On the composition of hybrid systems*

Sébastien Bornot and Joseph Sifakis
Sebastien.Bornot@imag.fr Joseph.Sifakis@imag.fr

VERIMAG, 2 rue Vignate, 38610 Gières, France

1 Introduction

Concurrent systems can be usually specified as systems of communicating processes obtained by composing sequential processes by means of binary parallel composition operators. The latter express process interaction in terms of action composition. Their semantics is usually defined by two types of rules.

- Synchronization rules that specify how an action of the product process is defined as the result of the (simultaneous) occurrence of two actions in two component processes.
- Interleaving rules, that specify how an action of a component process is an action of the product process. These rules allow some component processes to be idle while the others progress.

Combining synchronization and interleaving rules is essential for the specification of systems as process coordination requires both synchronization and waiting. However, their adequate combination must satisfy two conflicting requirements :

Deadlock-freedom : Deadlocks may appear in the product process as a result of enforcing synchronization, for instance, when two processes are at states from which only non matching synchronization actions can be performed. Such deadlocks can be avoided by using "escape" transitions generated by application of interleaving rules. However, the presence of both synchronization and interleaving actions may imply non maximal progress.

Maximal progress : When synchronization of two actions is possible, interleaving rules, used precisely to avoid deadlocks, may be applicable. Maximal progress means that synchronization is preferred to interleaving when both are possible. This is sometimes achieved by using restriction or hiding operators that prune out interleaving actions.

The above problems are amplified for timed or hybrid systems where time progress is synchronous and waiting times are bounded. This can be easily observed when hybrid specifications are obtained by adding timing constraints to untimed communicating systems specifications, as it has been pointed out in [SY96].

* in LNCS 1386, "Hybrid Systems : Computation and Control", Berkeley, April 1998, pp 49-63.

In [SY96,BS97b] it is claimed that specifying time progress conditions independently from discrete transitions may be source of inconsistencies in specifications. We propose a model where time progress constraints are associated with actions and thus time progress is directly related with the ability of a system to perform actions. This model satisfies the property of *time reactivity* in the sense that if no action is enabled at a state, time can progress.

Following the process algebra approach, we consider discrete (untimed) systems represented as terms generated from a set of abstract actions by using operators such as prefixing, non deterministic choice and parallel composition. We extend the semantics of these operators to hybrid actions.

For a given abstract action a, a *hybrid action* extension of a, is defined as a triple (g_a, d_a, f_a) where g_a and d_a are unary predicates and f_a is a total function on a continuous set of states. The predicate g_a is a *guard* characterizing the states from which a is enabled while d_a is a *deadline* satisfied by all the enabling states at which the action a becomes urgent (time progress is stopped). The function f_a represents the effect of the action when it is executed.

As usually, for a given n-ary operator *op*, the hybrid actions of the term $op(t_1, \ldots, t_n)$ are obtained by composing the hybrid actions of the arguments t_i. We show that the semantics of operators on abstract actions can be extended to hybrid actions in different manners. The extensions have the same semantics for discrete transitions but may differ in urgency (ability to perform actions within a given delay).

We assume that parallel composition of two discrete systems can be expressed as the non-deterministic choice of terms starting with interleaving or synchronization actions (by means of some expansion theorem [BK85]). The expansion theorem is extended to hybrid actions in the following manner :

- To guarantee maximal progress, non-deterministic choice is replaced by priority choice that gives higher priority to synchronization actions over interleaving actions.
- Synchronization operators between abstract actions are extended to hybrid actions. The guard and the deadline resulting from the synchronization of two hybrid actions depend on the guards and deadlines of the synchronizing hybrid actions. We show that for hybrid actions different synchronization operations of practical interest can be defined by taking as synchronization guards and deadlines modal formulas. In particular, we identify three important synchronization modes : AND-synchronization where the guards of the synchronization action is the conjunction of the guards of the contributing actions. MAX-synchronization used to model synchronization with waiting and for which the synchronization action occurs as soon as all of the contributing actions have been completed. MIN-synchronization where the synchronization action occurs as soon as one of the contributing actions is completed.

The paper is organized as follows. In section 2, we define hybrid extensions of discrete systems as a labeling homomorphism that extends prefixing and choice operators. Section 3 presents a framework for parallel composition of hybrid systems as an extension of parallel composition of untimed systems. For the three basic synchronization modes parallel composition rules are proposed that guarantee both local deadlock-freedom and maximal progress. We conclude by indicating possible application directions.

2 Hybrid extensions of discrete systems

We consider a simple (discrete) algebra of terms S_A with prefixing and non-deterministic choice. We show that a hybrid extension of S_A can be defined as a labeling of the underlying transition system associating with a state s, an evolution function $\triangleright_s$ and with any action a a hybrid action $h(a)$.

2.1 Discrete systems

Consider the language of terms S_A defined by the grammar

$$s ::= \ Nil \ \mid \ a.s \ \mid \ s + s$$

where Nil is a constant and $a \in A$, a set of atomic actions.

