The Fundamentals of Hybrid Systems Modelers

Albert Benveniste, INRIA-Rennes
Campus de Beaulieu
35042 Rennes cedex, France

Benoît Caillaud, INRIA-Rennes
Campus de Beaulieu
35042 Rennes cedex, France

Marc Pouzet, LIENS
École normale supérieure
45 rue d'Ulm
75230 Paris Cedex 05, France

***Abstract*—Hybrid systems modelers have become the corner stone of embedded system development, with Simulink a de facto standard and Modelica a new player. Such tools still raise a number of issues that, we believe, require more fundamental understanding. In this paper we propose using *non standard analysis* as a semantic domain for hybrid systems — non standard analysis is an extension of classical analysis in which infinitesimals (the $''$ and in the celebrated generic sentence $\forall '' \exists \; :::$ in college maths) can be manipulated as first class citizens. This allows us to provide a denotational semantics and a constructive semantics for hybrid systems, thus establishing simulation engines on a firm mathematical basis. In passing, we cleanly separate the job of the numerical analyst (solving differential equations) from that of the computer scientist (generating execution schemes).[1]**

I. Introduction

Hybrid systems modelers have become in the last two decades the corner stone of complex embedded system development, for computer controlled systems. Simulink[2] has become the de facto standard for physical system modeling and simulation. Noticeably, by building on top of the success of Simulink, The Mathworks was able to take over the market of embedded systems design, for many sectors. This speaks for itself regarding the importance of such tools. In this paper we focus on general modelers, aimed at modeling and simulation of any type of hybrid system and we refer the reader to [10] for an overview of all tools related to hybrid systems analysis. Besides Simulink with its state-based extension Stateflow, several such hybrid systems modelers have been developed. Scicos[3] is a freeware developed by Ramine Nikoukhah at INRIA [19], [17]. Modelica[4] is a non-proprietary, object-oriented, equation based language to conveniently model complex multi-physics systems. In Modelica, equations have no pre-defined causality. Hybrid systems modelers raise a number of difficult issues:

- Since simulations use a single, global, solver, the choice and tuning of the integration method is global to the system. This may cause undesirable interactions between sub-systems that seemingly should not interact [2].
- *Zero-crossings,* which trigger mode changes, can involve a combination of complex operations whose scheduling may be delicate.
- How discrete is the semantics of the discrete part of a hybrid system modeler? Often, discrete and continuous are not cleanly separated [2].
- What are the consequences for compilation of Modelica's "acausal" approach?

Overall, we think that, with the exception of [18], no fundamental answer has been provided to the difficulty raised by the following justified although contradictory requirements:

(a) The semantic function, mapping a hybrid systems specification to its executable mathematical model (its operational semantics), should be *statically* defineable.
(b) Computers can only run according to discrete steps, hence discretizing must be part of defining the semantic map.
(c) To achieve high computational quality with high flexibility, the discretization scheme must be fixed adaptively, *at run time* [18].

To reconcile requirements (a)–(c), we first propose using *non standard analysis* as a semantic domain for hybrid systems. Second, we develop a comprehensive *constructive semantics* for hybrid systems. — the constructive semantics was first proposed by G. Berry [7] as a mathematical theory on which to build synchronous languages' compilers.

The paper is organized as follows. Background on non standard analysis is provided in section II. Our mathematical formalism for hybrid systems specification is introduced in section III, we call it SimpleHybrid. Section IV is the core of our paper; a denotational semantics is given based on non standard analysis; then, a constructive semantics is provided. Related work is analysed in section V. More results can be found in the extended version of this paper [2].

II. Background on non-standard analysis

The key difficulty in reconciling requirements (a)–(c) is to give proper semantics to the inherently continuous time statement: $\dot{y} = x$. In non-standard analysis, this statement means, *by definition of the derivative of a function*: $\forall \partial \simeq 0$, $\forall t \in \mathbb{R}_+$, $\frac{1}{@}(y_{t+@} - y_t) = x_t$, where expression "$\partial \simeq 0$" is a *non-standard* expression that reads: *"∂ is infinitesimal"*.

Non-standard analysis has been proposed by Abraham Robinson in the 60's to allow handling explicitly "infinitesimals" in analysis [1], [16]. Robinson's approach is axiomatic, in that he proposes enriching the basic ZFC (Zermelo-Fraenkel) framework with three more axioms. There has been much debate in the community of mathematicians as to

[1]This work was supported by the INRIA "Action d'envergure" SYNCHRONICS.

[2]http://www.mathworks.com/products/simulink/

[3]http://www-rocq.inria.fr/scicos/

[4]http://www.modelica.org/

whether it is worth considering non-standard analysis instead of sticking with the traditional one. We won't enter this debate. One important thing for us, however, is that it allows using non-standard discretization of continuous dynamics "as if" it was operational. To our surprise, such an idea is indeed not new. Bliudze and Krob [9], [8] have used non-standard analysis as a mathematical support for defining a system theory for hybrid systems. The formalization they propose closely mimics that of Turing machines. The introduction to non-standard analysis in [8] is very pleasant and we take the liberty to borrow it. This presentation was originally due to Lindstrøm, see [15]. Its interest is that it does not require any fancy axiomatic material but only makes use of the axiom of choice — actually a weaker form of it.

