A Fully Parallel Calculus of Synchronizing Processes *

Diego Latella† Paola Quaglia‡

† CNR Ist. CNUCE, Pisa, email: latella@fdt.cnuce.cnr.it **
‡ Dip. di Informatica, Università degli Studi di Pisa, email: quaglia@di.unipi.it

Abstract. We propose a fully parallel calculus of synchronizing processes. The calculus was deeply inspired by LOTOS, of which it inherits *multi-party synchronization* in process parallel composition. On the other hand, its semantics is *not* interleaving whereas LOTOS one is. The model we propose is somehow in between Milner's SCCS and ASCCS in that independent actions are performed simultaneously, whereas synchronization is achieved by means of delay. Also, delay is controlled in the sense that no process can delay an action if the environment allows that action to be performed.
The calculus we propose here was originally designed as a first step towards a probabilistic one. Nevertheless we think that the pure version of the calculus has some features which are interesting on their own. As an example we use it to describe a quite simple system which may be thought of as a possible fault tolerant architecture for a hardware component.
We also provide a set of equational laws based on a notion of *strong bisimulation.*

1 Introduction

In this paper we present a fully parallel calculus of synchronizing processes. The calculus was originally designed as a first step towards a probabilistic one [11]. Several probabilistic models have been proposed in the literature [2,4,5,6,8,13,18,21,22]. They are derived mostly from SCCS [15] which, contrary to CCS [14], has a non-interleaving semantics.

In fact, in order to reason about probabilistic systems, it is a crucial point to have a direct correspondence between *choice operators* in behaviour expressions and the *branching structure* of the transition systems those expressions denote. This is so because on the syntactical level probabilities are associated *only* to the alternatives of choice expressions. On the other hand, from a semantic point of view, the transitions leaving from any state must define, *all together*, a stochastic experiment, which implies all of them being labeled by both an event and a probability value, so that the transition system can be thought of as a Markov system. This is not the case

* This work has been done within the ESPRIT Project Ref. 2304 - LOTOSPHERE and has been partially funded by CPR (Consorzio Pisa Ricerche) and the CNR-NATO Advanced Fellowships Program.

** Current address: Univ. of Twente, Dept. of Comp. Science - PO BOX 217 - 7500 AE Enschede - NL email: latella@cs.utwente.nl

with interleaving semantics since there are branches in transition systems which do not come from non-deterministic choice, but rather from parallel composition.

All the proposals for probabilistic process calculi mentioned above do not allow for *multi-party* synchronization, which is a main feature of CSP [1] and LOTOS [7] and is essential for modeling multi-/broad-casting. The only proposals for dealing with multi-party synchronization for probabilistic processes we know about are [3, 20]. Anyway they both are based on interleaving semantics and force to assign the same (fictitious!) probability to all transitions of branches generated by parallel expressions.

In our calculus, like in LOTOS, parallel composition looks like $B1|G|B2$, where $B1$ and $B2$ are *behaviour expressions* (i.e. processes) which can proceed in parallel but are compelled to simultaneously execute those actions belonging to the list G.

Departing from LOTOS standard semantics, we require (as in SCCS) that every transition corresponds to the *simultaneous* execution of an action by *every* component of the system. So the notion of a single atomic action at a time is replaced by the notion of *as soon as possible* and *composite event*, the latter being denoted by a multi-set. In other terms an action must be executed as soon as the environment makes it possible. For instance, being ';' the prefixing operator, the process $a;stop|[]|b;stop$ performs $\{\{a,b\}\}$ and becomes $stop|[]|stop$ (using '{{' and '}}' as multi-set brackets).

The situation is quite similar to Milner SCCS in that independent actions are executed simultaneously, but the actual action each process performs depends *both* on the actions it is able to perform *and* the synchronization constraints imposed by the parallel context in which it is put, i.e. its environment. In particular, when a process is ready to perform an action which is *not* allowed by the environment, the former is *delayed* and the action will be executed *if* and *as soon as* the synchronization constraints will allow it. In the meanwhile, the process will be forced to *idle*.

With this respect the model is similar to Milner's ASCCS [15] in that it does not force the specifier to explicitly insert idle actions in the specification in order to get the processes synchronized. On the other hand, it differs from the above mentioned calculus since delay is controlled: no process can delay an action if the environment allows that action to be performed. For instance consider the behaviour expression $B1|[a]|B2 = (a;stop)|[a]|(b;a;stop)$. When $B2$ performs b, process $B1$, whose initial action is a synchronization one, namely a, is delayed and executes the special *idling* action λ:

$$B1|[a]|B2 \xrightarrow{\{\{\lambda,b\}\}} B1'|[a]|B2' = (a;stop)|[a]|(a;stop)\,.$$

Now both $B1'$ and $B2'$ can perform action a, so:

$$B1'|[a]|B2' \xrightarrow{\{\{a,a\}\}} (stop|[a]|stop)\,.$$

In conclusion, we call our model *fully parallel* in the sense that it expresses the highest level of parallelism of actions which is allowed by synchronization constraints (i.e. everything which *can* be done *must* be done).

We formalize the concept of delay by means of an operational semantics which defines transition relations parametrized by *delay sets*, i.e. sets of actions which must be delayed. Given any finite set $\Delta \subseteq Gates$ ($Gates$ being the set of observable actions), $B \xrightarrow{\alpha}_{\Delta} B'$ informally means that B can produce the event α and transform in B' when all the actions belonging to Δ are delayed. Given a behaviour expression

its semantics is then the one generated by letting $\Delta = \emptyset$, the intended meaning being obvious. The reason why we do not consider only the transition relation $\longrightarrow_\emptyset$ is that, when a behaviour expression is put in a synchronization context, the delay set Δ is computed according to synchronization constraints. In other terms, in order to deduce $B1|G|B2 \xrightarrow{\alpha}_\emptyset B'$, we need to know $B1 \xrightarrow{\alpha 1}_{\Delta 1} B1'$ and $B2 \xrightarrow{\alpha 2}_{\Delta 2} B2'$ for suitable $\Delta 1$ and $\Delta 2$, as we shall see later.

