of the topic by T. Lindstrom [7], kind of “non-standard analysis for the axiom-averse”.

- 2006: used in Simon Bliudze’s PhD [4] where he proposes the counterpart of a “Turing machine” for hybrid systems (supervised by Daniel Krob)

- Recently used by Benveniste et al. [3, 2] as a semantical basis for a synchronous language extended with ODEs.
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;

• $\star\mathbb{R}$ contains elements that are infinitesimal, $0 < \partial < t$ for any $t \in \mathbb{R}_+$.

• $\star\mathbb{R}$ should obey the same algebra as $\mathbb{R}$: order, $+$, $\times$, $\ldots$, $f : \mathbb{R} \mapsto \mathbb{R}$ extends to $\star f : \star\mathbb{R} \mapsto \star\mathbb{R}$, etc.

In the same way, $\star\mathbb{N}$ is the non-standard extension of $\mathbb{N}$. $\star\mathbb{N}$ contains elements that are infinitely large ($\star n > n$ for any $n \in \mathbb{N}$)

Idea:

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

  close to 0 : $\{ \frac{1}{\sqrt{n}} \} > \{ \frac{1}{n} \} > \{ \frac{1}{n^2} \} > 0$

  close to $+\infty$ : $\{ \sqrt{n} \} < \{ n \} < \{ n^2 \}$
Non-standard Analysis:

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 such that the sets \( \{ n \mid x_n > y_n \} \), \( \{ n \mid x_n < y_n \} \), \( \{ n \mid x_n = y_n \} \), are all infinite.
Non-Standard Analysis: the beautiful idea of Lindstrom

Pick $\mathcal{F}$ an 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 $\mathcal{F}$
- sets not belonging to $\mathcal{F}$ are declared *neglectible*

Existence of an ultrafilter for any infinite set follows from Zorn’s lemma, which is equivalent to the axiom of choice

Convergence of sequences $\{x_n\}$ of reals?

- Let $X$ be its set of limit points. Given $x \in X$, define $\mathbb{N}_x = \{n_k \mid x_{n_k} \to x\}$ Then, one and exactly one $\mathbb{N}_x \in \mathcal{F}$: *say that* $x_n \to x$

- *In this setting, all sequences converge!*
Non-Standard Analysis: the beautiful idea of Lindstrom

For any two sequences \( \{x_n\} \) and \( \{y_n\} \) exactly one among the sets \( \{n \mid x_n > y_n\} \), \( \{n \mid x_n < y_n\} \), \( \{n \mid x_n = y_n\} \), belongs to \( \mathcal{F} \).

- Thus, *sequences can always be compared.*

- In particular, if \( \{n \mid x_n = y_n\} \in \mathcal{F} \), say that \( \{x_n\} \sim \{y_n\} \), which implies \( x_n \to x \leftarrow y_n \)

Define:

\[ *\mathbb{R} = \mathbb{R}^\mathbb{N} / \sim \]

- elements of *\( \mathbb{R} \) are written \([x_n]\)

- By pointwise extension, *\( \mathbb{R} \) possesses the same algebra as \( \mathbb{R} \)

- Say that:

\[ x = st([x_n]) \text{ if } x_n \to x \]

\( x \) is called the standard part of \([x_n]\).
Integrals and differential equations

- internal functions and sets by pointwise extension:

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

- Pick an infinite number \( N \in \mathbb{N}^* \) and consider the set

\[ T = \{ 0, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N}, 1 \} \]

By definition, if \( N = [N_n] \) then \( T = [T_n] \) with

\[ T_n = \{ 0, \frac{1}{N_n}, \frac{2}{N_n}, \ldots, \frac{N_n-1}{N_n}, 1 \} \]

- For \( f : [0, 1] \mapsto \mathbb{R} \) a continuous function and \(*f = [f, f, \ldots] \) its non-standard version

\[ st \left( \left[ \sum_{t \in T_n} \frac{1}{N_n} f(t_n) \right] \right) = st \left( \sum_{t \in T} \frac{1}{N} *f(t) \right) = \int_0^1 f(t) \, dt \]
Integrals and differential equations