The goal is to augment $\mathbb{R} \cup \{\pm\infty\}$ by adding, to each x in this set, a bunch of elements that are "infinitesimally close" to it, call ${}^\star\mathbb{R}$ the resulting set. Another requirement is that all operations and relations defined on $\mathbb{R}$ should extend to ${}^\star\mathbb{R}$. A first idea is to represent such additional numbers as convergent sequences of reals.[5] For example, elements infinitesimally close to the real number zero are the sequences $u_n = 1/n$, $v_n = 1/\sqrt{n}$ and $w_n = 1/n^2$. Observe that the above three sequences can be ordered: $v_n > u_n > w_n > 0$. How can this be made systematic? We explain it next.

A. Ultrafilters, ultraproducts, and the Transfer Principle

For I an arbitrary set, a *filter* $\mathcal{F}$ over I is a family of subsets of I such that:

1) the empty set does not belong to $\mathcal{F}$,
2) $P, Q \in \mathcal{F}$ implies $P \cap Q \in \mathcal{F}$, and
3) $P \in \mathcal{F}$ and $P \subset Q \subseteq I$ implies $Q \in \mathcal{F}$.

Consequently, $\mathcal{F}$ cannot contain both a set P and its complement P^c. A filter that contains at least one of the two for any subset $P \subseteq I$ is called an *ultra-filter*. At this point we recall Zorn's lemma, known to be equivalent to the axiom of choice:

Lemma 1 (Zorn's lemma): Any partially ordered set $(X, \leq)$ such that any chain in X possesses an upper bound has a maximal element.

It is easily seen that a filter $\mathcal{F}$ over I is an ultra-filter if and only if it is maximal with respect to set inclusion. By Zorn's lemma, any filter $\mathcal{F}$ over I can be extended to an ultra-filter over I. Now, if I is infinite, the family of sets $\mathcal{F} = \{P \subseteq I \mid P^c \text{ is finite}\}$ is a *free* filter, meaning it contains no finite set. It can thus be extended to a free ultra-filter over I:

Lemma 2: Any infinite set has a free ultra-filter.

Every free ultra-filter $\mathcal{F}$ over I uniquely defines, by setting $\mu(P) =$ if $P \in \mathcal{F}$ then 1 else 0, a finitely additive measure[6] $\mu : 2^I \mapsto \{0, 1\}$ such that

$$\mu(I) = 1 \text{ and } \mu(P) = 0 \text{ whenever } P \text{ is finite.}$$

Now, fix an infinite set I and a finitely additive measure μ over I as above. Let $\mathbb{X}$ be a set and consider the Cartesian product $\mathbb{X}^I = (x_i)_{i \in I}$. Say $(x_i) \sim (x'_i)$ iff $\mu\{i \in I \mid x_i \neq x'_i\} = 0$. Relation $\sim$ is an equivalence relation whose equivalence classes are denoted by $[x_i]$ and we define

$$ {}^\star\mathbb{X} = \mathbb{X}^I / \sim \tag{1}$$

$\mathbb{X}$ is naturally embedded into ${}^\star\mathbb{X}$ by mapping every $x \in \mathbb{X}$ to the constant tuple such that $x_i = x$ for every $i \in I$. Any algebraic structure over $\mathbb{X}$ (group, ring, field) carries over to ${}^\star\mathbb{X}$ by almost pointwise extension. In particular, if $[x_i] \neq 0$, meaning that $\mu\{i \mid x_i = 0\} = 0$ we can define its inverse $[x_i]^{-1}$ by taking $y_i = x_i^{-1}$ if $x_i \neq 0$ and $y_i = 0$ otherwise. This construction yields $\mu\{i \mid y_i x_i = 1\} = 1$, whence $[y_i][x_i] = 1$ in ${}^\star\mathbb{X}$. The existence of an inverse for any non-zero element of a ring is indeed stated by the following first order formula: $\forall x(x = 0 \vee \exists y(xy = 1))$. More generally:

Lemma 3 (Transfer Principle): Every first order formula is true over ${}^\star\mathbb{X}$ iff it is true over $\mathbb{X}$.