With a term of S_A we associate transition relations subsets of $S_A \times A \times S_A$ defined by

$a.s \xrightarrow{a} s$

$s_1 \xrightarrow{a} s_1'$ implies $s_1 + s_2 \xrightarrow{a} s_1'$ and $s_2 + s_1 \xrightarrow{a} s_1'$

We consider that $+$ is an associative commutative operator with Nil as zero element. Any term s is congruent (strongly bisimilar) to a term of the form :

$$s = \sum_{i \in I} a_i.s_i \qquad \text{(taken to be } Nil \text{ if } I = \emptyset)$$

2.2 Hybrid extension of S_A

A hybrid extension of S_A is defined as a pair (V,h) where

- V is a continuous state space isomorphic to $\mathbf{R}^n$ for some $n > 0$
- h is a labeling of S_A such that :
  - $h(s) = (s, \triangleright_s)$, where $\triangleright_s : V \times \mathbf{R}_+ \to V$ is an *evolution function*. We write $v \triangleright_s t$ for $\triangleright_s(v,t)$. We require that $\triangleright_s$ is *additive*, i.e., $\forall v \in V \ \forall t_1, t_2 \in \mathbf{R}_+. \ v \triangleright_s (t_1 + t_2) = (v \triangleright_s t_1) \triangleright_s t_2$.
  - $h(a) = (a, g, d, f)$ where g and d are two unary predicates on V and $f : V \to V$. We suppose that $d \Rightarrow g$. We call g, d, f the *guard*, the *deadline* and the *jump* respectively of the *hybrid action* $h(a)$ associated with a.

The hybrid extension of the term $s = \sum_i a_i.s_i$ is represented by the term $h(s) = \sum_i h(a_i).h(s_i)$.

We define hereafter the semantics of $h(s)$ in two steps. First, we associate transition relations with hybrid actions $h(a_i)$ on the continuous state space V. Then, we define the transition relation of the hybrid extension.

Definition 1. Let $b = (a, g, d, f)$ be a hybrid action associated with a in some transition $s \xrightarrow{a} s'$ of S_A. We define transition relations $\xrightarrow{t}$ for $t \in \mathbf{R}_+$ and $\xrightarrow{a}$ for $a \in A$ subsets of $V \times V$:

- $b : v \xrightarrow{t} v \triangleright_s t$ if $\forall t' < t.\ \neg d(v \triangleright_s t')$
- $b : v \xrightarrow{a} f(v)$ if $g(v)$

The two relations describe the behavior of b from a continuous state v. $b : v \xrightarrow{t} v \triangleright_s t$ means that the execution of b can be delayed for t time units and $b : v \xrightarrow{a} f(v)$ represents the effect of a jump.

Definition 2.
The semantics of $h(s) = \sum_i b_i.h(s_i)$ where $b_i = (a_i, g_i, d_i, f_i)$ and $h(s) = (s, \triangleright_s)$ is defined as a family of labeled transitions, subsets of $(S_A \times V) \times (A \cup \mathbf{R}_+) \times (S_A \times V)$ by the rules

- If $b_i : v \xrightarrow{a_i} v_i$ then $(s, v) \xrightarrow{a_i} (s_i, v_i)$
- If $\forall i \in I.\ b_i : v \xrightarrow{t} v \triangleright_s t$ then $(s, v) \xrightarrow{t} (s, v \triangleright_s t)$.

Remark 3.
Notice that the projection of the transition relations on discrete state components agrees with the transition relations of the associated discrete system. This justifies the use of the term "extension".

Time can advance in $h(s)$ for $s = \sum_i a_i.s_i$ only if all the hybrid actions $h(a_i)$ agree to let time advance. This rule determines a time progress condition associated with s similar to the "invariants" in [ACH+95] and "time progress conditions" in [KMP96]. Associating time progress with actions is an important feature of the presented model as it will be shown throughout the paper. For a given hybrid action, its guard characterizes the states from which the action is possible while its deadline characterizes the subset of the states where the action is enforced by stopping time progress.

The condition $d \Rightarrow g$ guarantees that if no action is enabled from a state then time can progress. In fact, time progress can stop only at states where a guard is enabled. Using terminology from synchronous language [JM94] we call this property *time reactivity*.

The relative position of d with respect to the corresponding g determines the *urgency* of an action. For a given g, the corresponding d may take two extreme values: $d = g$ which means that the action is *eager* and $d = false$ which means that the action is *lazy*. A particularly interesting case is the one of *delayable* action where d is the falling edge of g (cannot be disabled without enforcing its execution) (figure 1).

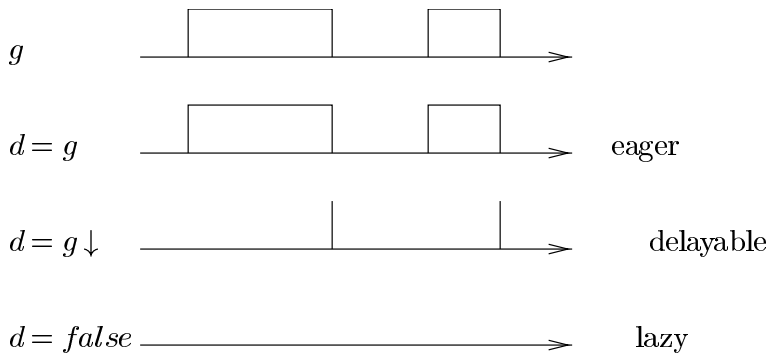

Fig. 1. using deadlines to specify urgency

2.3 Choice operators

Let $B = \{b_i\}_{i \in I}$ be a set of actions $b_i = (a_i, g_i, d_i, f_i)$ labeling transitions issued from a term with evolution function $\triangleright$. We use the modal operators $\Diamond_{\leq k}\ p$ (eventually p within k) and $\diamondsuit_{\leq k}\ p$ (once p since k) where p is a unary predicate on V, and $k \in \mathbf{R}_+ \cup \{\infty\}$.

