Compositional and Uniform Modelling of Hybrid Systems

Albert Benveniste

Abstract **This paper discusses fundamentals of hybrid system modelling. Emphazis is put on compositionality and the use of multiform time. Compositionality refers to the ability of freely composing hybrid systems. Since hybrid systems are considered, different time indices occur, and modularity calls for considering "time" as local to each module, this is what is called multiform time. The proposed framework is behavioral, hence a way is provided to automatically synthesize proper scheduling constraints for the joint simulation of discrete time, event based, and continuous time components. Finally, the relations of this model with the more traditional state based point of view are discussed.**

I. Introduction

Hybrid Systems have been a topic of growing interest and activity in the recent years. It is our opinion that relevant real-life applications of hybrid systems paradigm mainly consist of very large, complex, distributed, systems. For such systems, *modularity* in modelling, simulation, control design, and verification, is mandatory. Thus we shall request for our model that 1/ composition works uniformly for both discrete and continuous parts, and is based on easy and clean mathematical principles, and 2/ "time" shall be local to each module instead of global. Point 1 calls for a composition via the intersection of behaviours, or, equivalently, via conjunction of constraints or systems of equations. Point 2 leads us to consider that "time bases" should not be given once for all, but should rather be *variables* of our hybrid systems. A more extensive introduction and motivations can be found in [1].

This work has been supported in part by NSF/ESPRIT grant Nr. EC-US-043.

IRISA-INRIA, Campus de Beaulieu, 35042 Rennes cedex, France; benveniste@irisa.fr, fax +33 2 99 84 71 71

II. A General Hybrid Systems Model

A. Primitives

Time

Our time index set is $\mathbf{R} = (-\infty, +\infty)$ viewed as a totally ordered dense and continuous set, we also write $\overline{\mathbf{R}} = [-\infty, +\infty]$. This choice only specifies that we work with a continuous, totally ordered, and one-dimensional time. It does not imply, however, that we are bound to a single time t, as we shall see later.

Presences

"Time bases" will be used as index sets for signals, or for "task activation" (we do not define formally what we mean by a task here). A time basis sets is assigned to each signal. Time bases will be referred to in the sequel as "*presences*". A *presence* is any Borel [1] subset $\mathcal{T}$ of $\mathbf{R}$ such that $\mathcal{T} \cap (-\infty, x) = \emptyset$ for some finite x [2]. The family of presences are equipped with the following set theoretic operations: union, intersection, and set difference. Discrete time divergent sequences, as well as intervals, are particular cases of presences.

Signals

A *signal* is a map

$$X \ : \ \mathcal{T} \ \mapsto \ D_X, \quad \text{written } t \mapsto X_t \tag{1}$$

[1] The reader not willing to bother with technicalities can discard the "Borel" assumption. Note however that we need a class of subsets of $\mathbf{R}$ which is invariant under union, intersection, and set difference.

[2] This latter condition expresses that signals have some finite birth date.

where $\mathcal{T}$ is a presence (the presence of X), D_X is some topological space, the *type* of X, and map $t \mapsto X_t$ is *piecewise continuous.* Thus signals are partial functions of time, whose domain we call a "presence". Signals will be used to model trajectories, both discrete or continuous time. In the sequel, for X a signal, we shall denote by $\mathcal{T}_X$ the presence of X.

Delaying signals

If X is a signal with presence $\mathcal{T}_X$, we define its delayed version: $Y = \mathbf{pre}(X)$ **init** x_0, by setting:

$$\begin{aligned} \mathcal{T}_Y &= \mathcal{T}_X = \mathcal{T}, \qquad (2) \\ \forall t \in \mathcal{T} \ : \ Y_t &= \begin{cases} X_{t_-} & \text{if} \quad \mathcal{T} \cap (-\infty, t) \neq \emptyset \\ x_0 & \text{otherwise,} \end{cases} \end{aligned}$$

where X_{t_-} is the left limit of X at t, defined as follows. Let t_- denote the upper boundary of the set $\mathcal{T} \cap (-\infty, t)$. Thus if there are $s \in \mathcal{T}, s < t$ that are arbitrarily close to t, then simply $t_- = t$. In the other hand, if the distance from t to $\mathcal{T} \cap (-\infty, t)$ is strictly positive, then t_- equals the largest instant of $\mathcal{T} \cap (-\infty, t)$. Then we define

$$X_{t_-} = \lim_{s \in \mathcal{T}, s < t_-, s \nearrow t_-} X_s \ . \qquad (3)$$