The rest of the paper is organized as follows. In Section 2 we discuss the formal semantics of the calculus. An example of its application is given in Section 3. In Section 4 we provide a notion of *strong bisimulation equivalence* and a set of equational laws. Section 5 contains directions for future work.

For the sake of simplicity we consider here only a subset of the calculus [1] consisting of *inaction, action prefix, choice, parallel composition, hiding, relabeling,* and *process instantiation.* A process B has the following sintax:

$$B ::= stop \mid \mu; B \mid B[]B \mid B|G|B \mid hide\ g1, \ldots, gn\ in\ B \mid B[a1/f1, \ldots, an/fn] \mid P[a1, \ldots, an]$$

where $\mu, gj, aj, fj \in Gates$ for $1 \leq j \leq n$.

We collect now some notational conventions which are used throughout the paper. Given any synchronization list G, we let G denote also the set with the same elements of the list, furthermore $\alpha, \alpha 1, \alpha 2, \beta, \beta 1, \beta 2 \in \mathcal{E}v$ which is the set of the events, i.e. of the finite multi-sets on $Gates \cup \{\lambda\}$ where $\lambda \notin Gates$. $\{\{a\}\}^+$ denotes the multi-set which contains only a finite, non-zero number of occurrences of a while $\#(a, \alpha)$ denotes the number of occurrences of a in the multi-set α. ∇ is the union operator over multisets and finally $\alpha[a1/f1, \ldots, an/fn]$ denotes the multi-set obtained by simultaneously replacing in α all the occurrences of any fj with aj and, if $\Gamma = \{g1, \ldots, gn\}$, we use $\alpha[a/\Gamma]$ as a shorthand for $\alpha[a/g1, \ldots, a/gn]$.

In the sequel we shall assume the absence of unguarded recursion. Moreover, for the sake of notational simplicity, we shall often let the same symbol denote both a multiset and the set of its elements, the intended meaning being clear from the context.

2 Operational Semantics

We define the operational semantics [17] of the fully parallel calculus by means of an auxiliary set of axioms and deduction rules (Fig.1) [2] which define the relations $\longrightarrow_\Delta$. Let BE be the set of the behaviour expressions; formally $\longrightarrow_\Delta \subseteq (BE \times \mathcal{E}v \times BE)$ where Δ is a finite subset of $Gates$. Then the semantics of a behaviour expression (i.e. the transition relation $\xrightarrow{\alpha}$) can be derived only by means of the following rule:

$$\frac{B \xrightarrow{\alpha}_\emptyset B'}{B \xrightarrow{\alpha} B'}$$

In the sequel we shall suppose that $B, B1, B1', B2$ e $B2'$ range over BE. Also, $B \xrightarrow{\alpha}_\Delta$ is a shorthand for $\exists B' : B \xrightarrow{\alpha}_\Delta B'$.

[1] The reader interested in the whole calculus is referred to [11].

[2] A first, simpler, version of this semantics is presented in [10], there it is also shown that such a simple version is indeed unable to express external nondeterminism.

(st) $\quad stop \xrightarrow{\{(\lambda)\}}_{\Delta} stop$

$(a1)$ $\quad \mu;B \xrightarrow{\{(\mu)\}}_{\Delta} B \ , \ if \ \mu \notin \Delta$

$(a2)$ $\quad \mu;B \xrightarrow{\{(\lambda)\}}_{\Delta} \mu;B \ , \ if \ \mu \in \Delta$

$(c1)$ $\quad B1 \xrightarrow{\alpha}_{\Delta} B1' \ , \ \alpha \notin \{\{\lambda\}\}^+$ implies $B1[]B2 \xrightarrow{\alpha}_{\Delta} B1'$

$(c2)$ $\quad B2 \xrightarrow{\alpha}_{\Delta} B2' \ , \ \alpha \notin \{\{\lambda\}\}^+$ implies $B1[]B2 \xrightarrow{\alpha}_{\Delta} B2'$

$(c3)$ $\quad B1 \xrightarrow{\alpha 1}_{\Delta} B1' \ , \ B2 \xrightarrow{\alpha 2}_{\Delta} B2' \ , \ \alpha 1, \alpha 2 \in \{\{\lambda\}\}^+$ implies $B1[]B2 \xrightarrow{\{(\lambda)\}}_{\Delta} B1'[]B2'$

(pld) $\quad B1 \xrightarrow{\alpha 1}_{G \cup \Delta} B1', \ B2 \xrightarrow{\alpha 2}_{G \cup \Delta} B2', \ Del_{\Delta}(B1, G, \alpha 2)$
implies $B1|G|B2 \xrightarrow{\alpha 1 \nabla \alpha 2}_{\Delta} B1'|G|B2'$

(prd) $\quad B1 \xrightarrow{\alpha 1}_{G \cup \Delta} B1', \ B2 \xrightarrow{\alpha 2}_{G \cup \Delta} B2', \ Del_{\Delta}(B2, G, \alpha 1)$
implies $B1|G|B2 \xrightarrow{\alpha 1 \nabla \alpha 2}_{\Delta} B1'|G|B2'$

