Towards specification and proof of asynchronous systems

Boubakar Gamatié

INRIA

CENTRE DE RENNES

IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (3) 954 90 20

Rapports de Recherche

N°466

TOWARDS SPECIFICATION AND PROOF OF ASYNCHRONOUS SYSTEMS

Boubakar GAMATIÉ

Décembre 1985

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
Avenue du Général Leclerc
35042 - RENNES CÉDEX
FRANCE
Tél. : (99) 36.20.00
Télex : UNIRISA 95 0473 F

PUBLICATION INTERNE N° 271
NOVEMBRE 1985 - 30 PAGES

TOWARDS SPECIFICATION AND PROOF OF ASYNCHRONOUS SYSTEMS

B. GAMATIE

IRISA/INRIA Campus de Beaulieu
35042 RENNES CEDEX (FCE)

Abstract

A communication based language is presented which allows to specify asynchronous systems of broadcasting processes. Together with this language, we define an equivalence which is a congruence, merging both notions of complexity and non-determinism. It allows to provide the processes set with a fully defined (and convergent) minimal element, and is characterized by an axioms set together with finitary rules, that we proved to be sound and complete.

Résumé

Nous présentons un langage permettant de spécifier les systèmes asynchrones de processus communicants par diffusion. Ce langage est muni d'une relation d'équivalence que nous montrons être une congruence, et qui est complètement caractérisée par un système d'axiomes, comportant des règles d'induction finitaires. L'équivalence est engendrée par un préordre, lequel, compte-tenu d'une relation d'ordre définie sur l'ensemble des actions rend compte à la fois de la complexité (des langages acceptés) et de la notion de non déterminisme. Elle permet de décider de l'équivalence ou non de deux processus quelconques. En particulier l'ensemble des processus est muni d'un élément minimal qui est un processus fini et non-divergent.

Key-words

Aysnchronous systems, broadcasting processes, non-determinism, equivalence and congruence of processes, least process.

I- INTRODUCTION

Recently, several papers have been published, aiming in defining and formalizing the concept of process (AB, AN, DK, HBR, HM, HN, KM, M1...). This is essentially due to the fact that newly proposed languages integrate such notions as explicit communications and parallelism. And in fact a process can be intuitively seen as an object which can both communicate values (of data) and signals (for control), and proced simultaneously with others processes (including its "partners" for communications).

More precisely we consider that a process is an object wich can only communicate. Now this communication may pressuposes another processes (partners) ; that is the communicated items may be either visible by another process, there the communication is said to be external or addressed to some "component" of the process, and thus not visible from "outside", the communication is then said to be internal. However any action a process could perform is a communication one.

Generally, a set of processes is said to be asynchronous if it is "insensible" to time considerations, in contrast with synchronous ones, where time dependancies are a crucial question. Models for process languages resually relate these notions to the different possible states of a process, in stating that internal actions occurrences must not disturb the behaviour of a process, under hypothesis of asynchrony ; in models for C.C.S. based languages (for example (M1) , (D)), this amounts to set the equivalence of t and τt where τ represents internal actions, t a process and τt a process which first performs an internal action, and then behaves like t. However this equivalence is not valid in all contexts, particulary, considering non-deterministic environments.

As for us, we define an asynchronous system, as a set of processes where the inactivity of any one among them doesn't induce inactivity of all the system , in contrast with synchronous ones where the "death" of any component induces the "death" of the whole system. Relate this notion to process states amount for us to allow a process to remain idle (in fact to perform a null action different from τ) during a finite amount of time without changing it's state.

So, our approach is original in that we relate asvnchrony not with the possibility of performing internal action without state changes, but with that of not performing any action (during a finite amount of time). Formally that means that our operation of composition of actions will admits as neutral

element, not the internal action, but the absence of actions (to be defined).

Relying upon this idea we define structures for expressing asynchronous parallel algorithms (in fact these structures allowes us to define certain class of both synchronous and asynchronous applications (G1)). In our approach, as earlier stated, a process can change its state, exclusively by performing communication actions. Thus recursivity which allows process state changes without communication, is not to be considered any more. As a counterpart, we introduce a "guarded parallel iteration" operation, which allows to define infinite process using communication actions (which may be either internal or external). Another important point to mention here, is that our structures are intended to model "real" parallelism ; so the communication actions use a "multi-level" rendez-vous scheme ; that is any item to be communicated to the environment is in fact broadcasted, and thus several processes (more than two) can communicate simultaneously.

All these differences from usually proposed approaches, quickly lead us to problems of finding appropriate relations between systems of processes, that is finding an adequat equivalence relation over the set of processes. In fact this important question has been emphasized by all the proposals of model for processes ; more over the proposed equivalences have been axiomatized (as an example we mention the well known CCS calculus (M1)) , unfortunatly these resuls do not fit our approach.

Most of authors now agree to say that two processes are equivalent if no observation allow to distinguish one from the other. An observation being an interaction between the (observed) process and another process (or system) which is called the observer. In the case of communication based languages, an interaction reduces to a communication of the process. Thus an observation of a process can change it's state. Therefore, an observation can be modelled by a binary relation over the set of (states of) processes, which relates two states, before and after the communication. Now two processes are said to be congruent if any one can be replaced by the other in any context without observable differences, and the observational congruence is the largest congruence included in observational equivalence.

Our opinion is that observational equivalence must be a congruence, because it relies on observations which happens to be the only means a process has to interact with it's environment. Unfortunately this is not so general.

An earlier proposed equivalence reduces to identify of accepted languages ; but it has been quickly discarded, because of it's poorness ; in particular it does not capture such notions as deadlocks or divergence ([HN]). Milner proposed the recursive equivalence which takes into account the different intermediate states reached during computations [M] , [HM]. This equivalence, and all those based on the notion of bisimulation [AB] have been shown too discriminative [D] , [HN], because they sometimes differentiate processes whose internal activities have the same external effects. Moreover, this equivalence is not a congruence. However the congruence it contains has been axiomatized, leading to the C.C.S. calculus.

Later on, testing equivalence was proposed by Hennessy and De Nicola [HN] in order to identify the internally different but observationally equivalent processes. But this equivalence is still not a congruence. But the congruence it contains has been axiomatized, and in [HN], a strictly operational characterization of this congruence is provided. Unfortunately the set of processes is provided with a minimal element which can perform an infinity of (internal) actions, while the whole set of processes is made in an algebraic cpo, and that a process is greater than any of it's finite approximants. One immediately notices that the existence of a minimal (finite and non divergent) element would be very nice. since simple techniques would be usable to define process semantics. In [D1] Darondeau defined the implementation preorder, which generates a congruence, unfortunatly the minimal process is still the totally divergent one, and moreover this congruence allows the law $t = \tau t$, which we early stated as unacceptable in our framework.

In this paper, we define a preorder based on (pre)order relations we provide with the actions sets ; our preorder relies on two fundamental intuitions :

1) the more you do, the greater you are

that is processes will first be compared relying on their accepted languages.

2) the less the environment can influence your decisions, the greater you are

that is, non-determinism (evaluated as the ability the environment has to influence communication actions of processes) will also be taken into account.

As an important consequence, the least process will be deterministic, and will accept the least language. Moreover, the equivalence this preorder generates, turns out to be a congruence.
It is presented in chapter III, and a sound and complete axiomatization is provided in chapter IV. Right now, in chapter II, we present our language (Called Concurrently Functioning Agents), its syntax and operational semantics.

II- THE LANGUAGE

As earlier stated, the language we consider here, relies heavity on the ideas of communication, already used in other process language approaches : Milner's C.C.S. (M1), HOARE'S C.S.P. (H)... In fact except the essential point that in our approach, a process cannot change it's state without communicating, most of our processes could be easily expressed, using for example the asynchronous calculus ASCCS of (M2). In our ease, the actions a process can perform are exclusively communication actions (wether internal or visible ones), which may be composed of several (but finite anyway) basic events to be performed simultaneously.