Also, x_0 is an initial condition required when t is the initial instant of $\mathcal{T}$. See Figure 1 for an illustration of our notion of a delay. From (3) we see that, if t is not isolated from the left, X_{t_-} can be approximated by $X_{t-\varepsilon}$, where ε represents the (possibly variable) discretization step used at the execution of the program. In doing so, we guarantee that no instantaneous causality occurs from X_t to X_{t_-}.

Pointwise constraints on signals

Pick two signals X, Y with presences $\mathcal{T}_X$ and $\mathcal{T}_Y$ respectively. Let $\mathcal{T}$ be a presence. Let $C \subset D_X \times D_Y$ be a constraint defined on the types of X, Y. We shall write $C(x, y)$ to mean that $(x, y) \in C$. We shall say that signals (X, Y) satisfy the constraint

$$\text{on } \mathcal{T} \quad : \quad C(X, Y) \qquad (4)$$

if

$$C(X_t, Y_t) \qquad \forall t \in \mathcal{T} \ , \qquad (5)$$

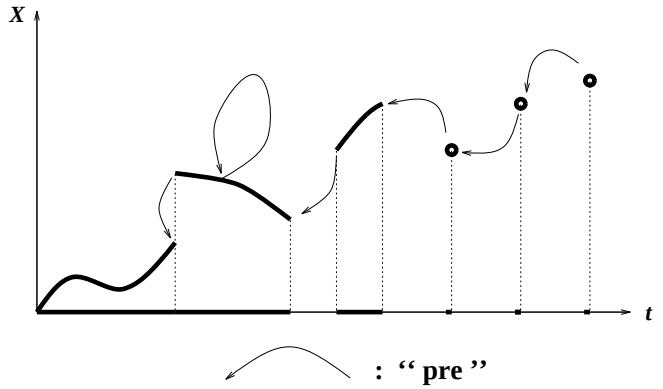

Fig. 1: *The delay operator.* For selected instants t, the backward arrow indicates which value is taken as a definition of $\mathbf{pre}(X)$ **init** x_0. For isolated t points, the previous value is taken and our delay operator behaves like a shift register. For points t in the interior of the presence of X, our delay delivers the value of X "just prior" to t.

this requires in particular that the involved signals are present at each instant of $\mathcal{T}$:

$$\mathcal{T} \quad \subseteq \quad \mathcal{T}_X \cap \mathcal{T}_Y \ . \qquad (6)$$

An interesting particular case is

$$\text{on } \mathcal{T} \quad : \quad Y = f(X) \qquad (7)$$

where f is a function.

Differential Equations and Differential Algebraic Equations (ODE,DAE)

In this subsection, we shall consider signals X of type real, with presence $\mathcal{T}_X$ which we assume to be equal to the closure of its interior. Assume also that $t \mapsto X_t$ is *continuously differentiable in* $\mathcal{T}_X$*, we denote by* $\dot{X}$ *this derivative*[3]. Pick two such signals X, Y with presences $\mathcal{T}_X$ and $\mathcal{T}_Y$ respectively, and a real signal Z with presence $\mathcal{T}_Z$. Let $\mathcal{T}$ be an open presence satisfying

$$\mathcal{T} \quad \subseteq \quad \mathcal{T}_X \cap \mathcal{T}_Y \cap \mathcal{T}_Z \qquad (8)$$

[3]If t belongs to the interior of $\mathcal{T}_X$, then $\dot{X}$ is just the usual derivative, otherwise there is some $\varepsilon > 0$ such that either $(t, t+\varepsilon) \subset \mathcal{T}_X$ or $(t-\varepsilon, t) \subset \mathcal{T}_X$; in the first case we define $\dot{x}_t = \lim_{s>t, s \searrow t} \dot{x}_s$ and in the second case we define $\dot{x}_t = \lim_{s<t, s \nearrow t} \dot{x}_s$.