(p_b) $\quad B1 \xrightarrow{\alpha 1}_{Cant_{\Delta}(B1|G|B2) \cup \Delta}$, $B2 \xrightarrow{\alpha 2}_{Cant_{\Delta}(B1|G|B2) \cup \Delta}$,
$B1 \xrightarrow{\beta 1}_{Cant_{\Delta}(B1|G|B2) \cup \Delta \cup \Gamma 1} B1'$, $B2 \xrightarrow{\beta 2}_{Cant_{\Delta}(B1|G|B2) \cup \Delta \cup \Gamma 2} B2'$,
$B_{\Delta}(B1, B2, G, \alpha 1, \alpha 2, \beta 1, \beta 2)$
implies $B1|G|B2 \xrightarrow{\beta 1 \nabla \beta 2}_{\Delta} B1'|G|B2'$
where $\Gamma 1 = (\alpha 1 \cap G) \setminus \alpha 2$, $\Gamma 2 = (\alpha 2 \cap G) \setminus \alpha 1$

(h) $\quad B \xrightarrow{\alpha}_{\Delta \setminus \{g1, \ldots, gn\}} B'$ implies $hide \ g1, \ldots, gn \ in \ B \xrightarrow{\alpha[i/g1, \ldots, i/gn]}_{\Delta} hide \ g1, \ldots, gn \ in \ B'$

(r) $\quad B \xrightarrow{\alpha}_{\Gamma} B'$ implies $B[a1/f1, \ldots, an/fn] \xrightarrow{\alpha[a1/f1, \ldots, an/fn]}_{\Delta} B'[a1/f1, \ldots, an/fn]$
where $\Gamma = (\Delta \setminus \{f1, \ldots, fn\}) \cup \{fj : aj \in \Delta\}$

(p) $\quad B[a1/f1, \ldots, an/fn] \xrightarrow{\alpha}_{\Delta} B'$ implies $P[a1, \ldots, an] \xrightarrow{\alpha}_{\Delta} B'$
where $P[f1, \ldots, fn] = B$

Fig. 1. Operational Semantics

The axiom (st) says that *stop* does not perform any action letting time pass.

The axioms for *action prefix* simply say that the atomic action μ, which is the only one ready to be executed, can actually be executed $(a1)$ only if it is not requested to be delayed. If it is not the case $(a2)$, then an *idle* action λ is performed and the process remains unchanged, so that the same action will be ready for execution later.

The interpretation of the rules for hiding, relabeling and process instantiation is straightforward. We simply want to point out that the language of our calculus has indeed also a special *unobservable* action i which we omitted in the presentation above for the sake of simplicity. It corresponds to the τ action of CCS and can never belong to a synchronization list. So in our framework, without making the notation dull, it is sufficient to know that any delay set can never contain i. And in fact the rule for hiding is such that the behaviour of *hide* $g1, \ldots, gn$ *in* B w.r.t. Δ is derived by the one of B w.r.t. a delay set which does not contain the actions of $\{g1, \ldots, gn\}$ which, just as the unobservable action i, have never to be delayed.

As far as relabeling is concerned, simply notice that, in order to infer the be-

haviour of $B[a1/f1,\ldots,an/fn]$ when delayed on Δ, what we must know is the behaviour of B when delayed on those elements belonging to Δ that will not be relabeled and those elements that will be relabeled by gates of Δ.

The semantics for the choice operator is such that the following requirements are met:

- in order for an alternative to be selected for execution it must *not* be the case it is *completely delayed* on Δ, i.e. it must produce an event which is *not* labeled by λs only ((c1), (c2));
- if both alternatives are completely delayed then the choice expression itself is delayed (c3).

For instance the *only* possible transitions of

$$B = (a; stop|[]|a; stop)[](b; stop|[]|b; stop)$$

when delayed respectively on $\emptyset$, $\{a\}$, and $\{a,b\}$, are the following ones:
B—$\{\{a,a\}\} \rightarrow_{\emptyset} (stop|[]|stop)$ and B—$\{\{b,b\}\} \rightarrow_{\emptyset} (stop|[]|stop)$,
B—$\{\{b,b\}\} \rightarrow_{\{a\}} (stop|[]|stop)$,
B—$\{\{\lambda\}\} \rightarrow_{\{a,b\}} B$.
The expression B above performs two transitions of size two when delayed on $\emptyset$ but it executes only one transition w.r.t. the delay sets $\{a\}$ and $\{a,b\}$ (respectively of size two and one).

Observation 1. An increase in the size of the delay set may induce a decrease on the number of transitions of a choice expression as well as the size of the involved events. The size of an event is to be intended as the potential *degree of parallelism* of the event, i.e. the number of processes contributing to its realization. The actual degree of parallelism is of course given by the number of non-λ actions in the event. •

In order to explain the rules for parallel composition we need to introduce the function $\mathcal{I}nit_{\Delta}$ (see Fig.2). $\mathcal{I}nit_{\Delta}(B)$ is recursively defined on the structure of behaviour expressions and contains [3] all the observable actions which B may perform in its first step when delayed on Δ. For instance $\mathcal{I}nit_{\emptyset}((c; stop[]b; stop)|[a,b]|(a; stop[]b; stop)) = \{c,b\}$ and in fact we do not expect that the parallel composition can perform a since a is a synchronization action and one of the partners cannot execute it.

Using $\mathcal{I}nit_{\Delta}$ we can now establish (Def.2) which are the synchronization actions that $B1|G|B2$ is not enabled to perform.

Definition 2. $\forall B1, B2 \in BE$, $\forall G$ synchronization list, $\forall \Delta \subseteq Gates$,
$\mathcal{C}ant_{\Delta}(B1|G|B2) = G \setminus (\mathcal{I}nit_{\Delta}(B1) \cap \mathcal{I}nit_{\Delta}(B2))$. •

Since $B1|G|B2$ cannot execute actions in $\mathcal{C}ant_{\Delta}$, in order to infer the behaviour of the parallel composition when delayed on Δ, we take under consideration *only* the transitions of $B1$ and $B2$ when they are delayed (at least) on $\Delta \cup \mathcal{C}ant_{\Delta}(B1|G|B2)$.

