ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

**He Jifeng, Oxford University Computing Laboratory, UK, John Cooke,
Loughborough University, UK, and Peter Wallis, University of Bath, UK (Eds)**

BCS-FACS 7th Refinement Workshop

Proceedings of the BCS-FACS 7th Refinement Workshop, Bath, 3-5
July 1996

On Using Syntactic Action Refinement to Derive Compositionally a Timed Efficient Implementation

A. Dekdouk and A. Schaff

Published in Collaboration with the
British Computer Society

On Using Syntactic Action Refinement to Derive Compositionally a Timed Efficient Implementation

Abdelkader Dekdouk *

and

André Schaff

CRIN/INRIA, Université Henri Poincaré

Nancy, 54500, France.

Abstract

In this paper, we present a durational process algebra incorporating syntactic action refinement. We study its correctness with respect to the semantic refinement in an interleaving model, namely the labelled transition systems. This defined refinement is compositional over the language operators and may provide at a given level of abstraction a timed efficient implementation for a specification.

1 Introduction

The hierarchical program design consists of constructing a program by a top-down analysis. Where at each analysis step, an abstract program is replaced by a concrete one. This technique is known as *program refinement*. A similar notion called *action refinement* has been proposed as a means of relating descriptions of concurrent systems at different levels of abstraction. It consists in implementing an abstract action in terms of a more concrete behaviour. This is generally expressed in a process algebraic setting by the term $P[a \rightsquigarrow Q]$ or $P[\rho]$ that is a more general form, with a refinement function ρ that maps actions to behaviours.

The action refinement has been investigated in depth previous years. The first attempts have considered the mathematical models. Indeed, it has been studied on three different models approaches, namely the interleaving [7, 14], causality [12, 30, 5, 6] and ST-models [13, 10, 11]. It has been also investigated in a linguistic approach. Where even in this framework, we distinguish two kinds of action refinement, which we call *semantic* and *syntactic*. In the former approach, consider for instance the transition systems as a semantics domain. The term $P[a \rightsquigarrow Q]$ is interpreted as a labelled transition system where every a -labelled action in the transition system of P is replaced by the transition system of Q . The syntactic approach interprets $P[a \rightsquigarrow Q]$ as a term obtained from P by replacing each a -action symbol by the term Q . Many examples show that the two approaches do not coincide. This is due to the interplay between refinement and parallel composition with communication (see [13, 23, 15, 29, 9]).

Pursuing the investigations on syntactic and semantic action refinement, here in this paper we explore them on a timed process algebraic setting. We will focus more on the syntactic action refinement. In fact, in the last few years many timed process algebras [21, 31, 4, 26, 17, 22, 18, 8] were developed in order to specify rigorously the timed behaviours of timed concurrent systems. These languages enable an explicit description of timing properties such as time delay or time response and most of them assume the instantaneity of actions. However, if we endow such languages with an operator that performs action refinement, it will be counter-intuitive to refine (for instance syntactically) an instantaneous action by a behaviour that contains explicit positive delays.

*For any remark or suggestion, contact please Abdelkader Dekdouk {email : dekdouk@loria.fr}

Consequently we need to have a language based on the assumption of actions non-instantaneity. Effectively, in this paper we begin by defining a durational actions process algebra with ACP sequential composition, TCSP multiway synchronisation and time delay. Then, we investigate the *timed syntactic action refinement* on this language. As already pointed out, when performing syntactic action refinement interpreted as a syntactic substitution, in a linguistic approach containing a parallel composition with communication mechanisms, we are faced with problems that consist in obtaining undesirable results. In other words, the syntactic substitution is not compositional over the multiway synchronisation. To ensure that, we constraint the semantics of parallel composition and propose a more disciplined version of syntactic substitution by introducing an new operator called *atomicity operator*. The disciplined syntactic refinement respects not only the functional behaviour of a system at a given level of abstraction but also its temporal behaviour, by requiring that any duration of execution of a behaviour refining an action is at most equal to the duration of this action. This defined syntactic refinement permits to provide compositionally at any design step an efficient implementation of a given specification.

The rest of the paper is organised as follows: the next section is devoted to the description of the durational actions process algebra, while the following investigates the action refinement on this timed language and the definition of a compositional and timed efficient syntactic action refinement. The investigation of the correctness of this disciplined timed syntactic action refinement with respect to the semantic one is done in section 3. Finally we draw some conclusions and discuss related work.

2 The description of $\mathbf{L}$

$\mathbf{L}$ is a process algebraic language for the description of timed distributed behaviours with durational actions [3, 16]. Conceptually, $\mathbf{L}$ is principally based on ACP and TCSP constructs. Indeed, in addition to the new concept of time delay, we can find the classic ones such as deadlock, successful termination, elementary visible action, sequential composition, choice, action hiding, parallel composition with multiway synchronisation, sequential composition and recursion. Note, that there is no construct to express the timeouts or watchdogs since the real-time expressivity is not the principal issue of the paper.

2.1 Syntax

Let $\mathbf{A}$ ranged over by $\alpha, \beta, \dots$ be an arbitrary set of *action labels* and containing respectively special elements i and $\surd \in \mathbf{A}$ which represent the "invisible action" and the successful termination. Let also $\mathbf{A} \subseteq \mathbf{A}$ be the set of visible actions ranged over by $a, b, \dots$. The set of $\mathbf{L}$ *process expressions* is generated by the following BNF definition:

$$E ::= \delta \mid \varepsilon \mid x \mid a \mid \text{wait}(d) \mid E; E \mid E + E \mid \partial_H E \mid E \parallel_S E \mid \mu x[E]$$

where X ranges over $x, y, z, \dots$ is a set of process variables. The constant d belonging to time domain $\mathbb{R}_*^+$ (the set of positive reals) is the duration of time delay, S and $H \subseteq \mathbf{A}$ are respectively the finite set of synchronising actions and the finite set of hidden actions.

An occurrence of variable x is said to be *bound* when it appears in the scope of the binder $\mu x[\cdot]$, otherwise it is said to be *free*. E is said to be a *closed term* when the set of free variables in E is empty.

A variable x is *guarded* if its occurrence is sequentially preceded by an occurrence of an action or a delay in $[E]$. So we say that E is guarded if all its free variables are guarded. In the following, to avoid any technical problem, we will consider only closed and guarded terms denoted by $P, Q, R, \dots$

Convention For reasons of convenience, we will use the symbols $|||$ and $||$ to denote particular cases of the general parallel composition combinator $||_S$, respectively when $S = \emptyset$ and $S = \mathbf{A}$. Moreover to avoid using cumbersome terms with parentheses, we assume the known operators precedences namely sequential composition $>$ choice $>$ parallel composition $>$ hiding.

2.2 Semantics

2.2.1 Basic intuitions

The semantics of the language **L** is much related to the ones attributed to the classic process algebras mentioned previously. However, there are some characteristics that are proper to it and that can be summarised in two issues:

- Any visible action a has a fixed duration $\phi(a)$ given by the mapping $\phi : \mathbf{A} \longrightarrow \mathbb{R}_*^+$. The mapping ϕ is extended to the successful termination by $\phi : \surd \mapsto 0$.
- **L** does not consider the process consisting in an elementary invisible action. However there is another way to express it, in fact we consider a refined version of this particular process that consist in assuming that time delay corresponds exactly to an internal execution or vice versa.

