An Operational Semantics for Hybrid Systems Involving Behavioral Abstraction

Simon Bliudze
École Polytechnique Fédérale de Lausanne
INJ Building, Station 14
1015 Lausanne, Switzerland
simon.bliudze@epfl.ch

Sébastien Furic
LMS Imagine S.A.
7 place des Minimes
42300 Roanne, France
sebastien.furic@lmsintl.com

Abstract

We discuss the challenges of building a simulation framework for hybrid systems, in particular the well-known Zeno effect and correct composition of models idealised by abstracting irrelevant behavioural details (e.g. the bounce dynamics of a bouncing ball or the process of fuse melting in an electrical circuit). We argue that the cornerstone of addressing these challenges is the definition of a semantic framework with an appropriate underlying model of time.

Using two simple examples, we illustrate the properties of such a model and explain why existing models are not sufficient. Finally, we propose a new Zeno-free semantic model that allows mixing discrete and continuous behaviour in a rigorous way and provides for the compositional behavioural abstraction.

Although it is based on non-standard analysis, we explain how our semantic model can be used to develop hybrid system simulators.

Keywords: Hybrid Modeling Languages; Non-Standard Analysis; Models of Signals; Behavioral Abstraction; Operational Semantics

1 Introduction

A large number of modelling, verification and simulation frameworks for hybrid systems have been designed in the past years. Although, a complete overview is beyond the scope of our paper, we observe that they broadly fall in two categories: those that put special emphasis on a rigorous model definition, such as, for instance, the Ptolemy project [6] (based on [14]), the Zélus synchronous language [3] (based on the semantics in [1]) and SpaceEx [7]; and those that have chosen a more pragmatic, informal approach, such as the Modelica language [8] and the associated tools, and the Scicos block-diagram modeller and simulator [4].

All the associated tools share the same basic model of execution alternating between continuous phases and sequences of 'run-to-completion' discrete actions [3] as formalised by the notion of hybrid automata [11]. None of these approaches attempts to include the operational semantics of differential equations in their core semantic model: execution of the continuous phases is delegated to numerical solvers, which are used to advance physical time and compute the values of physical signals.

Except for Zélus, none of the above semantic models are Zeno-free, which means that, as explained in the next sections, they do not reflect the fact that time diverges and rely on analysing the solver output to detect and advance past the Zeno points [13] (cf. Section 2). This poses a fundamental problem, since the solver behaviour at this point is usually unspecified.

Furthermore, none of the above proposals allows *compositional behavioural abstraction*: idealised models do not account for the physical nature of phenomena, in particular the fact that original, *high-fidelity* signals are *continuous*. However, this property is assumed by most users and validated by most real-life systems. Hence, it must be a fundamental property of signals and should be reflected in their idealisations.

In our view, to achieve maximum robustness of a simulation framework, it is crucial to define the semantic model before designing either the language or the simulator. Thus, the design of a hybrid simulation framework should involve the following steps.

First of all, one must define a semantic model that properly accounts for the expected elementary properties of systems to be simulated. This includes dynamic behaviour properties, but also "higher level" ones, such as modularity.

The second step consists in designing a simulator

capable of computing acceptable approximations of the dynamics of models conforming to the semantic model above. Successful completion of this step validates the semantic model by showing that all conforming models can be simulated.

The third step involves the design of a language powerful enough to express a useful subset of models that provably conform to the semantic model. In order to avoid errors at simulation time or, worse, spurious simulation results, the semantic validity of a model must be provable statically, e.g. by type-checking.

Finally, once both the semantic model and the language have been defined, one has to design a compiler for this language, which would perform the necessary static checks and reject the invalid models.

In this paper, we focus on the first step above, i.e. defining the appropriate semantic model. Indeed, in our opinion, current semantic models still lack some of the properties required by real-life systems.

In particular, modularity is an extremely important property for both correctness and practical usability of hybrid simulation frameworks. One of us has already discussed some modularity issues in Modelica [9]. In the present paper, we focus on another—probably the most important—aspect contributing to modularity, which is *compositional behavioural abstraction*.

Realistic hybrid models almost always require part of the physical behaviour to be abstracted by means of "ideal equations"—such as conditional and reset equations—that typically yield discontinuities in physical signals. Below, we will use the term *behavioural abstraction* more generally to designate any mechanism that enables concrete physical behaviour to be "hidden" by considering *idealised* models. Intuitively, from the point of view of an external observer, behavioural abstraction makes the model "jump" over instants corresponding to activations of abstraction mechanisms. Thus, idealisation consists in explicitly providing a constraint to be met *after* these instants.

It is well known from physical system modelling experts that, although extremely useful in practice, behavioural abstraction often leads to inconsistencies and even to singularities in simulation code, especially when abstractions are *composed*. Informally, we say that behavioural abstraction is *compositional* if substituting an ideal sub-model—instead of a high-fidelity one—within a larger model does not introduce new behaviours that were not present in the original model (a counter-example is given in Section 3.1).

The cornerstone of a semantic framework that would address the above issues is the underlying model of time.

Traditional definition of the simulation time-line either relies on a fixed basic step, provided as a simulation parameter, or a variable step adjusted based on events detected during the simulation. The former has the advantage of being Zeno-free by construction, since at each simulation step the time advances by a constant value, but produces wrong simulation results whenever the signal activity is higher than the fixed sampling frequency. Variable step simulation, on the other hand, improves the simulation precision by sacrificing Zeno-freeness. However, in both cases, the semantics of a model is only defined at simulation time depending on the choice of time steps. Hence, model validity cannot be statically proven.

In Section 3.1, we argue that, to ensure compositionality of behavioural abstraction and reflect the intrinsic continuity of physical phenomena, the time model must be *densely ordered*.[1] Indeed, as a result of behavioural abstraction, several causally dependent events happen at the same instant. Traditional time models above do not provide any possibility to preserve the causality information in the semantic model, leading to spurious behaviours.

The *super-dense* time approach [14] addresses this problem by defining time as a subset of the cartesian product $\mathbb{R} \times \mathbb{N}_0$. However, solutions of differential equations are obtained by means of standard operational semantics. Thus, although causality between instantaneous events can be preserved, the problems related to the Zeno effect persist.

Recent use of Non-Standard Analysis [12] in the design of operational semantics for hybrid systems [1, 2, 16] has led to the definition of a *linear* time that is 1) *discrete*, i.e. instants can be considered in isolation; 2) *well-ordered*, i.e. for each time instant, there is a uniquely defined *next instant* respecting the usual temporal order; and 3) it can be treated as a *continuum*. These properties make it suitable to express both discrete and continuous dynamics in a *unified* fashion and avoid the Zeno effects. However, as a consequence, such time model cannot be densely ordered. Indeed, it features *consecutive instants*, which, by definition, do not have any instants between them.

In this paper, we propose a new, Zeno-free semantic model also based on a non-standard model of time, which allows mixing discrete and continuous behaviour in a rigorous way. In addition, this new model also allows compositional behavioural abstrac-

[1] A partially ordered set is said to be densely ordered if for all elements x and y for which $x < y$ there exists a z such that $x < z < y$.

DOI 10.3384/ECP14096693

tion: density of time is reestablished so that physical signals remain continuous even when idealised, which avoids the emergence of "impossible behaviours".