Similarly, for every $0 < t \leq 1$:

\[
\int_0^t f(u)du = st\left(\sum_{u \in T, u \leq t} \frac{1}{N} f(t)\right)
\]

Set $\partial = \frac{1}{N}$ and consider the ODE:

\[
\begin{align*}
\dot{x} &= f(x, t) \quad \text{with} \quad x(0) = x_0 \\
x(t) &= st \left( x_0 + \sum_{u \in T, u \leq t} \frac{1}{N} f(x(u), u) \right)
\end{align*}
\]

And thus we get, for the ODE, the following non-standard version:

\[
\begin{align*}
x(t_n) &= x(t_{n-1}) + \partial \times f(x(t_{n-1}), t_{n-1}) \\
x(t_0) &= x_0
\end{align*}
\]

(1) only holds if the ODE has a solution (2) is always defined as a non-standard dynamical system
Non-Standard hybrid Systems

The base clock: Fix an infinitesimal base step \( \partial \)

\[
BaseClock(\partial) = \{ n\partial \mid n \in {}^*\mathbb{N} \}
\]

is isomorphic to \( {}^*\mathbb{N} \) as a total order. For every \( t \in \mathbb{R}_+ \) and any \( \epsilon > 0 \), there exists \( t' \in BaseClock(\partial) \) such that \( |t' - t| < \epsilon \) expressing that \( BaseClock(\partial) \) is dense in \( \mathbb{R}_+ \).

\( BaseClock(\partial) \) is thus a natural candidate for a time index set and \( \partial \) is the corresponding time basis.

Previous/next instant: Let \( \mathbb{T} \) be a subset of \( BaseClock(\partial) \). For \( t \in \mathbb{T} \), define:

- \( \bullet t = \max\{ s \mid s \in \mathbb{T}, s < t \} \)
- \( t^\bullet = \min\{ s \mid s \in \mathbb{T}, s > t \} \)

For \( t = t_n = n\partial \in BaseClock(\partial), \bullet t = t_{n-1} \) and \( t^\bullet = t_{n+1} \)

This allows specifying and manipulating hybrid systems “as if” time was discrete and global, as in synchronous languages!
A minimal Hybrid formalism

A static single assignment language mixing data-flow equations and ODEs.

Equations:

\( Eq_1 : \ y = f([x]) \)

\( Eq_2 : \ y = \text{last}(x) \)

\( Eq_3 : \ z = \text{up}(x) \)

\( Eq_4 : \ \dot{y} = x \text{ init } y_0 \text{ reset } u \)

\( Eq_5 : \ u = [x] \text{ every } [z] \text{ init } u_0 \)

\( Eq_6 : \ y = \text{pre}(x) \text{ init } y_0 \)

System:

\[
Eq \ ::= \ Eq_1 \ | \ ... \ | \ Eq_6 \\
S \ ::= \ Eq \ | \ S || S
\]
Notations

Symbols $x, y, z, u, ...$ denote variables, taken from an underlying set $X$ of variables, and having respective domains $D_x, D_y$, etc.

$\dot{y}$ denotes a derivative.

$[x] = [x^1, ..., x^n]$ is a tuple of variables. Symbols $x_0, y_1$, etc. denote immediate values (e.g., 42, 1.5).

$z$ denotes a zero-crossing variable taken from a set $Z \subset X$ of clock variables.

Clock variables take their values from the set of all clocks, where a clock is any subset of $BaseClock(\partial)$. Equations $Eq_1 - Eq_6$ define dynamical systems, or, equivalently, sets of behaviors with time index set $BaseClock(\partial)$.

E.g., equation $y = x$ (form $Eq_1$, taking $f$ as the identity function) means $\forall t \in BaseClock(\partial) : y_t = x_t$.