1) The actions set

First of all, we assume the existence of the following sets :

a) T a set of types, for example integers, boolean,...

b) X a set of identifiers, rangerd over by x,y,z,...

c) $V = \bigcup_{t \in T} V_t$ a set of typed values ; given t in T, V_t contains all t-typed values, and is provided with some basic functions which allow to construct new values from existing ones.

d) $\Lambda = \bigcup_{t \in T} \langle \Lambda_t \times \mathbb{N} \rangle$ a set of typed and numbered communication ports ; for each t in T and integer i, $\langle \Lambda_t \times \{i\} \rangle$ contains all t-typed of number (or level) i. A t-typed port can be used only for t-typed values communications. ($\mathbb{N}$ represents the set of positive integers).

e) $\mathcal{A}_0 = A \cup \overline{A} \cup \tilde{A} \cup \{\tau\}$ a set of basic events representing elementary communication events s.t.

* τ represents any internal communication action
* $A = \{\langle \lambda ? x \rangle \mid \lambda \in \langle \Lambda_t \times \mathbb{N} \rangle, x \in X\}$ contains receiving actions
* $\overline{A} = \{\langle \lambda ! v \rangle \mid \lambda \in \langle \Lambda_t \times \mathbb{N} \rangle, v \in V_t\}$ contains sending actions
* $\tilde{A} = \{\langle \lambda^V v \rangle \mid \lambda \in \langle \Lambda_t \times \mathbb{N} \rangle, v \in V_t\}$ contains broadcasting actions.

definition 1 we let "$\sqsupset$" and "$(\ )$" denote two functions respectively from $\mathcal{A}_0$ to Λ, and from $\mathcal{A}_0$ to $(V \cup X)$, s.t.

i) $\boxed{a} = \lambda$ iff $a \equiv \langle \lambda ? x \rangle$ or $a \equiv \langle \lambda ! v \rangle$ or $a \equiv \langle \lambda^V v \rangle$

ii) $\boxed{\tau}$ and $[\tau]$ are undefined

iii) $(a) = v$ if $a \in \tilde{A} \cup \overline{A}$ & $a \equiv \langle \lambda ! v \rangle$ or $a \equiv \langle \lambda^V v \rangle$ else
if $a \in A$ & $a \equiv \langle \lambda ? x \rangle$ then $(a) = x$■

definition 2 we let "$\sim$" denote a binary relation over the set of finite subsets of $\mathcal{A}_0$: $\mathcal{P}_f(\mathcal{A}_0)$ s.t. given a finite set $a \in \mathcal{P}_f(\mathcal{A}_0)$, let

$\overline{a} = \{\overline{\alpha} \mid \alpha \in a \cap A\}$ and $\underline{a} = \{\overline{\alpha} \mid \alpha \in a \cap (\overline{A} \cup \tilde{A})\}$, and $\overline{\underline{a}} = \overline{a} \cup \underline{a}$

then if $a \in \mathcal{P}_f(\mathcal{A}_0)$, $b \in \mathcal{P}_f(\mathcal{A}_0)$, $a \sim b$ iff

i) $\overline{a} = \overline{b}$
ii) $a \cap (\overline{A} \cup \tilde{A}) = b \cap (\overline{A} \cup \tilde{A})$ ■

One immediatly verify that "$\sim$" is a preorder relations over $\mathcal{P}_f(\mathcal{A}_0)$; we denote $\approx$ the equivalence relation it generates over $\mathcal{P}_f(\mathcal{A}_0)$; that is $a \approx b \Leftrightarrow a \sim b$ and $b \sim a$.

definition 3 actions represent finite subset of $\mathcal{A}_0$ which discards the possibility of sending (or broadcasting) simultaneously two (or more) different values via the same port. Two equivalent actions (under $\approx$) are denoted in the same way ; that is our actions set, denoted $\mathcal{A}$, represents $\mathcal{P}_f(\mathcal{A}_0)/\approx$ -

definition 4 we provide $\mathcal{A}$ with a composition operation denoted "." , which allows to form new actions, given compatible ones. Two actions a and b are said to be compatibles (well denote a # b) iff

$(\underline{a} \cap \underline{b} = \emptyset)$ or $(a = \{\tau\})$ or $(b = \{\tau\})$. Now composition is defined as follows :

i) $\forall a \in \mathcal{A}$, $a \neq \emptyset \Rightarrow a.\{\tau\} = a$
ii) $\forall a \in \mathcal{A}$, $a.\emptyset = a$
iii) $\forall a, b \in \mathcal{A}$, $(a \neq \{\tau\} \vee b \neq \{\tau\})$, $a \# b \Rightarrow a.b\ (a\backslash\{\tau\} \cup b\backslash\{\tau\})$ ■

One immediatly checks that the composition of two compatible actions is actually an action, and that for any actions a,b and c s.t. a # b # c, then

1) $a.(b.c) = (a.b).c$
2) $a.b = b.c$
3) $a.a = a$

That is, the subset of compatible actions, provided with composition is an abelian idempotent monoïd, $\emptyset$ being the neutral element. Note also that for the set of actions different from $\emptyset$, $\{\tau\}$ acts like a neutral element. This composition is different from those usually proposed for communication based languages because it's neutral element is no more the internal action $\{\tau\}$, but $\emptyset$, the absence of actions : neither internal nor external ones. Moreover it takes explicitly our multi-level rendez-vous scheme.

notation : henceforth, we denote the internal action $\{\tau\}$, in the sameway as the internal event τ (without brakets).

2) Syntax

definition 1 we let $\mathcal{R}$ denote a set of mappings over Λ. These mappings are generalized to $\mathcal{A}_0$ and $\mathcal{A}$ (and denoted the sameway) as follows :
1) given ϕ in $\mathcal{R}$ and α in $\mathcal{A}_0$, then

- $\phi(\alpha) = \tau$ if $\alpha = \tau$ or $\phi(\underline{\alpha}) \notin \Lambda$ and $\alpha \in \tilde{A}$
- $\phi(\alpha) = \emptyset$ if $\phi(\underline{\alpha}) \notin \Lambda$ and $\alpha \in A \cup \bar{A}$
- if $\phi(\lambda) \in \Lambda$, then $\phi(<\lambda ? x>) = <\phi(\lambda) ? x>$, $\phi(<\lambda ! v>) = <\phi(\lambda) ! v>$ and $\phi(<\lambda^{\sim} v>) = <\phi(\lambda)^{\sim} v>$.

2) given a and b in $\mathcal{A}$, and ϕ in $\mathcal{R}$ then

- $\phi(a) = \{\phi(\alpha) \mid \alpha \in a\}$ if this set belongs to $\mathcal{A}$,
  $= \emptyset$ otherwise.
- $\phi(a.b) = \phi(a).\phi(b).$ if both $\phi(a)$ and $\phi(b)$ belongs to $\mathcal{A}$,
  $= \emptyset$ otherwise

Now C.F.A.'s terms are those of the word algebra $w_{\sum}$ generates by the following signature ($\sum_i$ contains all n-ary operators).

1) $\sum_0 = \{nil\}$
2) $\sum_1 = \{a : \mid a \in \mathcal{A}\backslash\{\emptyset\}\} \cup \{a \rightarrow \mid a \in \mathcal{A}\backslash\{\emptyset\}\} \cup \{(\phi) \mid \phi \in \mathcal{R}\}$
3) $\sum_2 = \{+, \&\}$
4) $\sum_n = \emptyset \ \forall \ n \geq 3.$

Operators a : and a → are prefixed, (ϕ) is postfixed and + and & are infixed. Note that we do not use the internal action τ in the operators of set $\sum_1$ of our signature. This is essentially because we consider that τ represents the behaviour of parts of a system, whose activity one don't want to aware about ; simultaneously, we assume that what is internal or external within a system, is uniquely a matter of taste. That's why our internal actions only result from application of $\mathcal{R}$'s elements to external actions. In some sens $\mathcal{R}$'s elements allow changes to the level of abstraction from which one is observing the system, and thus may mask some parts of that system, generating then, internal actions. We'll formalize these notions while defining the semantics of our language. Before, let's pursue this informal presentation and examine intuitively what $\sum$'s operators are intended to do.

Nil operator (nil), guarding (a :), sum (+) and renamings ((ϕ)) are quite similar to those used in C.C.S. calculus. For example

$((a:nil) + (b:nil))(\phi)$ where $\phi(a) = a$ and $\phi(b) = \tau$, represents a term which can either perform an a-action and "die", or perform an internal action and then "die" ; that is, it may hever communicate with its environment.

Operator "&" represents parallel composition. t_1 & t_2 denotes a term composed of two subterms t_1 and t_2, which can proceed concurrently, and communicate. The communication is constrained by the usage of the same port, and a term can remain idle (for only a finite delay) though it might perform some action = the composition is asynchronous. The difference from for example C.C.S.'s composition operator resides in that "&" takes explicitly the "multi-level rendez-vous" scheme into account : any sent value is actually broadcasted toward the environment of the process, so that any other process which is ready to import a value via the sending-port, could receive it.