The key idea behind our proposal is to discretise the *value* of signals instead of time. Discretisation is necessary in order to associate *dates* with events [1, 2, 16]. However, discretising signal values allows us to choose these dates among all non-standard reals, rather than fixing a regular time-line in advance. Thus, we consider discrete signal *activity* in the frame of a densely-ordered time-line compatible with the requirements of compositional behavioural abstraction. This approach has something in common with [10], where well-ordered time scales are constructed as subsets of a common densely-ordered time reference.

The paper is structured as follows. In Section 2, we explain the need for an operational semantics that can cope with the Zeno paradox, and how non-standard analysis can be used to define such a semantics. In Section 3, we show why previous proposals based on non-standard analysis do not ensure compositional behavioural abstraction, by taking an example involving the *composition* of several abstractions. Finally, in Section 4, we present our proposal, also based on non-standard analysis, but where discretisation is no longer applied to the time-line but to signal values, leading to a more intuitive definition of time that, moreover, allows compositional behavioural abstraction.

2 Why non-standard semantics?

2.1 Behavioural abstraction example

The famous bouncing ball model, whereof a Modelica implementation is given in Listing 1, is probably one of the simplest models involving behavioural abstraction. In this model, the physics of the bounce have been abstracted away: we consider that, at bounce time, the velocity, which was negative just before the ball hits the ground, *instantaneously* becomes positive, with a magnitude decreased by 20%. Figure 1 shows the simulation results for `x` in the $[0, 1.1]$ time interval.

Despite its apparent simplicity, this model poses severe challenges to language theorists as we shall see.

2.2 The Zeno effect

We suppose here that physical time is based on standard reals (i.e. $\mathbb{R}$). How does our model behave with such a time model?

Let's introduce the following notations:

Listing 1: A bouncing ball model, in Modelica.

```
model BouncingBall
  Real v, x;
  constant Real g = 10;
initial equation
  v = 1.0;
  x = 0.0;
equation
  der(v) = -g;
  der(x) = v;
  when x < 0 then
    reinit(v, -0.8 * pre(v));
    reinit(x, 0.0);
  end when;
end BouncingBall;
```

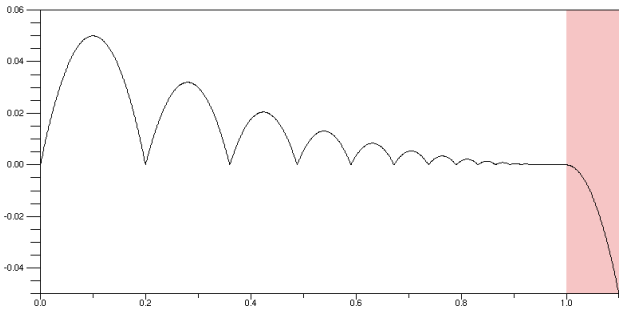

Figure 1: Simulation results for the bouncing ball model with the DASKR solver (altitude of the ball with respect to time)

- t_i, for $i \in \mathbb{N}_0$, denotes the initial instant of the i^{th} flight of the ball;
- v_i, for $i \in \mathbb{N}_0$, denotes the (positive) velocity of the ball at t_i.

According to the model, the trajectory of the ball in-flight has to verify:

$$\dot{v} = -g \tag{1a}$$

$$\dot{x} = v \tag{1b}$$

Equations (1a) and (1b) can be solved analytically for v and x, for $t \in [t_i, t_{i+1})$, giving:

$$v(t) = -g \cdot (t - t_i) + v_i \tag{2a}$$

$$x(t) = -\frac{1}{2} g \cdot (t - t_i)^2 + v_i \cdot (t - t_i) \tag{2b}$$

Since at t_{i+1} the ball hits the ground (that is, $x = 0$), we must have:

$$0 = -\frac{1}{2} g \cdot (t_{i+1} - t_i)^2 + v_i \cdot (t_{i+1} - t_i) \tag{3}$$

From (3) we deduce the duration of the i^{th} flight (notice that it only depends on the velocity of the ball at t_i):

$$t_{i+1} - t_i = \frac{2}{g} v_i \tag{4}$$

Also, from (2a) and (4), we deduce that:

$$\lim_{\substack{t \to t_{i+1} \\ t < t_{i+1}}} v(t) = -v_i \tag{5}$$

It follows that:

$$v_{i+1} = 0.8 v_i \tag{6}$$

From (4), (5) and (6), we deduce the total duration of all flights until the n^{th} bounce, for any $n \geq 0$:

$$\begin{aligned} t_n - t_0 &= \sum_{i=0}^{n-1} t_{i+1} - t_i = \frac{2}{g} \sum_{i=0}^{n-1} v_i = \frac{2v_0}{g} \sum_{i=0}^{n-1} 0.8^i \\ &= \frac{2v_0}{g} \cdot \frac{1 - 0.8^n}{1 - 0.8} \end{aligned} \tag{7}$$

Finally, we conclude from (7) that:

$$\lim_{n \to \infty} t_n - t_0 = \frac{10 v_0}{g} = 1 \tag{8}$$

In other terms time *converges*, meaning that the lifespan of the model is finite! This unexpected result is a manifestation of the *Zeno effect* which prevents the model from moving forward in time past the convergence limit, called the *Zeno point*. As a consequence, simulating a physical model past its Zeno point—which solvers kindly accept to do, as illustrated on Figure 1—means that we are no longer executing the alleged semantics of our favourite modelling language: we are observing a *free and necessarily wrong interpretation* of our program by the simulation tool. This is extremely embarrassing since Zeno effects cannot be spotted during simulation and since many practical models contain *abstraction mechanisms* such as Modelica's *reinit* operator, conditional equations, etc., whose composition is known to yield such issues: how to prove that we are not actually running a model out of its actual time domain?

2.3 Discussion of the example

Physical system theorists may argue that the issue with our bouncing ball model comes from the usage of inappropriate abstraction mechanisms. Indeed, application of a classical energetic approach (such as the standard Bond Graph for instance) instead of the more direct equation-based approach would have naturally lead to a time-diverging model: from this point of view, a bouncing ball model is no more than a damped *oscillator*, whose solution exhibits an exponential decay as time diverges.

However, in practice, the price to pay to benefit from the virtues of energetic modelling is often too high in terms of model complexity. Applied to our example for instance, such an approach would force us to express contact with the ground in a more detailed fashion (typically, by means of modulated C and R elements in Bond Graph, which poses the additional problem of finding an adequate modulation constraint). Quite often, we do not *know enough* of the phenomena to be able to write meaningful high-fidelity equations.

Furthermore, simulation performance may suffer dramatically from excess modelling details. For instance, if we simulate our original bouncing ball model with a solver based on the Trapezoidal rule (which is an order 2 method) we see that it performs very well (this is not very surprising since the flight trajectory is a parabola). On the other hand, performance dramatically collapses when the same solver is given a detailed version of the bouncing ball model. Interestingly, a closer look at the performance profile reveals that most of the CPU time is spent in solving the contribution of additional details of the high-fidelity model, which correspond precisely to the phenomenon we wanted to abstract away! Experimenting with real-world models brings us to similar conclusions: abstraction mechanisms help focusing on important parts of models, making the result often simpler and more efficient (but error-prone).