2.2.2 Operational semantics

The following rules present the operational semantics of **L** in the SOS style [25], associating to each process term P a transition system called *durational transition system* $(\mathcal{P}, \mathcal{A}, \longrightarrow, P_0)$, where $\mathcal{P} \subseteq \mathbf{L}$ is the set of states reached from the initial state P_0 that corresponds in fact to the process P , $\mathcal{A} \subseteq \mathbf{A} \times \mathbb{R}^+$ the set of labels and a transition relation $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{P}$ which respects the laws of Table 1. Instead of $(P, \alpha, \varphi, Q) \in \longrightarrow$ we write $P \xrightarrow[\varphi]{\alpha} Q$ to mean that process P executes an action α of duration φ and then evolves to process Q . We feel free to suppress the duration information from the transition relation each time, it is not relevant to consider it.

$(Ex) \frac{}{\varepsilon \xrightarrow[\delta]{\surd} \delta}$	$(Act) \frac{}{a \xrightarrow[\phi(a)]{\alpha} \varepsilon}$	$(Del) \frac{}{wait(d) \xrightarrow[d]{i} \varepsilon}$
$(SC1) \frac{P \xrightarrow[\varphi]{\alpha} P', \alpha \neq \surd}{P; Q \xrightarrow[\varphi]{\alpha} P'; Q}$	$(SC2) \frac{P \xrightarrow[\delta]{\surd} P', Q \xrightarrow[\varphi]{\alpha} Q'}{P; Q \xrightarrow[\varphi]{\alpha} Q'}$	
$(Ch) \frac{P \xrightarrow[\varphi]{\alpha} P'}{P + Q \xrightarrow[\varphi]{\alpha} P', Q + P \xrightarrow[\varphi]{\alpha} P'}$		
$(Hi) \frac{P \xrightarrow[\varphi]{\alpha} Q}{\partial_H P \xrightarrow[\varphi]{\alpha} \partial_H Q \mid \alpha \notin S, \partial_H P \xrightarrow[\varphi]{i} \partial_H Q \mid \alpha \in S}$		
$(PC1) \frac{P \xrightarrow[\varphi]{\alpha} P', \alpha \notin S \cup \{\surd\}}{P \parallel_S Q \xrightarrow[\varphi]{\alpha} P' \parallel_S Q, Q \parallel_S P \xrightarrow[\varphi]{\alpha} Q \parallel_S P'}$		
$(PC2) \frac{P \xrightarrow[\varphi]{\alpha} P', Q \xrightarrow[\varphi]{\alpha} Q', \alpha \in S \cup \{\surd\}}{P \parallel_S Q \xrightarrow[\varphi]{\alpha} P' \parallel_S Q'}$		
$(Rec) \frac{P\{\mu x[P]/x\} \xrightarrow[\varphi]{\alpha} Q}{\mu x[P] \xrightarrow[\varphi]{\alpha} Q}$		

Table 1. The operational Semantics of **L**

Definition 2.1

We will write $\llbracket P \rrbracket$ for the durational transition system $(\mathcal{P}, \mathcal{A}, \longrightarrow, P)$ of process term P generated by the above formal system.

3 Action refinement

In this section we tackle the technique of syntactic action refinement of the functional and temporal behaviour of a system. First of all we focus only on action refinement of the system functional behaviour. Then we treat the action

refinement of its temporal behaviour. So we have now a language denoted **RL** which is the one of **L** enriched with the action refinement operator.

$$P ::= \dots \mid P[\rho]$$

Where $\rho : \mathbf{A} \rightarrow \text{finite}(\mathbf{RL}^{-\varepsilon})$ is a refinement function which is explicitly extended to all visible actions by $\rho : \sqrt{} \mapsto \varepsilon$. Note that sometimes the action refinement will be expressed by terms of the form $[a_i \rightsquigarrow Q_i]_{i=1 \dots n}$. We assume that when an action has started it must finish after a positive duration, so our refining process is supposed to be finite, non-deadlocking and different from the empty or immediately terminating process.

Intuitively the action refinement $P[a \rightsquigarrow Q]$ consist in the execution of process P in which whenever a should be executed; the process Q of execution duration at most equal to the duration of a , is executed instead. In other words the semantics of $P[a \rightsquigarrow Q]$ represents a labelled transition system where every a -labelled transition is replaced by the labelled transition system of Q . This interpretation of action refinement should be regarded as a *semantics substitution*. So more formally $\llbracket P[a \rightsquigarrow Q] \rrbracket = \llbracket P \rrbracket[a \rightsquigarrow \llbracket Q \rrbracket]$ which is defined as follows: for every transition $s \xrightarrow{a}_{\varphi} s'$ in $\llbracket P \rrbracket$. Identify s with the root of $\llbracket Q \rrbracket$ and s' with the end nodes of $\llbracket Q \rrbracket$.

3.1 Syntactic action refinement of functional behaviour

Roughly speaking, the syntactic action refinement $P[a \rightsquigarrow Q]$ corresponds to the operation of *syntactic substitution* of process term Q for a -action symbol in process term P (i.e. by definition $P[a \rightsquigarrow Q] = P\{Q/a\}$ see Table 2). Unfortunately, this approach does not coincide in general with the intuitive one. More precisely when dealing with terms having parallel composition, i.e. namely in our case the TCSP-like multiway synchronisation.

$P\{R/a\}$	=	P where $P = \{\delta, \varepsilon, \text{wait}(d)\} \cup \mathbf{X}$
$b\{R/a\}$	=	$\begin{cases} R & \text{if } b \equiv a \\ b & \text{otherwise} \end{cases}$
$(P \odot Q)\{R/a\}$	=	$(P\{R/a\}) \odot (Q\{R/a\})$ where $\odot \in \{;, +\}$
$\partial_H P\{R/a\}$	=	$\begin{cases} \partial_H P & \text{if } a \in H \\ \partial_H (P\{R/a\}) & \text{otherwise} \end{cases}$
$(P _S Q)\{R/a\}$	=	$\begin{cases} (P\{R/a\}) _{S \setminus \{a\} \cup \text{label}(R)} (Q\{R/a\}) & \text{if } a \in S \\ (P\{R/a\}) _S (Q\{R/a\}) & \text{otherwise} \end{cases}$
$(\mu x[P])\{R/a\}$	=	$\mu x[P\{R/a\}]$

Table 2. The syntactic substitution in process terms

Where $\text{label}(P)$ is the set of action labels occurring in P .

3.1.1 Refining the non-synchronising actions

The more obvious example that illustrates the non agreement of the syntactic substitution with the intuitive action refinement (i.e. the semantic action refinement) is: consider the terms $(a|||b)[a \rightsquigarrow a_1; a_2]$ and $a[a \rightsquigarrow a_1; a_2]|||b[a \rightsquigarrow a_1; a_2]$. These terms are not equivalent upto the more reasonable equivalence, because intuitively the latter can perform the sequence $\langle a_1, b, a_2 \rangle$, while the former cannot. This example shows clearly the non-compositionality of the general syntactic substitution when operating on processes with parallel composition. We would propose an action refinement that behaves like a syntactic substitution and that respects the intuitive meaning of action refinement. To this aim, we begin by considering the general syntactic substitution and step by step present the different problems that may appear when this operation operates on multiway synchronisation. By remedying these problems, we obtain a disciplined version of syntactic substitution. This syntactic action refinement is compositional modulo a constrained version of multiway synchronisation.

