Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

ON THE REPRESENTATION OF TIME IN MODELING & SIMULATION

Fernando J. Barros

Universidade de Coimbra
Departamento de Engenharia Informática
3030 Coimbra, PORTUGAL

ABSTRACT

The representation of time plays a key role in the modeling and simulation of dynamic systems. Traditionally, time has been represented by real numbers for continuous and discrete event models, and integer numbers for what is commonly defined as discrete time models. These choices have been found to be insufficient for achieving deterministic models when representing systems subjected to changes in topology or when simultaneous events occur. In this paper we study the advantages of using the set of hyperreal numbers for the time base. For demonstrating the advantages of Hyperreals over the set of reals we use the Hybrid Flow System Specification (HyFlow) formalism. This formalism uses a single hyperreal time base to achieve a unifying representation of sampling and discrete event semantics. We show that a hyperreal time base (HRTB) enables the definition of deterministic, dynamic topology, hybrid systems, while a real time base cannot achieve these fundamental properties.

1 INTRODUCTION

Time representation plays an essential role in M&S of dynamic systems. Time is commonly used to classify models making a crisp separation between formalisms. Models are often classified into discrete and continuous time if they use integers or real numbers as the time base, respectively (Zeigler, Praehofer, and Kim 2000). However, complex models often require the ability to integrate both continuous an discrete time models raising the problem of defining a unifying time semantics. The use of discrete time is commonly used to represent numerical solvers with a fixed time step size. Continuous time is used for representing discrete events systems since signals can be created at non-regular time instants. The use of discrete time offers advantages over continuous time simplifying simulator development, since in general all components undergo a modification at every time step, avoiding the use of complex data-structures to efficiently manage asynchronous events. Time simultaneity has also been a major research challenge. Common approaches use tie-breaking rules in order choose one among the components that can be scheduled at a particular time (Zeigler, Praehofer, and Kim 2000). Other approaches schedule all events with the same due time simultaneously, introducing a change in the traditional time semantic(Preiss 1993). Time simultaneity has also been extend to enable events that occur at different instants but within the same time window to be considered simultaneous (Wieland 1999). These different time semantics raise the problem of model compatibility and reproduction of results.

Another topic, impacting simulation semantics is to decide whether the input value received by a model at time t influences its output at the same time t. Traditionally, discrete time machines use component input to compute its output at the same time. While Mealy machines exhibits this feature, Moore type machines exclude this possibility.

Time warp simulations require the ability to track causality in order to produce a correct sequence of events. If, for example, event A occurring at time t causes event A' occurring at the same time t, the parallel semantics cannot be applied since it would violate causality. For this case, and since A and A' are simultaneous they need to be triggered together, causing a rollback in time warp simulation. Models with

dynamic topology pose also causality challenges. An event occurring at time t causing the deletion of its transmission link at the same time t originates a causality issue. Actually, since the link is deleted at time t the event could not be transmitted and the link could not have been removed, causing a contradiction.

At a first glance all choices presented so far look valid and none seems to be intrinsically better and more accurate that the others. The choice among these alternative semantics looks an arbitrary decision that can be made in order to provide a more convenient representation for certain types of models. If we take, for example, discrete time models with a Moore type, if model output at a time t cannot take into account the input at time t the model will be forced to wait until the next time step so the input can take effect into the output. This situation is quite limiting for represent a large variety of systems since this delay can introduce large errors, in particular when numerical integrators are involved. On the other hand Mealy machines do not have this limitation, but they have problems in representing models with loops, requiring the use of priorities and tie-breaking mechanisms for solving these algebraic loops. Thus the parallel/simultaneous transition semantics mentioned before cannot be apply.

In this paper we describe the Hyperreal Time Base (HRTB) a representation of time based on the set of Hyperreal numbers $\mathbb{H}$. We show that the HRTB provides a simple solution for the semantics problems raised by a conventional time representation based on real or integer numbers, enabling a unifying time base for dynamic systems. To demonstrate the HRTB we use the Heterogeneous Flow System Specification Formalism (HyFlow), a formalism for describing systems with both discrete event and sampling semantics.

2 Preliminaries

In this section we describe the semantics of hyperreal numbers that are used for defining a time base for dynamic systems. We introduce also the concepts of flow, causality and partial states.

2.1 Hyperreal Numbers

Hyperreal numbers, $\mathbb{H}$, are an extension to real numbers, $\mathbb{R}$, with the addition of the constructs of infinitesimals and infinite. An infinitesimal can be defined as a number that is smaller than all non-zero positive real numbers. A number i is infinitesimal iff $i < a$ for $a \in \mathbb{R}^+$ (Goldblatt 1998). Conversely, a number l is infinite iff $l > a$ for $a \in \mathbb{R}$.