So it seems that we have to live with "dirty" abstractions: does it mean that we also have to accept unsound semantics for the sake of performance and simplicity? Fortunately no, as shown in the next sections: remember that our conclusions follow the initial premise assuming a time model based on $\mathbb{R}$.

2.4 Non-standard approach

A closer look at (7), which actually represents the *work* performed by our bouncing ball model in the course of the simulation, reveals the profound cause of our problem: we are assuming that *any* countable family of joined, nonempty sub-intervals (our $[t_i, t_{i+1})$) eventually partitions any simulation interval. Of course, this property does not hold: it would be equivalent to assuming that the sum of the terms of *any* sequence of positive reals would certainly diverge.

How could it be possible to define an operational semantics that would ensure time divergence even in presence of behavioural abstractions? Let us make the following experiment: We successively apply the fixed-step forward Euler method (see Listing 2) to our bouncing ball model with decreasing step sizes, as shown in Figure 2. We observe progressively better

DOI 10.3384/ECP14096693

approximations of the solution, since the problem is stable and since we can find a step size that ensures stability of the method itself. Notice that since we have chosen a fixed-step scheme, the process of simulation simply cannot exhibit Zeno effects: it terminates after at most $\lceil \frac{t_{end}-t_0}{h} \rceil$ steps, where h is the step size.

So why not define step-by-step calculation of the *ideal* trajectories generated by models—which is what operational semantics of hybrid systems is all about—in terms of the standard forward Euler method? The reason is that whatever the step size we choose this method only gives *approximations* of the trajectories of interest. Furthermore, as small as a candidate step size would be, it would always be possible to forge models that would require a smaller one, for instance, linear models having eigenvalues large enough to make the solution unstable and divergent (the first slope on Figure 2 shows the behaviour of the method in such a situation).

Notice also that this reasoning applies to *any* numerical integration method, not only forward Euler. The point is: numerical methods have to perform steps and there is no smallest possible step that would fit *all* models.

In order to define our reference calculation steps, we would have to choose a step that would be smaller than any positive real number. Furthermore, this reference step would have to be *positive* to ensure time divergence. This is precisely the definition of a positive *infinitesimal*, as these have been used in 17th and 18th centuries by mathematicians and physicists such as Leibniz and Newton.

The common idea of real numbers was different from the modern one. For instance, to compute the derivative of a given function $f(x)$, one would consider the *increment* of this function given an infinitesimal increment dx to x. Thus the derivative $f'(x)$ was defined by setting

$$f'(x) = \frac{f(x+dx) - f(x)}{dx}.$$

For example, applying this reasoning to $f(x) = x^2$, one obtains the following computation

$$f'(x) = \frac{(x+dx)^2 - x^2}{dx} = \frac{2x\,dx + dx^2}{dx} = 2x + dx \approx 2x,$$

where the last relation signifies that dx, being infinitesimal, *vanishes* in the final expression.

The notion of infinitesimal was formalised in 1960s by Robinson (see [15]), by defining a set ${}^*\mathbb{R}$ of *non-standard reals*, which is an ordered field extension of

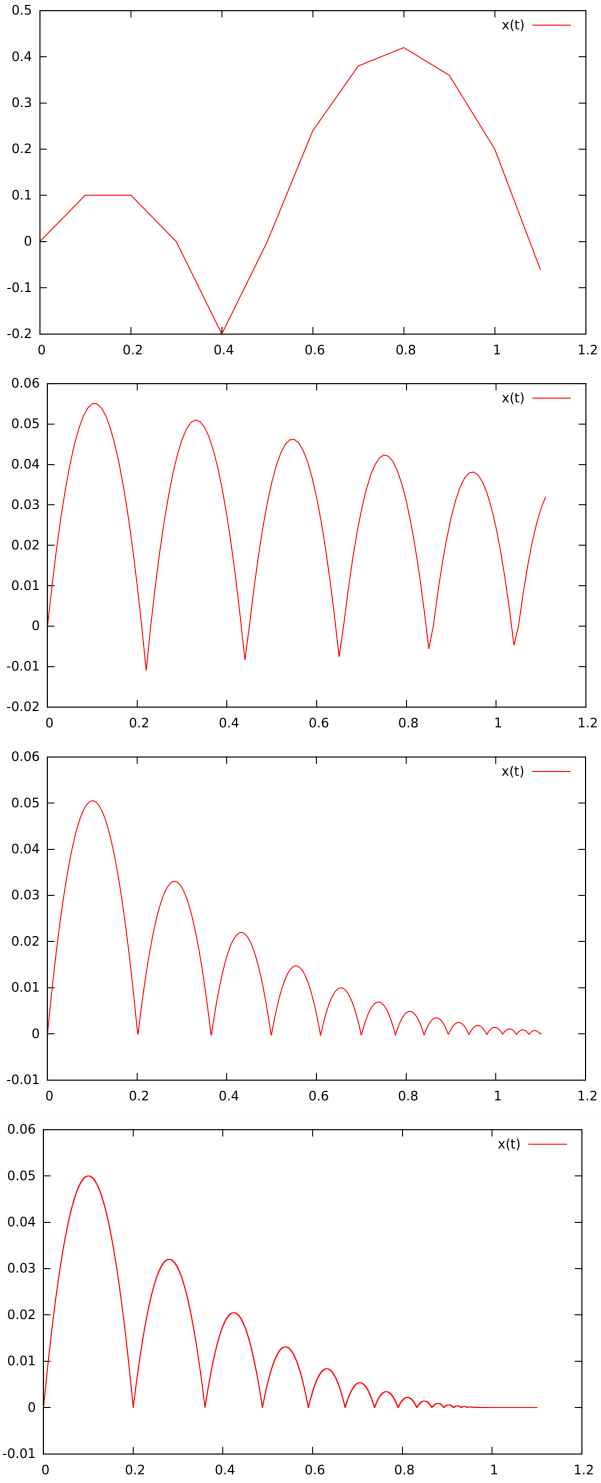

Figure 2: Simulation results for the bouncing ball model using forward Euler on $[0, 1.1]$ with step sizes 0.1, 0.01, 10^{-3} and 10^{-6} respectively (altitude of the ball).

$\mathbb{R}$ that contains all usual real numbers, but also the *infinitesimal* reals and their inverses, which are the *infinitely great* reals, i.e. those with an absolute value strictly greater than any usual real number $r \in \mathbb{R}$. One also speaks of *finite* (or *limited*) reals x, such that $|x| < y$ for *some* positive $y \in \mathbb{R}$;

In particular, two non-standard real numbers x and y are said to be *infinitely close* (denoted by $x \approx y$) if and only if $x - y$ is infinitesimal.

Among all non-standard real numbers, one can, of course, consider the set ${}^*\mathbb{Z}$ of *non-standard integers* that, on top of standard integers, contains infinitely great ones, having absolute value greater than any $n \in \mathbb{N}$. Similarly, it is possible to define the set of non-standard natural numbers ${}^*\mathbb{N}_0$ (including zero).

The field ${}^*\mathbb{R}$ of non-standard reals has been used by several authors to define operational semantics of continuous and hybrid systems [1, 2, 12, 16]. The key idea is to *change the definition of time*, by replacing standard reals with a subset of *non-standard* ones.