Let us first begin by the example just presented above. To preserve the intuitive meaning of action refinement we constrain the refining process to be executed atomically, i.e. it cannot be interleaved with the action of other concurrent processes. To do so, we introduce a new operator of atomicity denoted $[Q]_\ell$ where $\ell \in \{\perp, \top\}$ (Let $\mathbf{RL}_{at}$ denote $\mathbf{RL}$ endowed with the atomicity operator). When $\ell = \top$, it means that the refining process is active i.e. an action has been offered by it and therefore no other concurrent process must offer an action. Otherwise $\ell = \perp$ and it means that the refinement is at its initial phase. For instance $(a|||b)[a \rightsquigarrow a_1; a_2] = [a_1; a_2]_\perp ||| b$ and after offering a_1 -action becomes $[\varepsilon; a_2]_\top ||| b$. The latter process cannot carry out the b -action since the refining process has not yet finished its atomic execution. More generally, if we consider the concurrent process $P = Q|||R$ and suppose that Q has the possibility to offer an action then P performs this action when R has no active refining sub-process. To ensure that, we introduce a predicate $not_is_active(R)$ that permits to test if the set of the active refining sub-processes of R is empty. This predicate is defined more formally in detail subsequently.

3.1.2 Refining the synchronising actions

The situation becomes much more difficult if we allow the refinement of synchronising actions. Take for instance, the term $P = a|||_{\{a\}}a$. Naturally P can offer a synchronisation action a and then terminates successfully, yielding the transition $P \xrightarrow{a} \cdot \xrightarrow{\delta}$. However if we refine syntactically the a -action with a non-deterministic process $b; c + b; d$ we will obtain the process term $P_1|||_{\{b,c,d\}}P_2 = [b; c + b; d]_\perp |||_{\{b,c,d\}}[b; c + b; d]_\perp$ which may offer the b -action synchronisation yielding the transition $P \xrightarrow{b} P'_1|||_S P'_2 = [\varepsilon; c]_\top |||_{\{b,c,d\}}[\varepsilon; d]_\top$ and then falls in a deadlock state where no further action is possible. We notice that the syntactic refinement as it is actually defined may alter the structure of the communication channels and so the control structure of the refined process due to the interplay between the synchronisation and non-deterministic situation.

To solve such problem we constrain the semantics of the multiway synchronisation when refining processes communication. The condition consists of saying that a synchronisation action stemming from refining processes is allowed if after its appearance the residuals of these refining processes are syntactically equal. Therefore we introduce a predicate $is_res_eq(P'_1, P'_2, a)$ which test the syntactic equality of residuals of processes refining the a -action that appear in both synchronising processes P'_1 and P'_2 . However, when a refinement synchronisation is offered we have no information from which refining process it has stemmed. In other words we do not know the abstract action that has yielded this refinement synchronisation. To determine that, we label the atomicity operator by the name of the abstract action to refine, in order to keep track of the actions to carry out. So the syntactic action refinement can be defined as follows: $P[a \rightsquigarrow b; c + b; d] = [b; c + b; d]_\perp^a |||_a [b; c + b; d]_\perp^a$ which is unable to offer the synchronisation $[b]^a$ (i.e. the action a is the origin of the sub-action b) since the condition of synchronisation is not satisfied, i.e. $is_res_eq([\varepsilon; c]_\top^a, [\varepsilon; d]_\top^a, a) = false$ since $\varepsilon; c \neq \varepsilon; d$. However the two other communications sequences namely $\langle [b]^a, [c]^a, [\sqrt{ }]^a \rangle$ and $\langle [b]^a, [d]^a, [\sqrt{ }]^a \rangle$ are possible.

Now, consider the process term $P = (a|||a)|||_{\{a\}}a$. $P[a \rightsquigarrow a_1; a_2] = P_1|||_{\{a\}}P_2$ where $P_1 = [a_1; a_2]_\perp^a ||| [a_1; a_2]_\perp^a$ and $P_2 = [a_1; a_2]_\perp^a$. P_1 may offer the sub-action $[a_1]^a$ and then turns into $P'_1 = [\varepsilon; a_2]_\top^a ||| [a_1; a_2]_\perp^a$. Also P_2 may show its ability in communicating with P_1 by offering the same sub-action and then turns into $P'_2 = [\varepsilon; a_2]_\top^a$. However the synchronisation on $[a_1]^a$ may not be produced by the whole process $P[a \rightsquigarrow a_1; a_2]$ because the synchronisation condition may evaluate to false (i.e. $is_res_eq(P'_1, P'_2, a) = false$). As, rather than comparing the residual $\varepsilon; a_2$ of P'_2 with $\varepsilon; a_2$ of P'_1 , we establish the comparison between the former and $a_1; a_2$ which are effectively syntactically different. The problem here is that we confuse two different abstract events of sub-process $(a|||a)$ that are labelled by the same a -action label. To avoid this mismatching we have to reason in term of events execution rather than actions execution, by the refined process. In other words when a sub-action is offered, we must annotate it and then glue the annotation to the abstract action that has effectively offered it. Let us illustrate this issue on our previous example. We have $P_1 = [a_1; a_2]_\perp^a ||| [a_1; a_2]_\perp^a$ that executes the event $[a_1]_\omega^a$ and evolves to $P'_1 = [\varepsilon; a_2]_\omega^a ||| [a_1; a_2]_\perp^a$. Also P_2 offers the event $[a_1]_\pi^a$ and turns into $[\varepsilon; a_2]_\pi^a$. The synchronisation condition is now parametrised by the two abstract synchronising events rather than a single abstract action label. Concerning our example, we have $is_res_eq(P'_1, P'_2, \omega, \pi) = true$ because the residual of event ω is $\varepsilon; a_2$ and it equates the residual of event π .

To ensure the soundness of the annotation, in the sense that different events are annotated differently we use a technique that exploits the syntactic structure of the term. Indeed on the basis of this technique, we can easily show that two different abstract events can never be represented by the same annotations. This is due simply to the fact that they have not the same locations (w.r.t the combinators of parallel composition, choice, sequential composition) and do not belong to the same abstract design level (atomicity operator) in the process term. These annotations are sets of terms generated by the following grammar:

$$(\mathcal{L} \ni) \ell ::= \lambda \mid \bullet \mid \triangleleft \ell \mid \triangleright \ell$$

We shall use the set $\wp(\mathcal{L})$ ranged over by $\omega, \pi, \dots$ to represent these annotations (events). Intuitively $\{\lambda\}$ represents an event offered by a process that consists in either a single elementary action or a successful termination. $\bullet$ expresses the change of abstraction level. An event represented by $\triangleleft \omega = \{\triangleleft \ell \mid \ell \in \omega\}$ means that it is offered by the component P of $P \odot Q$ where $\odot \in \{;, +, ||\}$ and conversely for $\triangleright \omega$. We also use the expression $\omega_1 || \omega_2 = \triangleleft \omega_1 \cup \triangleright \omega_2$ to point out the synchronisation of two events represented by ω_1 and ω_2 .