[3] It can be proved [19] that $\forall \mu \in Gates$, if $\mu \in \mathcal{I}nit_{\Delta}(B)$ then $\exists \alpha \in \mathcal{E}v : B \xrightarrow{\alpha}_{\Delta}$ and $\mu \in \alpha$.

$\mathcal{I}nit_\Delta(stop) = \emptyset$

$\mathcal{I}nit_\Delta(\mu; B) = \{\mu\} \setminus \Delta$

$\mathcal{I}nit_\Delta(B1[]B2) = \mathcal{I}nit_\Delta(B1) \cup \mathcal{I}nit_\Delta(B2)$

$\mathcal{I}nit_\Delta(B1|G|B2) = (\mathcal{I}nit_\Delta(B1) \cap G \cap \mathcal{I}nit_\Delta(B2)) \cup (\mathcal{I}nit_\Delta(B1) \cup \mathcal{I}nit_\Delta(B2)) \setminus G$

$\mathcal{I}nit_\Delta(hide\ g1, \ldots, gn\ in\ B) = (\mathcal{I}nit_{\Delta \setminus \{g1,\ldots,gn\}}(B))[i/\{g1, \ldots, gn\}]$

$\mathcal{I}nit_\Delta(B[a1/f1, \ldots, an/fn]) = (\mathcal{I}nit_\Gamma(B))[a1/f1, \ldots, an/fn]$
where $\Gamma = (\Delta \setminus \{f1, \ldots, fn\}) \cup \{fj : aj \in \Delta\}$

$\mathcal{I}nit_\Delta(P[a1, \ldots, an]) = \mathcal{I}nit_\Delta(B[a1/f1, \ldots, an/fn])$
where $P[f1, \ldots, fn] = B$

Fig. 2. $\mathcal{I}nit_\Delta(B)$

How can we combine such transitions? There are essentially three cases to handle; we are going to discuss them letting

$$B1 \xrightarrow{\alpha 1}_{\mathcal{C}ant_\Delta(B1|G|B2) \cup \Delta} \quad \text{and} \quad B2 \xrightarrow{\alpha 2}_{\mathcal{C}ant_\Delta(B1|G|B2) \cup \Delta}$$

First case. Consider the following parallel expression and let $\Delta = \emptyset$:

$$(a; stop)|[a, b]|(b; stop)$$

Both $B1$ and $B2$ are completely delayed by $\mathcal{C}ant_\Delta(B1|G|B2) \cup \Delta$. In such a case also $B1|G|B2$ must be completely delayed. This behaviour can be obtained by pairing transitions of $B1$ and $B2$ delayed at least on $\mathcal{C}ant_\Delta(B1|G|B2) \cup \Delta$.

⊗

Second case. Both $\alpha 1$ and $\alpha 2$ are events not in $\{\{\lambda\}\}^+$. In such a case we can infer the behaviour of $B1|G|B2$ by combining transitions of the partners *provided* that such transitions are 'compatible'. More precisely, it *must not* be allowed to pair different synchronization actions of $B1$ and $B2$.

Take for instance the following expression with $\Delta = \emptyset$:

$$(a; stop[]b; stop)|[a, b]|(a; stop[]b; stop)$$

Of course we do not want to pair $\alpha 1 = \{\{a\}\}$ and $\alpha 2 = \{\{b\}\}$. Then the only transitions which we are allowed to take under consideration are those obtained augmenting the delay set of $B1$ and of $B2$ respectively by

$$\Gamma 1 = (\alpha 1 \cap G) \setminus \alpha 2 \quad \text{and} \quad \Gamma 2 = (\alpha 2 \cap G) \setminus \alpha 1$$

which are the sets of synchronization actions belonging to $\alpha 1$ and not to $\alpha 2$ or vice-versa (with a bit of overloading in the use of the variables αj which actually denote multisets).

Now recall the rules for choice expressions. For any delay set Γ, a choice expression may execute the same event performed by one of its components *only if* such a partner is not completely delayed by Γ. Thus augmenting Γ may result in 'dropping away' one alternative (or even both) of the choice expression (see Obs.1).

This is just the case for $(a; stop[]b; stop)|[a,b]|(a; stop[]b; stop)$ with $\Delta = \emptyset$, $\alpha 1 = \{\{a\}\}$ and $\alpha 2 = \{\{b\}\}$. In fact we have, for instance, $\Delta \cup Cant_{\Delta}((a; stop[]b; stop)|[a,b]|(a; stop[]b; stop)) \cup \Gamma 1 = \{a\}$ and $(a; stop[]b; stop) - \{\{b\}\} \rightarrow_{\{a\}}$. Such a transition has nothing to do with the event $\alpha 1$ previously considered and so we want to find a way for not picking it! Notice that the reason why we got such a 'wrong' transition was that the alternative $\alpha 2$.was such that $\alpha 1$ resulted completely delayed.

So we want to augment the delay set by $\Gamma 1$ (and respectively $\Gamma 2$) and *besides this* we require that $\Gamma 1$ ($\Gamma 2$) does not completely delay the alternative of a choice expression which possibly takes part in the execution of the event $\alpha 1$ ($\alpha 2$). Letting

$$B1 \xrightarrow{\beta 1}_{Cant_{\Delta}(B1|G|B2) \cup \Delta \cup \Gamma 1} \quad \text{and} \quad B2 \xrightarrow{\beta 2}_{Cant_{\Delta}(B1|G|B2) \cup \Delta \cup \Gamma 2}$$

we infer a transition for $B1|G|B2$ by pairing $\beta 1$ and $\beta 2$ only if the above requirement is met. We would like the condition ($\beta 1 = \alpha 1[\lambda/\Gamma 1]$ *and* $\beta 2 = \alpha 2[\lambda/\Gamma 2]$) hold. Actually this is not possible due to the fact that the number of λs can decrease when delay sets grow (see Obs.1). So we have to define a new relation, $\preceq$, which accounts for this (Def.3).