We shall say that X, Y, Z satisfy the differential equation

$$\text{on } \mathcal{T} \ : \ \frac{dY}{dX} = Z \tag{9}$$

if

$$\dot{Y}_t = \dot{X}_t \, Z_t \quad \text{holds } \forall t \in \mathcal{T} \ . \tag{10}$$

EXAMPLES :

- Of course, one can combine (10) with the equation " on $\mathcal{T} : Z = g(X,Y)$ " to get the Ordinary Differential Equation (ODE) " on $\mathcal{T} : dY/dX = g(X,Y)$ ". On the other hand, combining (10) with the equation " on $\mathcal{T} : C(Z,X,Y)$ " yields the Differential Algebraic Equation (DAE) " on $\mathcal{T} : C(dY/dX, X, Y)$ ".
- For $\mathcal{T}$ of the form $\{t \in \mathbf{R} \ : \ X_t \geq 0\}$, ODE " on $\mathcal{T} : dY/dX = g(X,Y)$ " specifies the differential equation $dy/dx = g(x,y)$, i.e., x is now taken as a "time variable", and the "time domain" in which this DE is defined is $(0, +\infty)$ since the presence of signal X was specified through the predicate $\{X_t \geq 0\}$. This gives an illustration of our mechanism of multiform time for differential equations.
- The standard chain rule for successive differentiations yields the usual chain rule for change of variables in differential equations :

$$\frac{dZ}{dX} = \frac{dZ}{dY}\frac{dY}{dX}$$

B. Hybrid Systems and their Composition

Our model makes use of "real time". However, only *relative* timing is relevant, not absolute time. Thus, prior to defining what is a hybrid system, we need to consider objects and operations "up to time change". This is what we formalize first[4].

1 Time changes

A *time change* is a map $\sigma \ : \ \overline{\mathbf{R}} \mapsto \overline{\mathbf{R}}$, which is bijective, increasing, and continuous with continuous inverse [5]. Time changes change signals in the following way :

$$Y \ =_{\text{def}} \ \sigma(X) \quad \text{defined by} \quad Y_t = X_{\sigma(t)}$$

Lemma 1 (invariance) *The following properties establish invariance under time change of all operations we have defined so far :*

$$\textit{on } S \ : \ C(X,Y) \ \Leftrightarrow \ \textit{on } \sigma(S) \ : \ C\left(\sigma(X), \sigma(Y)\right)$$

$$\textit{on } S \ : \ \frac{dY}{dX} = Z \ \Leftrightarrow \ \textit{on } \sigma(S) \ : \ \frac{d\sigma(Y)}{d\sigma(X)} = \sigma(Z)$$

As a consequence, all operations on signals we have defined so far are invariant under time changes.

2 Hybrid Systems

Consider a set $\{\mathtt{X}_1, \ldots, \mathtt{X}_k\}$ of *signal names,* also called *sort* in the sequel, and associated types $\{D_1, \ldots, D_k\}$.

A *hybrid system P, of sort* $\{\mathtt{X}_1, \ldots, \mathtt{X}_k\}$, is a set of k-tuples $(X_1, \ldots, X_k)$ of signals of respective types $\{D_1, \ldots, D_k\}$, which is *invariant under time change.* This means that, if $(X_1, \ldots, X_k) \in P$ and σ is a time change, then $(\sigma(X_1), \ldots, \sigma(X_k)) \in P$. Note that, in this definition, both the value and presence of signals are variables.

Thanks to the previous subsection, all relations on signals we have introduced so far specify hybrid systems : union/intersection/difference of presences, pointwise constraints on signals (4), differential equations (9).

3 Hybrid Systems Composition

If P and Q have the same sort, then

$$P \| \ Q \ =_{\text{def}} \ P \cap Q \, , \tag{11}$$

where $P \cap Q$ denotes the intersection of sets P and Q. Otherwise, if P has sort $\mathbf{X}$ and Q has sort $\mathbf{Y}$, then 1/ we consider equivalently P as a hybrid system of sort $\mathbf{X} \cup \mathbf{Y}$, which sets no constraint on the signals of names belonging to $\mathbf{Y} \setminus \mathbf{X}$ (where "\" denotes set difference), and symmetrically for Q, and then 2/ we apply definition (11). Note that, since intersection and sort extension both preserve invariance via time change, the composition of hybrid systems is a hybrid system.