$\Diamond_{\leq k}\ p\ (v)$ if $\exists t \in \mathbf{R}_+\ 0 \leq t \leq k.\ p(v \triangleright t)$
$\diamondsuit_{\leq k}\ p\ (v)$ if $\exists t \in \mathbf{R}_+\ 0 \leq t \leq k.\ \exists v' \in V.\ v = v' \triangleright t \wedge p(v')$

As usual, we write $\Diamond p$ and $\diamondsuit p$ for $\Diamond_{\leq \infty}\ p$ and $\diamondsuit_{\leq \infty}\ p$ respectively, and $\Box p$ and $\boxdot p$ for $\neg\Diamond\neg p$ and $\neg\diamondsuit\neg p$ respectively.

We have already defined a *non-deterministic* choice operator $\sum_i\ b_i.s_i$ which combines the semantics of hybrid actions in a very simple manner. The discrete transition relation is the union of the discrete transition relations of the hybrid actions b_i and the timed transition relation is the intersection of the timed transition relations of the b_i's. This semantics corresponds to a maximally urgent behavior in the sense that an action may occur when $\vee_i g_i$ holds and time progress stops as soon as $\vee_i d_i$ holds. In practice, it is often useful to define other choice operators with less prompt semantics ([BS97a]). We define a choice operator taking into account priorities between actions. Instead of considering non-deterministic choice between actions $b_i = (a_i, g_i, d_i, f_i)$, for $i = 1, 2$, one can consider that, for instance, b_2 has higher priority than b_1 which leads to restricting the guard and the deadline of b_1 to ${g_1}'$ and ${d_1}'$ respectively. One may take ${g_1}' = g_1 \wedge \neg g_2$ and ${d_1}' = d_1 \wedge {g_1}'$ to resolve conflicts between b_1 and b_2 in favor of b_2. This is a well-known manner to give priority to actions in untimed systems. However, for timed systems priority can concern not only instantaneous conflict resolution but also take into account possibility of waiting. For instance, if we take ${g_1}' = g_1 \wedge \Box\neg g_2$ and ${d_1}' = d_1 \wedge {g_1}'$, we restrict the enabling states of b_1 to only those states from which b_2 will never be enabled.

Definition 4. *priority order*
Consider the relation $< \subseteq A \times (\mathbf{N} \cup \{\infty\}) \times A$. We write $a_1 <_k a_2$ for $(a_1, k, a_2) \in <$ and suppose that

$<_k$ is a partial order relation for all $k \in \mathbf{N} \cup \{\infty\}$

$a_1 <_k a_2 \Rightarrow \forall k' < k.\ a_1 <_{k'} a_2$

$a_1 <_k a_2 \wedge a_2 <_l a_3 \Rightarrow a_1 <_{k+l} a_3$

Property : The relation $a_1 \ll a_2 = \exists k\ a_1 <_k a_2$ is an order relation.

Definition 5. *priority choice operator*
Given $<$, a priority order and $\{b_i.s_i\}_{i \in I}$, a set of term, we define the priority choice operator $\sum_<$ such that :

$$\sum\nolimits_< \{b_i.s_i\}_{i \in I} = \sum_{i \in I} b'_i.s_i$$

where if $b_i = (a_i, g_i, d_i, f_i)$ then $b'_i = (a_i, g'_i, d'_i, f_i)$ with $g'_i = g_i \wedge \bigwedge_{a_i <_k a_j} \neg \Diamond_{\leq k} g_j$ and $d'_i = d_i \wedge g'_i$.

Notice that if $a_i <_k a_j$ then in $\sum b'_i.s_i$ "a_j has higher priority than a_i in the interval $[0, k]$" that is, a_i is disabled if a_j will be enabled within k time units.