Operator $a \rightarrow$ (guarded parallel iteration) has no counterpart into C.C.S. It allows to generate an infinitu of terms which proceed concurrently. However this process of terms generating can be completely controlled by the environment : a new term is generated if and only if the "guard" a is "passed". Therefore gpi's allow to perform infinite sequences of actions. For example $(\alpha?x \rightarrow \alpha!\ x : nil)\ \&\ \alpha!\ v : nil$ can perform the infinite sequence $\langle \alpha, 0\rangle\ !\ v.\ \langle \alpha, 1\rangle\ !\ v.\ \langle \alpha, 2\rangle\ !\ v.\ \ldots\ \langle \alpha, n\rangle\ !\ v\ \ldots$ In fact each newly generates term has all its port's level incremented by one. This allows to distinguish different instances of generated terms. Now if we denote $\emptyset_0$ the renaming which changes all ports's level into zero, then

$((\alpha?\ x \rightarrow d\ !\ x : nil)\ \&\ \alpha!\ v : nil)\ (\emptyset_0)$ may perform the sequence $\alpha,o\ !\ v.\ \ \alpha,o\ !\ v\ \ldots\ \alpha,o\ !\ v\ \ldots$ That it is alway able to send value v via port α_0.

3) Operational semantics

The semantics of C.F.A.'s terms relies essentially on two notions : first that of interface which contains the immediately possible actions of a term, including internal ones. Secondly the notion of experimentation relations which are binary relations parameterized by actions. Let Ra be such a relation, with parameter a, and t_1 t_2 be two terms ; then t_1 Ra t_2 if t_2 represents a state t_1 can reach after having performed action a. The semantics is given by means of conditional rules (à la Plotkin (P)), using these experimentation relations.

definition 1 we denote $\xrightarrow{a}$ the relation Ra, and call it basic experimentation relation. Such a relation can be generalized to sequence of actions as follows : let's denote it $\xrightarrow[*]{s}$ where $s \in \mathcal{A}^*$ then

1) $t \xrightarrow[*]{\varepsilon} t$ and $t \xrightarrow[*]{\varepsilon} t' \Rightarrow t' \equiv t$, $\forall t \in$ C.F.A.

2) $t \xrightarrow[*]{sa} t_1 \iff \exists t' : t \xrightarrow[*]{s} t'$ and $t' \xrightarrow{a} t_1$ ■

definition 2 let I be a total mapping of C.F.A. to $\mathcal{P}_f(\mathcal{A})$ s.t. for any term t, I(t) denotes the set of actions t can immediately perform:

$\forall a \in \mathcal{A}$, $a \in I(t) \iff \exists t' : t \xrightarrow{a} t' \wedge a \neq \emptyset$. I(t) is called the interface of t.

definition 3 1) given a in $\mathcal{A}$ and α in $\wedge$ s.t. $\alpha \in \underline{a}$, we denote $V_a(\alpha)$ the value exported via port α, during action a.

2) for any port α in $\wedge$, any identifier x in X and any term t, x is said to be bound by α in t if t has expressions a : t' or a → t' as subterms, and a contains an event γ s.t. $\lceil\gamma\rceil = x$ and $\overline{\gamma} = \alpha$. The set of all identifiers bounded by α in t is denoted $X_t(\alpha)$.

remark for any $a \in \mathcal{A}$ and $\alpha \in \wedge$, $V_a(\alpha)$ is well defined since different values cannot be exported via the same port during an action.

We now give the rules defining the operational semantics of C.F.A.

R_0) $\forall t \in$ C.F.A. $t \xrightarrow{\emptyset} t$ and $t \xrightarrow{\emptyset} t' \Rightarrow t' \equiv t$.

R_1) $I(nil) = \emptyset$

R_2) $I(a:t) = \{a\}$

$a:t \xrightarrow{a} t$

R_3) $I(t_1+t_2) = I(t_1) \cup I(t_2)$

$t_1 \xrightarrow{a} t'_1$, $a \neq \emptyset \Rightarrow t_1+t_2 \xrightarrow{a} t'_1$, $t_2+t_1 \xrightarrow{a} t'_1$.

R_4) $I(t(\phi)) = \{\phi(a) \mid a \in I(t) \quad \phi(a) \neq \emptyset\}$

$t \xrightarrow{a} t' \wedge \phi(a) \neq \emptyset \Rightarrow t(\phi) \xrightarrow{\phi(a)} t'(\phi)$

R_5) $I(t_1 \,\&\, t_2) = I(t_1) \cup I(t_2) \cup \{a.b \mid a \in I(t_1), b \in I(t_2), a \# b\}$.

$t_1 \xrightarrow{a} t'_1$, $t_2 \xrightarrow{a} t'_2$, $a \# b \Rightarrow t_1 \,\&\, t_2 \xrightarrow{ab} t''_1 \,\&\, t''_2$, $t_2 \,\&\, t_1 \xrightarrow{ab} t''_2 \,\&\, t''_1$

where $t''_1 = t'_1\langle V_b(\alpha)/x\rangle$ for $\alpha \in \underline{b} \cap \overline{a}$ and $x \in X_{t_1}(\alpha)$

$t''_2 = t'_2\langle V_a(\alpha)/x\rangle$ for $\alpha \in \underline{a} \cap \overline{b}$ and $x \in X_{t_2}(\alpha)$

and $t\langle v/x\rangle$ represents the term obtained by substituting v for x in t.

R_6) $I(a \to t) = \{a\}$

$(a \to t) \xrightarrow{a} ((a \to t) \,\&\, t)$ (ψ^1)

where $\forall \langle\lambda,i\rangle \in \wedge$, $\psi^1(\langle\lambda,i\rangle) = \langle\lambda, i+1\rangle$.

notice that process nil cannot change it's state ; it cannot be rewritten.

III- EQUIVALENCE AND CONGRUENCE OF C.F.A. TERMS

In this section we define an equivalence relation over C.F.A. As we stated earlier, this equivalence is generated by a preorder, so that two terms could be related even if they do not behave in exactly the sameway, but if one among them "approximates" the other, and more over, we impose that two equivalent terms be interchangeable in any context without visible differences. So, while comparing C.F.A. terms, we consider that internal moves which do not affect "external" behaviour must be ignored, and we set that (finite) sequences of internal moves cannot be differentiated from a single internal move.

Therefore our equivalence (and preorder) relies on binary relations called observation relations which are experimentation relations restricted in such a way that they encompasses the fact that internal actions are not visible from "outside" a term.

definition given a in $\mathcal{A}$, $\stackrel{a}{\Longrightarrow}$ denotes a binary relation over C.F.A. called basic a-observation relation, and s.t. $\stackrel{a}{\Longrightarrow} = \xrightarrow[*]{\tau^*} \xrightarrow{a} \xrightarrow[*]{\tau^*}$. These relations are extended to sequences of non τ-actions, and denoted $\stackrel{s}{\Longrightarrow}$ s.t.

1) $\stackrel{\epsilon}{\Longrightarrow} = \xrightarrow[*]{\tau^*}$

2) $\stackrel{sa}{\Longrightarrow} = \stackrel{s}{\Longrightarrow} \stackrel{a}{\Longrightarrow}$.

Our equivalence also relies on relations provided with the actions set :

1) Ordering the actions set

definition 1 we let " $\sqsubset$ " and " $\lesssim$ " denote two binary relations respectively over $\mathcal{A}$ and $\mathcal{P}_f(\mathcal{A})$ s.t.

1) $\forall a, b \in \mathcal{A}$, $a \sqsubset b \iff (a \equiv b) \vee (a \equiv \emptyset) \vee (b \equiv \tau)$

2) $\forall X, Y \in \mathcal{P}_f(\mathcal{A})$, $X \lesssim Y \iff \forall a \in X, \exists b \in Y : a \sqsubset b$ ▪

note that in part 1) of definition 1, $\emptyset$ represent the null action of $\mathcal{A}$.

proposition 1 relation "$\lesssim$" is a preorder relation over $\mathcal{P}_f(\mathcal{A})$, and moreover given X and Y in $\mathcal{P}_f(\mathcal{A})$, the following hold :

1) $\emptyset \lesssim X \lesssim \{\tau\}$

2) $X \lesssim Y \cup \{\tau\}$

3) $X \lesssim Y \wedge \tau \in X \Rightarrow \tau \in Y$

4) $X \subseteq Y \Rightarrow X \lesssim Y$ -

proof immediate.