[4] For different purposes, the importance of invariance via time change has been recognized and utilized in [6].

[5] In fact, the statement "continuous with continuous inverse" is a consequence of σ being bijective and increasing.

$$
\begin{array}{rrcl}
 & \mathcal{T}_X & = & \mathcal{T}_Y \;=\; \mathcal{T} \\
\| & \mathcal{S} & = & \mathcal{T} \setminus \mathcal{U} \\
\| & \textbf{on } \mathcal{S} & : & dY/dX = f(X,Y) \\
\| & \textbf{on } \mathcal{U} & : & Y = g(U) \\
\| & \mathcal{U} & = & \textbf{true}(\,\textbf{pre}(Y)\ \textbf{init}\ 0 \le 0\,)
\end{array}
$$

TABLE 1: *a* HYBRID *example.* In this example, we assume $f < 0$ and $g > 0$. The meaning of this program is the following. X acts as a "time variable". Y decreases according to differential equation $dY/dX = f(X,Y)$. When Y reaches zero, then immediately signal U is read and differential equation $dY/dX = f(X,Y)$ is reset so that Y takes value $g(U)$. Thus Y has a discontinuity when it reaches zero, hence "reaching zero" must be defined using the left limit of Y, which, by definition, is $\textbf{pre}(Y)$ **init** 0 (we have initialized $\textbf{pre}(Y)$ to the value zero, thus starting immediately by reading U).

EXAMPLES :

- Resetting mechanisms in ODE/DAE. Take X, Y, Z three signals such that $\mathcal{T}_Y$ is a left closed interval, $\mathcal{T}_Z$ is discrete and divergent, and is contained in $\mathcal{T}_Y$, and $\mathcal{T}_X = \mathcal{T}_Y \setminus \mathcal{T}_Z$. Then the composition of statements

$$
\begin{array}{rrcl}
 & \text{on } \mathcal{T}_Y & : & \dfrac{dY}{dX} = g(X,Y) \qquad (12) \\
\| & \text{on } \mathcal{T}_Z & : & Y = h(Z) \qquad (13)
\end{array}
$$

  specifies that Y satisfies ODE (12) with resetting by (13) at the instants of $\mathcal{T}_Z$. In this way we encompass the usual setting (15) of hybrid systems as a particular case.

- For B a boolean signal with presence $\mathcal{T}$, we define the presence

$$
\textbf{true}(B) \;=_{\text{def}}\; \{t \in \mathcal{T} \;:\; B_t = \mathit{true}\}
$$

  to be the instants t at which B_t is true, and similarly for $\textbf{false}(B)$.

The formalism consisting of the previously introduced primitives, plus the latter ones $\textbf{true}(B)$ and $\textbf{false}(B)$, is called HYBRID. Table 1 shows an example of hybrid system specified with HYBRID.

III. TIMING AND CAUSALITY

At this point we have provided a hybrid systems *specification* formalism. It is behavioral in style. This is in accordance with the modern way of buidling simulation models of physical systems from first principles of the physics, as advocated by J.C. Willems. Since it is a formalism of behavioral style, it generally results in programs that are not in operational form (they cannot be directly executed). In this section we investigate issues related to this fact.

Clock calculus and causality analysis are key issues in the compilation of synchronous languages [4]. On the other hand, causality analysis in bond graphs [5] helps preparing and facilitating the work of the ODE/DAE solvers. We propose here a framework which generalizes these notions (we follow the technique presented in [4]). In Table 2, we associate to each HYBRID primitive statement, its *presence calculus* and *causality calculus*. The presence calculus involves only presences and boolean signals, it summarizes the constraints on presences and their relations to boolean signals. Causality calculus uses a new statement, which we term a *causality*, and introduce informally now (a formal definition is provided in Appendix A) :

$$
\text{on } \mathcal{T} \;:\; X \longrightarrow Y \qquad (14)
$$

Its intuitive meaning is : *At each instant of* $\mathcal{T}$*,* Y *cannot be produced before* X*.* The statement "on $\mathcal{T}$" is optional. When it is omitted, then causality $X \longrightarrow Y$ always hold.