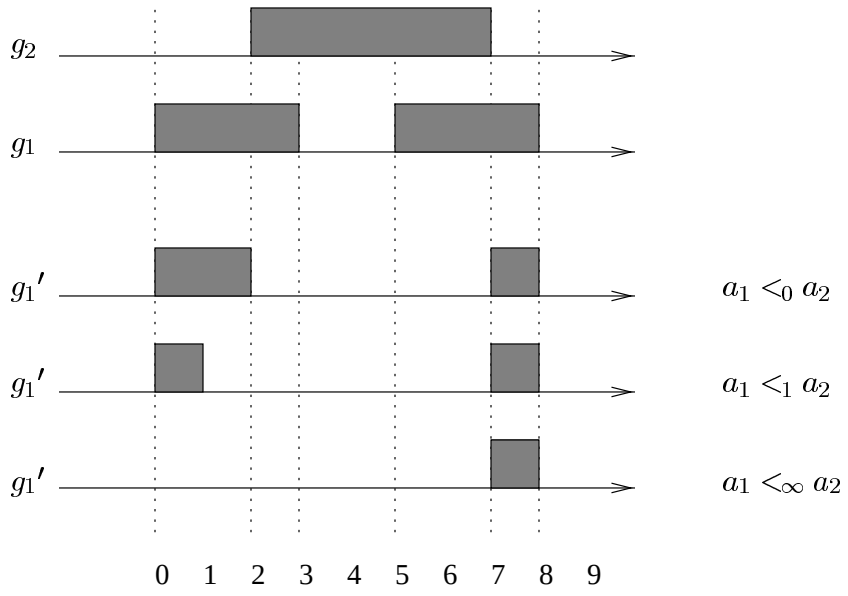

Fig. 2. Different priorities for a_2 over a_1

Consider the guards g_1, g_2 of the actions a_1, a_2. Figure 2 gives the guards g'_1 obtained when g_1 is restricted by considering the priority orders $a_1 <_0 a_2$, $a_1 <_1 a_2$, $a_1 <_\infty a_2$.

Proposition 6. *The priority choice operators defined above satisfy the following properties.*

1. $\Diamond g_i \Rightarrow \Diamond(g'_i \vee \bigvee_{a_i \ll a_j} g_j)$
2. $\Diamond \bigvee_{i \in I} g_i = \Diamond \bigvee_{i \in I} g'_i$

The first property means that if action a_i can occur in the non-prioritized choice then either a_i can occur in the prioritized choice or some action of higher priority.
The second property is a consequence of the first and simply says that $\sum_<$ preserves (local) deadlock-freedom : if some action can be executed in the non-prioritized choice then some action can be executed in the prioritized choice and vice versa.

3 Parallel composition

In this section we define parallel composition operators by following the same approach as in the previous section. First, we show how parallel composition on hybrid systems can be defined as an extension of parallel composition on untimed systems. We thus obtain general composition rules for which some practically interesting cases are discussed later.

3.1 Extending parallel composition from untimed to hybrid systems

Untimed systems We consider a general framework for the composition of untimed terms. For this, we suppose that the vocabulary of actions A contains a distinguished element $\perp$ and consider the set $A_{|}$ of the words generated from A with a commutative operator $|$ such that for all a, $a|\perp = \perp$. The operator $|$ is usually called communication function [BK85]. The words are used to represent synchronization actions that is, actions that result from the synchronous occurrence of atomic actions. $a_1|a_2 = \perp$ means impossibility of synchronization.

In the sequel, we suppose that there are no other simplification rules for $|$ but the rule for $\perp$ and that a word $a_i|a_j$ is given in reduced form.

Consider the language of terms $S_{A_|}$ defined by the grammar

$$s ::= s \in S_A \mid s \| s$$

The semantics of the parallel composition operator is defined by the rules

$$\left\{\begin{array}{l} s_1 \stackrel{a_1}{\rightarrow} s_1' \\ s_2 \stackrel{a_2}{\rightarrow} s_2' \end{array}\right\},\ a_1 | a_2 \neq \bot \text{ implies } \left\{\begin{array}{l} s_1 \| s_2 \stackrel{a_1 | a_2}{\rightarrow} s_1' \| s_2' \\ s_2 \| s_1 \stackrel{a_2 | a_1}{\rightarrow} s_2' \| s_1' \end{array}\right.$$

$$s_1 \stackrel{a_1}{\rightarrow} s_1' \text{ implies } \left\{\begin{array}{l} s_1 \| s_2 \stackrel{a_1}{\rightarrow} s_1' \| s_2 \\ s_2 \| s_1 \stackrel{a_1}{\rightarrow} s_2 \| s_1' \end{array}\right.$$

$\|$ is a commutative operator that can be expressed in terms of non-deterministic choice. It is well-known that for $q_1 = \sum_i\ a_i.s_i$ and $q_2 = \sum_j\ a_j.s_j$,

$$q_1 \| q_2 = \sum_i\ a_i.(s_i \| q_2)\ + \sum_j\ a_j.(s_j \| q_1)\ + \sum_{i,j}\ a_i | a_j.(s_i \| s_j)$$

The first two summands start with interleaving actions while the last one starts with synchronization transitions (only terms such that $a_i | a_j \neq \bot$ appear).

Hybrid extension of $S_{A_\mid}$ For given (V_i, h_i) hybrid extensions of q_i for $i = 1, 2$, a hybrid extension (V,h) for $q_1 \| q_2$ is defined by :

- $V = V_1 \times V_2$
- If $\tau_i = s_i \stackrel{a_i}{\rightarrow} s_i'$ is a transition of q_i then $q_1 \| q_2$ has transitions of the form $\tau = s_1 \| s_2 \stackrel{\lambda}{\rightarrow} s_1' \| s_2'$ where $\lambda = a_i$ or $\lambda = a_1 | a_2$. We take $h(\tau) = (s_1 \| s_2, \triangleright_{s_1} \times \triangleright_{s_2}) \stackrel{h(\lambda)}{\rightarrow} (s_1' \| s_2', \triangleright_{s_1'} \times \triangleright_{s_2'})$ where
  - $h(\lambda) = h_i(a_i)$ if $\lambda = a_i$ and $h(\lambda) = h_1(a_1) | h_2(a_2)$ if $\lambda = a_1 | a_2$ (we extend the communication function in an appropriate manner to hybrid actions, see below).
  - $\triangleright_{s_1} \times \triangleright_{s_2} : (V_1 \times V_2) \times \mathbf{R}_+ \rightarrow V_1 \times V_2$ is such that $(v_1, v_2)(\triangleright_{s_1} \times \triangleright_{s_2})t = (v_1 \triangleright_{s_1} t, v_2 \triangleright_{s_2} t)$.