The proposal in [2, 16] consists in defining time as:

$$\mathbb{T} \stackrel{def}{=} \{\varepsilon \cdot n \mid n \in {}^*\mathbb{N}_0\} \tag{9}$$

for *some* reference positive infinitesimal ε (notice that *any* positive infinitesimal fits our constraints). It should be noted that $\mathbb{T}$ contains arbitrary large numbers: in particular, it is possible, given any positive real t, to find $n \in {}^*\mathbb{N}_0$ such that $\varepsilon \cdot n$ is greater than t. This property is due to ${}^*\mathbb{R}$ being Archimedean in the non-standard sense[2], as shown in [2]. Also, it should be noted that although $\mathbb{T}$ may contain absolutely no standard real (for some "unfortunate" choice of ε), it is possible to *approximate* any standard real x by a non-standard real of the form $\varepsilon \cdot n$ such that:

$$|x - \varepsilon \cdot n| < \varepsilon$$

which means that the error committed in the approximation is *negligible* in comparison to any positive element of $\mathbb{R}$ (recall that ε is a positive *infinitesimal*).

We are now ready to define the meaning of a differential equation:

$$\dot{x} = f(x,t) \quad \stackrel{def}{\equiv} \quad x_{next} = x + \varepsilon \cdot f(x,t) \tag{10}$$

where:

- ε is the reference time step introduced in (9);
- x_{next} is the "next" value of x, that is $x(t+\varepsilon)$.

[2]It can be shown, however, that ${}^*\mathbb{R}$ is *not* Archimedean in the standard sense, because of the existence of infinite elements.

Listing 2: A Haskell program implementing the forward Euler method to the bouncing ball model.

```
euler t0 tEnd (v0, x0) h = step 0 (v0, x0)
  where
    step i (vNow, xNow)
      | tNow > tEnd = []
      | otherwise   =
          (tNow, xNow) : step (i + 1) (vNext, xNext)
      where
        tNow           = t0 + h * fromInteger i
        (vNext, xNext) = advance tNow (vNow, xNow)
    advance _ (vNow, xNow)
      | xNow < 0.0 = (-0.8 * vNow, 0.0)
      | otherwise  = (vNow - h * 10.0, xNow + h * vNow)
```

Why would an operational semantics based on (9) and (10) prevent Zeno effects? For exactly the same reason why standard forward Euler method prevents them: because time is forced to advance by fixed—although infinitesimal—steps and because the non-standard Archimedean property of ${}^*\mathbb{R}$ allows arbitrarily large times to be overstepped.

However, we are not completely done yet. Indeed, due to the multiplication by ε in (10), our operational semantics maps time instants as well as values of real signals to *non-standard* reals, although we would like our models to eventually yield standard real values. Fortunately, any limited non-standard real can be *uniquely* represented as the sum of their standard and infinitesimal parts [2, 12, 16]. All we need to do is to eventually discard infinitesimal parts of non-standard reals in order to construct our final standard signals.

Operational semantics based on this idea lead to *uniform* treatment of discrete and continuous dynamics: differential equations, reset equations (e.g. Modelica's *reinit* operator) and difference equations are actually all treated as non-standard difference equations.[3] As an illustration of this, the program of Listing 2 implements the operational semantics of our bouncing ball model, provided

- `v0` and `x0` are standard reals and `h` is an arbitrary infinitesimal;
- integers (used as step indexes) are non-standard;
- infinitesimal parts of the output are discarded.

Notice that, past $t = 1$, as experiments with standard Euler method suggest, the ball sticks to the ground—more exactly, standardisation of the model's variables

[3]It is interesting to contrast this uniformity with Modelica's informal semantics as given in the current language specification.

DOI 10.3384/ECP14096693

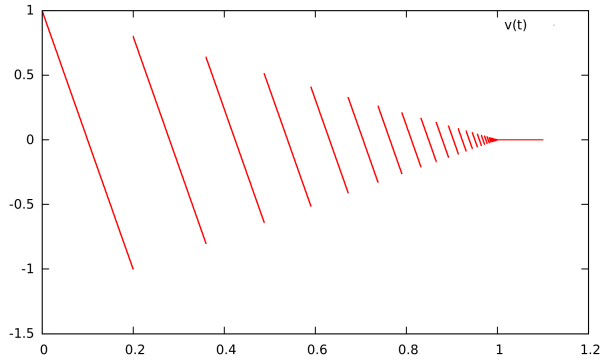

Figure 3: Velocity of the ball on $[0, 1.1]$ resulting from interpretation of the bouncing ball model with non-standard semantics.

yields two standard functions of time which map every $t \geq 1$ with zero. Indeed, from (6) we know that the standard part of v converges to zero as t converges to 1. It follows that for every $t \geq 1$, dynamic equations of the model, as a consequence of (10), only produce infinitesimal values since the product of ε by any limited real yields an infinitesimal number.

3 Behavioural abstraction issues

As shown in [1, 2], operational semantics built on time model (15) successfully explain the behaviour of programs like the bouncing ball model of Listing 1—even beyond $t = 1$. In particular, abstraction of the bounce phenomenon meets our expectations: velocity and altitude *instantaneously* take an explicitly specified value at bounce time and then *continuously* evolve from this new starting point until the next bounce.

It should be noted that, in this model, while altitude keeps its continuous character around bounce instants, velocity *looses* it, as illustrated in Figure 3. Nevertheless, interpretation of the bouncing ball model remains satisfactory with respect to the requirements which only demand a high-fidelity behaviour between bounces and a correct (and instantaneous) repositioning of velocity and altitude at bounce time for the next flight to start.

However, as shown below, loss of continuity due to abstraction may cause practical models of systems to fail unexpectedly.

3.1 A problematic example

We consider in Listing 3 a simple fuse sub-model, which behaves like an electrical switch that is closed by default but that can eventually become open if the branch current exceeds a limit.

Listing 3: A fuse sub-model, in Modelica.

```
import Modelica.Electrical.Analog.*;

model Fuse
  extends Interfaces.OnePort;
  parameter Real iMax;
  parameter Real Ron;
  parameter Real Roff;
protected Real R;
protected Boolean on;
initial equation
  on = true;
equation
  when i > iMax then
    on = false;
  end when;
  R = if on then Ron else Roff;
  v = R * i;
end Fuse;
```

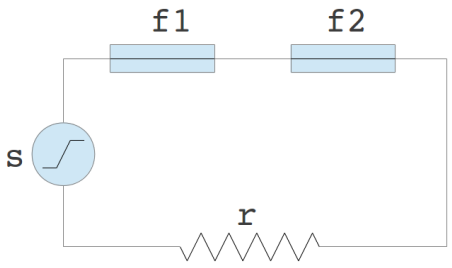

Figure 4: An electrical circuit using two instances of the fuse sub-model defined in Listing 3.

Listing 4: A ramp voltage source sub-model, in Modelica.

```
import Modelica.Electrical.Analog.*;

model RampVoltageSource
  extends Interfaces.OnePort;
  parameter Real startTime;
  parameter Real k;
  parameter Real vMax;
equation
  v =
    if time >= startTime then
      min(k * (time - startTime), vMax)
    else 0.0;
 end RampVoltageSource;
```

Using this sub-model, we build a model of the system shown in Figure 4, which is composed of a ramp voltage source (defined in Listing 4), two fuses and a simple linear resistor, in series. We suppose in this model that the fuses have *distinct* rated currents. Notice that, according to the definition of the fuse sub-model, the time required by a fuse to break the circuit is negligible with respect to the time required by the ramp voltage source to reach its maximum value[4].