Definition 3. $\forall \alpha, \beta \in \mathcal{E}v$, $\beta \preceq \alpha$ iff $\forall \mu \in Gates$,
$\#(\mu, \beta) = \#(\mu, \alpha)$ *and* $\#(\lambda, \beta) \leq \#(\lambda, \alpha)$. •

$\beta \preceq \alpha$ informally means that β is essentially the same as α except for the fact that it may contain less λs than α. So the above requirement can be formalized as follows:

$$\beta 1 \preceq \alpha 1[\lambda/\Gamma 1] \quad \textit{and} \quad \beta 2 \preceq \alpha 2[\lambda/\Gamma 2]$$

⊗

Third case. Consider the following expression and let $\Delta = \emptyset$, $\alpha 1 = \{\{c\}\}$:

$$(c; stop[](a; stop[]b; stop))|[a,b]|(a; stop[]b; stop)$$

$\alpha 1$ contains observable actions (i.e. $\alpha 1 \notin \{\{\lambda\}\}^+$), and they are not involved in synchronization (i.e. $\alpha 1 \cap G = \emptyset$); on the other hand $B2$ can only synchronize (i.e. $\mathcal{I}nit_{\Delta}(B2) \subseteq G$).

In such a case we expect $B2$ to be completely delayed and the whole process to perform all the actions belonging to $\alpha 1$. Such a behaviour cannot be obtained as above, since the method used in the 'second case' does not work now. It is too weak, so to speak. It lacks of a global view on *all* the alternatives of the partner which must be delayed. In fact, looking at the transitions of $B2$ one at a time ($\alpha 2 = \{\{a\}\}$ and $\alpha 2 = \{\{b\}\}$), we find them, one at a time, completely delayed. In fact we have $\beta 2 \not\preceq \alpha 2$ in both cases, so that no transition of the parallel composition can be inferred for $\alpha 1 = \{\{c\}\}$.

In the present case, and in the symmetric one, the correct behaviour is obtained, indeed, by pairing transitions of $B1$ and $B2$ delayed on $G \cup \Delta$. In fact under the above assumptions $G \cup \Delta$ completely delays the component which is only able to

synchronize ($B2$ in the example) but it does not delay the actions which the other partner can perform alone. Thus the above expression executes the event $\{\{c, \lambda\}\}$.

⊗

As a result of the above discussion we may handle parallel composition by means of three rules. Two of them, which are symmetric, are devoted to situations like those described in the third case. Such rules are (*pld*), i.e. 'parallel left delayed', and (*prd*), i.e. 'parallel right delayed'. They have, among the hypotheses, a predicate (Def.4) which establishes whether the rule is suitable for the case under consideration, that is it says whether a process (partner of a parallel composition) is completely delayed by a given event α (chosen by the other component, which is not delayed), relative to synchronization list G.

Definition 4. $\forall B \in BE$, $\forall G$ synchronization list, $\forall \alpha \in \mathcal{E}v$, $\forall \Delta \subseteq Gates$,
$Del_\Delta(B, G, \alpha) = \alpha \notin \{\{\lambda\}\}^+$ *and* $\alpha \cap G = \emptyset$ *and* $\mathcal{I}nit_\Delta(B) \subseteq G$. •

Finally, the last rule ((*p_b*), i.e. 'parallel both') deals simultaneously with the first and the second cases discussed above. Also this rule has a boolean condition (Def.5) among its hypotheses. It just formalizes our previous discussion.

Definition 5.
$\forall B1, B2 \in BE$, $\forall G$ synchronization list, $\forall \Delta \subseteq Gates$, if
$B1 \xrightarrow{\alpha 1}_{Cant_\Delta(B1|G|B2) \cup \Delta}$, $B2 \xrightarrow{\alpha 2}_{Cant_\Delta(B1|G|B2) \cup \Delta}$,
$\Gamma 1 = (\alpha 1 \cap G) \setminus \alpha 2$, $\Gamma 2 = (\alpha 2 \cap G) \setminus \alpha 1$,
$B1 \xrightarrow{\beta 1}_{Cant_\Delta(B1|G|B2) \cup \Delta \cup \Gamma 1}$, $B2 \xrightarrow{\beta 2}_{Cant_\Delta(B1|G|B2) \cup \Delta \cup \Gamma 2}$, then
$B_\Delta(B1, B2, G, \alpha 1, \alpha 2, \beta 1, \beta 2) =$ *not* $Del_\Delta(B1, G, \alpha 2)$ *and*
not $Del_\Delta(B2, G, \alpha 1)$ *and*
$(\beta 1 \preceq \alpha 1[\lambda/\Gamma 1]$ *and* $\beta 2 \preceq \alpha 2[\lambda/\Gamma 2])$

•

It can be proven [19] that if one of the three conditions $Del_\Delta(B1, G, \alpha 2)$, $Del_\Delta(B2, G, \alpha 1)$, and $B_\Delta(B1, B2, G, \alpha 1, \alpha 2, \beta 1, \beta 2)$ is true then the others are false. Moreover, if $\alpha 1$ and $\alpha 2$ make all the three predicates above false then we *must* deduce *no* transition for $B1|G|B2$. In fact, under that hypothesis, $\alpha 1$ and $\alpha 2$ are such that $(\beta 1 \preceq \alpha 1[\lambda/\Gamma 1]$ *and* $\beta 2 \preceq \alpha 2[\lambda/\Gamma 2])$ does not hold. Therefore at least one of the two transitions (or a choice sub-component) would be completely delayed by the selection of the other, even if it is not the case that one (or both) partner(s) of the parallel composition has (have) to be delayed. This simply means that the particular pairing $\alpha 1$, $\alpha 2$ is not allowed, like $\{\{a\}\}$, $\{\{b\}\}$ in $(a; stop[]b; stop)|[a, b]|(a; stop[]b; stop)$.