Hybrid systems are specified via sets of equations of the form $Eq_1 - Eq_6$, taken conjunctively.
Discrete/non discrete signals

For each signal $x : T \rightarrow V$, we assume a clock $\tau_x$ such that $x$ is guaranteed constant on the complement of $\tau_x$:

- $\tau_x \subseteq T$
- If $x_t$ is defined then $x_{t'}$ is also defined for all $t' < t$.
- $x_t$ is defined for all $t \in \tau_x$ and $x_{t'} = x_{\bullet t'}$ for $t' \not\in \tau_x$.

We call $\tau_x$ the clock of $x$ and take the following convention:

A signal is termed discrete if it has been declared as such, or if its clock is a zero-crossing. Otherwise it is termed continuous.

E.g., $Eq_3$ defines a discrete clock. $Eq_5$ and $Eq_6$ define discrete signals.

A “discrete” signal is thus piece-wise constant. Note that this definition of clock is different from that of synchronous languages.

Here, the clock of $s$ indicates a possible change of the value of $s^a$.

---

*a*In Lustre, it would consist in writing `current x` every time $x$ is the result of a sampling.
Discrete and Continuous

Any system $S$ made by composing equations $Eq_1 - Eq_6$ must verify the following constraints:

1. An equation $Eq_2 (y = \text{last}(x))$ is well formed if the variable $x$ is defined by an equation of the form $Eq_4$ or $Eq_5$.

2. An equation $Eq_4 (\dot{y} = x \text{ init } x_0 \text{ reset } u)$ is well formed if the variable $z$ is defined by an equation of the form $Eq_5$.

3. An equation $Eq_6 (y = \text{pre}(x) \text{ init } y_0)$ is well formed if the variable $x$ is defined by an equation of the form $Eq_5$.

There extra constraints ensure that discrete computations (e.g., changing the value of a register) only occur at a zero-crossing instant.

Remark: These well formation rules are too restrictive for a practical language. This can be done in a better way with a type-system.
## Non-standard Semantics

<table>
<thead>
<tr>
<th>Statement</th>
<th>Non-standard semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = f([x])$</td>
<td>$y_t = f([x]_t)$</td>
</tr>
<tr>
<td>$y = \text{last}(x)$</td>
<td>$y_t = x \cdot t$</td>
</tr>
<tr>
<td>$y = \dot{x}$</td>
<td>$y_t = \frac{1}{\partial} (x_t - x \cdot t)$</td>
</tr>
<tr>
<td>$z = \text{up}(x)$</td>
<td>$z_t \cdot = [x \cdot t &lt; 0] \land [x_t \geq 0]$</td>
</tr>
<tr>
<td>$\dot{y} = x \text{ init } y_0 \text{ reset } u$</td>
<td>$\tau_u \text{ discrete}$</td>
</tr>
<tr>
<td></td>
<td>$t \in \tau_u \Rightarrow y_t = u_t$</td>
</tr>
<tr>
<td></td>
<td>$t \not\in \tau_u \Rightarrow y_t = y \cdot t + \partial x \cdot t$</td>
</tr>
<tr>
<td>$u = [x] \text{ every } [z] \text{ init } u_0$</td>
<td>$\tau_u = \bigcup_i z^i$</td>
</tr>
<tr>
<td></td>
<td>$t &lt; \min(\bigcup_i z^i) \Rightarrow u_t = x_0$</td>
</tr>
<tr>
<td></td>
<td>$t \in z^i \setminus (\bigcup_{j&lt;i} z^j) \Rightarrow u_t = x_t^i$</td>
</tr>
<tr>
<td>$y = \text{pre}(x) \text{ init } y_0$</td>
<td>$\tau_y = \tau_x \text{ discrete}$</td>
</tr>
<tr>
<td></td>
<td>$t &lt; \min(\tau_y) \Rightarrow y_t = y_0$</td>
</tr>
<tr>
<td></td>
<td>$t \in \tau_y \Rightarrow y_t = x \cdot t \quad \tau_y = \tau_x$</td>
</tr>
</tbody>
</table>
Examples

We give three examples with ODEs and reset. They are written in a prototype language called Zelus.