Suppose we are given these parameter bindings:

$$\texttt{src.startTime} = 0.1 \tag{11a}$$
$$\texttt{src.k} = 2 \tag{11b}$$
$$\texttt{src.vMax} = 1 \tag{11c}$$
$$\texttt{f1.iMax} = 0.005 \tag{11d}$$
$$\texttt{f1.Ron} = 10^{-6} \tag{11e}$$
$$\texttt{f1.Roff} = 10^{6} \tag{11f}$$
$$\texttt{f2.iMax} = 0.006 \tag{11g}$$
$$\texttt{f2.Ron} = 10^{-6} \tag{11h}$$
$$\texttt{f2.Roff} = 10^{6} \tag{11i}$$
$$\texttt{r.R} = 100 \tag{11j}$$

and this initial state:

$$\texttt{time} = 0 \tag{12a}$$
$$\texttt{f1.on} = \mathsf{T} \tag{12b}$$
$$\texttt{f2.on} = \mathsf{T} \tag{12c}$$

According to our non-standard semantics, we have to solve the following dynamic equations to determine the dynamic behaviour of our model:

$$\texttt{src.v} = \begin{cases} \min(2(\texttt{time} - 0.1), 1) & \text{if } \texttt{time} \geq 0.1 \\ 0 & \text{otherwise} \end{cases} \tag{13a}$$
$$\texttt{src.i} = i \tag{13b}$$
$$\texttt{f1.R} = \begin{cases} 10^{-6} & \text{if } \texttt{f1.on} \\ 10^{6} & \text{otherwise} \end{cases} \tag{13c}$$
$$\texttt{f1.v} = \texttt{f1.R} \cdot i \tag{13d}$$
$$\texttt{f1.i} = i \tag{13e}$$
$$\texttt{f2.R} = \begin{cases} 10^{-6} & \text{if } \texttt{f2.on} \\ 10^{6} & \text{otherwise} \end{cases} \tag{13f}$$
$$\texttt{f2.v} = \texttt{f2.R} \cdot i \tag{13g}$$
$$\texttt{f2.i} = i \tag{13h}$$
$$\texttt{r.v} = 100i \tag{13i}$$
$$\texttt{r.i} = i \tag{13j}$$
$$\texttt{time}_{next} = \texttt{time} + \varepsilon \tag{13k}$$
$$\texttt{f1.on}_{next} = \texttt{f1.on} \wedge \neg(i > 0.005) \tag{13l}$$
$$\texttt{f2.on}_{next} = \texttt{f2.on} \wedge \neg(i > 0.006) \tag{13m}$$

where

$$i = \frac{\texttt{src.v}}{100 + \texttt{f1.R} + \texttt{f2.R}} \tag{13n}$$

Notice that this model has *three* state variables, namely `time`, `f1.on` and `f2.on`, hence the three non-standard difference equations (13k), (13l) and (13m).

Figure 5 shows the corresponding results. Notice that, as expected, only the first fuse melts (see second slope in Figure 5) since its rated current is lower than that of the first fuse (see (11d) and (11g)).

In this first experiment, only the behaviour of fuses has been abstracted away by considering their melt duration to be negligible with respect to the raise duration of the ramp source. But what happens if we also abstract the behaviour of the ramp source, considering its raise duration negligible with respect to the entire operating duration of the system?

Abstracting the ramp source in this context means replacing it with a *step* source, leading to a new system of dynamic equations, where (13a) is replaced with:

$$\texttt{src.v} = \begin{cases} 1 & \text{if } \texttt{time} > 0.1 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

[4] This property is enforced by the use of a *when* clause in the definition of the fuse sub-model in Listing 3.

DOI 10.3384/ECP14096693

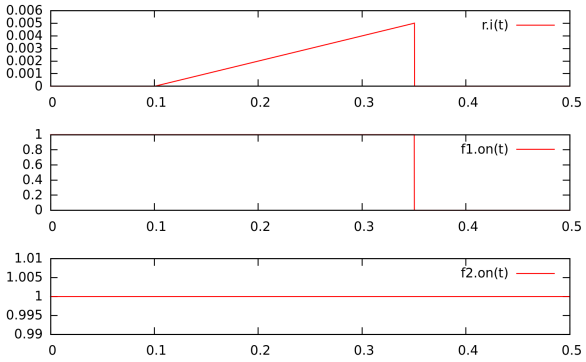

Figure 5: Solution of (13) with parameter bindings (11) and initial state (12).

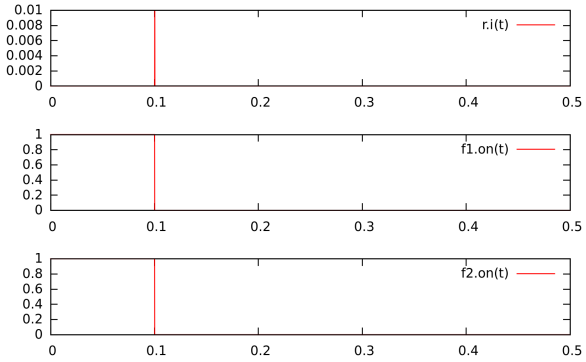

Figure 6: Solution of the system of dynamic equations resulting from the abstraction of the ramp voltage source.

Figure 6 shows the corresponding results. We can see that *both* fuses melt as a consequence of the raise of the voltage source signal. This fact contradicts our initial assumption of a model where only one fuse may melt, if ever. This last experiment has shown that an operational semantics based on a global, fixed infinitesimal step does not preserve *strict ordering* of events when behavioural abstractions are composed.

3.2 Conclusions from the experiments

The reason why we observe an accidental synchronisation of events after abstracting the behaviour of the ramp voltage source is that the time-line is "running out of available free slots" where to insert possible intermediate events. In our example, events corresponding to melts, whenever they eventually occur, must necessarily do so *strictly before* the ramp signal of the voltage source reaches its maximum value.[5] However, in a fixed-step non-standard operational semantics, when a standard difference equation is activated to reset the value of a state variable, or when a conditional equation switches (as in our example), new values are scheduled for the *next* instant, leaving no room for possible intermediate events to occur during this transition. So it seems that we should *densify* time in the neighbourhood of "urgent" actions.

[5] Because it is the voltage raise itself, after its "conversion" to current through the load `r`, that triggers an eventual melt.

4 Proposed semantic model

In this section, we introduce some basic notions that will be used to construct our model of a signal as well as defining the meaning of a differential equation.

We suppose given a positive infinitesimal real ε that we call the *real activity threshold*. Notice that, contrary to traditional fixed-step simulation, the semantics of the system is independent from the specific choice of ε.

For $r \in {}^*\mathbb{R}$, we denote $r + \varepsilon \cdot {}^*\mathbb{Z} \stackrel{def}{=} \{r + \varepsilon \cdot k \,|\, k \in {}^*\mathbb{Z}\}$, where ${}^*\mathbb{Z}$ is the set of non-standard integers. We define the base time-line ${}^*\mathbb{T}$ as a the non-negative values of ${}^*\mathbb{R}$:

$$ {}^*\mathbb{T} \stackrel{def}{=} {}^*\mathbb{R}_0^+ \tag{15} $$

Thus, ${}^*\mathbb{T}$ is a *linear continuum* (under the standard temporal order $<$), which contains all non-negative standard real numbers. An *instant* is an element of ${}^*\mathbb{T}$. In particular, we have non-standard instants.