Table 2 is commented now. For a signal X, the *status* of X a instant t shall be "absent" if $t \notin \mathcal{T}_X$, and the value of X_t otherwise.

1. One cannot produce a signal before knowing whether it is present or not at the current instant.

2. Enforcing causality "$X \longrightarrow Y$" cannot be performed before knowing the status of $\mathcal{T}$ at the current instant.

3. Here "**op**" denotes union, intersection, or set difference, and $(\mathcal{S},\mathcal{U}) \longrightarrow \mathcal{T}$ indicates that both $\mathcal{S} \longrightarrow \mathcal{T}$ and $\mathcal{U} \longrightarrow \mathcal{T}$ hold.

4. Exact translation of (7) with $\mathcal{T}_X = \mathcal{T}_Y = \mathcal{T}$, induced causality is clear.

5. Exact translation of (4) with $\mathcal{T}_X = \mathcal{T}_Y = \mathcal{T}$, the resulting causality constraints exhibits a circuit

	statement	presence calculus causality calculus
1.		$\mathcal{T}_X \longrightarrow X$
2.	on $\mathcal{T} : X \longrightarrow Y$	$\mathcal{T} \longrightarrow Y$
3.	$\mathcal{T} = \mathbf{op}(\mathcal{S},\mathcal{U})$	$\mathcal{T} = \mathbf{op}(\mathcal{S},\mathcal{U})$ $(\mathcal{S},\mathcal{U}) \longrightarrow \mathcal{T}$
4.	on $\mathcal{T} : Y = f(X)$	$\mathcal{T} \subseteq \mathcal{T}_X \cap \mathcal{T}_Y$ on $\mathcal{T} : X \longrightarrow Y$
5.	on $\mathcal{T} : C(X,Y)$	$\mathcal{T} \subseteq \mathcal{T}_X \cap \mathcal{T}_Y$ on $\mathcal{T} : X \longleftrightarrow Y$
6.	$Y = \mathbf{pre}(X)$ **init** x_0	$\mathcal{T}_Y = \mathcal{T}_X$
7.	on $\mathcal{T} : dY/dX = Z$	$\mathcal{T} \subseteq \mathcal{T}_X \cap \mathcal{T}_Y \cap \mathcal{T}_Z$
8.	$\mathcal{T} = \mathbf{true}(B)$	$\mathcal{T} = \mathbf{true}(B)$ on $\mathcal{T}_B : B \longrightarrow \mathcal{T}$

TABLE 2: *Timing and causality.* In the third column, clock (top) and causality (bottom) calculi associated with each basic statement are shown.

(notation $\leftrightarrow$ means that both $\leftarrow$ and $\rightarrow$ hold), which expresses that none of the two signals X, Y can be computed before the other at the current instant.

6. Exact translation of (2). No instantaneous causality results since X influences only the future of Y, as the discussion following equation (3) shows.

7. Exact translation of (9). No instantaneous causality results since X, Z influence only the future of Y.

8. Selfexplanatory.

Table 2 is used as follows for executing a program. First, for each statement of the program, add to this program the associated presence and causality calculi, following Table 2. This yields a new program, in which relations between the various presences, as well as causality constraints, have been made explicit.

Table 3 depicts the presence and causality calculi associated with the HYBRID program of table 1.

For all instants, at which causality circuits are in force [6], we must solve (differential) algebraic equations, i.e., sophisticated and fragile solvers must be used. This cannot be avoided in general, but is better performed most seldomly.

Since causality constraints, as derived from Table 2, depend on the *syntax* of the program, we can try to rewrite this program differently, with the objective of breaking most possible circuits [4]. Rewriting part of the program involving timing and booleans, can be performed using exactly the techniques of SIGNAL language compilation, since the SIGNAL clock calculus has identical algebraic structure as our presences and boolean signals. Rewriting DAE's (which would cause instantaneous cycles in the graph) into ODE's can sometimes be performed using symbolic calculus, but this may not be desirable in general, due to possible numerical illconditioning of the resulting ODE. A study related to our present notion of causality has been independently performed in the AI-related area in [9] and [10], for static hybrid systems (involving no dynamics, i.e., no ODE), and corresponding software has been developed.