We conclude this section with a remark pertaining to the fact that idling gives rise to transitions, a questionable choice 'a priori'. Indeed the presence of transitions labeled by λs only has the major advantage that *any* process *always*, i.e. w.r.t. any delay set, performs an event. Vice-versa, if we remove $\{\{\lambda\}\}^+$ from $\mathcal{E}v$, then any behaviour expression B such that $\mathcal{I}nit_\Delta(B) = \emptyset$ executes *no* event when delayed on Δ. As a result of this, idling transitions could be removed from our model only at the price of having more rules for parallel composition. In fact, on the one hand, rules (*pld*) and (*prd*) would need only some refinements; for instance (*prd*) should become something like $B1 \xrightarrow{\alpha 1}_{G \cup \Delta} B1'$, $\mathcal{I}nit_\Delta(B2) \subseteq G$ implies $B1|G|B2 \xrightarrow{\alpha 1}_\Delta B1'|G|B2$.

But, on the other hand, there should be three different 'versions' of rule (p_b). Such versions should cope with the distinct situations which could arise out of the addition of $\Gamma 1$ and, respectively, $\Gamma 2$ to $Cant_{\Delta}(B1|G|B2) \cup \Delta$. In fact it could be the case that both $B1$ and $B2$ give rise to a transition w.r.t. the new delay sets, as well as the case that *only* $B1$ (or *only* $B2$) can perform an event w.r.t. $Cant_{\Delta}(B1|G|B2) \cup \Delta \cup \Gamma 1$ ($Cant_{\Delta}(B1|G|B2) \cup \Delta \cup \Gamma 2$).

3 An Example

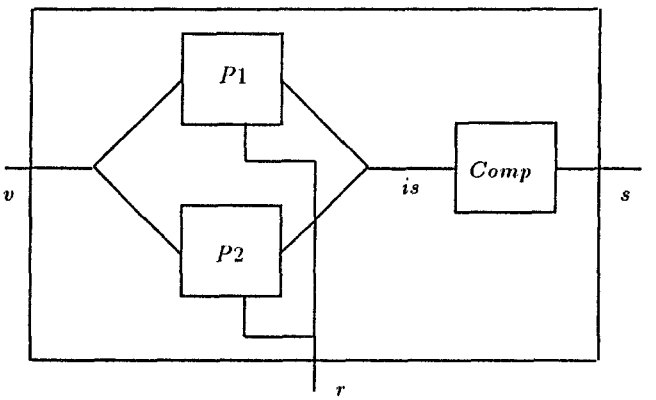

Fig. 3. Overall system

In this section a simple system is described. Its topology is shown in Fig.3. The overall system may be thought of as a possible fault tolerant architecture for a hardware component.

The sub-systems $P1$ and $P2$ are such that each of them, on request from outside (r) reads a value w which has been broadcasted to them via gate v and sends it back on gate *is*. However, this last value is non-deterministically affected by errors ek (Fig.4). We are using here a slightly richer version of the language in which we allow to deal with data values too. Anyway, we assume type IV of values received by the two processes, as well as type OV of their output values, be finite. Under this assumption the specification using data values can be proven equivalent to one without data values.

The third component of the system is a comparator $Comp$ (Fig.5). It receives values w_1 and w_2 from $P1$ and $P2$ respectively and compares them. If $|w_1 - w_2|$ is smaller than or equal to e, for suitable e, then it returns w_1, otherwise it returns $FAIL$.

Finally the formal specification of the overall system is given in Fig.6 whereas Fig.7 shows the labeled transition system of the overall system under the simplifying assumption $IV = \{1, 2\}, OV = \{1, 2, 3\}$, $m = 1$, $e1 = 1$, $e = 0$.

```
process P1[r,v,is] =
 r;v?x1:IV;
 (   i;is!x1?x2:OV;P1[r,v,is]
  [] i;is!(x1+e1)?x2:OV;P1[r,v,is]
  [] i;is!(x1+e2)?x2:OV;P1[r,v,is]
  ...
  ...
  [] i;is!(x1+em)?x2:OV;P1[r,v,is]
 )
endproc

process P2[r,v,is] =
 r;v?x2:IV;
 (   i;is?x1:OV!x2;P2[r,v,is]
  [] i;is?x1:OV!(x2+e1);P2[r,v,is]
  [] i;is?x1:OV!(x2+e2);P2[r,v,is]
  ...
  ...
  [] i;is?x1:OV!(x2+em);P2[r,v,is]
 )
endproc
```

Fig. 4. Specification of processes $Pj, j = 1, 2$

```
process Comp[is,s]
 is?w1:OV;w2:OV;
 ( [|w1-w2|<=e] --> s!w1;Comp[is,s]
   []
   [|w1-w2|> e] --> s!FAIL;Comp[is,s] )
endproc
```

Fig. 5. Specification of process *Comp*

4 Strong Bisimulation Equivalence

In the sequel we propose an adaptation of the notion of *strong bisimulation equivalence* [14,16] to our model.