Finally, we close this section by defining formally the predicates $not_is_active : \mathbf{L}_{at} \rightarrow \{true, false\}$ and $is_res_eq : \mathbf{L}_{at} \times \mathbf{L}_{at} \times \wp(\mathbf{L}) \times \wp(\mathbf{L}) \rightarrow \{true, false\}$. Concerning the former, note that when a refining process is active then it is annotated by $\omega \in \wp(\mathcal{L})$, therefore we do not need to use the symbol $\top$ to show that it is active. Let us first define the set of annotations occurring in a process. This is given by the function $ann : \mathbf{L}_{at} \rightarrow \wp(\wp(\mathcal{L}))$ which is defined as follows:

$ann(P)$	$=$	$\emptyset$ where $P \in \mathbf{A} \cup \mathbf{X} \cup \{\delta, \varepsilon, wait(d), [Q]_{\perp}^a\}$
$ann([P]_{\omega}^a)$	$=$	$\{\omega\}$ where $\omega \in \wp(\mathcal{L})$
$ann(\partial_H P)$	$=$	$ann(P)$
$ann(P \odot Q)$	$=$	$ann(P) \cup ann(Q)$ where $\odot \in \{;, +, _s\}$
$ann(\mu x[P])$	$=$	$ann(P)$

Table 3. The definition of the set of a process annotations

$not_is_active(P) \Leftrightarrow ann(P) = \emptyset$. $is_res_eq(P, Q, \omega, \pi) \Leftrightarrow P[\omega] \equiv_{\alpha} Q[\pi]$ where $P[\omega]$ represents the sub-term R of the abstract event $[R]_{\omega}^a$ of process P and $P \equiv_{\alpha} Q \Leftrightarrow P[\alpha] \preceq Q$ such that $\alpha : ann(P) \rightarrow ann(Q)$ is an isomorphism. $P \preceq Q$ is defined inductively on the syntactic structure of the process term thus:

$P \equiv Q$	$\Rightarrow$	$P \preceq Q$
$[P]_{\omega}^a \preceq [Q]_{\pi}^a$	$\Leftrightarrow$	$(\omega \equiv \pi) \text{ and } (P \equiv_{\alpha} Q)$
$P \odot Q \preceq R \odot S$	$\Leftrightarrow$	$(P \preceq R) \text{ and } (Q \preceq S)$ where $\odot \in \{;, +, _s\}$
$\partial_H P \preceq \partial_H Q$	$\Leftrightarrow$	$P \preceq Q$

Table 4. The process syntactic equivalence upto an annotations renaming

3.1.3 Syntactic action refinement of temporal behaviour

The action refinement we have developed to this point considers only the system functional behaviour and does not take into account its timed behaviour which can be modestly described by our language. To define an action refinement that respects also the timed behaviour of the system to refine, we need to constrain further this operation. To do so, we require that the refining process lasts no longer than the action it refines. Therefore, we need to introduce a means that permits computing the duration of a refining process. This is realized by the function $dur_{\phi} : \text{finite}(\mathbf{RL}^{-\delta}) \times \wp(\mathbf{A}^{\vee} \times \mathbb{R}^+) \rightarrow \wp(\mathbb{R}^+)$ which is defined inductively on the syntactic structure of the process as follows:

$dur_\phi(\varepsilon)$	$=$	$\{0\}$
$dur_\phi(a)$	$=$	$\{\phi(a)\}$
$dur_\phi(wait(d))$	$=$	$\{d\}$
$dur_\phi([R]_\omega^a)$	$=$	$dur_\phi(R)$
$dur_\phi(P + Q)$	$=$	$dur_\phi(P) \cup dur_\phi(Q)$
$dur_\phi(\partial_H P)$	$=$	$dur_\phi(P)$
$dur_\phi(P _S Q)$	$=$	$\{max(t, t') \mid t \in dur_\phi(P), t' \in dur_\phi(Q)\}$
$dur_\phi(P; Q)$	$=$	$\{t + t' \mid t \in dur_\phi(P), t' \in dur_\phi(Q)\}$
$dur_\phi(P[a \rightsquigarrow Q])$	$=$	$dur_{\phi'}(P) \text{ where } \phi'(x) := \begin{cases} Max(dur_\phi(Q)) \\ \text{if } x = a \\ \phi(x) \text{ otherwise} \end{cases}$

Table 5. The duration of a finite non-deadlocking process

Definition 3.1 *let $P \in \mathbf{RL}$ a refining process of action a : P is called temporally valid if $\forall t \in dur_\phi(P), t \leq \phi(a)$*

The fact that the duration of the refining process must at most be equal to the duration of the action it refines seems to us intuitive. Indeed at early system design stage we have no sufficient information on how much exactly a given activity may last. This information becomes more precise when we concretise the abstract action stepwise by the refining processes.

Before defining the operational semantics of our $\mathbf{L}_{at}$ processes, we establish syntactically the meaning of the refinement term $P[\rho]$. To do so, we introduce a mapping $\sigma : \mathbf{RL}_{at} \longrightarrow \mathbf{L}_{at}$ that removes refinement operators from the given expression, so that the disciplined syntactic substitution $\{\rho\}^* : \mathbf{L}_{at} \longrightarrow \mathbf{L}_{at}$ is only applied to terms already reduced (also called flat terms).

$P\sigma$	$=$	$P \text{ where } (P \in \{\delta, \varepsilon\} \cup \mathbf{A} \cup \mathbf{X})$
$([P]_\omega^a)\sigma$	$=$	$[P\sigma]_\omega^a$
$(\partial_H P)\sigma$	$=$	$\partial_H(P\sigma)$
$(P_1 \odot P_2)\sigma$	$=$	$(P_1)\sigma \odot (P_2)\sigma \text{ where } \odot \in \{;, +, _S\}$
$(\mu x[P])\sigma$	$=$	$\mu x[P\sigma]$
$(P[\rho])\sigma$	$=$	$(P\sigma)\{\rho\sigma\}^*$
$P\{\rho\}^*$	$=$	$P \text{ where } (P \in \{\delta, \varepsilon\} \cup \mathbf{X})$
$a\{\rho\}^*$	$=$	$[\rho(a)]_\perp^a \text{ where } (\forall t \in dur_\phi(\rho(a)) : t \leq \phi(a))$
$([P]_\omega^a)\{\rho\}^*$	$=$	$[P\{\rho\}^*]_\omega^a$
$(\partial_H P)\{\rho\}^*$	$=$	$\partial_H(P\{\rho \upharpoonright (dom(\rho) \setminus H)\}^*)$
$(P_1 \odot P_2)\{\rho\}^*$	$=$	$(P_1\{\rho\}^*) \odot (P_2\{\rho\}^*) \text{ where } \odot \in \{;, +, _S\}$
$(\mu x[P])\{\rho\}^*$	$=$	$\mu x[P\{\rho\}^*]$

Table 6. The formal definition of timed syntactic action refinement

Where $\rho\sigma$ is s.t. $\rho\sigma(a) = \rho(a)\sigma$ and $\rho \upharpoonright A$ is the restriction of the function ρ to the set A .

The operator of timed syntactic action refinement is distributive over the operators of the language especially over the parallel composition with communications. It is modulo an observational (i.e. without considering the origins of sub-actions) isomorphism. However, to prove the correctness of these equations we are content only with the strong bisimulation-like equivalence that is defined in the section of semantic equivalence.