B. The sets ${}^\star\mathbb{R}$ and ${}^\star\mathbb{N}$ of non-standard reals and integers

We just apply the above general construction to $\mathbb{X} = \mathbb{R}$ and $I = \mathbb{N}$ and we denote by ${}^\star\mathbb{R}$ the result, which is then a field according to the transfer principle. By the same principle, ${}^\star\mathbb{R}$ is totally ordered by $[u_n] \leq [v_n]$ iff $\mu\{n \mid v_n > u_n\} = 0$. For u an arbitrary sequence of real numbers, let $\lim(u) \subseteq \overline{\mathbb{R}} =_{\text{def}} \mathbb{R} \cup \{-\infty, +\infty\}$ denote the (possibly empty) set of all limit points of sequence u: for $x \in \lim(u)$, let $v_k = u_{n_k}$ be a subsequence of u converging to x. If $\lim(u) \neq \emptyset$, there exists exactly one limit point $x \in \lim(u)$ such that $\mu\{n_k\} = 1$, and any other limit point yields a μ-measure 0 for the corresponding subsequence.[7] Call x the *standard part* of $[x_n]$ and we write $x = st([x_n])$. Infinite $x \in {}^\star\mathbb{R}$ have no standard part in $\mathbb{R}$. It is also of interest to apply the general construction (1) to $\mathbb{X} = I = \mathbb{N}$, which results in the set ${}^\star\mathbb{N}$ of *non-standard integers*. ${}^\star\mathbb{N}$ differs from $\mathbb{N}$ by the addition of *infinite integers,* which are equivalence classes of sequences of integers whose essential limit is $+\infty$.

C. Integrals and differential equations

Any sequence (g_n) of functions $g_n : \mathbb{R} \mapsto \mathbb{R}$ pointwise defines a function $[g_n] : {}^\star\mathbb{R} \mapsto {}^\star\mathbb{R}$ by setting

$$[g_n]([x_n]) = [g_n(x_n)]$$

A function ${}^\star\mathbb{R} \to {}^\star\mathbb{R}$ which can be obtained in this way is called *internal*. Properties of and operations on ordinary functions extend pointwise to internal functions of ${}^\star\mathbb{R} \to {}^\star\mathbb{R}$. For $g : \mathbb{R} \to \mathbb{R}$, its *non-standard version* is the internal function ${}^\star g = [g, g, g, \ldots]$. The same notions apply to sets. An internal set $A = [A_n]$ is called *hyperfinite* if $\mu\{n \mid A_n \text{ finite}\} = 1$; the *cardinal* $|A|$ of A is defined as $[|A_n|]$.

[5] Indeed, the proposed construction bears some resemblance with the construction of $\mathbb{R}$ as the set of equivalence classes of Cauchy sequences in $\mathbb{Q}$ modulo the equivalence relation $(u_n) \approx (v_n)$ iff $\lim_{n\to\infty}(u_n - v_n) = 0$.

[6] Observe that, as a consequence, μ cannot be sigma-additive (in contrast to probability measures or Radon measures) in that it is *not* true that $\mu(\bigcup_n A_n) = \sum_n \mu(A_n)$ holds for an infinite denumerable sequence A_n of pairwise disjoint subsets of $\mathbb{N}$.

[7] So far this was a bit of hand waving. To prove this, let $x = \sup\{x \in \mathbb{R} \mid [x] \leq [x_n]\}$, where $[x]$ denotes the constant sequence equal to x. Since $[x_n]$ is finite, x exists and we only need to show that $[x_n] - [x]$ is infinitesimal. If not, then there exists $y \in \mathbb{R}, y > 0$ such that either $[y] < [x_n] - [x]$ or $[y] < [x] - [x_n]$, a contradiction. The unicity of x is clear.

Now, consider an infinite number $N \in {}^?\mathbb{N}$ and the set

$$T = \left\{ 0, \frac{1}{N}, \frac{2}{N}, \frac{3}{N}, \dots \frac{N-1}{N}, 1 \right\} \tag{2}$$

By definition, if $N = [N_n]$, then $T = [T_n]$ with

$$T_n = \left\{ 0, \frac{1}{N_n}, \frac{2}{N_n}, \frac{3}{N_n}, \dots \frac{N_n - 1}{N_n}, 1 \right\}$$

hence $|T| = [|T_n|] = [N_n + 1] = N + 1$. Next, consider an internal function $g = [g_n]$ and a hyperfinite set $A = [A_n]$. We can then define the *sum* of g over A by

$$\textstyle\sum_{a \in A} \; g(a) \;\; =_{\text{def}} \;\; \left[\sum_{a \in A_n} \; g_n(a) \right]$$

If t is as above and $f : \mathbb{R} \to \mathbb{R}$ is a standard function, we get

$$\textstyle\sum_{t \in T} \frac{1}{N} {}^?f(t) = \left[\sum_{t \in T_n} \frac{1}{N_n} f(t_n) \right] \tag{3}$$

Now, f continuous implies $\sum_{t \in T_n} \frac{1}{N_n} f(t_n) \to \int_0^1 f(t)dt$, so,

$$\textstyle\int_0^1 f(t)dt = st \left(\sum_{t \in T} \; \frac{1}{N} {}^?f(t) \right) \tag{4}$$

Under the same assumptions, for any $t \in [0, 1]$,

$$\textstyle\int_0^t f(u)du = st \left(\sum_{u \in T; u \leq t} \; \frac{1}{N} {}^?f(t) \right) \tag{5}$$