Note that in assertion 1) $\emptyset$ denotes the empty subset of $\mathcal{A}$.

Before going on and define our equivalence, let's define a relation which will be usefull later on : the divergence predicate :

2) Divergence predicate

definition 1 a term is said to be divergent (we'll denote $t\uparrow$), if it can perform an infinite sequence of internal actions. That is, $t \xrightarrow[*]{\tau\omega}$.

Formally a term t diverges if and only if it verifies the following predicate :

$$\Delta(X) = \exists X' : X \xrightarrow{\tau} X' \wedge \Delta(X')$$

For example the following terms t diverge :

t ≡ 1) $((a \rightarrow t') \,\&\, t_1)\ (\phi) \wedge t_1 \overset{\bar{a}}{\Longrightarrow} t'_1 \wedge t' \,\&\, t'_1 \overset{\bar{a}}{\Longrightarrow}$

and $\phi(a) = \phi(\bar{a}) = \emptyset$, $a \# \bar{a}$ and $\phi(a.\bar{a}) = \tau$.

or 2) $((a \rightarrow t') \,\&\, t_1)\ (\phi) \wedge t_1 \overset{\bar{a}}{\Longrightarrow} \wedge\ a \# \bar{a} \wedge t\uparrow$.

or 3) $t_1 \,\&\, t_2 \wedge (t_1\uparrow \vee t_2\uparrow)$

or 4) $t_1 + t_2 \wedge (t_1\uparrow \vee t_2\uparrow)$

or 5) $t_1\ (\phi) \wedge t_1\uparrow$

or 6) $(a : t_1 \,\&\, t_2) \wedge t_2 \overset{\bar{a}}{\Longrightarrow} \wedge t_1\uparrow \wedge a \# \bar{a}$ ■

definition 2 a term t is said to be s-divergent, we denote $t\uparrow s$ $s \in (\mathcal{A} \setminus \{\tau\})^*$ if there exist t' s.t. $t \overset{s}{\Longrightarrow} t' \wedge t'\uparrow$. We let $t\uparrow$ denote $t\uparrow\varepsilon$ ■

We now define the preorder which generates our equivalence.

3) Ordering C.F.A. terms :

Our preorder relation relies essentially on two notions : first of all that of accepted language. Given two related terms (we'll also use processor agent while speaking about C.F.A. terms), the greater one admits at least the same language as the least one (we use the notion of accepted language of a process in it's usual sens). Secondly the preorder encompasses the notion of non-determinism : if process t is greater than process t', then t' is at least as deterministic as t, in the sens that while accepting a sequence, that

both can accept, the influence of the environment is greater while using the least one, than the greater one.
In fact our preorder is a less restrictive version of that of (G2) and allows us to merge in a single relation, both notions of safe implementation and complexity : the safest agent which implements language L is the least one among all those whose accepted language includes L.

definition 1 given t_1 and t_2 , two C.F.A. terms, t_1 is said to be less than t_2, we denote $t_1 \sqsubset_{\approx} t_2$ iff :

1) $I(t_1) \lesssim I(t_2)$

2) $\forall s \in \mathcal{L}_+^* , t_1 \stackrel{s}{\Rightarrow} \Rightarrow t_2 \stackrel{s}{\Rightarrow} \wedge$

a) $t_1 \uparrow s \Rightarrow t_2 \uparrow s \wedge$

b) $(t_2 \stackrel{s}{\Rightarrow} t'_2 \Rightarrow t'_2 \stackrel{a}{\Rightarrow}) \Rightarrow (t_1 \stackrel{s}{\Rightarrow} t'_1 \Rightarrow t'_1 \stackrel{a}{\Rightarrow}) \vee (t_1 \stackrel{sa}{\not\Rightarrow}) \blacksquare$

Informally, $t_1 \sqsubset_{\approx} t_2$ iff any string t_1 may accept, may also be accepted by t_2, and moreover, any string that t_2 must accept, must be accepted by t_1, provided t_1 may accept it. Here we use the terms may and must under their definitions of (HN). That is t may accept s iff there exist t' s.t. $t \stackrel{s}{\Rightarrow} t'$ and t' must accept a after having accepted s iff $\forall t' : t \stackrel{s}{\Rightarrow} t'$, then $t' \stackrel{a}{\Rightarrow}$ ($t \stackrel{s}{\not\Rightarrow}$ means t' cannot accept s). Notice that when t may diverge after s, then t must not anything after s. So our preorder allows to evaluate both possibilities for terms to accept a given string, and abilities for the environment to influence this string acceptance.

Examples

1) nil $\sqsubset_{\approx}$ t for any t in C.F.A.

proof : 1) $I(nil) = \emptyset \lesssim I(t) \quad \forall t$

2) $\forall s \in \mathcal{L}_+^* , nil \stackrel{s}{\Rightarrow} \iff s = \varepsilon \wedge$ not $(nil \uparrow \varepsilon)$ and
$\forall t \in$ C.F.A. $t \stackrel{\varepsilon}{\Rightarrow} t$, and obviously,
$(t \stackrel{\varepsilon}{\Rightarrow} t' \Rightarrow t' \stackrel{a}{\Rightarrow}) \Rightarrow nil \stackrel{a}{\not\Rightarrow}$ if $a \neq \emptyset$ $\blacksquare$

This example states that the language accepted by nil is also accepted by any other term (nil accepts the language reduced to the empty string) ; moreover, while accepting this language, nil is at least as deterministic as any other agent. That is, the safest term for recognizing the empty string is nil.

2) $a : nil \sqsubseteq_{\sim} a : nil + b : nil$

proof : 1) $I(a : nil) = \{a\} \lesssim \{a,b\} = I(a : nil + b : nil)$.

2) by definition of t,

$a:nil \xRightarrow{s} t \Longrightarrow a:nil + b:nil \xRightarrow{s} t$ for any s and t

and $(a:nil + b:nil \xRightarrow{s} t \Longrightarrow t \xRightarrow{c}) \Longleftrightarrow$

$a:nil \xRightarrow{s} t$ and $t \xRightarrow{c}$ or $b:nil \xRightarrow{s} t \xRightarrow{c}$ that is $s = b\,s'$ thus

$a:nil \xRightarrow{sc} \not\Rightarrow$ -

This example is an instance of a more general statement we'll prove later : summing any process with t, deliver a process greater than t. This can be intuitively justified by noting that sum operator allows to increase both the accepted language and the influence of environment for accepting this language.

3) $a:nil \sqsubseteq_{\sim} ((\alpha? :a :nil) \,\&\, \alpha! : nil)(\phi)$ where $\phi(\alpha) = \emptyset$ and $\phi(a) = a$.

proof let's denote p and q the left and right handside of $\sqsubseteq_{\sim}$; then :

1) $I(p) = \{a\} \lesssim \{\tau\} = I(q)$

2) $p \xRightarrow{s} t \Longrightarrow q \xRightarrow{s} t(\phi)$, and moreover p is completly deterministic.

Here we have a particular form of a more general law well state later : $t \sqsubseteq_{\sim} \tau t$; intuitively one can convince himself of the validity of this law, by noting that t and τt accept the same language. However one can easily exhibit an environment which has more influence on t than on τt (think to $a:nil + \tau: b:nil$ and $\tau:$ anil $(+ \tau: b:nil)$-

For relation "$\sqsubseteq_{\sim}$" to generate an equivalence relation, it must enjoy certain properties ; we state them in the following

proposition 1 relation "$\sqsubseteq_{\sim}$" is a preorder relation over C.F.A. that is $\forall\, t_1, t_2, t_3$ in C.F.A. then

1) $t_1 \sqsubseteq_{\sim} t_1$

2) $(t_1 \sqsubseteq_{\sim} t_2 \wedge t_2 \sqsubseteq_{\sim} t_3) \Rightarrow t_1 \sqsubseteq_{\sim} t_3$.

proof part 1) is obvious. Let's show part 2).