3.1.4 Operational semantics

We proceed with the definition of the meaning of $\mathbf{L}_{at}$ terms using the SOS style. As previously stated the model associated to each term is a durational labelled transition system $(\mathcal{P}_1, \mathcal{A}', \longrightarrow^1, P_0)$. With respect to transition system

defined in Table 1. The difference here, is that the labels are enriched with information to deal with syntactic action refinement, in order to ensure the compositionality of this operation. So, a transition is written as $P \xrightarrow[\phi]{\alpha}^\beta Q$ where $\alpha \in \mathbf{A}$, $\beta \in \mathbf{A} \cup \{\sqrt{\cdot}, \perp\}$ and $\omega \in \wp(\mathcal{L})$. $\beta = \perp$ is introduced to express the unknown origin of the internal action caused by the time delay. $\xrightarrow[\phi]{\alpha}$ is the smallest relation satisfying the given formal system (consisting of some axioms and some rules). $P\{\pi/\omega\}$ is the replacement of the event ω by π in process P .

$(Ex) \frac{}{\varepsilon \xrightarrow[\phi]{\alpha}^\beta \delta}$	$(Act) \frac{}{a \xrightarrow[\phi(a)]{\alpha}^\beta \varepsilon}$	$(Del) \frac{}{wait(d) \xrightarrow[d]{\alpha}^\beta \varepsilon}$
$(SC1) \frac{P \xrightarrow[\phi]{\alpha}^\beta P' \ (\alpha \neq \sqrt{\cdot})}{P; Q \xrightarrow[\phi]{\alpha}^\beta (P'\{\triangleleft \omega/\omega\}); Q}$	$(SC2) \frac{P \xrightarrow[\phi]{\alpha}^\beta P', \ Q \xrightarrow[\phi']{\alpha}^\beta Q'}{P; Q \xrightarrow[\phi']{\alpha}^\beta Q'\{\triangleright \omega'/\omega'\}}$	
$(Ch) \frac{P \xrightarrow[\phi]{\alpha}^\beta P'}{P + Q \xrightarrow[\phi]{\alpha}^\beta P'\{\triangleleft \omega/\omega\}, \ Q + P \xrightarrow[\phi]{\alpha}^\beta P'\{\triangleright \omega/\omega\}}$		
$(Hi) \frac{P \xrightarrow[\phi]{\alpha}^\beta P'}{\partial_H(P) \xrightarrow[\phi]{\alpha}^\beta \partial_H(P') \ (\beta \in H), \ \partial_H(P) \xrightarrow[\phi]{\alpha}^\beta \partial_H(P') \ (\beta \notin H)}$		
$(PC1) \frac{P \xrightarrow[\phi]{\alpha}^\beta P' \ (\beta \notin S \cup \{\sqrt{\cdot}\} \text{ and } not_is_active(Q))}{P \parallel_S Q \xrightarrow[\phi]{\alpha}^\beta (P'\{\triangleleft \omega/\omega\}) \parallel_S Q, \ Q \parallel_S P \xrightarrow[\phi]{\alpha}^\beta Q \parallel_S (P'\{\triangleright \omega/\omega\})}$		
$(PC2) \frac{P \xrightarrow[\phi]{\alpha}^\beta P', \ Q \xrightarrow[\phi']{\alpha}^\beta Q' \ (\beta \in S \cup \{\sqrt{\cdot}\} \text{ and } is_res_eq(P', Q', \omega, \omega'))}{P \parallel_S Q \xrightarrow[\phi]{\alpha}^\beta (P'\{\omega \parallel \omega'/\omega\}) \parallel_S (Q'\{\omega \parallel \omega'/\omega'\})}$		
$(At) \frac{P \xrightarrow[\phi]{\alpha}^\beta P'}{[P]_\pi^a \xrightarrow[\phi]{\alpha}^\beta [P']_\bullet^a}$	$(Rec) \frac{P\{\mu x[P]/x\} \xrightarrow[\phi]{\alpha}^\beta Q}{\mu x[P] \xrightarrow[\phi]{\alpha}^\beta Q}$	

 Table 7. The operational semantics of $\mathbf{L}_{at}$

Definition 3.2

We will write $\llbracket P \rrbracket$ for the extended durational transition system $(\mathcal{P}_1, \mathcal{A}', \xrightarrow[\phi]{\alpha}^\beta, P)$ of process term P generated by the above formal system.

3.2 Equivalence semantic and soundness of timed syntactic action refinement

In this section we are interested in proving formally the correctness of timed syntactic action refinement w.r.t the semantic action refinement. The temporal soundness of timed syntactic refinement is obtained by definition of the latter. However the case of the functional refinement is not direct. We introduce an equivalence relation based on the well known notion of bisimulation [24, 19] as a comparison means. The formal comparison of these two kinds of action refinement needs to define formally in an operational style the meaning $P[\rho]$. Its definition is based on a restricted form of atomicity operator. The complete formal system defining the operational semantics of the corresponding language¹, that is denoted $\mathbf{RL}_{at}$ is provided in appendix A.

¹This language is similar to $\mathbf{RL}_{at}$ with the difference that its atomicity operator is not annotated

Definition 3.3 $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are called *I-durational bisimilar*, denoted $\llbracket P \rrbracket \sim^1 \llbracket Q \rrbracket$, if there exist a symmetric relation $\mathcal{R} \subseteq \mathcal{P}_1 \times \mathcal{P}_1$ such that $P \mathcal{R} Q$ and if $P_1 \mathcal{R} Q_1$ then,

$$P_1 \xrightarrow[\varphi]{[\alpha]_\omega^\beta} P'_1 \text{ then there exists a } Q'_1 \in \mathcal{P}_1 \text{ such that } Q_1 \xrightarrow[\varphi]{[\alpha]_\pi^\gamma} Q'_1 \text{ and } P'_1 \mathcal{R} Q'_1$$

Definition 3.4 $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are called *II-durational bisimilar*, denoted $\llbracket P \rrbracket \sim^2 \llbracket Q \rrbracket$, if there exist a symmetric relation $\mathcal{R} \subseteq \mathcal{P}_2 \times \mathcal{P}_2$ such that $P \mathcal{R} Q$ and if $P_1 \mathcal{R} Q_1$ then

$$P_1 \xrightarrow[\varphi]{[\alpha]^\beta} P'_1 \text{ then there exists a } Q'_1 \in \mathcal{P}_2 \text{ such that } Q_1 \xrightarrow[\varphi]{[\alpha]^\gamma} Q'_1 \text{ and } P'_1 \mathcal{R} Q'_1$$

Proposition 3.1 $\sim^1$ and $\sim^2$ are equivalences.

Proof. It is similar to the one given in [20].

Proposition 3.2 $\equiv_\alpha$ is a I-durational bisimulation.

Proof. We have to show that $\{(P, Q) \mid P \equiv_\alpha Q, P, Q \in \mathbf{L}_{at}\} \cup \sim^1$ is I-durational bisimulation.