Notice that, contrary to [1, 2, 16], we propose here a base time-line that is neither discrete, nor well-ordered. However, since ${}^*\mathbb{T}$ is *densely ordered*, intermediate instants exist between any two given distinct instants.

The question that arises now is: how to recover discreteness (as required to express the notion of *next* instant) as well as time divergence? The key idea lies in a *generalisation* of the usual concept of *clock* as encountered in synchronous languages.

4.1 Time, signals and dates

4.1.1 Time

A *time signal* t with the initial value $t_0 \in {}^*\mathbb{R}$ is defined as the following right-continuous step map:

$$ \begin{aligned} t : {}^*\mathbb{T} &\to t_0 + \varepsilon \cdot {}^*\mathbb{Z} \\ \tau &\mapsto t_0 + \varepsilon \cdot n_\tau \end{aligned} \tag{16} $$

with

$$ n_\tau = {}^*\left\lfloor \frac{\tau}{\varepsilon} \right\rfloor \in {}^*\mathbb{N}_0 $$

where ${}^*\lfloor \cdot \rfloor$ denotes the non-standard floor function.

In particular, for any $\tau \in \mathbb{R}_0^+$ we have:

$$st\big(t(\tau) - t_0\big) = \tau$$

meaning that the standardisation of t coincides with a standard time signal that would start from $st(t_0)$.

By definition, a *date* is an element of the co-domain of a time signal.

Notice the fundamental change of perspective with respect to [1, 2, 16]: instead of discretising the *domain* of the time signal, we discretise its *co-domain*. Thus, we can associate a discrete and well-ordered set of dates with a time signal, despite its densely ordered domain. Notice also that dates and instants are different concepts: while instants are densely ordered, dates are not. In the next sections, we show how to extend this idea to *any* signal.

4.1.2 Signals

We call a *(non-standard) signal* x a map from a subset of ${}^*\mathbb{T}$—the *domain* of the signal, denoted $\mathrm{Dom}\,x$—to a *discrete* set—its *co-domain*, denoted $\mathrm{Codom}\,x$. A signal maps instants to *values*.

We distinguish two families of signals:

discrete-time signals have their domain (called their *clock*) in a discrete subset of ${}^*\mathbb{T}$;

dense-time signals are right-continuous step signals, whereof the domain is ${}^*\mathbb{T}$.

A few remarks can be made regarding this taxonomy:

- We do not require zero (i.e. the least element of ${}^*\mathbb{T}$) to belong to the domain of a discrete-time signal: zero is only guaranteed to be a lower bound of the domain. In other terms, the birth instant of a discrete-time signal does not necessarily coincide with the birth instant of the model.
- Physical signals introduced in [1, 2] belong to the *discrete-time* family of signals.
- In some sense, compared with [2], we loose some uniformity by introducing two kinds of signal domains, with distinct topologies. However, practical implementations of modelling languages have to make a semantic distinction between signals at some point (to avoid ill-posed problems). In [1], for instance, the type system classifies signals in "discrete" and "continuous" ones (although they share the same representation).

Notice that, according to the definition of a signal, the co-domain of any signal is discrete. This raises the question of the representation of *real* signals in general, and *continuous* (or *physical*) signals in particular. We impose the following restrictions:

real signals have co-domains of the form $r + \varepsilon \cdot {}^*\mathbb{Z}$, where ε is the real activity threshold and r the start value of the signal;

continuous (or physical) signals are dense-time real signals that can only change their value by $\pm\varepsilon$.

4.1.3 Signal activity

The *activity* of a signal x, denoted $\mathrm{Act}\,x$, is the discrete (possibly finite, but often non enumerable) subset of $\mathrm{Dom}\,x$ defined as:

$$\mathrm{Act}\,x = \{\tau \mid x(\tau^-) \neq x(\tau)\} \tag{17}$$

where

$$x(\tau^-) = \lim_{\substack{\tau' \to \tau \\ \tau' < \tau}} x(\tau')$$

Intuitively, the activity of a signal can be seen as the set of instants corresponding to its "perceptible changes" from the point of view of an external observer. Notice that we have required the co-domain of signals to be discrete: this implies that the activity of a signal x contains *all* the instants at which "something happened" to x from an external observer point of view, whatever the accuracy of the measure.

4.2 Differential equations

If one assumes the existence of a *maximal* clock such as

$$\{\varepsilon \cdot n \mid n \in {}^*\mathbb{N}_0\}$$

then the sole concept of non-standard *difference equation* suffices to express dynamic behaviour of models. Indeed, as explained in previous sections, the solution yielded by the (non-standard) forward Euler scheme coincides with the actual, standard solution of the corresponding differential equation, after standardisation. But we have also seen that this uniform approach reaches its limits when composition of abstract models comes into play.

Density of possibly detectable events should be ideally correlated to the behaviour of signals instead of being imposed by a rigid, fixed-step scheme: indeed, more *activity* potentially implies more events.

Also, if one would be able to *predict* the occurrence of an event from the behaviour of the implied

DOI 10.3384/ECP14096693

signal(s) instead of waiting for the *next* time instant and then realise that *several* pending events need to be (wrongly) treated simultaneously, one would avoid the issues mentioned in previous sections.

The key idea is to define differential equations in such a way that activity of signals can be predicted, and then to use the concept of activity instead of the concept of clock to represent (mutually desynchronised) event sources. This idea is not new: B. P. Zeigler [17] already introduced most of the necessary concepts in his theory of systems. QSS solvers [5] are practical applications of this theory.

Recall that in [1, 2], the semantics of a differential equation was defined by means of a non-standard version of the forward Euler method, which is the simplest order-one method based on the classical time slicing approach. In the following, we will define the meaning of a differential equation by means of a non-standard order-one QSS-like method and show how this new approach solves behavioural abstraction issues.

Assume that the following are given:

- a non-empty interval $[\tau_a, \tau_b) \subseteq {}^*\mathbb{T}$,
- a non-standard real number $x_0 \in {}^*\mathbb{R}$,
- a non-standard real signal $y : [\tau_a, \tau_b) \to y_0 + \varepsilon \cdot {}^*\mathbb{Z}$.

Then the differential equation

$$der(x, x_0) := y \tag{18}$$

defines x on $[\tau_a, \tau_b)$ as follows.