In our calculus the minimal observational unit is the composite event, then:

Definition 6. A symmetric binary relation $\mathcal{R} \subseteq BE \times BE$ is a *strong bisimulation* iff $B1\mathcal{R}B2$ implies that $\forall \alpha \in \mathcal{E}v$,
if $B1 \xrightarrow{\alpha} B1'$ then $\exists B2' : B2 \xrightarrow{\alpha} B2'$ and $B1'\mathcal{R}B2'$. •

Definition 7. $B1$ and $B2$ are strong bisimulation *equivalent*, $B1 \sim B2$, iff exists a strong bisimulation $\mathcal{R}$ containing $(B1, B2)$. •

```
process PP[r,v,s]
 hide is in
  (P1[r,v,is]|[r,v,is]|P2[r,v,is])|[is]|Comp[s,is]
endproc
```

Fig. 6. Specification of the overall system

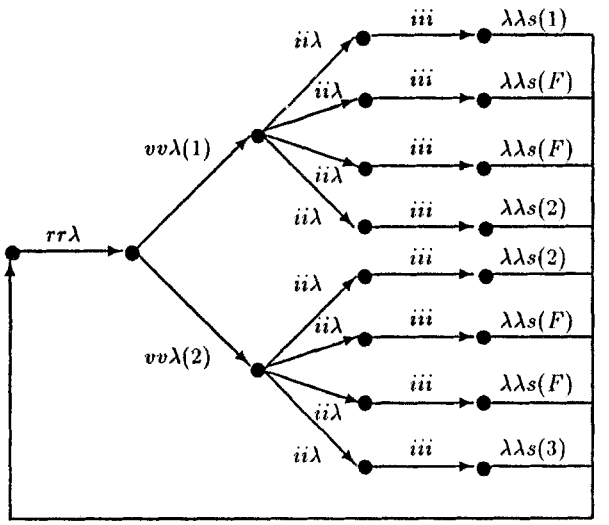

Fig. 7. Labeled Transition System of PP

Then $\sim = \bigcup\{\mathcal{R} | \mathcal{R}$ *is a strong bisimulation*$\}$. In order to prove such an assertion we have to define a function $\mathcal{F}$ on the relations $\mathcal{R}$ such that $(B1, B2) \in \mathcal{F}(\mathcal{R})$ iff Def.6 holds. $\mathcal{F}$ is monotone and $\mathcal{R}$ is a strong bisimulation iff $\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})$, therefore $\sim$ is the greatest fixed point of $\mathcal{F}$ under set inclusion. Moreover, in [19] the following lemmata have been proven:

Lemma 8. *The relation* $\sim$ *is a congruence.* •

Lemma 9. *The following laws hold.*

$B1[]B2 \sim B2[]B1$
$B1[](B2[]B3) \sim (B1[]B2)[]B3$
$B1|G|B2 \sim B2|G|B1$
$B1|G|B2 \sim B1|G'|B2$ if $G = G'$
hide $g1, \ldots, gn$ *in* $B \sim$ *hide* $g1', \ldots, gn'$ *in* B if $\{g1, \ldots, gn\} = \{g1', \ldots, gn'\}$
hide $g1, \ldots, gm$ *in hide* $g1', \ldots, gn'$ *in* $B \sim$ *hide* $f1, \ldots, fh$ *in* B
  if $\{f1, \ldots, fh\} = \{g1, \ldots, gm\} \cup \{g1', \ldots, gn'\}$
hide $g1, \ldots, gn$ *in* $g; B \sim g;$ *hide* $g1, \ldots, gn$ *in* B if $g \notin \{g1, \ldots, gn\}$

$hide\ g1,\ldots,gn\ in\ g; B \sim i; hide\ g1,\ldots,gn\ in\ B$ if $g \in \{g1,\ldots,gn\}$
$hide\ g1,\ldots,gn\ in\ B1[]B2) \sim (hide\ g1,\ldots,gn\ in\ B1)[](hide\ g1,\ldots,gn\ in\ B2)$
$hide\ g1,\ldots,gn\ in\ (B1|G|B2) \sim (hide\ g1,\ldots,gn\ in\ B1)|G|(hide\ g1,\ldots,gn\ in\ B2)$
if $\{g1,\ldots,gn\} \cap G = \emptyset$
$stop[S] \sim stop$
$(\mu; B)[S] \sim \mu[S]; B[S]$
$(B1[]B2)[S] \sim B1[S][]B2[S]$
$B[S] \sim B$ if $[S]$ is the identity on the set of label of B
$P[a1,\ldots,an] \sim B[a1/f1,\ldots,an/fn]$ if $P[f1,\ldots,fn] = B$

•

All the laws of the above list have a counterpart in standard LOTOS, but our notion of bisimulation is too strong, for instance, to get an absorption law. This is due to the fact that the equivalence is sensitive to the number of occurrences of the idling action in the performed events. As an example notice that $B = a; stop|[a,b]|b; stop$ and $B[]B = (a; stop|[a,b]|b; stop)[](a; stop|[a,b]|b; stop)$ are not bisimilar, since B—$\{\{\lambda,\lambda\}\} \rightarrow B$ while $B[]B$—$\{\{\lambda\}\} \rightarrow B[]B$. Thus the absorption law, as well as a law equating $B[]stop$ and B, could be recovered by a weaker notion of bisimulation abstracting from idle moves. Such a bisimulation should be still stronger than the usual *weak bisimulation*.