Definition 3.5 Let $P, Q \in \mathbf{L}_{at}$ and $R, S \in \mathbf{RL}_{at}$. Then

$$\begin{aligned} P \xrightarrow[\varphi]{[\alpha]_\omega^\beta} Q &\stackrel{def}{=} P \xrightarrow[\varphi]^\alpha Q, \\ R \xrightarrow[\varphi]{[\alpha]^\beta} S &\stackrel{def}{=} R \xrightarrow[\varphi]^\alpha S. \end{aligned}$$

In the sequel we suppose that all transition system is transformed to the one with the transition relation $\rightarrow \subseteq (\mathcal{P}_1 \cup \mathcal{P}_2) \times \mathbf{A} \times \mathbb{R}^+ \times (\mathcal{P}_1 \cup \mathcal{P}_2)$. Moreover, the bisimulations $\sim^1$ and $\sim^2$ may also be adapted to it.

Definition 3.6 $\llbracket P \rrbracket$ and $\llbracket Q \rrbracket$ are called *bisimilar*, denoted $\llbracket P \rrbracket \cong \llbracket Q \rrbracket$, if there exist $\mathcal{R} \subseteq 2^{\mathcal{P}_1 \cup \mathcal{P}_2}$ such that $P \mathcal{R} Q$ and if $P_1 \mathcal{R} Q_1$ then

1. $P_1 \xrightarrow[\varphi]^\alpha P'_1$ then there exists a $Q'_1 \in \mathcal{P}_1 \cup \mathcal{P}_2$ such that $Q_1 \xrightarrow[\varphi]^\alpha Q'_1$ and $P'_1 \mathcal{R} Q'_1$;
2. $Q_1 \xrightarrow[\varphi]^\alpha Q'_1$ then there exists a $P'_1 \in \mathcal{P}_1 \cup \mathcal{P}_2$ such that $P_1 \xrightarrow[\varphi]^\alpha P'_1$ and $P'_1 \mathcal{R} Q'_1$.

Proposition 3.3 $\cong$ is an equivalence.

Proof. (see [20]).

Proposition 3.4 $\sim^1$ and $\sim^2$ are bisimulations.

Proof. It is obvious.

Lemma 3.1 Let $P_1, P_2 \in \mathbf{L}_{at}$ such that, $P_1 \equiv_\alpha P_2$ then,

$$P_1 \xrightarrow[\varphi]^\beta P'_1 \Rightarrow P_2 \xrightarrow[\varphi]^\beta P'_2 \text{ and } P'_1 \equiv_\alpha P'_2$$

Proof. by induction on $P_1 \equiv_\alpha P_2$ and $P_1 \xrightarrow[\varphi]^\beta P'_1$. The base cases are when $P_1 = a$ or $[Q]_\omega^a$. The case of $P_1 = a$ is trivial. $P_1 = [Q_1]_\omega^a$ then $P_2 = [Q_2]_\omega^a$ and $Q_1 \equiv_\alpha Q_2$. $P_1 \xrightarrow[\varphi]^\beta P'_1$ if $Q_1 \xrightarrow[\varphi]^\beta Q'_1$, then by induction $Q_2 \xrightarrow[\varphi]^\beta Q'_2$ and $Q'_1 \equiv_\alpha Q'_2$. Therefore $P_2 \xrightarrow[\varphi]^\beta P'_2$ and $P'_1 \equiv_\alpha P'_2$. The other inductive cases are easy to prove. Consider for instance, $P_1 = R_1 + R_2$, so $P_2 = S_1 + S_2$ such that $R_1 \equiv_\alpha S_1$ and $R_2 \equiv_\alpha S_2$. Suppose that $P_1 \xrightarrow[\varphi]^\beta P'_1 \equiv R'_1$ if $R_1 \xrightarrow[\varphi]^\beta R'_1$, so by induction we have $S_1 \xrightarrow[\varphi]^\beta S'_1$ and $R'_1 \equiv_\alpha S'_1$. Consequently $P_2 \xrightarrow[\varphi]^\beta P'_2 \equiv S'_1$ and obviously $P'_1 \equiv_\alpha P'_2$.

Lemma 3.2 Let $P_1, P_2, P_3 \in \mathbf{L}_{at}$ such that $P_1 \equiv_\alpha P_2 \equiv_\alpha P_3$.

$$\llbracket [P_1]_\pi^a \rrbracket \sim^1 \llbracket [P_2]_\omega^a \rrbracket_{\{a\}} [P_3]_\omega^a$$

Proof. We have to prove that $\{([P_1]_\pi^a, [P_2]_\omega^a \rrbracket_{\{a\}} [P_3]_\omega^a) \mid P_1 \equiv_\alpha P_2 \equiv_\alpha P_3\} \cup \sim^1$ is a I-durational bisimulation. The case of initial states is trivial. Suppose that $P_1 \xrightarrow[\varphi]{\beta} P'_1$, then by lemma 3.1 we have $P_2 \xrightarrow[\varphi]{\beta} P'_2$ and $P_3 \xrightarrow[\varphi]{\beta} P'_3$. $P'_2 \equiv_\alpha P'_3$, the synchronisation constraint is satisfied. This leads to the transition $[P_2]_\omega^a \rrbracket_{\{a\}} [P_3]_\omega^a \xrightarrow[\varphi]{\beta} [P'_2]_{\omega'}^a \rrbracket_{\{a\}} [P'_3]_{\omega'}^a$ and $([P'_1]_{\pi'}^a, [P'_2]_{\omega'}^a \rrbracket_{\{a\}} [P'_3]_{\omega'}^a) \in \mathcal{R}$. The proof of other direction is similar.

The following corollary shows that our syntactic refinement does not affect the communication structure.

Corollary 3.1 $\llbracket [a \rightsquigarrow P] \sigma \rrbracket \sim^1 \llbracket (a \rrbracket_{\{a\}} a) [a \rightsquigarrow P] \sigma \rrbracket$

Lemma 3.3 Let $P, Q \in \mathbf{RL}$

$$\llbracket (P \parallel_S Q) [\rho] \rrbracket \cong \llbracket (P \parallel_S Q) [\rho] \sigma \rrbracket$$

Proof. We present only a sketch of it. We have to prove that $\mathcal{R} = \{((P \parallel_S Q) [\rho], (P \parallel_S Q) [\rho] \sigma) \mid P, Q \in \mathcal{P}_1 \cup \mathcal{P}_2\} \cup \cong$ is a bisimulation. There are many cases to demonstrate. We only content to prove the equation $\llbracket (P \parallel_S Q) [a \rightsquigarrow R] \rrbracket \cong \llbracket (P \parallel_S Q) [a \rightsquigarrow R] \sigma \rrbracket$ for the following hypothesis: P, Q, R are flat terms, $a \in S$. Suppose that $(P \parallel_S Q) \xrightarrow[\varphi]{a} (P' \parallel_S Q')$ if $P \xrightarrow[\varphi]{a} P'$ and $Q \xrightarrow[\varphi]{a} Q'$. If $R \xrightarrow[\varphi']{\beta} R'$ then $(P \parallel_S Q) [a \rightsquigarrow R] \xrightarrow[\varphi']{\beta} [R']_\omega^a$; $((P' \parallel_S Q') [a \rightsquigarrow R])$. Moreover, we have $(P \parallel_S Q) [a \rightsquigarrow R] \sigma = P \{R/a\}^* \parallel_S Q \{R/a\}^* \xrightarrow[\varphi']{\beta} [R']_\omega^a$; $P' \{R/a\}^* \parallel_S [R']_\omega^a$; $Q' \{R/a\}^* \cong$ (by a generalisation of corollary 3.1 and proposition 3.4) $[R']_\omega^a$; $((P' \parallel_S Q') \{R/a\}^*) = [R']_\omega^a$; $((P' \parallel_S Q') [a \rightsquigarrow R] \sigma) = [R']_\omega^a$. We have the sequential composition is monotone w.r.t $\cong$. So $([R']_\omega^a; ((P' \parallel_S Q') [a \rightsquigarrow R]), [R']_\omega^a; ((P' \parallel_S Q') [a \rightsquigarrow R] \sigma)) \in \mathcal{R}$.