This definition leads, by taking $b_i = h_1(a_i)$ and $b_j = h_2(a_j)$, to a scheme of expansion theorem for parallel composition where $\oplus$ and $\bigoplus$ are arbitrary choice operators (as defined in the previous section and in [BS97a]) :

$$\begin{array}{rl} h(q_1 \| q_2) = h_1(q_1) \| h_2(q_2) = \sum_i\ b_i.h_1(s_i)\ \|\ \sum_j\ b_j.h_2(s_j) & \\ = \bigoplus_i\ b_i.(h_1(s_i) \| \sum_j\ b_j.h_2(s_j))\ \oplus \bigoplus_j\ b_j.(h_2(s_j) \| \sum_i\ b_i.h_1(s_i)) & \\ \oplus \bigoplus_{i,j}\ (b_i | b_j).(h_1(s_i) \| h_2(s_j)) & \end{array}$$

If $\oplus$ and $\bigoplus$ are non-deterministic choice operators then maximal progress is not guaranteed as an interleaving action may be executed when synchronization is possible. For this reason, we define parallel composition as the priority choice of the expanded terms with infinite priority to synchronization actions $b_i | b_j$ over the interleaving actions b_i and b_j. This corresponds to priority choice for the

minimal order $<$ such that $a_i <_\infty a_{i,j}$ and $a_j <_\infty a_{i,j}$ for any i, j. By using the notations

$$B = \{b_i.(h_1(s_i) \| \sum_j\ b_j.h_2(s_j))\}_i \cup \{b_j.(h_2(s_j) \| \sum_i\ b_i.h_1(s_i))\}_j \\ \cup \{(b_i \natural b_j).(h_1(s_i) \| h_2(s_j))\}_{i,j}$$

and $h_1(q_1) = \sum_i\ b_i.h_1(s_i)$ and $h_2(q_2) = \sum_j\ b_j.h_2(s_j)$, we have $h_1(s_1) \| h_2(s_2) = \sum_< B$ which is equivalent to

$\sum_i\ b_i'.(h_1(s_i) \| h_2(s_2)) + \sum_j\ b_j'.(h_2(s_j) \| h_1(s_1)) + \sum_{i,j}\ b_i \natural b_j.(h_1(s_i) \| h_2(s_j))$
(figure 3)

In the above term, b_i', b_j' are the actions obtained by restricting b_i and b_j due

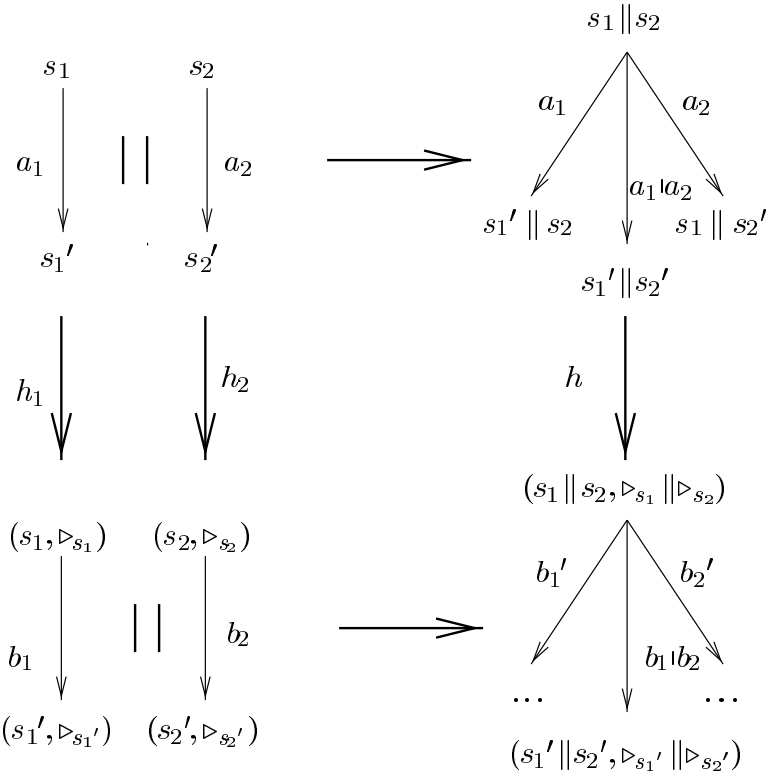

Fig. 3. Hybrid extension for parallel composition

to priority. We now define $b_i \natural b_j$.

Suppose that $h(a_i) = b_i = (a_i, g_i, d_i, f_i)$ for $i \in I$. If $a_i \natural a_j = \perp$ then we take $b_i \natural b_j = \perp$. Otherwise, we write $b_{i,j} = b_i \natural b_j = h(a_i \natural a_j) = h_1(a_i) \natural h_2(a_j) = (a_i \natural a_j, g_{i,j}, d_{i,j}, f_i \times f_j)$ where

$f_i \times f_j : V_1 \times V_2 \to V_1 \times V_2$ such that $(f_i \times f_j)(v_1, v_2) = (f_i(v_1), f_j(v_2))$.
We propose in the next subsection a method for defining $g_{i,j}$ and d_i, j by respecting the requirements $g_{i,j} \Rightarrow g_i \vee g_j$ and $d_{i,j} \Rightarrow d_i \vee d_j$ which mean that $b_{i,j}$ may be caused only by b_i or b_j.