In our discussion we use one infinitesimal ε and one infinite ∞ for defining the time base for dynamic systems. The use of a single infinitesimal enables a simplified operational semantics for hyperreals without compromising expressiveness (Barros 1998), (Barros 2000), (Barros 2008). Since time advance requires mainly the definition of the addition and comparison operators, we assume hyperreals of the form $(a, z\varepsilon)$ where $a \in \mathbb{R}$ and $z \in \mathbb{Z}$.

Given two numbers $a + m\varepsilon$ and $b + n\varepsilon$ we define the add operation by:

$$(a + m\varepsilon) + (b + n\varepsilon) = (a + b) + (m + n)\varepsilon \tag{1}$$

The subtract operation is defined by:

$$(a + m\varepsilon) - (b + n\varepsilon) = (a - b) + (m - n)\varepsilon \tag{2}$$

The number zero can be written by:

$$0 \equiv 0 \pm 0\varepsilon \tag{3}$$

The equality operator is defined for two numbers $x, y \in \mathbb{H}$ by:

$$x = y \quad \text{if } x - y = 0 \tag{4}$$

The relational operator $<$ is defined for $r + i\varepsilon \in \mathbb{H}$ and 0 by:

$$r + i\varepsilon < 0 \quad \text{if } r < 0 \vee (r = 0 \wedge i < 0) \tag{5}$$

The relational operator $<$ is defined for two numbers $x, y \in \mathbb{H}$ by:

$$x < y \quad \text{if } x - y < 0 \tag{6}$$

Given an hyperreal $a + z\varepsilon$, the real (standard) part is given by $(a + z\varepsilon)_{std} = a$.

The set of positive hyperreals is defined by:

$$\mathbb{H}_0^+ = \{h : h \in \mathbb{H}, h \geq 0\} \tag{7}$$

The use of hyperreals has been proposed in (Iwasaki, Farquhar, Saraswat, Bobrwo, and Gupta 1995). This work however, did not define the semantics of hyperreals, making its implementation in a modeling formalism or in a programming language an open issue. In particular, this work does not assume hyperreals in the form $z\varepsilon, z \in \mathbb{Z}$, letting semantics of the time base undefined. The same problem is present in (Mosterman, Simko, Zander, and Han 2014) that does not define the semantics of hyperreals failing to identify a unique infinitesimal required, in our work to enable an operational definition of the non-instantaneous assumption. Nevertheless this work fails to define the operational semantics of a formalism based on hyperreals numbers (Mosterman, Simko, Zander, and Han 2014). This contrast with the HyFlow formalism whose operational semantics is fully defined in Section 3.

2.2 Partial States

A component can be in a state $s \in S$ with $S = \{(p,e) |\ p \in P,\ 0 \leqslant e \leqslant \rho(p)\}$ where P is the set of partial states (p-states), e is the time already elapsed in p-state $p \in P$, and $\rho : P \longrightarrow \mathbb{H}_0^+$ is the maximum time interval a component can stay in p-state $p \in P$. The elapsed time in a p-state p is measured from the time instant when the component has entered p-state p until the current time. We constrain our study to digital machines where the number of p-state transitions that a component can undergo is finite in a finite time interval. Under this constraint, p-states are piecewise constant, since variables in a digital computer cannot be updated in a continuous manner. Analog computers do not impose these constraints and can provide a continuous update of p-states, but are out scope of this study. Although, analog computers are nowadays virtually extinct, their model is still used in some formalisms that are based on the existence of continuous variables , (Bliudze and Krob 2009), (Henzinger 1996), (Praehofer 1991). Nevertheless, these formalisms need to be translated into some representation amenable to be described by digital computers, if simulation is required.

2.3 Causality: Non-Instantaneous Change Assumption (NIA)

We consider the non-instantaneous assumption (causality) that describes the time semantics of modeling instances (components) when subjected to a transition (Barros 1998).

Assumption. A component undergoing a transition at time t only changes its p-state at time $t + \varepsilon$ (Barros 1998).

Taking as an example a component undergoing one single transition at time $t \in (-\infty, +\infty)$, the component will be in p-state p during the interval $(-\infty, t]$ and in a new p-state p' in the interval $(t, +\infty) = [t + \varepsilon, +\infty)$ as depicted in Figure 1. The infinitesimal ε provides an alternative notation for discontinuities, where the traditional right discontinuity time t^+ can be expressed by $t + \varepsilon$. The infinitesimal enables a simple notation to specify the occurrence time of a discontinuity that happens after another discontinuity. If, for example, a discontinuity occurs after t^+, its occurrence time can be expressed by $t + 2\varepsilon = t^{++} = (t^+)^+$.

We note that the p-state is well defined during the transition, occurring at time t, that causes the change $p \to p'$. Component p-state is given by p during the interval $[t, t + \varepsilon)$ and p' at time $t + \varepsilon$. More generally, the next p-state p' is computed by a transition function that takes also into account the current input value and the time elapsed in p-state p, as described in Section 3.1.