Theorem 3.1 Let $P \in \mathbf{RL}$ then,

$$\llbracket P \rrbracket \cong \llbracket P \sigma \rrbracket$$

Proof. We have to prove that $\mathcal{R} = \{(P, P \sigma) \mid P \in \mathcal{P}_1 \cup \mathcal{P}_2\} \cup \cong$ is a bisimulation. The complete proofs are given in the full version of the paper.

4 Conclusion and related work

In this paper we have reported a durational actions process algebra equipped with action refinement combinator. This latter has been investigated in its two sides, syntactic and semantic. We have defined a timed syntactic action refinement that is based on a disciplined timed syntactic substitution, which is correct w.r.t the timed semantic action refinement. This result is interesting for applications of action refinement a) to know how to implement a timed semantic action refinement by a disciplined timed syntactic substitution, and b) how a syntactically refined specification can be implemented in a distributed and timed efficient way. This work is considered as a follow-up to the various studies done on this topic. Here in the following, we are going to compare our contribution to only some of them that are much related to it.

In [15], the authors study the same question for an un-timed language interpreted on flow event structures. Consequently, the approach they consider is denotational and of causality-oriented model. Then, they establish the necessary and sufficient *semantic* and sufficient *syntactic* conditions for that the syntactic refinement based on the general syntactic substitution coincides with the semantic refinement. Indeed we can say that they have showed the reasons of the problem but they have not remedy it, since they have not proposed any syntactic refinement that considers the terms that do not satisfy the condition and that agrees with the semantic refinement.

In [28], the authors tackle the problem of syntactic action refinement in presence of multiway synchronisation. This constitutes the common issue and the starting point for our study of the equivalence between the timed syntactic and semantic action refinement.

The approach of syntactic refinement, as it is investigated in [1, 2] for a CCS-like language, is fundamentally different from ours. Indeed, they consider the syntactic substitution without limitations and try to find the equivalence relation that is congruent w.r.t the syntactic refinement operator. Their approach allows refinements that are not intuitive and therefore not correct w.r.t to the semantic refinement. Consider for instance, the process $P = (a.nil|\bar{a}.nil)$. P can behave either asynchronously by executing the sequence $\langle a, \bar{a} \rangle$ or $\langle \bar{a}, a \rangle$ or in a synchronous way by executing $\langle \tau \rangle$. If we refine syntactically a -action with $a_1; a_2$ and following the approach in [1], we obtain $(a_1.a_2.nil|\bar{a}_1.\bar{a}_2.nil)$ that may carry out the sequence $\langle a_1, \bar{a}_1, \tau \rangle$ that we consider as an non-intuitive execution. Their syntactic refinement does not respect the semantics of parallelism and so that of process term in high abstract level. The above execution is not allowed by our syntactic refinement. Indeed, following our approach, the possible execution sequences are: $\langle a_1, a_2, \bar{a}_1, \bar{a}_2 \rangle$, $\langle \bar{a}_1, \bar{a}_2, a_1, a_2 \rangle$ or $\langle \tau, \tau \rangle$.

We have defined a timed syntactic action refinement on a durational process algebra, we have not shown that it is monotone w.r.t to the I-durational bisimulation but we conjecture that it is so. An important issue to study is to find the appropriate vertical implementation relation $\lesssim^\rho$ [27] which is formally a triple consisting of specification, implementation and refinement function. This relation may be a formalisation of the notion of vertical design step for the timed efficient transformation of timed specifications.

References

- [1] L. Aceto and M. Hennessy. Adding Action Refinement to a Finite Process Algebra. In *Proceedings of International Colloquium on Automata, Languages and Programming, LNCS*, 1991.
- [2] L. Aceto and M. Hennessy. Towards action-refinement in process algebras. *Information and Computation*, (103):204–269, 1993.
- [3] L. Aceto and D. Murphy. On the Ill-timed but Well-caused. In Eike Best, editor, *In Proceedings of CONCUR’93, LNCS 715*, pages 97–111. Springer-Verlag, 1993.
- [4] L. Chen. An Interleaving Model for Real-Time System. Technical Report ECS-LFCS-91-184, Laboratory for Foundations of Computer Science, University of Edinburgh, November 1991.
- [5] Ph. Darondeau and P. Degano. Event Structures, Causal Trees, and Refinements. In *Proceedings of Mathematical Foundations of Computer Science, LNCS 452*, 1990.
- [6] Ph. Darondeau and P. Degano. About Semantic Action Refinement. In *Fundamenta Informaticae XIV*, pages 221–234, 1991.
- [7] P. Degano and R. Gorrieri. Atomic refinement in process description languages. In A. Tarlecki, editor, *Proceedings of Mathematical Foundations of Computer Science*, pages 121–130. Springer-Verlag, 1991.
- [8] A. Dekdouk and Schaff A. Towards an Algebraic Semantics of a Timed LOTOS. In *AMAST Workshop on Real-Time Systems, Models and Proofs*, Bordeaux, France, June 1995.
- [9] A. Dekdouk and Schaff A. Syntactic Action Refinement in a Timed Process Description Language. In Dominique Méry, editor, *In Proceeding of First International Workshop on Formal methods for parallel programming: theory and applications*, Honolulu, April 1996.
- [10] R. J. Van. Glabbeek. *Comparative Concurrency Semantics and Refinement of Actions*. PhD thesis, Vrije Universiteit te Amsterdam, 1990.
- [11] R. J. Van. Glabbeek. The Refinement Theorem For ST-bisimulation Semantics. In M. Broy and C.B. Jones, editors, *Proceedings of IFIP Working Conference on Programming Concepts and Methods*, Sea of Gallilee, Israel, 1990. North-Holland.
- [12] R. J. Van. Glabbeek and U. Goltz. Refinement of actions in a Causality Based Models. In W.P. de Roever J.W. de Bakker and G. Rozenberg, editors, *Proceedings of Stepwise Refinement of Distributed Systems, LNCS 430*, pages 267–300, 1989.
- [13] R. J. Van. Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency. In A.J. Nijman J.W. de Bakker and P.C. Treleaven, editors, *Proceedings of PARLE, LNCS 259*, pages 224–242. Springer-Verlag, 1987.
- [14] R. J. Van. Glabbeek and W.P. Weijland. Refinement in Branching Time Semantics. In *International Conference on Algebraic Methodology and Software Technology*, pages 197–201, Iowa City, 1989.
- [15] U. Goltz, R. Gorrieri, and A. Rensink. On Syntactic and Semantic Action Refinement. In Hagiya and Mitchell, editors, *Proceedings of Theoretical Aspects of Computer Science, LNCS 789*, pages 385–404. Springer-Verlag, 1994.
- [16] R. Gorrieri, M. Roccetti, and E. Stancampiano. A Theory of Processes with Durational Actions. In *Theoretical of Computer Science*, 1994.