Now, consider the ODE with initial condition

$$\dot{x} = f(x, t), \quad x(0) = x_0 \tag{6}$$

where $f : \mathbb{R}^m \times [0, 1] \to \mathbb{R}^m$ is bounded and continuous. Rewriting (6) in integral form $x(t) = x_0 + \int_0^t f(x(u), u)du$ and then using (5), we get

$$x(t) = st \left(x_0 + \textstyle\sum_{u \in T; u \leq t} \; \frac{1}{N} {}^?f(x(u), u) \right) \tag{7}$$

Set in (7) $\partial = 1/N$ which is > 0 and infinitesimal, so that $T = \{ t_n = n\partial \mid n = 0, \dots, N \}$. Then, the expression in parentheses at the right hand side of (7) is the piecewise constant right continuous function ${}^?x(t), t \in [0, 1]$ such that, for $n = 1, \dots, N$:

$$\left\{ \begin{array}{lcl} {}^?x(t_n) & = & {}^?x(t_{n-1}) + \partial \times f({}^?x(t_{n-1}), t_{n-1}) \\ {}^?x(t_0) & = & x_0 \end{array} \right. \tag{8}$$

Formula (8) can be seen as a *non-standard operational semantics* for the ODE (6). In particular, the following holds:

Principle 1 (Standardisation Principle): Non-standard dynamical system (8) can always be considered, for any non-standard function $f : {}^?\mathbb{R} \mapsto {}^?\mathbb{R}$. If, however, f is internal and ODE (6) has a unique solution, then (7) holds, meaning that the standardisation of dynamical system (8) is solution of (6).

D. Non-Standard Analysis as a semantic domain

Using non-standard analysis has the following advantages:

1) We will be able to use a time set $\mathbb{T}$ that is both *dense* in $\mathbb{R}$ and *discrete* in that each instant in $\mathbb{T}$ possesses a unique previous and next instant — e.g., in formula (8).
2) Since $\mathbb{T}$ is discrete, we can specify dynamical systems over $\mathbb{T}$ in full generality, without the need for referring to any kind of smoothness condition — e.g., as in (8).
3) Did the problem with the smoothness condition miraculously disappear? Not quite so. But it is postponed to the very end, at run time, thanks to Standardisation Principle 1: if the considered hybrid system has a unique solution in the usual mathematical sense, then the standardisation of our operational semantics does compute it.

III. The SimpleHybrid formalism[8]

In this section we develop a small "mathematical language" for hybrid systems, we call it SimpleHybrid. Primitives of SimpleHybrid are equations of the following form:

$$\begin{array}{lclcl} Eq_1 & : & y & = & f(x, \dot{x}) \\ Eq_2 & : & y & = & \mathbf{last}\,(x) \\ Eq_3 & : & \tau & = & \mathbf{up}(z) \\ Eq_4 & : & y & = & \mathbf{pre}\,(x) \;\; \mathbf{init} \;\; y_0 \\ Eq_5 & : & u & = & [z] \;\; \mathbf{every} \;\; [\tau] \;\; \mathbf{init} \;\; u_0 \\ Eq_6 & : & \dot{y} & = & x \;\; \mathbf{init} \;\; y_0 \, \mathbf{reset} \, u \end{array} \tag{9}$$

In (9) symbols u, x, y, z denote *variables* taken from an underlying set $\mathcal{X}$ of variables, and symbol τ denotes a *clock variable* taken from an underlying set $\mathcal{T}$ of clock variables. Symbols y_0 and u_0 denote values. Finally, dotted variables $\dot{x}$ and $\dot{y}$ indicate derivatives. Hybrid systems are specified via conjunctions of equations of the form Eq_1–Eq_6.

In the following we give an informal explanation of the above primitives, without expliciting the needed continuity or smoothness assumptions for them to make sense. The corresponding precise meaning will be given in the next section. In the sequel, a *clock* is a subset of $\mathbb{R}_+$. We identify clock variable τ with the boolean predicate it defines:

$$\tau_t \quad = \quad \text{if } t \in \tau \text{ then T else F} \tag{10}$$

Eq_1 means $\forall t \in \mathbb{R} : y_t = f(x_t, \dot{x}_t)$.

Eq_2 means $y_t = x_{t\_} =_{\text{def}} \lim_{s \nearrow t} x_s$, i.e., y_t is the *left-limit* of x_s when s approaches t from below.

Eq_3 defines the clock τ such that, using convention (10): $\tau_t = [z_{t\_} < 0] \wedge [z_t \geq 0]$. Thus τ selects the instants t at which z_t crosses zero from below, we call such a clock a *zero-crossing*. We will need to consider tuples $[\tau] \;=_{\text{def}}\; (\tau_1 \dots \tau_k)$ of zero-crossings.[9]

For each signal x, we assume a clock τ_x, the *clock of* x, such that x is guaranteed constant on the complement of τ_x.