So let t_1 , t_2 and t_3 be CFA terms s.t. $t_1 \sqsubseteq_{\sim} t_2$ and $t_2 \sqsubseteq_{\sim} t_3$ then

1) $I(t_1) \lesssim I(t_3)$ from transitivity of relation "$\lesssim$"

2) suppose $t_1 \xRightarrow{s}$ for $s \in \mathcal{A}^*_+$ then $t_2 \xRightarrow{s}$ and thus $t_3 \xRightarrow{s}$

and obviously $t_1\!\uparrow\!s \Longrightarrow t_2\!\uparrow\!s \Longrightarrow t_3\!\uparrow\!s$.

now let $a \in \mathcal{A}_+$ be such that $\forall t'_3 : t_3 \stackrel{s}{\Longrightarrow} t'_3 , t'_3 \stackrel{a}{\Longrightarrow}$ then two cases are possible :

a) $t_2 \stackrel{sa}{\Longrightarrow\!\!\!\!\!/}$ and then $t_1 \stackrel{sa}{\Longrightarrow\!\!\!\!\!/}$ or

b) $\forall t'_2 : t_2 \stackrel{s}{\Longrightarrow} t'_2, t'_2 \stackrel{a}{\Longrightarrow}$; then by definition

either $\forall t'_1 : t_1 \stackrel{s}{\Longrightarrow} t'_1 , t'_1 \stackrel{a}{\Longrightarrow}$ or $t_1 \stackrel{sa}{\Longrightarrow\!\!\!\!\!/}$ -

Now in order " $\underset{\sim}{\sqsubset}$ " to generate a congruence over C.F.A., it must enjoy some other properties, with regard to $\sum$'s operators. It is well known that preorders preserved by operators of a signature, generate congruence over the word algebra generated by that signature. Actually $\underset{\sim}{\sqsubset}$ is not preserved by all $\sum$'s operators ; however because of its structure, the equivalence it generates is still a congruence over C.F.A.

proposition 2 let t_1, t'_1, t_2 and t'_2 be C.F.A. terms s.t. $t_1 \underset{\sim}{\sqsubset} t'_1$ and $t_2 \underset{\sim}{\sqsubset} t'_2$ then the following hold

1) $a : t_1 \underset{\sim}{\sqsubset} a : t'_1 \quad \forall a \in A_+$

2) $t_1 (\phi) \underset{\sim}{\sqsubset} t'_1 (\phi) \quad \forall \phi \in \mathcal{R}$

3) $a \rightarrow t_1 \underset{\sim}{\sqsubset} a \rightarrow T'_1 \quad \forall a \in A_+$

more over, if $\{s \in A_+^* : (t'_1 \Rightarrow_\wedge t'_2 \stackrel{s}{\Rightarrow}) \wedge (t_1 \stackrel{s}{\Rightarrow} \vee t_2 \stackrel{s}{\Rightarrow})\} \subseteq \{ s \in A_+^* : t_1 \stackrel{s}{\Rightarrow}_\wedge t_2 \stackrel{s}{\Rightarrow} \}$

then 4) $t_1 + t_2 \underset{\sim}{\sqsubset} t'_1 + t'_2$

5) $t_1 \,\&\, t_2 \underset{\sim}{\sqsubset} t'_1 \,\&\, t'_2$

4) Equivalence relation over C.F.A.

definition 1 two C.F.A. terms t_1 and t_2 are said to be equivalent, we denote $t_1 \simeq t_2$ iff $t_1 \underset{\sim}{\sqsubset} t_2 \wedge t_2 \underset{\sim}{\sqsubset} t_1$ -

example $nil \simeq nil\,(\phi) \quad \forall \phi \in \mathcal{R}$

proof 1) $nil \underset{\sim}{\sqsubset} nil(\phi)$ has been shown in part 1) of the examples of previous section-

2) $nil\,(\phi) \underset{\sim}{\sqsubset} nil$ since $I(nil(\phi)) = \emptyset \lesssim I(nil) = \emptyset$, and $nil(\phi) \stackrel{s}{\Longrightarrow} t' \iff s = \varepsilon \wedge t' \equiv nil(\phi)$ and $nil(\phi) \not\equiv \varepsilon$ and nil must not any action (it cannot perform any non-null one) ∎

lemma $\forall t_1, t_2 \in$ C.F.A. , $t_1 \simeq t_2 \Rightarrow \forall s \in \mathcal{A}_+^* \; t_1 \stackrel{s}{\Rightarrow} \iff t_2 \stackrel{s}{\Rightarrow}$.

proof immediate.

Theorem 1 relation "$\simeq$" is a congruence over C.F.A.

proof immediate from previous lemma and proposition 2 of previous section -

This result is an important one among those which differentiate our equivalence from, for example testing equivalence (HN) or those based on bisimulations (HM) which are not preserved by sum operator.

Now, to sum up, we define an equivalence relation over C.F.A., which first of all encompasses our intuition about processes behaviour, and moreover, happens to be a congruence. So we get means for discussing whether or not two processes can be interchangeable in any context. However it may sometimes be difficult to directly deal with the definition of the equivalence, since one must then, examine all the accepted languages, and in case of infinite processes, these are infinite too. So we need another mecanism, in order to finitely carry out properties about these infinite objects.

IV- AXIOMATIZATION OF THE CONGRUENCE :

In this section, we provide a system of equalities and inequalities between C.F.A. terms, which allows to decide whether or not two given terms are equivalent. Our system relies on finitary induction rules, and is proved to be sound and complete with regard to preorder $\sqsubseteq_{\simeq}$. Actually this proof relies on an alternative characterization of relations $\simeq$ and $\sqsubseteq_{\simeq}$ -

1) Alternative characterization of "$\sqsubseteq_{\simeq}$"

definition 1 given n in , we let $\sqsubseteq_{\simeq n}$ denote a binary relation over C.F.A. s.t. : $\forall t_1, t_2 \in$ C.F.A. , $t_1 \sqsubseteq_{\simeq n} t_2 \iff$

1) $I(t_1) \lesssim I(t_2)$

2) $\forall s \in \mathcal{A}_+^{(n)}$, $t_1 \stackrel{s}{\Longrightarrow} \Longrightarrow t_2 \stackrel{s}{\Longrightarrow} \wedge$

a) $t_1 \uparrow s \Longrightarrow t_2 \uparrow s \wedge$

b) $(t_2 \stackrel{s}{\Longrightarrow} t'_2 \Longrightarrow t'_2 \stackrel{a}{\Longrightarrow}) \Rightarrow (t_1 \stackrel{s}{\Longrightarrow} t'_1 \Longrightarrow t'_1 \stackrel{a}{\Longrightarrow}) \vee (t_1 \stackrel{sa}{\Longrightarrow})$

Where $\mathcal{A}_+^{(n)}$ denotes the set of sequences of elements of $\mathcal{A}_+$, whose lenght is less than n. Relations $\sqsubseteq_{\simeq n}$ look very similar to $\sqsubseteq_{\simeq}$, except that $\sqsubseteq_{\simeq n}$ is concerned only with experiments with lenght n at most. Now if we use (the usual) tree-representation for processes, $\sqsubseteq_{\simeq}$ "compare" the n first levels of tress and examine the leaves going out from nodes of level n. This constitutes the fundamental difference from the principle suuporting Milner's recursive equivalence, which is not concernes with interfaces at level n.

proposition 1 $\underset{\sim}{\sqsubseteq}_n$ is a preorder relation over C.F.A., for any n in .
proof quite similar to that for $\underset{\sim}{\sqsubseteq}$. Left to the reader.

proposition 2 $\forall n$, $\underset{\sim}{\sqsubseteq}_n \subseteq \underset{\sim}{\sqsubseteq}_{n-1}$ that is $t_1 \underset{\sim}{\sqsubseteq}_n t_2 \Rightarrow t_1 \underset{\sim}{\sqsubseteq}_{n-1} t_2$, $\forall t_1, t_2$.
proof immediate, noting that $\mathcal{A}_+^{(n-1)} \leq \mathcal{A}_+^{(n)}$.

This proposition states that relations $\underset{\sim}{\sqsubseteq}_n$ constitute a chain, and allows us to relate $\underset{\sim}{\sqsubseteq}$ with relations $\underset{\sim}{\sqsubseteq}_n$.

proposition 3 $\underset{\sim}{\sqsubseteq}$ is the least upper bound of the set of relations $\underset{\sim}{\sqsubseteq}_n$ ie

$$\underset{\sim}{\sqsubseteq} = \bigcap_n \underset{\sim}{\sqsubseteq}_n .$$

proof left to the reader. (Notice that $\forall n$, $\mathcal{A}_+^{(n)} \leq \mathcal{A}_+^*$ and $\mathcal{A}_+^* = \bigcup_n \mathcal{A}_+^{(n)}$).