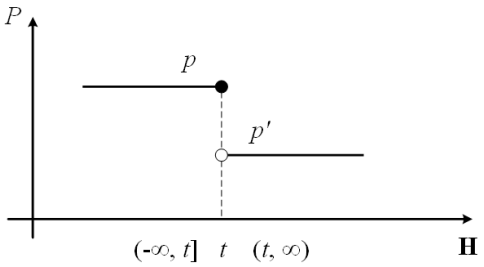

Figure 1: Non-instantaneous assumption(NIA).

This assumption postulates that all transitions take the same amount of time ε to take effect. This postulate enables the use of the operations defined in Section 2.1. Without this assumption it would become virtually impossible to provide a simple operational semantics for representing dynamics systems. For example, the concept of simultaneity would be very difficult to express if the infinitesimals were allowed to be different.

Non-instantaneous change gives an operational definition of causality by defining the time between the cause and its effect as ε. Given the operations on infinitesimals it is possible to define a total order for transitions and to provide deterministic semantics for simultaneous transitions as we show in the next sections. The non-instantaneous assumption enables the use of feedback loops of zero-delays and changes of topology without introducing non-determinism and inconsistencies (Barros 1998).

3 THE HYFLOW FORMALISM

The Heterogeneous Flow System Specification (HyFlow) is a formalism created to represent hybrid modular systems (Barros 2003). HyFlow defines two types of models: basic and network. Basic models provide state representation and state transition functions. Network models are a composition of basic models and/or other network models. To the best of our knowledge HyFlow was the first hybrid formalism to use a HyperReal time-base (Barros 2008). Additionally, the HyFlow formalism merges discrete event system with sampling-based systems that have been traditionally represented by a real time base and a discrete time base, respectively (Zeigler, Praehofer, and Kim 2000), (Manna and Pnueli 1993). The HyFlow formalism proposes a hyperreal time base in order to seamlessly unify these two paradigms.

3.1 HyFlow Basic Model

We describe HyFlow basic models and their semantics. A HyFlow basic model associated with name $B \in \widehat{B}$ is defined by:

$$M_B = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \bar{X} \times \ddot{X}$ is the set of input flow values

$\bar{X}$ is the set of continuous input flow values

$\ddot{X}$ is the set of discrete input flow values

$Y = \bar{Y} \times \ddot{Y}$ is the set of output flow values

$\bar{Y}$ is the set of continuous output flow values

$\ddot{Y}$ is the set of discrete output flow values

P is the set of partial states (p-states)

$\rho : P \longrightarrow \mathbb{H}^0_{+\infty}$ is the time-to-input function

$\omega : P \longrightarrow \mathbb{H}^0_{+\infty}$ is the time-to-output function
$S = \{(p,e) | p \in P, 0 \leq e \leq \nu(p)\}$ is the state set
with $\nu(p) = \min\{\rho(p), \omega(p)\}$, representing the time to transition function
$\delta : S \times X^{\varnothing} \longrightarrow P$ is the transition function
where $X^{\varnothing} = \bar{X} \times (\ddot{X} \cup \{\varnothing\})$
and $\varnothing$ represents the null value (absence of value)
$\bar{\Lambda} : S \longrightarrow \bar{Y}$ is the continuous output function
$\lambda : P \longrightarrow \ddot{Y}$ is the partial discrete output function

HyFlow models describe independent entities that can only communicate through input/output interfaces, defined by sets X and Y, respectively. These sets are structured into continuous and discrete parts since the corresponding HyFlow components can accept and produce hybrid signals. The transition function, δ, describes how a component changes from the current p-state to the next one. Transitions can be triggered by different conditions. A component can change its p-state due the arrival of a discrete flow. A component can also change the p-state according to its autonomous behavior specified by the time-to-input/output functions ρ and τ. When the time elapsed in the current p-state reaches a value specified by one of these time functions, the component changes its p-state. A change can also be triggered by any combination of the previous conditions. The output function $\bar{\Lambda}$ describes the continuous flow output of a component, while the function λ describes the discrete flow. This latter function can only be non-null when the time elapsed in the current p-state reaches the time-to-output function τ.

3.2 HyFlow Basic Component

Models themselves do not provide the semantics of HyFlow entities since they do not specify how rules are applied to dynamic entities that evolve over time. To obtain timed semantics we can use the concept of HyFlow component (Barros 2008). Components have their behavior governed by HyFlow models but they introduce the rules on how models are interpreted.

A HyFlow basic component obeys to the *non-instantaneous assumption* defined in Section 2.3, implying that a transition occurring at time t only takes effect at time $t^+ = t + \varepsilon$. This constraint enables the deterministic simulation of HyFlow models (Barros 2008), particularly at times when a network model undergoes a change in the topology or when there are loops of zero-time delay.