Definition 1: A signal is typed discrete *if either it has been declared so, or if its clock is some zero-crossing. Otherwise it is typed* continuous.

The following operators define or involve discrete signals:

Eq_4 assumes x *discrete* and defines y as the delayed version of x by setting $\tau_y = \tau_x$ and the nth new value for y equal to the $n - 1$st one of x; an initial condition y_0 is provided.

Eq_5 For u a signal, $u_0 \in \mathbb{R}^n$ a value, and $[\tau] = (\tau_1 \dots \tau_k)$ and $[z] = (z_1 \dots z_k)$ two matching[10] tuples of zero-crossings and signals, Eq_5 states that u has clock $\tau = \bigcup_{i=1}^k \tau_i$

[8]The term "simple" refers to the fact that DAE are not supported. The study of formalisms à la Modelica is deferred to a subsequent paper.

[9]The use of brackets in $[\tau]$ is not to be confused with its use in Section II.

[10]Say that two tuples $(u_1 \dots u_k)$ and $(v_1 \dots v_l)$ are *matching* if they possess identical number of components: $k = l$.

— hence u is *discrete* — and, for every $t \in \tau$, $u_t = z_t$ holds, and $u_t = u_0$ for $t < t_1$, the first instant of τ. Note that we do not require that z is discrete.

Eq_6 For y, x two signals, y_0 a value, and u a *discrete* signal, Eq_6 states that ODE $\dot{y}_t = x_t$ holds with initial condition y_0 and this ODE is reset to the value given by u at each instant of the discrete clock of u.

As said before, hybrid systems are specified in SIMPLEHYBRID via sets of equations of the form Eq_1–Eq_6, taken conjunctively. For example, composing ODE Eq_6 with statement $x = f(y, v)$ of the form Eq_1, and reset Eq_5, yields the ODE

$$\dot{y} = f(y, v) \ \textbf{init} \ y_0 \ \textbf{reset} \ [z] \ \textbf{every} \ [\tau] \tag{11}$$

which means that ODE $\dot{y}_t = f(y_t, v_t)$ holds with initial condition y_0 and this ODE is reset to a value given by z_i each time zero-crossing τ_i occurs. It is easily checked that generic form (11) for a SIMPLEHYBRID equation is closed under parallel composition, and that SIMPLEHYBRID allows to encode hybrid automata with their locations [2].

IV. A SEMANTICS OF SIMPLEHYBRID

Throughout this section we fix a basic infinitesimal base step ∂. Following [9], as our universal time base we replace $\mathbb{R}_+$ by the non-standard set

$$\mathbb{T} \quad = \quad \{t_n = n\partial \mid n \in {}^{?}\mathbb{N}\}$$

For $t \in \mathbb{T}$, define

$$\begin{array}{rcl} {}^{\bullet}t & = & \max\{s \mid s \in \mathbb{T}, s < t\} \\ t^{\bullet} & = & \min\{s \mid s \in \mathbb{T}, s > t\} \end{array} \tag{12}$$

We thus have ${}^{\bullet}t_n = t_{n-1}$ and $t_n^{\bullet} = t_{n+1}$. The key fact about $\mathbb{T}$ is that for every $u \in \mathbb{R}_+$ there exists a unique $t \in \mathbb{T}$ such that ${}^{\bullet}t < u \leq t$ and $t - u$ is infinitesimal. Thus $\mathbb{T}$ is, at the same time, dense in $\mathbb{R}_+$, and can still be handled as if it was discrete and totally ordered.

A. Non-standard semantics

We assume an underlying set $\mathcal{T}$ of *clock variables*. Elements and subsets of $\mathcal{T}$ are generically denoted by τ and T, respectively. We identify τ with the boolean predicate it defines, see (10). We assume an underlying set $\mathcal{X}$ of *variables* and, a domain D_x for every $x \in \mathcal{X}$. For $X \subseteq \mathcal{X}$ finite, a *state* over X is an element $s \in D_X$, where $D_X = \prod_{x \in X} D_x$ and a *behaviour* over X is an element $\sigma \in \mathbb{T} \to D_X$. We write x_t instead of $\sigma \to \sigma(t, x)$ and τ_t instead of $\sigma \to \sigma(t, \tau)$.