2) The axiom set

Henceforth relations $\underline{\sqsubseteq}_n$, $\underline{\sqsubseteq}$ stand for preorder relations over terms , which generate equivalence relations " $=_n$" and " $=$ " respectively.
Our system is intended to be used in conjunction whith the following rules :
let t_1 t_2 and t be C.F.A. terms then :

(R_o) $\forall n,\ t_1 \underline{\sqsubseteq}_n t_2 \Rightarrow t_1 \underline{\sqsubseteq}_{n-1} t_2$

(R_1) $t \underline{\sqsubseteq}_n t$

(R_2) $t \underline{\sqsubseteq}_n t_1 \wedge t_1 \underline{\sqsubseteq}_n t_2 \Rightarrow t \underline{\sqsubseteq}_n t_2$

(R_3) $t_1 \underline{\sqsubseteq}_n t_2 \wedge t_2 \underline{\sqsubseteq}_n t_1 \Longleftrightarrow t_1 =_n t_2$

(R_4) $t_1 \underline{\sqsubseteq}_n t_2 \wedge t'_1 \underline{\sqsubseteq}_n t'_2 \Rightarrow$

i) $a : t_1 \underline{\sqsubseteq}_{n+1} a : t_2$

ii) $(\{ s \in A_+^{(n)} : t_2 \overset{s}{\Rightarrow} \wedge t'_2 \overset{s}{\Rightarrow}) \wedge (t_1 \overset{s}{\Rightarrow} \vee t'_1 \overset{s}{\Rightarrow})\} \subseteq \{s \in A_+^{(n)} : (t_1 \overset{s}{\Rightarrow} t'_1 \overset{s}{\Rightarrow})\} \Rightarrow$
$t_1 + t'_1 \underline{\sqsubseteq}_n t_2 + t'_2$

We let "Ω" denote fully divergent terms of C.F.A.: that is $\forall n, \Omega \xrightarrow[*]{\tau^n} \Omega$.
Our system (called so) is the following :

$A_o)$ nil $\underline{\sqsubseteq}_n$ X

$A_1)$ $X + Y =_n Y + X$

$A_2)$ $X + (Y + Z) =_n (X + Y) + Z$

$A_3)$ $X + nil =_n X$

$A_4)$ $X + X =_n X$

$A_5)$ $aX + aY =_n a(\tau X + \tau Y)$

$A_6)$ $X + \tau(Y + Z) =_n \tau(X + Y + Z)$ if $(I(X) \subseteq I(Y) \cup I(Z)) \vee (\tau \in I(Y) \cup I(Z))$
$=_n \tau(X + Y + Z) + \tau(Y + Z)$ anyway

$A_7)$ if $X \equiv \sum_{a \in L1} a:X_a$ and $Y \equiv \sum_{b \in 12} b:Y_b$ then

$$X \,\&\, Y =_n \sum_{a \in L1} a(X_a \,\&\, Y) + \sum_{b \in L2} b\,(X \,\&\, Y_b) + \sum_{a \# b} ab\,(X_a \,\&\, Y_b)$$

$A_8)$ $(a : X)(\phi) =_n \phi(a):X(\phi)$ if $\phi(a) \neq \emptyset$, nil otherwise

$A_9)$ $(X + Y)(\phi) =_n X(\phi) + Y(\phi)$

$A_{10})$ $\mathrm{nil}(\phi) =_n \mathrm{nil}$

$A_{11})$ $a \to X =_n a((a \to X) \,\&\, X)$ (ψ^1)

$A_{12})$ $\Omega + X =_n \Omega + \tau X$

$A_{13})$ a) $X \subseteq_n \tau X$ b) $\tau X \subseteq_n X$ if $\tau \in I(t)$

$A_{14})$ $X \subseteq_n X + Y$ ∎

Interfaces of terms (X, Y, Z,...) are inductively defined as stated in chapter II. Notation $\sum_{a \in L} aX_a$ stands for $a_1 X_1 + \ldots + a_n X_n$ if $L = \{a_1, a_2, \ldots, a_n\}$, ψ^1 is the renaming already defined, $\equiv$ denotes syntactical identity and τX is put for terms like $(a : t \,\&\, \bar{a}:\mathrm{nil})(\phi)$ where $a \# \bar{a}$, $\phi(a) = \phi(\bar{a}) = \emptyset$, and $\phi(a.\bar{a}) = \tau$, and ϕ preserves all actions of t.

Most of these axioms are those usually proposed, when axiomatizing congruence of C.C.S. based languages. For example the four first ones allow to make the set of processes, together with sum operator, into an abelian monoïd with absorption, nil being the neutral element. However there exists some significant differences that we note in the followings :

i) axioms A_5-A_6 are similar to τ-laws of C.C.S., but are not identical, because here, we use the notion of interface. In fact τ-laws of (HM) and those which stand for them in (HN)and (D) , can be deduced from these axioms.

ii) axiom A_{13} in its fullness, has no counterpart : it departs from the famous $t = \tau t$, because of its part b), and considering preorder which only allow $\tau t \subseteq t$, then this axiom allows our preorder to be finer.

iii) axiom A_o is usually absent for other systems, since they generally set Ω to be the least process, so is A_{14}.

Now S_o axioms are rather basic, and more complex derived ones can be deduced from our system. We propose some useful ones in the followings :

proposition 4 given system So, the following hold :

$$d_1)\ X + \tau Y =_n \tau(X + Y) + \tau Y$$

$$d_2)\ a\ \tau X =_n a\ X$$

$$d_3)\ a\ X + a(Y + Z) =_n a\ X + a\ (X + Y) + a\ (X + Z) + a\ (X + Y + Z)$$

$$d_4)\ X + \tau X =_n \tau X$$

$$d_5)\quad X + \tau\ (X + Y) =_n \tau\ (X + Y)$$

$$d_6)\ a\ X + a\ Y + a(X = Y) =_n a\ X + a\ Y \quad \blacksquare$$

notice that $d_1)$ and $d_2)$ are axioms that usually stand for τ-laws -

Recall that we are looking for mechanisms which will allow us to reason about terms equivalence, without directly dealing with the definitions of our relations. Fortunatly system So has properties which allow this.

Theorem 2 system So, together with rules R_0-R_4 is sound and complete for relations $\underset{\sim}{\sqsubseteq}_n$ over C.F.A. That is

$$\forall t_1, t_2 \in \text{C.F.A.} \quad t_1 \sqsubseteq_n t_2 \iff t_1 \underset{\sim}{\sqsubseteq}_n t_2 \quad \text{for any } n \text{ -}$$

Now the axiomatization of relation $\underset{\sim}{\sqsubseteq}$ becomes quite easy. Let's, before augment our rules R_0-R_3 with the following two :

$$R_5)\ \forall n\ ,\ t(\triangleright n) \sqsubseteq t$$

$$R_6)\ (t_1 \sqsubseteq_0 t_2 \wedge t_1 \sqsubseteq_n t_2 \Rightarrow t_1 \sqsubseteq_{n+1} t_2) \Rightarrow t_1 \sqsubseteq t_2$$

where $t(\triangleright n) \overset{s}{\Longrightarrow} t'\ (\triangleright m) \iff t \overset{s}{\Longrightarrow} t' \wedge |s| = n - m \geqslant 0$ -

We allow, here, some kind of "renamings" which restrict strings accepted by a process s.t. their lenght be bounded. Rule $R_5)$ relate relations $\underset{\sim}{\sqsubseteq}_n$ and $\underset{\sim}{\sqsubseteq}$, allowing a finitary form of complete induction (remind that our least element is nil).

Theorem 3 system So, together with rules R_0-R_6 is sound and complete for relation $\underset{\sim}{\sqsubseteq}$ over C.F.A. that is $\forall t_1, t_2 \in \text{C.F.A.}\ t_1 \underset{\sim}{\sqsubseteq} t_2 \iff t_1 \sqsubseteq t_2$ -

The proof of these results is relagated to appendix ; it heavily relies on existence, for C.F.A. term, of normal form, based on convex-primitive sets of finite sets of external actions.