Example 1: reset an integrator on a zero-crossing event

```plaintext
let hybrid main () =
    let rec der x = 1.0 init -1.0
        and der y = 0.0 init -1.0 reset 1.0 every up(x) in
    (x,y)
```

![Diagram of the example](image-url)
Example 2: Unbounded cascades of zero-crossing

let hybrid main () =
  let rec der x = 0.0 init -1.0
      reset -. 1.0 every up(y) | 1.0 every up(-. y) | 1.0 every up(z)
  and der y = 0.0 init -1.0
      reset 1.0 every up(x) | -1.0 every up(-. x)
  and der z = 1.0 init -1.0 in
  (x,y,z)

• $\varepsilon$ represent a “very small” step size in that finitely many $\varepsilon$’s sum up to $\approx 0$.

• At $t = 1$, $x$ and $y$ starts an infinite cascade of zero-crossing while time remains blocked. This is certainly pathological.

MPRI Systèmes Synchrone, Marc Pouzet
October 31, 2012, page 23/28
Example 3: Sliding mode control

let hybrid main (y0) =

let rec der x = 0.0 init -. sgn(y0) reset -1.0 every up(y)

| 1.0 every up(-. y)

and der y = x init y0 in

y
• Suppose that $y_0 < 0.0$. Thus $x_0 > 0$.

• $y$ increases at constant speed until its first zero-crossing, just after $t = |y_0|$.

• Then, $y$ chatters infinitesimally around 0 as its speed alternate between $-1$ and $+1$ with infinitesimal step $\varepsilon$.

• This example is not pathological. It is equivalent to:

```plaintext
let hybrid main (y0) =
  let rec der y = z init y0
  and der x = 0.0 init -. sgn(y0) reset 0 every up(y) in
  y
```

Unbounded cascades of zero-crossings:

• Time did not progress in example (2) while the very same zero-crossing condition has been taken twice.

• Is-it a run-time error? What about a causality analysis which accept programs (1) and (3) but reject program (2)?
Conclusion/open problems

This is a shortcut. Read [2] carefully.

What non-standard semantics is good for:

- Reason “as if” time was discrete and global, considering the base clock as infinitely precise.
- Define an ideal semantics; reason about systems and their equivalence; deal with cascades of zero-crossings.
- Do not pollute the semantics with extra conditions (e.g., lipshitz, non-zenoness)

Limitations/open questions:

- Static typing: can we/shoud we ensure by typing that only integration occurs during continuous steps. E.g., reject programs such as:

  \[ x = (\text{last}(x) + 1) \text{ init } 0 \]

  This program sticks. Well formation rules reject such programs but are too syntactical. A more general system is proposed in [1].

- Can we detect simple zeno effects (typically unbounded discrete cascades)?
• Or instrument the code or program more robust zero-crossing detection (e.g., with an hysteresis)?

• What would be a Clock calculus for Hybrid systems with a proper separation of discrete and piece-wise continuous signals?

• Non-standard semantics does not account for rounding errors and the fact that numerical solvers do approximation. E.g., the two signals $x$ and $y$ are considered equivalent:

\[
\begin{align*}
\dot{p} &= 1 \text{ init } -1 \text{ reset } -1 \text{ every } \text{ up}(p - 1) \\
\dot{x} &= 1 \text{ init } -1 \\
\dot{y} &= 1 \text{ init } -1 \text{ reset } \text{ last}(y) \text{ every } \text{ up}(p)
\end{align*}
\]

But this is not necessarily true in practice. In particular, resetting a solver clears its internal memory and modifies its accuracy for detecting zero-crossing events.

• The solver games with the system, choosing on its own a subset of the base clock for computing intermediate values.

• Can we account for that, still with a synchronous data-flow semantics?
References