A *hybrid system* is a tuple $S = (X, T, \Sigma)$, where $X \subseteq \mathcal{X}$ and $T \subseteq \mathcal{T}$ are finite and Σ is a set of behaviours over $X \cup T$. For $Y \supseteq X \cup T$, we can lift Σ to Y, written $\Sigma^{\uparrow Y}$, by taking all behaviours over Y whose projection over $X \cup T$ are in Σ. Then, for $S_i = (X_i, T_i, \Sigma_i), i = 1, 2$, we define the *parallel composition*

$$S_1 \parallel S_2 = (\, X,\, T,\, {\Sigma_1}^{\uparrow X \cup T} \cap {\Sigma_2}^{\uparrow X \cup T} \,), \tag{13}$$

where $X = X_1 \cup X_2$ and $T = T_1 \cup T_2$. The hybrid systems we shall consider are the parallel composition of a finite set of statements of one of the forms Eq_1–Eq_6. Call $Clocks(S)$ the (finite) set of all discrete clock variables involved in the specification of S. A *clock configuration* for S is a map

$$\kappa : \ Clocks(S) \mapsto \{\mathrm{F}, \mathrm{T}\}, \tag{14}$$

assigning a truth value to each discrete clock variable of S. Clock configurations are used to indicate the presence/absence of each discrete clock of S at a given instant t. A clock configuration κ for S is called *reachable* if there exists a behavior σ and an instant t such that $\sigma(t)(T) = \kappa(T)$ for every $T \in Clocks(S)$. The non-standard semantics of SIMPLEHYBRID is given in table I, second column.

B. Constructive semantics

As for any synchronous language, the *constructive semantics* [7], [4] formalizes how the different actions should be scheduled at a considered instant.

Scheduling constraints: Let $\perp$ be a special value not belonging to any domain D_x, to be interpreted as "not evaluated yet".[11] Define, for any $x \in \mathcal{X}$, $D_x^{\perp} = D_x \cup \{\perp\}$. Write $x = \top$ to mean that $x \neq \perp$. Let $\rhd$ be the following *scheduling constraint* relating any two variables u and v with domains $D_u^{\perp}$ and $D_v^{\perp}$, respectively:

$$u \rhd v \quad =_{\mathrm{def}} \quad [u = \top \ \vee \ v = \perp] \tag{15}$$

that is, $u \rhd v$ means $[v = \top \Rightarrow u = \top]$. It formalizes that "$v$ cannot be evaluated strictly before u". In particular, for τ any clock,

$$\forall t \in \tau \Rightarrow x_t \rhd y_t \quad =_{\mathrm{def}} \quad [x_t = \top] \vee [y_t = \perp] \vee [\tau_t = \mathrm{F}]$$

where τ_t is defined in (10). Observe that statement $v = f(u)$, where f is a function, abstracts as $u \rhd v$ since v can be substituted by its evaluation $f(u)$ everywhere. Relation $\rhd$ captures causality constraints within a system of equations.

The constructive semantics is obtained by abstracting, in the non-standard semantics (second column of table I), any statement of the form $y_t = exp$ where expression exp involves variables x_s, u_s, τ_s for $s = {}^{\bullet}t, t, t^{\bullet}$, by the scheduling constraints $x_s \rhd y_t$, $u_s \rhd y_t$, or $\tau_s \rhd y_t$, respectively. For example, $y_t = f(x_t)$ is abstracted as $x_t \rhd y_t$.

Observe that the semantics of $\tau = \mathbf{up}(z)$ corresponds to a "weak preemption" in that the change in the sign of z at instant t results in emitting a zero-crossing at the next instant $t^{\bullet}$. Hence, no clock occurs on any consequent part of a zero-time causality constraint. Therefore, preconditions such as "$\forall t \in \tau \Rightarrow$" in the mid column of table I do not impair the validity of the above mentioned abstractions.

Pre- and post-variables: In writing the constructive semantics, we would like to abstract away dummy time index t. To this end, for each variable $x \in X$ of the considered system S, we augment X with the two auxiliary variables ${}^{\bullet}x$ and $x^{\bullet}$, such that ${}^{\bullet}x_t = x_{{}^{\bullet}t}$ and $x_t^{\bullet} = x_{t^{\bullet}}$ hold for every t. Using these auxiliary variables and clock variables, time index t can

[11] This notation deviates from the historically established use of symbol $\perp$ in synchronous languages to denote absence. Absence of a signal in a reaction is a well defined status that is the result of evaluating the considered reaction. "Absence" and "not evaluated yet" should therefore not be confused.