Some conclusions

In this paper, we defined a communication based language which allows to express parallel agents, communicating by broadcasting policy, in an asynchronous framework. Our approach completely relies on intuitions we have about process behaviour :

1) in our opinion, a process can be seen as an abstract computation station, with its own clock, and which is only capable of performing communication actions ; these actions may be either external ie they require participation of some others processes, or internal ie they only involve subcomposents of the process.

2) We define asynchrony, as the ability for a process to perform a finite sequence of null actions (ie not to communicate) without changing it's state. That is, on our case the neutral element of the composition of actions is no more the internal one, but the absence of action. And in fact internal actions are actually "real" actions, restricted in such a way that they do not involve participation of the environment.

3) A process cannot change its state without communicating. So in our approach, infinite processes cannot be expressed recursively. As a counterpart, we use some kind of iteration which allows to generate different (possibly infinite) instances of the same process ; however, any new generation activity is preceded by a "guard passing".

All these points make our approach rather original, and as a consequence, discard the possibility of immediately apply the results till now proposed, in order to formalize processes behaviour.

We then defined an equivalence relation which turns out to be a congruence. This equivalence is generated by a preorder which relies on an ordering we provide with the actions set, and allows us to merge both notions of complexity and non determinism : the least term of a set of terms, accepts as language the intersection of all their accepted language and is at less as deterministic as any one among them. As an important consequence, we proved nil to be the least process, show that sum increases processes ($X \underset{\sim}{\sqsubset} X + Y, \forall Y$) and depart from the law $\tau t = t$.

Moreover, we provide a set of axioms, together with finitary induction rules, that we proved to be sound and complete for our equivalence.

This language allows us to express certain class of parallel systems, and we used it in a previous paper [G1] to specify "data-flow" and "reduction"

applications. However it would be interesting to formally characterize the complexity, and the possible extentions, in order to express more basic classes. Another quite interesting direction, we undertake is to provide a model of asynchronous processes, based on these ideas. In this way, the normal form we defined for C.F.A. terms (based on convex-primitive sets) in some extends easies the task.

REFERENCES

AB : AUSTRY D., BOUDOL G. - Algèbre de processus et Synchronisation
TCS 30,1 (pp. 91-131) - 1984 -

AN : ARNOLD A., NIVAT M. - Comportement de processus
Colloque AFCET : les mathématiques de l'informatique - 1982 -

D : DARONDEAU Ph.- An enlarged definition and complete axiomatization of observational congruence of finite processes
LNCS 137 - 1982 -

DI : DARONDEAU Ph. - Infinitary languages and Fully abstract models of fair asynchrony
R. n° 230 IRISA - 1984 -

DK : DARONDEAU Ph., KOTT L. - On the observational semantics of fair parallelism
Proc ICALP 83, LNCS 154 - 1983 -

G1 : GAMATIE B. - Systèmes de processus communicants et interprétation parallèle de langages fonctionnels
R. n° 320 INRIA - 1984 -

G2 : GAMATIE B. - Observational congruence of non-deterministic and communicating finite processes in asynchronous systems
R. n° 254 IRISA - 1985 -

H : HOARE C.A.R. - Communicating Sequential Processes
CACM 21, vol 8 - 1978 -

HBR : HOARE C.A.R., BROOKES S.D., ROSCOE A.W. - A theory of communicating processes
PRG-16 Oxford University - 1981 -

HM : HENNESSY M., MILNER R. - Algebraic laws for non-determinism and Concurrency
JACM 32,1 - 1985 -

HN : HENNESSY M. , DE NICOLA R. - Testing equivalence for processes
Proc ICALP 83, LNCS 154 - 1983 -

KM : KAHN G., MC QUEEN D.B. - Coroutines and networks of parallel processes
Gilchrist B. (Ed.) Information processing - Amsterdam - 1977 -

M1 : MILNER R. - A calculus of communicating systems
LNCS 92 - 1980 -

M2 : MILNER R. - Calculi for synchrony and asynchrony
TCS 25,3 - 1983 -

P : PLOTKIN G. - A structural approach to operational semantics
DAIMI FN-19 Comp. Sc. Dept. Aarhus University - 1981 -

I) This section is devoted to the proof of results from chapter III.

1)Proposition2 section 3)

A) $a{:}t_1 \underset{\sim}{\subset} a{:}t'_1$

1) $I(a{:}t_1) = [a] \leqslant [a] = I(a{:}t'_1)$

2) $a{:}t_1 =^{s}\Rightarrow \iff s{=}as' \wedge t_1 =^{s'}\Rightarrow \Rightarrow t'_1 =^{s'}\Rightarrow \iff a{:}t'_1 =^{s}\Rightarrow$

$a{:}t_1 \uparrow s \iff s{=}as' \;\&\; t_1 \uparrow s' \Rightarrow t'_1 \uparrow s' \iff a{:}t'_1 \uparrow s$

$(a{:}t'_1 =^{s}\Rightarrow t''_1 \Rightarrow t''_1 =^{b}\Rightarrow) \iff (s{=}as' \;\&\; t'_1 =^{s'}\Rightarrow t''_1 \Rightarrow t''_1 =^{b}\Rightarrow) \Rightarrow$

$(t_1 =^{s'}\Rightarrow p_1 \Rightarrow p_1 =^{b}\Rightarrow) \vee (t_1 =^{s'b} \not\Rightarrow)$ ie

$(a{:}t_1 =^{s}\Rightarrow p_1 \Rightarrow p_1 =^{b}\Rightarrow) \vee (a{:}t_1 =^{sb} \not\Rightarrow)$

B) $t_1 + t_2 \underset{\sim}{\subset} t'_1 + t'_2$

1) $a \in I(t_1+t_2) \iff (a \in I(t_1), \Rightarrow \exists b \in I(t'_1): a \sqsubset b)$

$\vee (a \in I(t_2), \Rightarrow \exists b \in I(t'_2): a \sqsubset b)$

ie: $I(t_1+t_2) \leqslant I(t'_1+t'_2)$

2) $*(t_1+t_2) =^{s}\Rightarrow \iff t_1 =^{s}\Rightarrow \Rightarrow t'_1 =^{s}\Rightarrow \iff (t'_1+t'_2) =^{s}\Rightarrow$

$\vee\ t_2 =^{s}\Rightarrow \Rightarrow t'_2 =^{s}\Rightarrow \iff (t'_1+t'_2) =^{s}\Rightarrow$

$*(t_1+t_2)\uparrow s \iff (t_1\uparrow s \vee t_2\uparrow s) \Rightarrow (t'_1\uparrow s \vee t'_2\uparrow s) \iff (t'_1+t'_2)\uparrow s$

$*((t'_1+t'_2) =^{s}\Rightarrow p \Rightarrow p =^{a}\Rightarrow) \iff (t'_1 =^{s}\Rightarrow p_1 \Rightarrow p_1 =^{a}\Rightarrow) \wedge (t'_2 =^{s}\Rightarrow p_2 \Rightarrow p_2 =^{a}\Rightarrow$

now, since $[s \mid (t'_1 =^{s}\Rightarrow \wedge t'_2 =^{s}\Rightarrow) \wedge (t_1 =^{s}\Rightarrow \vee\ t_2 =^{s}\Rightarrow)] \subseteq [s \mid t_1 =^{s}\Rightarrow \wedge t_2 =^{s}\Rightarrow]$

then obviously $((t_1+t_2) =^{s}\Rightarrow q \Rightarrow q =^{a}\Rightarrow) \vee ((t_1+t_2) \not=^{sa}\Rightarrow)$

C) $(t_1 \;\&\; t_2) \underset{\sim}{\subset} (t'_1 \;\&\; t'_2)$

In order to prove this part, we need the following lemmas:

lemma1: $(t \;\&\; t') -^{s}\rightarrow_{*} \iff (t -^{s1}\rightarrow_{*} p) \wedge (t' -^{s2}\rightarrow_{*} p') \wedge (s \in (s \otimes s2)) \wedge (q = p \;\&\; p')$

$\wedge (\forall i, s1[i] \neq s2[i] \wedge s[i] = s1[i].s2[i])$.($\otimes$ denotes the shuffle product).