Let $(w_i)_{i \in {}^*\mathbb{N}_0}$ be a family of ${}^*\mathbb{R}$-valued maps defined as:

$$w_0(\tau) = x_0 + (\tau - \tau_0) \cdot y(\tau_0) \tag{19a}$$
$$w_{n+1}(\tau) = w_n(\tau_{n+1}) + (\tau - \tau_{n+1}) \cdot y(\tau_{n+1}) \tag{19b}$$

where

$$\tau_0 = \tau_a \tag{20a}$$
$$\tau_{n+1} = \mathrm{Inf}_{[\tau_a, \tau_b]} \Big(\{ \tau \mid \tau > \tau_n \wedge event_n(\tau) \} \cup \{ \tau_b \} \Big) \tag{20b}$$

and

$$event_n(\tau) = (\tau \in \mathrm{Act}y) \vee \Big(\big(w_n(\tau) \in x_0 + \varepsilon \cdot {}^*\mathbb{Z} \big) \wedge \big(w_n(\tau) \neq w_n(\tau_n) \big) \Big) \tag{21}$$

Defining the co-product

$$w = \bigoplus_{i \in {}^*\mathbb{N}_0} w_i \big|_{[\tau_i, \tau_{i+1})} \tag{22}$$

then, finally:

$$x : [\tau_a, \tau_b) \to x_0 + \varepsilon \cdot {}^*\mathbb{Z}, \quad \tau \mapsto w\big(last(\tau)\big) \tag{23}$$

where

$$last(\tau) = \mathrm{Sup}_{[\tau_a, \tau_b)} \{ \tau' \mid \tau' \leq \tau \wedge w(\tau') \in x_0 + \varepsilon \cdot {}^*\mathbb{Z} \} \tag{24}$$

The idea is the following:

- w represents the "private" behaviour of the defined signal x: starting from x_0 at τ_a, it evolves linearly between two consecutive event instants ((19a) and (19b)) until τ_b;
- as stated by (21), events affecting the behaviour of w have two possible causes:
  - y (defining the right-hand side of the differential equation) has evolved perceptibly, i.e. its value has drifted by $\pm\varepsilon$, since the previous event (including initialisation),
  - w itself has evolved perceptibly since the previous event (including initialisation);
- each time an event affecting w occurs, its gradient is reevaluated, then w evolves linearly until the next event;
- the "public" behaviour of x (i.e. as seen by an external observer) is a piece-wise constant approximation of its "private" behaviour w such that both maps coincide at event instants corresponding to perceptible changes of w, as stated by (24).

Figure 7 illustrates the process (the w_i appear as black linear slopes, and x is the blue slope):

- blue bullets indicate events resulting from the evolution of w itself;
- red bullets indicate events resulting from the evolution of y (notice that they are not necessarily "synchronised" with ε-steps);
- green bullets indicate "skipped targets" (i.e. events that would have resulted from the sole evolution of w but that have been "intercepted" by an event caused by y).

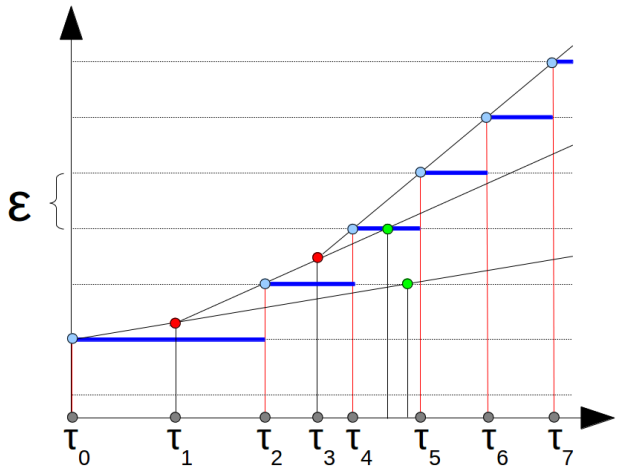

Figure 7: Semantics of a differential equation.

Notice that we have defined the semantics of a differential equation on an interval of the form $[\tau_a, \tau_b)$ instead of defining it on the whole time-line ${}^*\mathbb{T}$. The reason is that we want differential equations to be defined modularly, as required in particular by behavioural abstraction: it should be possible to define signals where ideal phases alternate with high-fidelity ones.

Clearly, the set of continuous signals is closed under the application of continuous functions. Therefore any family of signals defined solely by means of differential equations and by algebraic constraints like

$$y = f(x_1, \ldots, x_n) \tag{25}$$

only contains continuous signals. Regarding (25) we have in particular:

$$\mathrm{Act}\, y \subseteq \mathrm{Act}\, x_1 \cup \cdots \cup \mathrm{Act}\, x_n \tag{26}$$

meaning that such an algebraic constraint acts as an "activity filter".

4.3 Behavioural abstraction support

Below, we suppose given an additional positive infinitesimal constant δ.

We model a reset equation constraining a continuous signal x on $[\tau_a, \tau_a + \delta)$ by means of the following *differential* equation:

$$der(x, x_0) := \frac{x_1 - x_0}{\delta} \tag{27}$$

where x_0 is the value of x at τ_a, and x_1 is the "target value" to be reached by x at $\tau_a + \delta$ as a result of the reset operation.[6]

[6] In Modelica, this is given by the second argument of `reinit` (see Listing 1 for an example).

Notice that the right-hand side of (27) is constant and infinite. By the definition of a differential equation given in Section 4.2, x reaches x_1 (more precisely: the first element of $x_0 + \varepsilon \cdot {}^*\mathbb{Z}$ that is greater than or equal to x_1) after δ units of time (i.e. the length of $[\tau_a, \tau_a + \delta)$). Notice that this definition of a reset equation involves a *duration* (i.e. the time spent by the signal to reach its target value). Since this duration is infinitesimal, we actually observe the desired behaviour: the reset is infinitely fast compared to any standard phenomenon.

Recall, however, that we want to support proper composition of such abstraction mechanisms. In order to refine the above definition to achieve this goal, we first associate with each continuous signal defined in the model an *abstraction level* $n \in \mathbb{N}_0$[7] which represents the maximal number of nested reset equations that might contribute to the definition of the signal:

- a independent signal that is not reset in the model has the abstraction level 0;
- a signal that is reset in the model by equations involving only signals whose abstraction level is at most $n-1$ has abstraction level n;
- a signal that is defined by an algebraic constraint inherits the highest abstraction level among those of the signals appearing in its definition.

Notice that, in a language implementing our semantic model, the abstraction level defined as above can always be statically computed by elementary data-flow analysis for an arbitrary model without circular dependencies between reset equations.

We are now in the position to refine (27) as follows. A signal x with abstraction level n and the current value x_0 is reset to x_1 at τ_a by means of the following differential equation activated on $[\tau_a, \tau_a + \delta^n)$:

$$der(x, x_0) := \frac{x_1 - x_0}{\delta^n} \tag{28}$$

The idea is that signals with higher abstraction levels should reset infinitely faster than others: by taking successive powers of δ, we achieve precisely this effect.

What about the Zeno effect under behavioural abstraction in such a semantic model?

Notice that we can assume that continuous signals do not diverge in value during integration (otherwise the model is considered singular) and let the highest level of abstraction in a given model be n. By (28), this

[7] A standard natural number.

DOI 10.3384/ECP14096693

means that in this model, the fastest reset terminates in δ^n units of time. Hence, integration of each signal terminates in at most $\lceil(\tau_{end}-\tau_0)/\delta^n\rceil$ steps, where τ_0 and τ_{end} denote respectively the start time and the end time of the integration. Thus, provided that the maximum abstraction level in the model is finite, continuous signals always diverge in time.

4.4 Application Example

Let us consider again the model of Figure 4, where we suppose that the behaviour of the voltage source and of the fuses have been abstracted. How does our proposal fare in such a case?

Before we answer this question, we need to examine the case of conditional equations.

Conditional equations are used to express "mode changes" in physical models, typically leading to equation *switching*, as in our electrical model example. As a consequence, conditional equations potentially lead to the exact same issues encountered with reset equations. Same symptoms, same medication: we just need to find a way to introduce the concept of infinitely fast transition in the semantics of conditional equations. This is achieved as follows.