[6] for example: "on $\mathcal{T} : X \longrightarrow Y$" and "on $\mathcal{S} : Y \longrightarrow X$" and $\mathcal{S} \cap \mathcal{T} \neq \emptyset$.

$$
\begin{array}{lrcll}
 & \mathcal{T}_X & = & \mathcal{T}_Y = \mathcal{T} & (1)\\
\| & \mathcal{S} & = & \mathcal{T} \setminus \mathcal{U} & (2)\\
\| & \mathcal{U} & = & \mathbf{true}(\,\mathbf{pre}(Y)\ \mathbf{init}\ 0 \leq 0\,) & (3)\\
\| & & & \mathcal{T} \longrightarrow (X, Y, \mathbf{pre}(Y)) & (4)\\
\| & & & \mathcal{U} \longrightarrow U & (5)\\
\| & & & (\mathcal{T},\mathcal{U}) \longrightarrow \mathcal{S} & (6)\\
\| & (\ \mathbf{on}\ \mathcal{T} & : & \mathbf{pre}(Y) \longrightarrow \mathcal{U} & \\
 & \| & & \mathcal{T} \longrightarrow \mathcal{U}\ ) & (7)\\
\| & (\ \mathbf{on}\ \mathcal{U} & : & U \longrightarrow Y & \\
 & \| & & \mathcal{U} \longrightarrow Y\ ) & (8)\\
\| & (\ \mathbf{on}\ \mathcal{S} & : & (X, Y) \longrightarrow dY/dX & \\
 & \| & & \mathcal{S} \longrightarrow dY/dX\ ) & (9)
\end{array}
$$

TABLE 3: *The example of table 1: presence_and_causality calculus.* This HYBRID program is obtained by applying the rules of Table 2 until fixpoint is reached. To apply these rules, we should have introduced the intermediate signal $Z = f(X, Y)$; this intermediate signal is simply denoted by dY/dX in the above program. Instructions (1–3) are copied from the full program of table 1, these are the constraints relating presences. The rest is the causality calculus. Instructions (4,5) result from rule 1. Instruction (6) follows from rule 3. Instructions (7,8,9) show two causality constraints on each line. These result from rules 4 and 8 (first causality), followed by the subsequent application of rule 2 (second causality). The resulting program exhibits no circuit, hence correct scheduling to execute the simulation follows as explained next. At each instant, 1/ test if this current instant belongs to $\mathcal{T}$, if yes, then 2/ compute $\mathbf{pre}(Y)$ $\mathbf{init}$ 0 and then if the current instant belongs to $\mathcal{U}$ or $\mathcal{S}$. 3/ If it belongs to $\mathcal{U}$ reset the ODE using U. Otherwise simulate a step of the ODE.

IV. Data-flow vs. state based models of hybrid systems

A. *State based models of hybrid systems: the classical view*

According to the most commonly accepted approach today [2], a hybrid system is a continuous time dynamical system having persistent locations and instantaneous transitions occurring at a divergent sequence of instants. Such a triple

$$\boxed{\text{location}} \xrightarrow[\text{transition}]{} \boxed{\text{new location}} \quad (15)$$

has the following form ($\dot{x}$ denotes the time derivative of function $x(t)$, and u, a function of time, is the exogeneous control):

$$
\begin{array}{rcl}
\text{location} & : & \left\{ \begin{array}{l} \dot{x} = F(x,u) \\ x_{\text{init}} = ... \end{array} \right. \\
\text{transition} & : & \dfrac{\mathcal{P}(x,u)\ \vee\ ?\,\text{interrupt}\, I(x_0)}{!\,\text{emit}\, S} \longrightarrow \\
\text{location} & : & \left\{ \begin{array}{l} \dot{x} = G(x,u) \\ x_{\text{init}} = x_0 \end{array} \right.
\end{array}
$$