A basic component associated with name $B \in \widehat{B}$ and with model $M_B = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda)$ is defined by:

$$\Xi_B = (\langle s, v \rangle, \mathrm{T}, \mathrm{V}, \Lambda, \Delta)$$

where

$s = (p,t) \in S$, is the component state, where
$S = \{(p,t) | p \in P, t \in \mathbb{H}\}$ is the component state set
p is the component p-state
t is the time of component last transition

$v \in Y^{\varnothing}$, is the component output value, with $Y^{\varnothing} = \bar{Y} \times (\ddot{Y} \cup \{\varnothing\})$

$\mathrm{T} : \{\} \longrightarrow \mathbb{H}$, is the time of the next transition function, defined by:
$\mathrm{T}() = t + \nu(p)$

$\mathrm{V} : \{\} \longrightarrow Y^{\varnothing}$, is the component output function, defined by:

$\mathrm{V}() = v$

$\Lambda : \mathbb{H}$, is the component output action, defined by:

$\Lambda(\tau) \triangleq$

if $(\tau - t = \omega(p))$

$v \leftarrow (\bar{\Lambda}(p, \tau - t), \lambda(p))$

else

$v \leftarrow (\bar{\Lambda}(p, \tau - t), \varnothing)$

$\Delta : \mathbb{H} \times X^{\varnothing}$, is the component transition action, defined by:

$\Delta(\tau, (\bar{x}, \ddot{x})) \triangleq$

if $(\tau \neq \mathrm{T}() \wedge \ddot{x} \neq \varnothing)$ **return**

$s \leftarrow (\delta((p, \tau - t), (\bar{x}, \ddot{x})), \tau + \varepsilon)$,

where by the non-instantaneous propagation, the component state changes only at $\tau + \varepsilon$

3.3 Example: Mealy Machines

HyFlow defines Moore-type machines keeping inputs isolated from outputs. Mealy-type machines, on the contrary, compute output based on its current input. This feature makes it difficult to handle loops formed by connecting machines and do not enable a parallel semantics where all components schedule for the same time can be triggered simultaneously. Since Mealy machines are mainly targeted to represent discrete time systems it looks a good solution to let outputs change instantaneously, as represented in Figure 2, since otherwise the input could only be translated into the output at the next time step. This behavior can hardly be acceptable in many cases, like numerical solvers, since it could compromise accuracy. Given the Hyperreal time base we can express Mealy machines into an equivalent HyFlow model. Let us consider an hypothetical Mealy-type HyFlow discrete output function defined by:

$$\lambda : X \times P \longrightarrow \ddot{Y}$$

We can modify model transition function and schedule a new event in zero time after the input, so the output can reflect it after an infinitesimal as shown in Figure 3. The ouput value can be made a function of the current input by introducing a new transition to a transitory state p'' that takes into account input x at time t. The output at time $t + 2\varepsilon$ corresponds to the Meal-type machine of Figure 2. Thus in practice, a Mealy-type machine can be approximated by a Moore-type machine with the extra cost of an additional transitory (zero-time) transition. Although this semantics departs from fixed time step behavior, it enables self-loops and parallel transitions, as discussed in Section 3.7. Under the hyperreal time base (HRTB), Mealy-type machines are obviously flawed and based on naive considerations of synchronism that can be adopted to simplify simulator development.

Given the semantics enabled by hyperreals it becomes clear that real numbers provide a coarse approximation that have been used when a rigorous representation is relaxed. Although the infinitesimal part seems to add up reaching potential very large values, the we may choose a non-standard number for specifying the time advance, making transitions occur at standard numbers. If, for example, we require a model to enter into a new p-sate at time $7 + 0\varepsilon$, a time advance of $7 - \varepsilon$ can be specified so the transition takes affect at time 7.

3.4 HyFlow Network Model

HyFlow network models are compositions of HyFlow models (basic or other HyFlow network models). Let $\widehat{N}$ be the set of names corresponding to HyFlow network models, with $\widehat{N} \cap \widehat{B} = \{\}$. Formally, a HyFlow

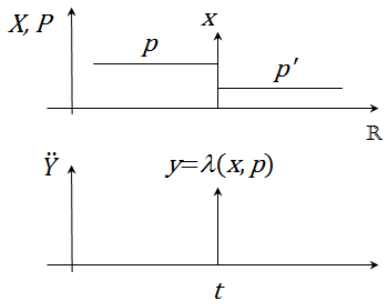

Figure 2: Mealy-type semantics.