Suppose that $c : {}^*\mathbb{T} \to \{\mathsf{F},\mathsf{T}\}$ is a boolean signal whose value is F until the event instant τ_0, after which it takes the value T. Suppose also that $x : {}^*\mathbb{T} \to x_0+\varepsilon\cdot{}^*\mathbb{Z}$ and $y : {}^*\mathbb{T} \to y_0+\varepsilon\cdot{}^*\mathbb{Z}$ are two given physical signals. Then the semantics of a conditional real *function* can be defined as:

$$\text{if } c \text{ then } x \text{ else } y \stackrel{def}{=} \lambda\cdot x+(1-\lambda)\cdot y \qquad (29)$$

where

$$\lambda = \begin{cases} 0 & \text{on } \{\tau \mid \tau \le \tau_0\} \\ \frac{\tau-\tau_0}{\delta} & \text{on } \{\tau \mid \tau_0 < \tau \le \tau_0+\delta\} \\ 1 & \text{on } \{\tau \mid \tau > \tau_0+\delta\} \end{cases} \qquad (30)$$

It can easily be shown that signals defined by conditional real functions are continuous signals.

Coming back to our example, what happens when the voltage source makes a step? The voltage `src.v` *gradually* evolves from its start value 0 to its maximum value 1. During the rise, it traverses the surface corresponding to

$$\texttt{src.v} = \texttt{f1.iMax}\cdot(\texttt{r.R}+\texttt{f1.R}+\texttt{f2.R})$$

because, given our parameter settings and the values of the internal resistances of both fuses before the first melt, we have:

$$\texttt{f1.iMax}\cdot(\texttt{r.R}+\texttt{f1.R}+\texttt{f2.R}) = 0.500000001$$

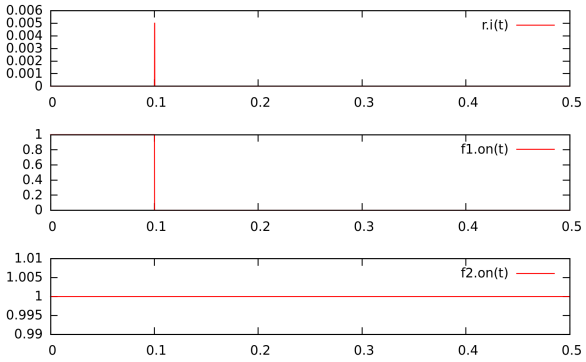

Figure 8: Solution of the system of dynamic equations resulting from the abstraction of the ramp voltage source.

which belongs to $[0,1]$.

But, according to (13n), this is precisely the voltage required to induce a current equal to the rated current of `f1` in the circuit. Consequently, `f1` melts, but not `f2`. Figure 8 illustrates the result of interpreting the model with our semantics.

5 Conclusion and Future Work

In this paper, we have extended the pioneer work of [2, 16] by proposing a non-standard operational semantics supporting compositional behavioural abstraction. As demonstrated by [1], non-standard semantics can be used to give rigorous interpretation of hybrid modelling languages such as Modelica.

The most obvious practical application of our work would certainly be the design and development of a simulator that would conform to our semantic model: one of us (S. Furic) is currently working in this direction in the course of the French funded project AGéSys.

6 Acknoledgements

We are indebted to Albert Benveniste, Benoît Caillaud, Marc Pouzet and Timothy Bourke for having launched the topic: their impressive work and their availability for exchanging on modelling language semantic topics has greatly helped us in this work. Many thanks also to Ramine Nikoukhah, for his help in understanding hybrid modelling issues and for having brought to our attention the central role of some synchronous language concepts, especially those of the SIGNAL language. Many thanks also to Belgacem Ben Hedia and to Étienne Hamelin, from CEA for the many interesting discussions. This paper results in large parts from the

fruitful collaboration between CEA and LMS Imagine in the course of the ITEA2 project OpenProd.

References

[1] Albert Benveniste, Timothy Bourke, Benoit Caillaud, and Marc Pouzet. Non-standard semantics of hybrid systems modelers. *Journal of Computer and System Sciences*, 78:877–910, May 2012. doi: 10.1016/j.jcss.2011.08.009.

[2] Simon Bliudze and Daniel Krob. Modelling of complex systems: Systems as dataflow machines. *Fundamenta Informaticae*, 91:1–24, 2009. doi: 10.3233/FI-2009-0001.

[3] Timothy Bourke and Marc Pouzet. Zélus: A synchronous language with ODEs. In *16th International Conference on Hybrid Systems: Computation and Control (HSCC'13)*, pages 113–118, Philadelphia, USA, March 2013.

[4] Stephen L Campbell, Jean-Philippe Chancelier, and Ramine Nikoukhah. *Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4*. Springer, 2010. ISBN 978-1-4419-5527-2.

[5] François E. Cellier, Ernesto Kofman, Gustavo Migoni, and Mario Bortolotto. Quantized state system simulation. *Proceedings of Grand Challenges in Modeling and Simulation (GCMS'08)*, pages 504–510, 2008.

[6] J. Eker, J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. Taming heterogeneity: The Ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, 2003.

[7] Goran Frehse, Colas Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer Berlin Heidelberg, 2011. doi: 10.1007/978-3-642-22110-1_30.

[8] Peter Fritzson. *Introduction to modeling and simulation of technical and physical systems with Modelica*. Wiley-IEEE Press, 2011. ISBN 978-1-118-01068-6.

[9] Sébastien Furic. Enforcing model composability in Modelica. In *Proceedings of the 7th International Modelica Conference, Como, Italy*, pages 868–879, 2009.

[10] Boris Golden, Marc Aiguier, and Daniel Krob. Modeling of complex systems II: A minimalist and unified semantics for heterogeneous integrated systems. *Applied Mathematics and Computation*, 218(16):8039–8055, 2012.

[11] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science, LICS'96*, pages 278–292. IEEE Society Press, 1996.

[12] H. Jerome Keisler. *Foundations of Infinitesimal Calculus*. On-line Edition, 2007. URL `http://www.math.wisc.edu/~keisler/foundations.html`.

[13] Michal Konečný, Walid Taha, Jan Duracz, Adam Duracz, and Aaron Ames. Enclosing the behavior of a hybrid system up to and beyond a Zeno point. In *Cyber-Physical Systems, Networks, and Applications (CPSNA), 2013 IEEE 1st International Conference on*, pages 120–125, 2013. doi: 10.1109/CPSNA.2013.6614258.

[14] Edward A. Lee and Haiyang Zheng. Operational semantics of hybrid systems. In Manfred Morari and Lothar Thiele, editors, *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 25–53. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25108-8. doi: 10.1007/978-3-540-31954-2_2.

[15] Abraham Robinson. *Non Standard Analysis*. North Holland, 1966.

[16] Heinrich Rust. *Operational Semantics for Timed Systems: A Non-standard Approach to Uniform Modeling of Timed and Hybrid Systems*, volume 3456 of *Lecture Notes in Computer Science*. Springer, 2005. ISBN 3-540-25576-1. doi: 10.1007/978-3-540-32008-1.

[17] Bernard P. Zeigler and Jong Sik Lee. Theory of quantized systems: formal basis for DEVS/HLA distributed simulation environment. In *SPIE Proceedings*, volume 3369, pages 49–58, 1998.