Thus, locations are characterized by constraints on continuous trajectories — expressed here in the form of differential equations. Transitions are triggered either by *guards,* i.e., predicates on state variables ($\mathcal{P}(x,u)$), or by external interrupts possibly carrying reset values for the new location ($?\,\text{interrupt}\, I(x_0)$). Transitions can output signals of any type (bool, integer, real,...: $!\,\text{emit}\, S$). Here follow some features of this model:

- it is a continuous time systems model, its time index is global and unique, and is $\mathbf{R}_+ = [0, +\infty)$,
- discrete time "timers" occur in the form of the various sequences of "interrupts" or "emit" events,
- events trigger transitions, hence they are the vehicle for switching from one continuous behaviour to another one,
- continuous behaviours in turn can create events (by testing guards over continuous time state variables), this is the mechanism for creating discrete time from continuous time,

– such systems compose only via their discrete parts : discrete signals and events are the only vehicle for systems interaction ; in contrast, continuous state variables are local. At least, ODE specifying locations cannot be freely combined, since their combination generally results in DAE.

As we have noticed, our approach generalizes the classical one.

B. *State based models for* HYBRID

State variables in dynamical systems abstract trajectories according to the Nerode equivalence relation, in which, at each instant t, two trajectories having the same future are equivalent. For classical continuous time dynamical systems, state variables can be chosen by selecting those variables to which derivatives apply. In our case of the HYBRID formalism, state variables originate from the following two items :

1. the "$Y = \mathbf{pre}(X)\ \mathbf{init}\ x_0$" delay operator makes X a state variable ;
2. the statement "on $\mathcal{T}$: $dY/dX = Z$" makes X, Y to be state variables.

Keeping state variables of type (1), and then testing the validity of predicates over these variables is all what is needed to build "locations" in the sense of (15), as clearly shown in the example of Table 1.

V. CONCLUSION

We have proposed a framework for hybrid systems in which continuous and discrete parts are handled in a uniform way including system composition. Our model seems quite general. Clearly, issues related to discretization schemes of differential equations have not been considered. A feature of our model is its behavioral nature. This makes compositionality much easier, which facilitates specification. In turn, this makes program execution nontrivial, and timing and causality calculi are needed. Finally, one can suspect that the techniques of separate compilation and desynchronization developed in particular for SIGNAL could be extended to provide distributed hybrid systems simulation, in which different ODE/DAE solvers cooperate for the simulation of a hybrid systems, and interact at discrete events.

Finally, I should mention the remarkable work of Favret [7], who developed so-called "fake real-time simulation" for high-fidelity joint hybrid simulation of the plant and its control/supervision, taking into account real-time durations for computing control and reactions. "Fake" real-time simulation refers to the ability to perform this without building hardware. The prototype system developed by Favret implements the ideas of multiform time and hybrid simulation we have presented, although our theory has not been used for this.

ACKNOWLEDGEMENT : *Oded Maler is gratefully acknowledged for his highly stimulating, inspiring, and nearly theological remarks, which I did not fully exploit yet. Also, thanks are due to Paul Caspi and Eric Rutten for improving early versions of this manuscript. Also, discussions with André Arnold have clarified the notion of a state in* HYBRID. *Finally, thanks are due to the reviewers for improving an earlier version of the manuscript.*

A APPENDIX : FORMAL DEFINITION OF THE CAUSALITY RELATION

Causality relations have been investigated for several years in the past in the area of models of distributed systems and computations, see for example [4], from which we borrow the essentials of the present technique.

We consider an hybrid system P of sort $\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}$, where $\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}$ denotes the set of the signal names of P.

1. A *preorder* on the set $\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}$ is a relation (generically denoted by $\preceq$) which is reflexive ($x \preceq x$) and transitive ($x \preceq y$ and $y \preceq z$ imply $x \preceq z$). To $\preceq$ we associate the equivalence relation $\asymp$, defined by $x \asymp y$ iff $x \preceq y$ and $y \preceq x$. If equivalence classes of $\asymp$ are singletons, then $\preceq$ is said to be a *partial order.*

   We shall denote by $D_{\preceq}^{\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}}$ the set of all preorders defined on the set $\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}$.