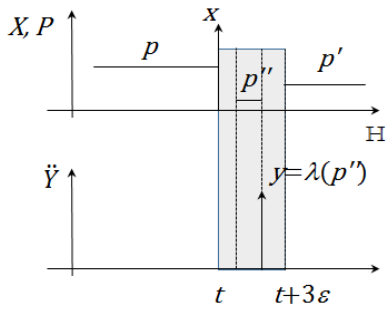

Figure 3: Equivalent HyFlow semantics.

network model associated with name $N \in \widehat{N}$ is defined by:

$$M_N = (X, Y, \eta)$$

where

N is the network name
$X = \bar{X} \times \ddot{X}$ is the set of network input flows
$\bar{X}$ is the set of network continuous input flows
$\ddot{X}$ is the set of network discrete input flows
$Y = \bar{Y} \times \ddot{Y}$ is the set of network output flows
$\bar{Y}$ is the set of network continuous output flows
$\ddot{Y}$ is the set of network discrete output flows
$\eta \in \widehat{\eta}$ is the name of the dynamic topology network executive
with
$\eta \in \widehat{\eta}$ representing the set of all names associated with HyFlow executive models, constrained to $\widehat{\eta} \cap \widehat{B} = \widehat{\eta} \cap \widehat{N} = \{\}$

Executives are uniquely assigned to network models, i.e.,

$\forall_{i,j \in \widehat{N}, i \neq j} \eta_i \neq \eta_j$ with $M_k = (X, Y, \eta)_k, \forall_{k \in \widehat{N}}$

The model of an executive $\eta \in \widehat{\eta}$, is a modified HyFlow basic model, defined by:

$$M_\eta = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda, \widehat{\Sigma}, \gamma)_\eta$$

where

$\widehat{\Sigma}_\eta$ is the set of network topologies
$\gamma_\eta : P_\eta \longrightarrow \widehat{\Sigma}_\eta$ is the topology function

The network topology $\Sigma_\alpha \in \widehat{\Sigma}_\eta$, corresponding to the p-state $p_\alpha \in P_\eta$, is given by the 3-tuple

$$\Sigma_\alpha = \gamma(p_\alpha) \quad = (C_\alpha, \{I_{i,\alpha}\} \cup \{I_{\eta,\alpha}, I_{N,\alpha}\}, F_{i,\alpha} \cup \{F_{\eta,\alpha}, F_{N,\alpha}\})$$

where

C_α is the set of names associated with the executive state p_α
for all $i \in C_\alpha \cup \{\eta\}$
  $I_{i,\alpha}$ is the sequence of asynchronous influencers of i
  $F_{i,\alpha}$ is the input function of i
$I_{N,\alpha}$ is the sequence of network influencers
$F_{N,\alpha}$ is the network output function

For all $i \in C_\alpha$

$M_i = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda)_i$ if $i \in \widehat{B}$
$M_i = (X, Y, \eta)_i$ if $i \in \widehat{N}$

The executive is a special component that controls the network topology. Topology depends on the current executive p-state and it is established by the *topology function* γ. Changes in network topology include the ability to modify composition and coupling through add and delete operations. Although HyFlow relies on a central component to manage the topology of each network, the decision to change this topology can be made by any arbitrary component or in cooperative manner by several components. However, this decision needs to be communicated to the executive so it can become effective. HyFlow topology management guarantees well defined and deterministic behavior when changes in topology occur (Barros 2008).

3.5 Executive Component

Before describing the network component we define the executive component, an extension to the basic component, that introduces the topology function required to establish network current topology. A HyFlow executive component Ξ_η with $\eta \in \widehat{\eta}$, associated with the executive model $M_\eta = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda, \widehat{\Sigma}, \gamma)$ is defined by:

$$\Xi_\eta = (\langle (p, e), v \rangle, \mathrm{T}, \mathrm{V}, \Lambda, \Delta, \Gamma)$$

where

$\Gamma : \{\} \longrightarrow \widehat{\Sigma}$, is the executive component topology function defined by:
  $\Gamma() = \gamma(p)$

Executive component function Γ defines a short notation to describe the current network topology and it is used in the next section to define network semantics and in particular network dynamic topology.

3.6 HyFlow Network Component

A network is composed by the executive component and a set of other components. These components and their interconnections can change according to the current state of the executive. Components can be basic or other HyFlow networks, making possible to define networks hierarchically. A HyFlow network component Ξ_N with $N \in \widehat{N}$, associated with the network model $M_N = (X, Y, \eta)$, and executive $M_\eta = (X_\eta, Y_\eta, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda, \widehat{\Sigma}, \gamma)$, is defined by:

$$\Xi_N = (\langle v \rangle, \mathrm{T}, \mathrm{V}, \Lambda, \Delta)$$

where

$\mathrm{T} : \{\} \longrightarrow \mathbb{H}$, is the maximum time allowed in the current state defined by:

$\mathrm{T}() = \min\{\mathrm{T}_j() | j \in C \cup \{\eta\}\}$
with $(C, \{I_i\} \cup \{I_\eta, I_N\}, F_i \cup \{F_\eta, F_N\}) = \Gamma_\eta()$

$\mathrm{V} : \{\} \longrightarrow Y^\varnothing$, is the component output function, defined by:
$\mathrm{V}() = v$

$\Lambda : \mathbb{H}$, is the network component output action defined by:
$\Lambda(\tau) \triangleq$
forall $i \in C \cup \{\eta\}$ **do** $\Lambda_i(\tau)$
$v \leftarrow F_N(\times_{i \in I_N} \mathrm{V}_i())$
with $(C, \{I_i\} \cup \{I_\eta, I_N\}, F_i \cup \{F_\eta, F_N\}) = \Gamma_\eta()$

$\Delta : \mathbb{H} \times X^\varnothing$, is the component transition action defined by:
$\Delta(\tau, (\bar{x}, \ddot{x})) \triangleq$
if $(\tau \neq \mathrm{T}() \wedge \ddot{x} \neq \varnothing)$ **return**
forall $i \in C$ **do** $\Delta_i(\tau, F_i(\times_{j \in I_i} v_j))$ [A]
do $\Delta_\eta(\tau, F_\eta(\times_{j \in I_\eta} v_j))$ [B]
with

$$v_j = \begin{cases} \mathrm{V}_j() & \text{if } j \neq N \\ (\bar{x}, \ddot{x}) & \text{if } j = N \end{cases}$$

and
$(C, \{I_i\} \cup \{I_\eta, I_N\}, F_i \cup \{F_\eta, F_N\}) = \Gamma_\eta()$, before line B
$(C, \{I_i\} \cup \{I_\eta, I_N\}, F_i \cup \{F_\eta, F_N\})' = \Gamma_\eta()$, after line B

In the component transition action Δ, line A computes the transition in all the components excluding the executive. Since HyFlow components are modular these transitions can be taken in parallel (or in any arbitrary order) since they are independent. Executive transition is only performed in Line B. Thus, changes in topology will only affect the next transition as imposed by the non-instantaneous propagation assumption. Line B also establishes the new network topology that comprises now the current set of components C'. The executive transition removes the set of components $C \setminus C'$ and introduces the new set of components $C' \setminus C$.

3.7 Time-Base for Network Models

We discuss the implications of the non-instantaneous assumption (NIA) in network semantics. An obvious problem if NIA was not valid would be the support for dynamic topologies (Barros 1998). If executive p-state would be allowed to change at time t when, for example, an input arrives, the topology could be modified in such a way that the input would not reach the executive. In this case we would obtain a erroneous behavior since causality would be violated. We have used infinitesimals in 2.1 in order to guarantee that changes in topology cannot affect causality. From this problem we consider that NIA is actually correct, and that formalisms not complying with this assumption are intrinsically flawed since they cannot guarantee causality when changes in topology are involved. We can consider that a universal representation needs to provide a solution for dynamic topologies, and obviously a formalism violating the non-instantaneous assumption cannot provide a correct behavior. We also consider that NIA should either be considered valid or non valid. Pretending that it can be valid only when structural changes are involved is certainly a non-sense.

As mentioned before, the NIA enables also parallel semantics in the presence of algebraic loops. In its simple case a self-loop with zero-time delay introduces a problem when using Mealy machines. Solutions require the use of solvers to decide what is the actual component output, preventing also a parallel semantics. Using the NIA, outputs become decoupled from inputs preventing the very existence of algebraic-loops, greatly simplifying simulation. Actually, the solution of algebraic loops in Mealy machines, like Causal Block Diagrams (Denckla and Mosterman 2005), prevents models from being hierarchical simulated since they need to be flattened before simulation to enable the solution for these loops (Denil, Meyers, Meulenaere, and Vangheluwe 2015).

Another issue is whether transitions can be made sequentially, i.e., one component at a time, or if they must to be done in parallel, i.e., all components scheduled to change at a time t must be triggered at that time. This problem has been open to interpretation and both approaches are considered to be correct, depending on the conveniences of modeling and/or simulator implementation (Zeigler, Praehofer, and Kim 2000). Actually, assuming NIA to be true, triggering sequentially means that for each transition an infinitesimal is added to the simulation time. Thus, if components are triggered sequentially the transition time will depend on the number of components with the same scheduling time. Essentially, a component will undergo a transition at some random time above its specified scheduling time. This non-deterministic semantics can hardly be accepted, i.e., if one does not accept a component to be triggered at a different (standard) time of its specification, one should also not accept that it can be triggered at a different hyperreal time. Given this arbitrary and non-deterministic semantics we assume that the sequential triggering of components that have scheduled to trigger at the same time is intrinsically flawed.