statement S	non-standard semantics of S	$[\![S]\!]$: constructive semantics
$Eq_1: \quad y = f(x; \dot{x})$	$\forall t \in \mathbb{R}_+ \quad \Rightarrow \quad y_t = f\left(x_t; \frac{x_t - x_{\bullet t}}{@}\right)$	**on** $_y : x \rhd y$
$Eq_2: \quad y = \mathbf{last}(x)$	$\forall t \in \mathbb{R}_+ \quad \Rightarrow \quad y_t = x_{\bullet t}$	**on** $_y : {}^\bullet x \rhd y$
$Eq_3: \quad = \mathbf{up}(z)$	$_{t\bullet} = [z_{\bullet t} < 0] \wedge [z_t \geq 0]$	**on** $_y : z \rhd \ ^\bullet$
$Eq_4: \quad y = \mathbf{pre}(x)$ **init** y_0	$_y = \ _x$ discrete $\forall t < \min(\ _y) \quad \Rightarrow \quad y_t = y_0$ $\forall t \in \ _y \quad \Rightarrow \quad y_t = x_{\bullet t}$	**on** $_y : \ _y = \ _x$ discrete **on** $_y : {}^\bullet x \rhd y$
$Eq_5: \quad u = [z]$ **every** $[\ ]$ **init** u_0	$_u = \quad$ discrete $\forall t < \min(\bigcup_i \ _{u_i}) \quad \Rightarrow \quad u_t = u_0$ $\forall t \in \ _{u_i} \setminus (\bigcup_{j<i} \ _{u_j}) \quad \Rightarrow \quad u_t = z_{i;t}$	**on** $[\ ] : [z] \rhd u$
$Eq_6: \quad \dot{y} = \quad x$ **init** y_0 **reset** u	$\forall t \notin \ _u \quad \Rightarrow \quad y_t = y_{\bullet t} + @ \times x_{\bullet t}$ $\forall t \in \ _u \quad \Rightarrow \quad y_t = u_t$	**on** $_u$ *then* $u \rhd y$ *else* ${}^\bullet x \rhd y$
$Eq_7: \quad S_1 \parallel S_2$ $S_1 = (X_1; \Sigma_1)$ $S_2 = (X_2; \Sigma_2)$	$\left(X; {\Sigma_1}^{\uparrow X} \cap {\Sigma_2}^{\uparrow X}\right); \ X = X_1 \cup X_2$	$[\![S_1]\!] \ \parallel \ [\![S_2]\!]$

Table I
Non-standard semantics (mid column) and constructive semantics (right column) of SIMPLEHYBRID.

be abstracted away from the constructive semantics. As an example, the constructive semantics for statement 2 writes as ${}^\bullet x \rhd y$, and the constructive semantics for Eq_5 writes as "**on** τ_u *then* $u \rhd y$ *else* ${}^\bullet x \rhd y$". Using the above notations, the constructive semantics is given in table I, last column.

Avoiding causality circuits: Using the above abstraction, for each given clock configuration of S, the transitive closure of relation $\rhd$ is a *pre-order* on X — by abuse of notation, we call it also $\rhd$. If S is such that $\rhd$ is a *partial order* for any reachable clock configuration (see (14) and below), this means that no causality circuit occurs in S and the different variables can be evaluated according to any order compatible with $\rhd$.

Since no clock occurs on any consequent part of a zero-time causality constraint, the only possible cause of circuits in relation $\rhd$ is via sets of statements of the form Eq_1. We thus formally justify here the rule that no delay-free, derivative-free, data flow circuit should exist in the considered program. If this condition is satisfied, topological sorting yields the due scheduling. We assume this condition to be in force in the remainder of this section.

Single assignment condition: Say that system S obeys the single assignment condition if no variable of S sits on the left hand side of two or more equations. The following holds:

Lemma 4: If S possesses no causality circuit and obeys the single assignment condition, then it is deterministic and partial order $\rhd$ at each clock configuration specifies all correct schedulings for the execution of S.

V. RELATED WORK

Studies on hybrid systems modelers from a semantics point of view are not so numerous. We discuss the few we consider relevant for comparison. First of all, we recall the legacy wo6 0 d [(of)-29Td [55 Td [(partiastico9.967son455(tcnly)stiction)-(ys)gen used in table I, replacing the multi-dimensional instants $(t, 0)$ and $(t, 1)$ of [13], [14]. On another aspect, the work [13], [14] is made complicated by issues of smoothness, Lipschitzness, existence and uniqueness of solutions, Zenoness, etc (see section 6 of [13] on "Ideal Sover Semantics" and section 7 of [14] on "Continuous Time Models"). In our approach those issues do not disappear from the whole process, but they are, sort of, postponed to run time, as wished in our introduction.

The work performed by P. Mosterman and his co-workers at The Mathworks [18] is also very interesting, in its attempt to establish the Simulink modeler on a solid semantic basis. The contribution of the paper is to show how (a restricted class of) variable step solvers can be given a functional *stream* semantics [11]. To achieve this, the class of solvers is first restricted to those relying on *explicit schemes,* as *implicit* ones cannot be put in explicit functional form. The second difficulty consists in the use of iterative solving in order to on-line adapt the variable step size. This mechanism, again, does not have a functional shape since several successive integrations with different step sizes are compared, for a same time interval, in order to select the appropriate step size. [18] proposes to re-cast the above procedure to a functional form by replacing a repeated integration with smaller step size, by its increment with respect to the previous integration. If explicit schemes are used, then an explicit form for this increment can be found and added to the previous integration. Observe that this technique requires using the mechanism of super-dense time since a same time interval is processed several times until adequate step size is found. While this indeed provides a hybrid systems modeler with a stream semantics, this semantics is extremely complex since it explicits the discretization method — in particular, changing the latter changes the semantics. This approach forbids using implicit schemes, although they are valuable from the numerical analysis point of view. We also believe that this method cannot easily support the kind of clock configuration dependent causality analysis such as the one provided by our constructive semantics.