2. A *labelled preorder* $\preceq$ on the set $\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}$ is

a signal of type $D_{\preceq}^{\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}}$, i.e., a map

$$\preceq \quad : \quad \mathcal{T}_{\preceq} \longmapsto D_{\preceq}^{\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}}$$

Thus, as usually for signals, for $t \in \mathcal{T}_{\preceq}$, $\preceq_t$ shall denote the value of signal $\preceq$ at instant t, i.e., some preorder on the set $\{\mathtt{X}_1,\ldots,\mathtt{X}_k\}$.

3. For X, Y signals of the considered hybrid system P, we define

$$\text{on } \mathcal{T} \quad : \quad X \longrightarrow Y \tag{16}$$

as

$$\mathtt{X} \preceq_t \mathtt{Y} \quad \text{holds} \quad \forall t \in \mathcal{T} \tag{17}$$

which consequently requires

$$\mathcal{T} \quad \subseteq \quad \mathcal{T}_X \cap \mathcal{T}_Y \ . \tag{18}$$

The rationale for infering causality constraints in Table 2 is simple.

Actions of computing are abstracted as term rewriting: the computation action $y = f(x)$ is abstracted as "y can be substituted by $f(x)$", which is encoded as the preorder $x \preceq y$, also written $x \longrightarrow y$. For a system of numerical equations, having the resulting graph being circuitfree means that simple substitutions would solve the system — this does not imply that solving the system should be performed in this way, it just expresses that the considered system of equations is not implicit but already explicit. This explains the rules 3 and 4 of Table 2. In particular, in rule 4, no substitution is possible and this is reflected by the occurrence of the circuit $X \longrightarrow Y \longrightarrow X$.

Other rules in this table cannot be explained in this way. But they are justified by the following "pseudo-theorem" (in which we skip the necessary discussion on smoothness assumption required for the ODE solvers to work properly), see [4] for related details on discrete time systems:

Theorem 1 (executable hybrid systems) *Let P be a hybrid system such that*

1. *Referring to (17), for each instant t, the preorder $\preceq_t$ is indeed a partial order.*
2. *There is no double definition, i.e., no presence $\mathcal{T}$ is defined via two different expressions, and there is no instant t such that X_t is multiply defined for some X.*

Then P can be executed using ODE solvers only (meaning that there is no need for a DAE solver or even an Algebraic Equation solver for such an execution).

REFERENCES

[1] A. Benveniste, Compositional and Uniform Modelling of Hybrid Systems (extended version), Inria Res. Rep. to appear, 1997.

[2] R. Alur, C. Courcoubetis, T. Henzinger, and P. Ho, *Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems*, in Hybrid systems, Lecture Notes in Computer Science, vol 736, Springer Verlag, 1993, pp. 209–229.

[3] A. Benveniste, and P. Le Guernic, *Hybrid dynamical systems theory and the* Signal *language,* IEEE Trans. on Autom. Control, AC-35/5, 535–546, 1990.

[4] A. Benveniste, P. Caspi, N. Halbwachs, and P. Le Guernic, Data-flow synchronous languages. In *A Decade of Concurrency, reflexions and perspectives, REX School/Symposium*, pages 1–45, LNCS Vol. 803, Springer Verlag, 1994.

[5] P. Breedveld, R. Rosenberg, and T. Zhou, *Bibliography of bond graph theory and applications*, Journal of the Franklin Institute, 328 (1991), pp. 1067–1109.

[6] T. Henzinger, *Hybrid Automata with Finite Bisimulations,* preprint, Cornell University, 1995.

[7] F. Favret, *Aide à la communication modeleurs/solveurs et contribution à l'analyse des systèmes techniques complexes,* Thesis Ecole Centrale de Paris and Gaz de France, DR, 1995.

[8] P. L. Guernic, T. Gautier, M. L. Borgne, and C. L. Maire, *Programming real-time applications with* Signal, Proceedings of the IEEE, 79 (1991), pp. 1321–1336.

[9] N. Rouquette, *Operationalizing engineering models of steady-state equations into efficient simulation programs,* PhD thesis, USC, Dept. of Computer Science, 1995.

[10] N. Rouquette, *Hierarchical feedback decomposition of steady-state equation models,* Technical Report, Artificial Intelligence Group, Caltech, 1995.

List of Figures