We conclude the section with some remarks on the possibility of stating an expansion law. Indeed, in order to get it, we should include composite events as arguments of the prefixing operator. But this is not enough. Let us consider, for instance, the expression $B = a; stop|[]|b; stop$. Of course we would equate it to the process $\{\{a,b\}\}; (stop|[]|stop)$. On the other hand, consider the parallel composition $B|[a,b]|(a; stop[]b; stop)$. Due to synchronization constraints such a process can only perform the events $\{\{a,\lambda,a\}\}$ and $\{\{\lambda,b,b\}\}$. So, in order to be able to state an expansion of the whole behaviour, it should hold the equality $B = \{\{a,\lambda\}\}; (stop|[]|b; stop))[]\{\{\lambda,b\}\}; (a; stop|[]|stop)$, instead of the above one. As a result, stating an expansion law is really far from obvious, it would seem that 'auxiliary' expansion laws (unfortunately depending upon delay sets) are needed. In other words, in the bisimulation semantics, as well as in the operational one, the interaction of non-determinism and parallelism turns out to be intrinsically intricate.

5 Future work

In view of its synchrony, the proposed model seems to be quite appropriate for describing real-time systems or, to some extent, hardware systems, rather than distributed programs.

Anyway as we have already mentioned before, this work is a part of the definition of a probabilistic calculus [11] which can be considered as an extension of the language defined in this paper. In the complete work [12] also examples of applications as well as relation with Markov theory are presented.

With respect to the pure nondeterministic calculus we can refine the definition of the proposed operational semantics in order to represent synchronization with only one occurrence of the gate (in a way similar to LOTOS). We only need to redefine the rules for the parallel composition operator in such a way that multiple

occurrences of any action occurring in G are replaced by a single occurrence of the same action. This should be of particular importance when the event is in $\{\{\lambda\}\}^+$ in order to equate all deadlock processes.

Finally, the proposed semantic model could be taken as a starting point for the development of a general framework for reasoning about parallelism degree. This would give raise to a *parallelism spectrum* the end-points of which would be the (standard LOTOS) interleaving semantics, where no parallelism at all is allowed, and the fully parallel semantics proposed in the present paper, where the only constraints imposed on parallelism are those implied by synchronization. A framework like this could be obtained by means of parameterizing the transition relation also with a number representing the maximal parallelism degree allowed by the system.

Acknowledgements. We would like to thank an anonymous referee whose stimulating observations led us to a more careful exposition of the topics of the paper.

References

1. S.D. Brookes, C.A.R. Hoare, A.W. Roscoe. A Theory of Communicating Sequential Processes. Journal of the ACM, Vol.31, No.3, pp. 560-599, 1984.
2. I. Christoff. Testing Equivalences and Fully Abstract Models for Probabilistic Processes. CONCUR 90, LNCS 458, Springer-Verlag, 1990.
3. I. Christoff. Testing Equivalences for Probabilistic Processes. Ph. D. Thesis, Dept. of Comp. Science, Uppsala Univ., ISSN 0283-0574, 1990.
4. A. Giacalone, C.C. Jou, S.A. Smolka. Algebraic reasoning for probabilistic concurrent systems. Proc. of Working Conference on Programming Concept and Methods IFIP TC 2, 1990.
5. R. van Glabbeek, S.A. Smolka, B. Steffen, C. Tofts. Reactive, Generative and Stratified Models of Probabilistic Processes. Proc. of 5th LICS, 1990.
6. H. Hansson, B. Jonsson. A Calculus for Communicating Systems with Time and Probabilities. IEEE RTSS, 1990.
7. ISO. Information processing systems - Open systems interconnection - LOTOS - A formal description technique based on the temporal ordering of observational behaviour. ISO 8807, 1989.
8. C.C. Jou, S.A. Smolka. Equivalences, congruences and complete axiomatizations for probabilistic processes. CONCUR 90, LNCS 458, Springer-Verlag, 1990.
9. J. Keilson. Markov Chain Models-Rarity and Exponentiality. Applied Mathematical Sciences, Vol.28, Springer-Verlag, 1979.
10. D. Latella, P. Quaglia. A fully parallel semantics for LOTOS. LotoSphere reference Lo/WP1/T1.2/CNUCE/N0023/V1, 1991.
11. D. Latella, P. Quaglia. A Proposal for a Calculus of Probabilistic Processes. Internal Report CNUCE-CNR C91-27, 1991.
12. D. Latella, P. Quaglia. A Calculus of Probabilistic Synchronizing Processes and Some Applications. Internal Report CNUCE-CNR C92-17, 1992.
13. K. G. Larsen, A. Skou. Bisimulation through probabilistic testing. Proc. POPL, 1989.
14. R. Milner. A Calculus of Communicating Systems. LNCS 92, Springer-Verlag, 1980.
15. R. Milner. Calculi for Synchrony and Asynchrony. Theoretical Computer Science, 25(3), 1983.
16. D. Park. Concurrency and automata on infinite sequences. Proc. 5th GI-Conference, LNCS 104, Springer-Verlag, 1981.

17. G. D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DAIMI-FN-19, Comp. Science Dep., Aarhus University, 1981.
18. S. Purushothaman, P.A. Subrahmanyam. Reasoning about probabilistic behavior in concurrent systems. IEEE Trans. Software Engineering, Vol. SE-13, N.6, pp.740-745, 1987.
19. P. Quaglia. Proposta per una variante probabilistica di LOTOS. Tesi di Laurea in Scienze dell'Informazione, Universita' degli Studi di Pisa, 1991.
20. N. Rico, G.v. Bochmann. Performance description and analysis for distributed systems using a variant of LOTOS. Proc. of XI Int. IFIP Symp. on Protocol Specification, Testing and Verification, North Holland, 1991.
21. S.A. Smolka, B. Steffen. Priority as extremal probability. CONCUR 90, LNCS 458, Springer-Verlag, 1990.
22. C. Tofts. A synchronous calculus of relative frequency. CONCUR 90, LNCS 458, Springer-Verlag, 1990.