$\mathbb{H}$ also enable a simple description of time warp simulations. Assuming a parallel semantics, an event schedule from time t to the same time t would be considered as simultaneous in a real time base, imposing a rollback. This rollback is actually a causality error since the second event has occurred after the first one. On the contrary, a solution based on the HRTB guarantees causality, preventing a rollback in this situation. So called superdense semantics (SDS) can also be used to prevent this causality error. Differences between HRTB and SDS are described in Section 4.

4 RELATED WORK

Our approach relates to the work on super dense semantics (SDS) (Liu, Matsihoudis, and Lee 2006), (Manna and Pnueli 1993) but there are some key differences. First, we have intended to represent the physical concept of causality, and the infinitesimal time becomes a natural choice. However, no direct physical meaning is associated with the integer tag representation used in SDS. This integer tag is interpreted with the step taken at time t (Manna and Pnueli 1993), implying that after time t the tag is reset to zero. Actually, (Lee and Zheng 2005)[Section 5], defines the *initial value function* at time t to set the tag to zero. Moreover, the tags are constrained to be positive, departing from an hyperreal interpretation where negative values for tags could be used, as described in Section 3.3. We consider that SDS semantics for tags has no physical or mathematical meaning in the perspective of the non-instantaneous assumption defined in Section 2.3. In particular, we found no evidence that (Manna and Pnueli 1993) and (Lee and Zheng 2005) use the hyperreal addition (1) in SDS. In our semantics, the sum of two numbers $a+m\varepsilon$ and $b+n\varepsilon$ would be defined, using SDS, by:

$$(a+m\varepsilon)+(b+n\varepsilon)=\begin{cases}(a+b)+(m+n)\varepsilon & \text{if } a=0\vee b=0\\ (a+b)+0\varepsilon & \text{otherwise}\end{cases} \tag{8}$$

Addition semantics defined in (8) have obviously no mathematical or physical meaning in the perspective of the non-instantaneous assumption defined in Section 2.3. In particular the reset of infinitesimals when adding non-zero numbers can be regarded as a mathematically flawed. Moreover, a trace of a HyFlow component simulated using the hyperreal semantics is, in general different from the traces obtained with super dense semantics. Interesting noting that in a recent work SDS has been dropped and real numbers

have been reintroduced (Tripakis, Stergiou, Shaver, and Lee 2013), for describing the same class of hybrid systems described in(Lee and Zheng 2005) .

We found it also puzzling that, after defining SDS (Manna and Pnueli 1993), authors declare they are not going to apply it: *"... in this paper we continue to use the sampling-computation semantics. The advantages of sampling semantics are that it is simpler than the super-dense semantics..."*. In fact, (Manna and Pnueli 1993) considers discrete time as the most adequate representation for sampling computations. On the contrary, in the HyFlow formalism, sampling computations are achieved under a the hyperreal time-base that are also used to describe discrete event semantics.

In earlier versions of modeling formalisms, like classical DEVS (Zeigler, Praehofer, and Kim 2000), components were triggered sequentially. New versions, like (P)arallel-DEVS make transitions in parallel (Zeigler, Praehofer, and Kim 2000). However, since P-DEVS uses real numbers as the time base, the net effect is that, after a transition, infinitesimals are set to zero, introducing a similar resetting semantics to SDS. As mentioned before, time warp simulations require also the use of a HRTB to support causality. When, at time t, a component schedules an event with a zero-time interval the event occurs at time $t+\varepsilon$, making it non-simultaneous with time t, preventing simulation rollback to time t and thus avoiding a violation of causality.

On conservative parallel simulations (Fujimoto 2000) time management has been extended to include several constructs for guaranteeing deterministic executions in the presence of simultaneous events. *Hidden Time Stamps Fields* (HTSFs) can be used to support a tie-breaking mechanism enabling events to have a distinctive (extended) time. This approach contradicts the HRTB that requires simultaneous events to be handled at the same time. Another approach enables all events occurring at the same time to be delivered simultaneously so they can be handled together. This mechanism conforms to the HRTB. However, since no additional constructs are provided, a zero-time event is considered to have the same time-stamp t of the originating one. If this event is sent again to the producing component it will violate the assumption that all events occurring at time t were delivered. For this approach to work, all components requesting simultaneous events must have a non-zero lookahead (Fujimoto 2000, pg. 86). At this point it becomes clear that the simultaneous treatment of events occurring at the same time is optional, contradicting HRTB. Actually, HRTB lookahead is never zero (and no option is given) since it has a minimum value (by design) of ε. We note that HRTB has a simple and sound mathematical/physical basis, while the time base commonly used in conservative simulation has an *ad-hoc* and inconsistent semantics.

The representation of time using integer tags was originally proposed in (Lamport 1978) with a physical interpretation related to special relativity space-time diagrams, introducing, in our perspective, an unnecessary complexity.