Proposition 7. *If $g_{i,j} \Rightarrow g_i \vee g_j$, the above definition guarantees the following properties*

1. local deadlock-freedom preservation *that is,*

$$\Diamond(\bigvee_{i \in I} g_i \vee \bigvee_{j \in J} g_j) = \Diamond(\bigvee_{i \in I} g'_i \vee \bigvee_{j \in J} g'_j \vee \bigvee_{i \in I, j \in J} g_{i,j})$$

2. maximal progress *that is, interleaving actions are executed only if synchronizations $b_{i,j}$ are disabled forever.*

It is important to notice that these properties hold independently of the way the guards and deadlines of the synchronization actions are defined.

3.2 Synchronization modes of hybrid actions

Given two hybrid actions b_1, b_2 we define the guard $g_{1,2}$ and the deadline $d_{1,2}$ of the hybrid action $b_1 | b_2 = (a_1 | a_2, g_{1,2}, d_{1,2}, f_{1,2})$ resulting from their apropriate synchronization.

Composition of guards : synchronization modes As already discussed in [SY96,BS97b], for timed and hybrid systems the guard $g_{1,2}$ can be in general a modal formula in terms of the guards g_1 and g_2. We consider in particular three important synchronization modes :

AND-synchronization requires that synchronization takes place only when both synchronized transitions can be executed. This means $g_{1,2} = g_1 \wedge g_2$. Consider the example of two synchronizing actions with guards g_1 and g_2. Then, in general interleaving actions are needed to avoid deadlock. Their guards in this case will be $g_1' = g_1 \wedge \Box\neg(g_1 \wedge g_2)$ and $g_2' = g_2 \wedge \Box\neg(g_1 \wedge g_2)$.

MAX-synchronization requires that the first of the two synchronized actions that becomes enabled awaits for the other to become enabled. The enabling of the latest action triggers synchronization. A consequence of this assumption is that waiting may be unbounded. For a given execution trace, the time interval in which the synchronized action is enabled has as lower bound the **maximum** of the times they become enabled and as upper bound the **maximum** of the times they become disabled. The corresponding guard $g_{1,2}$ is defined by $g_{1,2} = (\diamondsuit g_1 \wedge g_2) \vee (g_1 \wedge \diamondsuit g_2)$. For this condition to express synchronization with waiting, it is necessary that if s_1 and s_2 are the source states of the transitions labeled by b_1 and b_2, these states should always be reached with values v_1 and v_2 such that $v_i \models_{s_i} \Diamond g_i$ (remember that the meaning of $\Diamond$ depends of the evolution

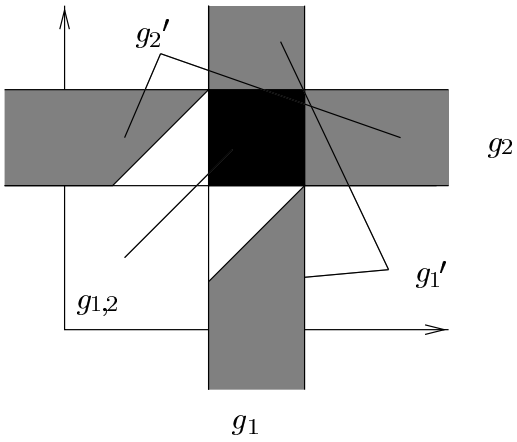

Fig. 4. AND-synchronization

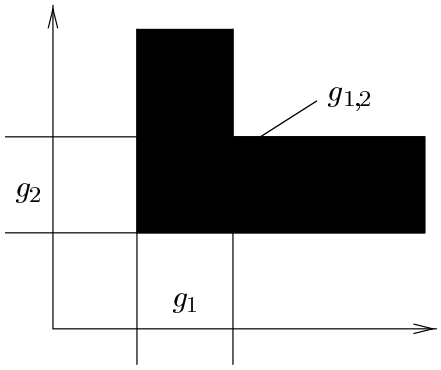

Fig. 5. MAX-synchronization

function $\triangleright_{s_i}$). In the case where there are only two synchronizing actions whose guards are g_1 and g_2, the interleaving actions will have guards $g_1{}' = g_1 \wedge \Box\neg g_{1,2}$ and $g_2{}' = g_2 \wedge \Box\neg g_{1,2}$, which can be simplified into $g_1{}' = g_1 \wedge \Box\boxminus \neg g_2$ and $g_2{}' = g_2 \wedge \Box\boxminus \neg g_1$.