- [17] J.F. Groote. Specification and Verification of Real Time Systems in ACP. In R. Probert L. Logrippo and H. Ural, editors, *Proceedings of Protocol Specification, Testing and Verification*, pages 261–274, North-Holland, Amsterdam, 1990.
- [18] L. Léonard, G. Leduc, D. de Frutos, C. Llana, L. Miguel, J. Quemada, and G. Rabay. Belgian-Spanish Proposal for a Time Extended LOTOS. in: J. Quemada, ed., *Revised Draft on Enhancements to LOTOS, ISO/IEC JTC1/SC21/WG1 N..*, 1994.
- [19] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-verlag edition, 1980.
- [20] R. Milner. *Communication and Concurrency*. Prentice-Hall International, Englewood Cliffs edition, 1989.
- [21] F. Moller and C. Tofts. A Temporal Calculus of Communicating Processes. In J.C.M. Baeten and J.W. Klop, editors, *Proceedings of CONCUR'90, LNCS 458 (Theories of Concurrency: Unification and Extension)*, pages 401–415, Amsterdam, The Netherlands, August 1990. Springer-Verlag.
- [22] X. Nicollin and J. Sifakis. The Algebra of Timed Processes ATP: Theory and Application. Technical Report RT-C26, LGI-IMAG, November 1991.
- [23] M. Nielsen, U. Engberg, and K.S. Larsen. Fully Abstract Models for a Process Language with Refinement . In *Proceedings Rex School on Linear Time, Branching Time and Partial Order in Concurrency, LNCS 354*, pages 523–548, 1989.
- [24] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proc. 5th GI conference on Theoretical Computer Science, LNCS 104*, pages 167–183. Springer-Verlag, 1981.
- [25] G. D. Plotkin. A Structured Approach to Operational Semantics. Technical Report DAIMI-FN-19, Aarhus University, Denmark, 1981.
- [26] G. M. Reed and A. W. Roscoe. A Timed Model for Communicating Sequential Processes. *Theoretical Computer Science*, 58:249–261, 1988.
- [27] A. Rensink. *Models and Methods for Action Refinement*. PhD thesis, University of Twente, August 1993.
- [28] D. E. Saïdouni and J.P. Courtiat. Syntactic Action Refinement in Presence of Multiway Synchronisation. In *Proceedings of Workshop on Semantics of Specification Languages*, Utrecht, Netherland, 1993. Springer-Verlag.
- [29] D.E. Saïdouni. *Sémantique de maximalité: Application au raffinement d'action dans LOTOS*. PhD thesis, LAAS, France, January 1996.
- [30] W. Vogler. Failures semantics based on interval semiwords is a congruence for refinement. In *STACS 90, LNCS*. Springer-Verlag, 1989.
- [31] Y. Wang. Real-time behaviour of asynchronous agents. In J.C.M Baeten and J.W. Klop, editors, *LNCS 458, Proceedings of CONCUR'90 (Theories of Concurrency: Unification and Extension)*, pages 502–520, Amsterdam, Springer-Verlag, August 1990..

A The operational semantics of $\mathbf{RL}_{\widehat{at}}$

Definition A.1

We will write $\llbracket P \rrbracket$ for the transition system $(\mathcal{P}_2, \mathcal{A}'', \rightarrow^2, P)$ of process term P generated by the following formal system. $\mathcal{A}'' \in \mathbf{A} \times \mathbf{A} \cup \{\sqrt{\cdot}, \perp\} \times \mathbb{R}^+$.

$$\begin{array}{c}
 (Ex) \frac{}{\varepsilon \xrightarrow[\phi(\varepsilon)]{\sqrt{\cdot}}^2 \delta} \quad (Act) \frac{}{a \xrightarrow[\phi(a)]{\alpha}^2 \varepsilon} \quad (Del) \frac{}{wait(d) \xrightarrow[d]{\perp}^2 \varepsilon} \\
 (SC1) \frac{P \xrightarrow[\varphi]{[\alpha]^\beta} P' \quad (\alpha \neq \sqrt{\cdot})}{P; Q \xrightarrow[\varphi]{[\alpha]^\beta} P'; Q} \quad (SC2) \frac{P \xrightarrow[\varphi]{[\alpha]^\beta} P', \quad Q \xrightarrow[\varphi']{[\alpha]^\gamma} Q'}{P; Q \xrightarrow[\varphi']{[\alpha]^\gamma} Q'} \\
 (Ch) \frac{P \xrightarrow[\varphi]{\alpha} P'}{P + Q \xrightarrow[\varphi]{\alpha} P', \quad Q + P \xrightarrow[\varphi]{\alpha} P'} \\
 (Hi) \frac{P \xrightarrow[\varphi]{[\alpha]^\beta} P'}{\partial_H(P) \xrightarrow[\varphi]{[\alpha]^\beta} \partial_H(P'), \quad \partial_H(P) \xrightarrow[\varphi]{[\alpha]^\beta} \partial_H(P') \quad (\beta \notin H)} \\
 (PC1) \frac{P \xrightarrow[\varphi]{[\alpha]^\beta} P' \quad (\beta \notin S \cup \{\sqrt{\cdot}\})}{P \parallel_S Q \xrightarrow[\varphi]{[\alpha]^\beta} P' \parallel_S Q, \quad Q \parallel_S P \xrightarrow[\varphi]{[\alpha]^\beta} Q \parallel_S P'} \\
 (PC2) \frac{P \xrightarrow[\varphi]{[\alpha]^\beta} P', \quad Q \xrightarrow[\varphi]{[\alpha]^\beta} Q' \quad (\beta \in S \cup \{\sqrt{\cdot}\})}{P \parallel_S Q \xrightarrow[\varphi]{[\alpha]^\beta} P' \parallel_S Q'} \\
 (\widehat{At}) \frac{P \xrightarrow[\varphi]{[\alpha]^\beta} P'}{[P]^a \xrightarrow[\varphi]{[\alpha]^a} [P']^a} \quad (Rec) \frac{P \{ \mu x[P]/x \} \xrightarrow[\varphi]{[\alpha]^\beta} Q}{\mu x[P] \xrightarrow[\varphi]{[\alpha]^\beta} Q} \\
 (Ref1) \frac{P \xrightarrow[\varphi]{[\mu]^\nu} P' \text{ and } \rho(\mu) \xrightarrow[\varphi']{[\alpha]^\beta} Q \text{ and } (\alpha = \sqrt{\cdot})}{P[\rho] \xrightarrow[\varphi']{[\alpha]^\nu} P'[\rho]} \\
 (Ref2) \frac{P \xrightarrow[\varphi]{[\mu]^\nu} P' \text{ and } \rho(\mu) \xrightarrow[\varphi']{[\alpha]^\beta} Q \text{ and } (\alpha \neq \sqrt{\cdot})}{P[\rho] \xrightarrow[\varphi']{[\alpha]^\nu} [Q]^\nu; P'[\rho]} \\
 (Ref3) \frac{P \xrightarrow[\varphi]{[\alpha]^\beta} P' \quad \alpha \notin dom(\rho)}{P[\rho] \xrightarrow[\varphi]{[\alpha]^\beta} P'[\rho]}
 \end{array}$$