5 CONCLUSION

Time plays an essential role for describing the behavior of dynamic systems. We have shown that the common representations of time based on real or natural numbers cannot correctly describe basic properties of systems like causality. Simulation semantics can be accurately described by hyperreal numbers that supports causal semantics. We have shown that an hyperreal time base enables loops of zero-time delay components, and the correct simulation of dynamic topologies. The HyFlow formalism defines its operational semantics on a hyperreal time base enabling a unified description of hybrid systems- a combination of discrete event and sampling based systems- while guaranteeing causality and determinism.

REFERENCES

Barros, F. 1998. "Handling Simultaneous Events in Dynamic Structure Models". In *SPIE Proceedings: Enabling Methodologies for Simulation*, Volume 3369, 355–363.

Barros, F. 2000. "A Framework for Representing Numerical Multirate Integration Methods". In *AI, Simulation and Planning in High Autonomy Systems*, 149–154.

Barros, F. 2003. "Dynamic Structure Multiparadigm Modeling and Simulation". *ACM Transactions on Modeling and Computer Simulation* 13 (3): 259–275.

Barros, F. 2008. "Semantics of Discrete Event Systems". In *Distributed Event-Based Systems*, 252–258.

Bliudze, S., and D. Krob. 2009. "Modelling of Complex Systems as Dataflow Machines". *Fundamenta Informaticae* 91 (2): 251–274.

Denckla, B., and P. Mosterman. 2005. "Formalizing Causal Block Diagrams for Modeling a Class of Hybrid Dynamic Systems". In *Proceedings of the 44th IEEE Conference on Decision and Control*, 4193–4198.

Denil, J., B. Meyers, P. D. Meulenaere, and H. Vangheluwe. 2015. "Explicit Semantic Adaptation of Hybrid Formalisms for FMI Co-Simulation". In *Proceedings of the Symposium on Theory of Modeling and Simulation*, 99–106.

Fujimoto, R. 2000. *Parallel and Distributed Simulation Systems*. Wiley.

Goldblatt, R. 1998. *Lectures on the Hyperreals: An Introduction to Nonstandard Analysis*. New York: Springer.

Henzinger, T. 1996. "The Theory of Hybrid Automata". In *11th Annual IEEE Symposium on Logic in Computer Science*, 278–292.

Iwasaki, Y., A. Farquhar, V. Saraswat, D. Bobrwo, and V. Gupta. 1995. "Modeling Time in Hybrid Systems: How Fast is Instantaneous?". In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1773–1780.

Lamport, L. 1978. "Time, Clocks, and the Ordering of Events in a Distributed System". *Communications of the ACM* 21 (7): 558–565.

Lee, E., and H. Zheng. 2005. "Operational Semantics of Hybrid Systems". In *Hybrid Systems Computation and Control*, Volume 3414 of *LNCS*, 392–406.

Liu, X., E. Matsihoudis, and E. Lee. 2006. "Modeling Timed Concurrent Systems". In *Proceedings of the 17th International Conference on Concurrency Theory*, 1–15.

Manna, Z., and A. Pnueli. 1993. "Verifying Hybrid Systems". In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, 4–35.

Mosterman, P., G. Simko, J. Zander, and Z. Han. 2014. "A Hyperdense Semantic Domain for Hybrid Dynamic Systems to Model Different Classes of Discontinuities". In *17th International Conference on Hybrid Systems: Computation and Control*, 83–92.

Praehofer, H. 1991. *System Theoretic Foundations for Combined Discrete-Continuous System Simulation*. Ph.d. diss., University of Linz, Austria.

Preiss, B. R. 1993. "The YADDES Distributed Discrete Event Simulation Specification Language and Execution Environment". In *Formal Techniques in Real-Time and Fault-Tolerant Systems*, 4–35.

Tripakis, S., C. Stergiou, C. Shaver, and E. Lee. 2013. "A Modular Formal Semantics for Ptolemy". *Mathematical Structures in Computer Science* 23:834–881.

Wieland, F. 1999. "The Threshold of Event Simultaneity". *Transactions of The Society for Computer Simulation International* 16 (1): 23–31.

Zeigler, B., H. Praehofer, and T. Kim. 2000. *Theory of Modeling and Simulation, 2nd Edition*. Academic Press.

AUTHOR BIOGRAPHIES

Fernando J. Barros is professor at the University of Coimbra, Portugal. His research interests include theory of modeling & simulation and hybrid dynamic topology systems. He published more than 70 papers in journals, book chapters and conference proceedings, and he has organized several conferences and workshops in the area of simulation. Fernando Barros is a member of the editorial board of the Int. J. Simulation and Process Modelling and associate editor of the Int. J. Agent Technologies and Systems. His email address is barros@dei.uc.pt.