MIN-synchronization is the dual of the previous synchronization mode, and it implies that the synchronization action $a_1 | a_2$ can occur when one of the two synchronizing actions is enabled and the other will be eventually enabled. That is, synchronization may occur in a time interval whose lower bound is the **minimum** of the times they become enabled and the upper bound is the **minimum** of the times they become disabled. The corresponding guard $g_{1,2}$ is described by the formula $g_{1,2} = (\Diamond g_1 \wedge g_2) \vee (g_1 \wedge \Diamond g_2)$. In the case where

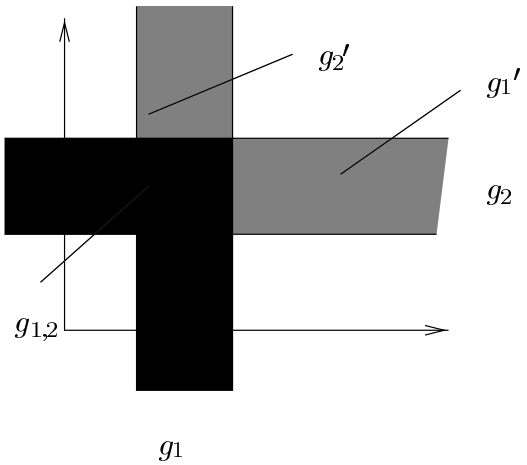

Fig. 6. MIN-synchronization

there are only two synchronizing actions with guards g_1 and g_2, the interleaving actions will have guards $g_1{}' = g_1 \wedge \Box\neg g_{1,2} = g_1 \wedge \Box\neg g_2$ and $g_2{}' = g_2 \wedge \Box\neg g_1$.

Composition of deadlines : typed transitions For two given hybrid actions $b_1 = (a_i, g_i, d_i, f_i)$, $i = 1,2$ the deadline $d_{1,2}$ corresponding to $b_1 | b_2$ must satisfy the following condition

$$d_{1,2} \Rightarrow g_{1,2} \wedge (d_1 \vee d_2)$$

Of course, the most urgent solution is to take $d_{1,2} = g_{1,2} \wedge (d_1 \vee d_2)$ but this often leads to situations where the computed deadline $d_{1,2}$ does not correspond to the intuition [BS97a]. For this reason but also to introduce a simple model where deadlines are defined from guards by means of simple assumptions about urgency of the actions, we slightly modify our model.

We suppose that the deadline d_i of a hybrid action $b_i = (a_i, g_i, d_i, f_i)$ is defined by a function $\delta_i : 2^V \to 2^V$ such that $\delta_i(g_i) = d_i$.

An example of such a function is $\downarrow$ (*falling edge*). When $d_i = g_i\downarrow$ we have a delayable action according to our terminology. Another example is the identity function $\mathbf{1} = \lambda g.g$ which can be used to define eager actions. Finally, a trivial case is the function $\mathbf{0} = \lambda g.false$ that allows to define lazy actions.

We call the function $\delta_i \in \{\mathbf{0}, \downarrow, \mathbf{1}\}$ *types* of the action. Types characterize the urgency of an action which is minimal for $\mathbf{0}$ and maximal for $\mathbf{1}$. Clearly, for synchronization between b_1 and b_2 it is necessary to define $\delta_{1,2}$ such that

$$\boxed{\delta_{1,2}(g_{1,2}) \Rightarrow g_{1,2} \wedge (\delta_1(g_1) \vee \delta_2(g_2))} \qquad (\alpha)$$

Proposition 8. *The following table gives the most urgent type $\delta_{1,2}$ satisfying (α) for any mode (AND, MAX, MIN) in terms of δ_1, δ_2.*

δ_2 \ δ_1	$\mathbf{0}$	$\downarrow$	$\mathbf{1}$
$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
$\downarrow$	$\mathbf{0}$	$\downarrow$	$\downarrow$
$\mathbf{1}$	$\mathbf{0}$	$\downarrow$	$\mathbf{1}$

This result allows to reason only in terms of types of actions and drastically simplifies the general framework.

To complete the results we show that the type of a transition is preserved by priorities and thus the type of interleaving actions is the same as the type of the corresponding synchronizing transitions.

Proposition 9. *If $d_i = g_i$ or $d_i = g_i\downarrow$ and $g_i' = g_i \wedge \Box\neg g$ for some g, then $d_i' = d_i \wedge g_i'$ is such that $d_i' = g_i'$ or $d_i' = g_i'\downarrow$ respectively.*

4 Applications

As an application of the above results, we define a parallel composition operator for typed hybrid actions that is, actions $b_i = (a_i, g_i, \delta_i, f_i)$ such that $\delta_i \in \{\mathbf{0}, \downarrow, \mathbf{1}\}$.

We suppose that for each pair of actions (a_1, a_2) the synchronization mode is given. The resulting interleaving and synchronization actions depend on the synchronization mode. The synchronization action $b_{1,2}$ is $b_{1,2} = (a_1|a_2, g_{1,2}, \delta_{1,2}, f_{1,2})$ where $g_{1,2}$ is defined in 3.2 according to the synchronization mode and $\delta_{1,2}$ is as specified in the table given in 3.2. The interleaving actions b'_i are of the form $b'_i = (a_i, g'_i, \delta'_i, f_i)$ where $g'_i = g_i \wedge \Box\neg g_{1,2}$ and $\delta'_i = \delta_i$ (by proposition 9) for $i = 1, 2$.

Some applications of this general framework can be found in [SY96] where it is shown that for timed Petri nets the underlying synchronization mode is MAX-synchronization. This allows to represent state machine decomposable timed Petri nets as the MAX-parallel composition of timed automata with delayable

actions and makes possible the application of efficient timing analysis techniques to timed Petri nets.