VI. Conclusion

We have proposed a novel approach to give a semantics to hybrid systems modelers. In doing so, we wanted:

- To keep the choice of integration method totally free;
- To ensure that hybrid systems are a conservative extension of discrete time systems;
- To give semantic support for the following:
  - Scheduling the actions triggered by zero-crossings;
  - Using typing to separate discrete from continuous;
  - Rejecting programs with causality circuits;

More objectives are addressed in [2]. Achieving these objectives was made possible thanks to the use of non-standard analysis as our semantic domain. The key point is that non-standard semantics allows cleanly separating the tasks of the computer scientist (answering the above questions) from that of the numerical analyst (tuning the solvers). Also, we believe that non-standard semantics is not a fancy thing for math addicts. It is rather a very natural way of viewing continuous time and hybrid systems from the syntactic side, as the computer scientist usually likes. While the first author was aware of non-standard analysis since the mid eighties, it is only the presentation [15] by Lindstrøm, as reported in [8], that allowed the authors to become familiar with the subject. In [2] we develop a small single-assignment language for hybrid systems modelers, with minimal type system to properly manage the discrete/continuous separation.

Next steps are the study of DAE compliant hybrid systems modelers, such as Modelica, with the same objectives.

Acknowledgement*: The authors are indebted to Ramine Nikoukhah and Sébastien Furic for detailed discussions regarding Modelica, and to Daniel Krob and Simon Bliudze for comments on their work.*

References

[1] A. Robinson, *Non Standard Analysis*. Princeton Landmarks in Mathematics, 1996, ISBN 0-691-04490-2.
[2] A. Benveniste, B. Caillaud, and M. Pouzet, "extended version of this paper," 2010.
[3] A. Benveniste, "Compositional and uniform modelling of hybrid systems," *IEEE Trans. on Automatic Control*, vol. 43, no. 4, pp. 579–584, April 1998.
[4] A. Benveniste, B. Caillaud, and P. L. Guernic, "Compositionality in dataflow synchronous languages: Specification and distributed code generation," *Inf. Comput.*, vol. 163, no. 1, pp. 125–171, 2000.
[5] A. Benveniste, P. Caspi, S. A. Edwards, N. Halbwachs, P. L. Guernic, and R. de Simone, "The synchronous languages 12 years later," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 64–83, 2003.
[6] A. Benveniste, P. L. Guernic, Y. Sorel, and M. Sorine, "A denotational theory of synchronous reactive systems," *Inf. Comput.*, vol. 99, no. 2, pp. 192–230, 1992.
[7] G. Berry, "Constructive Semantics of Esterel: From Theory to Practice (Abstract)," in *AMAST '96: Proceedings of the 5th International Conference on Algebraic Methodology and Software Technology*. London, UK: Springer-Verlag, 1996, p. 225.
[8] S. Bliudze, "Un cadre formel pour l'étude des systèmes industriels complexes: un exemple basé sur l'infrastructure de l'UMTS," Ph.D. dissertation, Ecole Polytechnique, 2006.
[9] S. Bliudze and D. Krob, "Modelling of complex systems: Systems as dataflow machines," *Fundam. Inform.*, vol. 91, no. 2, pp. 251–274, 2009.
[10] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Sangiovanni-Vincentelli, "Languages and tools for hybrid systems design," *Foundations and Trends in Electronic Design Automation*, vol. 1, no. 1/2, 2006.
[11] P. Caspi and M. Pouzet, "A co-iterative characterization of synchronous stream functions," *Electr. Notes Theor. Comput. Sci.*, vol. 11, 1998.
[12] E. A. Lee and A. L. Sangiovanni-Vincentelli, "A framework for comparing models of computation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 17, no. 12, pp. 1217–1229, 1998.
[13] E. A. Lee and H. Zheng, "Operational semantics of hybrid systems," in *HSCC*, 2005, pp. 25–53.
[14] ——, "Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems," in *EMSOFT*, 2007, pp. 114–123.
[15] T. Lindstrom, "An invitation to non standard analysis," in *Nonstandard analysis and its applications*, N. Cutland, Ed. Cambridge Univ. Press, 1988.
[16] N. Cutland (ed.), *Nonstandard analysis and its applications*. Cambridge Univ. Press, 1988.
[17] M. Najafi and R. Nikoukhah, "Modeling Hybrid Automata in Scicos," in *IEEE Multi-conference on Systems and Control*, 2007.
[18] Pieter J. Mosterman and Justyna Zander and Gregoire Hamon and Ben Denckla, "Towards Computational Hybrid System Semantics for Time-Based Block Diagrams," in *3rd IFAC Conference on Analysis and Design of Hybrid Systems (ADHS'09)*, Zaragoza, Spain, September 2009, pp. 376–385, keynote paper.
[19] Stephen L. Campbell and Jean-Philippe Chancelier and Ramine Nikoukhah, *Modeling and Simulation in Scilab/Scicos*. Springer, 2006, ISBN 0-387-27802-8.