An application domain for our results is modeling of multimedia systems where combinations of the different synchronization modes are necessary for a natural description of timing constraints. Several formalisms used in this area offer such possibilities. One of the most general seems to be the model of Time Stream Petri Nets, by Diaz et al[SDLdSS96]. These are Petri nets with interval time constraints where nine different synchronization modes can be associated with delayable transitions. It can be shown that the guards corresponding to the different synchronization modes can be expressed compositionally as modal formulas in terms of the guards of the components.

We are currently studying the application of the results to define the semantics of the language used in the MADEUS tool for the specification of multimedia documents [JLSIR97]. This language allows the description of timing constraints by means of logical and relational operators used to express causality and synchronization relations. The interesting fact is that very often a combination of the three synchronization types is necessary to specify coordination. The results of the study will be published in [BST97].

5 Discussion

We present a general framework for the composition of hybrid automata. We show that from elementary hybrid actions, choice and parallel composition, complex systems can be defined.

The main difference with other approaches is that we associate with actions time progress conditions which specify for how long an enabled action may wait. Time progress conditions at a given state depend on the urgency of the enabled actions.

The big variety of choice and parallel composition operators results from the different ways enabledness and urgency of components can be combined. Contrary to untimed systems, it is necessary to use modalities to express different kinds of composition that are of practical interest. However, for many tractable subclasses of hybrid automata modal operators can be eliminated, e.g. for linear hybrid automata ([ACH+95]). In that case, modalities are used just for notation convenience and do not modify the basic model.

Different choice operators can be expressed in terms of a basic non-deterministic choice operator which combines the behaviors of the contributing actions so as to obtain maximum urgency. Restricting guards to respect priorities leads to the definition of less prompt choice operators. Other kinds of restrictions remain to be investigated.

Priority choice plays an important role for the definition of a parallel composition operator that respects maximal progress and avoids deadlock by means of appropriate interleaving actions.

The proposed framework is very general. Validation by practice is necessary. It is important to notice that so far AND-synchronization has been used for timed

process algebras and the different timed extensions of the language Lotos [LL95] as well as for timed and hybrid automata. MAX-synchronization is implicitly used in the different extensions of timed Petri nets.

We believe that AND-synchronization is more appropriate for responsive synchronization, where process coordination is supposed to be strong enough to impose that all the timing constraints of the contributing actions are respected. This is often the case for input/output, sender/receiver synchronization where one of the actions is not submitted to deadline constraints. For example, in the train-gate example often mentioned in the literature [ACH+95] communication between the two processes (train and gate) is responsive as the gate reacts to input signals sent by the train. Applying AND-synchronization to obtain the product automaton means that the deadlines and upper bounds of each process must be respected. On the contrary, synchronization between the gate process and a car stopped before the gate should allow for waiting and MAX-synchronization seems more appropriate in this case. We believe that MAX-synchronization should be used to extend parallel composition of asynchronous processes à la CSP. When a hybrid system is obtained as the hybrid extension of an untimed system of communicating automata, it is seems natural to use MAX-synchronization for actions that can wait indefinitely before synchronizing.

Finally, MIN-synchronization corresponds to a kind of (symmetric) interrupt and one can hardly imagine examples where the use of this synchronization mode alone suffices.

Acknowledgement : We thank S. Graf, S. Tripakis, E. Olive as well as M. Jourdan of the Opera project of INRIA for fruitful discussions about possible applications.

References

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[BK85] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, May 1985. Fundamental studies.

[BS97a] S. Bornot and J. Sifakis. *On the composition of hybrid systems (complete version)*. 1997.

[BS97b] S. Bornot and J. Sifakis. Relating time progress and deadlines in hybrid systems. In *International Workshop, HART'97*, pages 286–300, Grenoble, France, March 1997. Lecture Notes in Computer Science 1201, Spinger-Verlag.

[BST97] S. Bornot, J. Sifakis, and S. Tripakis. *Modeling Urgency in Timed Systems*. To appear in COMPOS'97, LNCS, September 1997.

[JLSIR97] M. Jourdan, N. Layaida, L. Sabry-Ismail, and C. Roisin. An integrated authoring and presentation environment for interactive multimedia docu-

ments. In *4th Conference on Multimedia Modeling*, Singapore, November 1997. World Scientific Publishing.

[JM94] M. Jourdan and F. Maraninchi. Studying synchronous communication mechanism by abstractions. In *IFIP Working Conference on Programming Concepts, Methods and Calculi*, San Miniato, Italy, June 1994. Elsevier Science Publishers.

[KMP96] Y. Kesten, Z. Manna, and A. Pnueli. Verifying clocked transition systems. In *School on Embedded Systems*, Veldhoven, The Nederlands, November 1996.

[LL95] G. Leduc L. Léonard. An extended lotos for the design of real-time systems. In *workshop DARTS'95*, Bruxelles, Belgium, November 1995.

[SDLdSS96] P. Sènac, M. Diaz, A. Léger, and P. de Saqui-Sannes. Modeling logical and temporal synchronization in hypermedia systems. In *Journal on Selected Areas in Communications*, volume 14. IEEE, jan. 1996.

[SY96] J. Sifakis and S. Yovine. Compositional specification of timed systems. In *13th Annual Symposium on Theoretical Aspects of Computer Science, STACS'96*, pages 347–359, Grenoble, France, February 1996. Lecture Notes in Computer Science 1046, Spinger-Verlag.