Proof: by induction on sequences's length.

base: $s = \varepsilon$; obvious

hypothesis: let the assertion be true for any s s.t. $|s| \leqslant n$

step: $(t \;\&\; t') -^{sa}\rightarrow_{*} q \iff (t \;\&\; t') -^{s}\rightarrow_{*} (p1 \;\&\; p2) -^{a}\rightarrow q$

now by operational semantics of operator &, three cases are possible:

$*p1 -^{a}\rightarrow p'1 \wedge p2 -^{\Phi}\rightarrow p2$,

$*p1 -^{\Phi}\rightarrow p1 \wedge p2 -^{a}\rightarrow p'1$,

$*p1 -^{a1}\rightarrow p'1 \wedge p2 -^{a2}\rightarrow p'2$, $a1 \neq a2$, $q = (p'1 \;\&\; p'2)$ ∎

corollary : $(t \,\&\, t')=^{s}\Rightarrow p \iff (t=^{s1}\Rightarrow p1) \wedge (t'=^{s2}\Rightarrow p2) \wedge (s\in s1\otimes s2) \wedge (p=p1 \,\&\, p2)$ ∎

lemma2: $a \sqsubset a' \wedge b \sqsubset b' \Rightarrow a.b \sqsubset a'.b'$.

Proof: immediate; left to the reader ∎

we now engage in the proof of part C).

1) $a \in I(t_1 \,\&\, t_2) \Rightarrow$

- $a \in I(t_1)$ and then $\exists\, b \in I(t'_1) : a \sqsubset b$
- $a \in I(t_2)$ and then $\exists\, b \in I(t'_2) : a \sqsubset b$
- $a=a1.a2$, where $a1 \in I(t_1), a2 \in I(t_2)$ then $\exists\, b=b1.b2, b1 \in I(t'_1), b2 \in I(t'_2)$ and $a \sqsubset b$

2) $^{*}(t_1 \,\&\, t_2)=^{s}\Rightarrow \iff (t_1=^{s1}\Rightarrow \wedge\, t_2=^{s2}\Rightarrow \wedge\, s \in (s1\otimes s2)) \Rightarrow (t'_1=^{s1}\Rightarrow \wedge\, t'_2=^{s2}\Rightarrow) \iff (t'_1 \,\&\, t'_2)=^{s}\Rightarrow$

$^{*}(t_1 \,\&\, t_2)\uparrow s \iff (t_1\uparrow s \vee t_2\uparrow s) \Rightarrow (t'_1\uparrow s \vee t'_2\uparrow s) \iff (t'_1 \,\&\, t'_2)\uparrow s$

$^{*}((t'_1 \,\&\, t'_2)=^{s}\Rightarrow p' \Rightarrow p'=^{a}\Rightarrow) \iff (t'_1=^{s1}\Rightarrow p'_1 \Rightarrow p'_1=^{a1}\Rightarrow) \wedge (t'_2=^{s2}\Rightarrow p'_2 \Rightarrow p'_2=^{a2}\Rightarrow) \wedge (s\in(s1\otimes s2), a=a1.a2)$

now, since $[s \mid (t'_1=^{s}\Rightarrow \wedge t'_2=^{s}\Rightarrow) \wedge (t_1=^{s}\Rightarrow \vee\, t_2=^{s}\Rightarrow)] \subseteq [s \mid t_1=^{s}\Rightarrow \wedge t_2=^{s}\Rightarrow]$

then obviously $((t_1 \,\&\, t_2)=^{s}\Rightarrow q \Rightarrow q=^{a}\Rightarrow) \vee ((t_1 \,\&\, t_2)\neq^{sa}\Rightarrow)$

D) $t_1[\varphi] \underset{\sim}{\subset} t'_1[\varphi] \quad \forall\, \varphi \in \mathcal{R}$

Here, we also need some preliminary results:

lemma1: $\forall\, t \in C.F.A.\ t[\varphi]=^{s}\Rightarrow t' \iff t'=t1[\varphi] : t=^{s'}\Rightarrow t1 \wedge s=\varphi(s')$, where $\varphi(s')=(\varphi(s'[i]))_{1\leq i\leq j} \wedge \varphi(s'[i])\neq\Phi, \forall\, i \wedge j \leq |s'|$ ∎

Proof by induction on the size of s.

base: $s=\varepsilon$ obvious.

hypothesis: let's suppose the result be valid for s s.t. $|s|\leq n$

step $t[\varphi]=^{sa}\Rightarrow t' \wedge |s|=n$; that is $t[\varphi]=^{s}\Rightarrow t1[\varphi]=^{a}\Rightarrow t' \wedge |s|\leq n$

by hypothesis $t=^{s'}\Rightarrow t1 \wedge s=\varphi(s')$

by definition of $=^{a}\Rightarrow$, $\exists b$: $t1=^{b}\Rightarrow t'' \wedge a=\varphi(b)\neq\Phi \wedge t'=t''[\varphi]$ ie

$sa=\varphi(s'b) \wedge t=^{s'b}\Rightarrow t'' \wedge t'=t''[\varphi]$ ∎

lemma2: $a \sqsubset b \Rightarrow \varphi(a) \sqsubset \varphi(b)$ ∎

Proof immediate; left to the reader ∎

we now examine proof of part D).

1) $a \in I(t_1[\varphi]) \iff (a=\varphi(b)\neq\Phi \wedge b \in I(t_1)) \Rightarrow (\exists\, b' \in I(t'_1) : b \sqsubset b') \Rightarrow \varphi(b) \sqsubset \varphi(b')$ and thus $\varphi(b')\neq\Phi \Rightarrow \exists\, a' \in I(t'_1[\varphi]) : a \sqsubset a'$.

2)*$(t_1[\varphi]=^{s}\Rightarrow)$ $\Leftrightarrow(t_1=^{s'}\Rightarrow \wedge\ s=\varphi(s'))\Rightarrow(t'_1=^{s'}\Rightarrow)\Leftrightarrow(t'_1[\varphi]=^{s}\Rightarrow)$

* $(t_1[\varphi]\uparrow s)\Leftrightarrow(t_1\uparrow s' \wedge s=\varphi(s'))\Rightarrow(t'_1\uparrow s')\Leftrightarrow(t'_1[\varphi]\uparrow s)$

*$(t'_1[\varphi]=^{s}\Rightarrow p[\varphi] \Rightarrow p[\varphi]=^{a}\Rightarrow)\Leftrightarrow(t'_1=^{s'}\Rightarrow p' \Rightarrow p'=^{b}\Rightarrow \wedge\ a=\varphi(b)\neq\Phi)$

$\Rightarrow(t_1[\varphi]=^{s}\Rightarrow q[\varphi] \Rightarrow q[\varphi]=^{a}\Rightarrow) \vee (t_1[\varphi]\neq^{sa}\Rightarrow)$ ∎

E) $a\rightarrow t_1 \underset{\sim}{\subset} a\rightarrow t'_1$

we use the alternative characterisation of $\underset{\sim}{\subset}$, together with the following

lemma: $(t\&t')[\Psi^1]\simeq t[\Psi^1]\ \&\ t'[\Psi^1]$ ∎

whose proof is immediate, since $\Psi^1(a)=\Phi \Leftrightarrow a=\Phi$.

now as a corollary,

$a\rightarrow t \simeq_n a:(a_1:(\ldots a_{n-1}:(t[\Psi^{n-1}]\&t[\Psi^{n-2}])\ldots)\&t[\Psi^2])\&t[\Psi^1])$ where a_i represents action a at level i ie $\Psi^i(a)$.

The reader can convince himself that the result follows using parts A),C),and D) of the present proof and definition of $\underset{\sim}{\subset}$ in terms of $\underset{\sim}{\subset}_n$ ∎

Proposition4 section 4)

d1) $X+\tau Y=\tau(X+Y)+\tau Y$

an instance of A6),when Z=nil, using A3)∎

d2) $a\tau X=aX$

$aX=aX+aX$

$=a(\tau X+\tau X)=a\tau X$ ∎

d3) $aX+a(Y+Z)=aX+a(X+Y)+a(X+Z)+a(X+Y+Z)$

$aX+a(Y+Z)=a(\tau X+\tau(Y+Z))=a(\tau(Y+Z+\tau X)+\tau(Y+Z))$

$=a(\tau(\tau(X+Y)+\tau X+Z)+\tau(Y+Z))=a(\tau(\tau(X+Y)+\tau(X+Z)+\tau X)+\tau(Y+Z))$

$=a(\tau(X+Y)+\tau(X+Z)+\tau X+\tau(Y+Z))=a(\tau(X+Y)+\tau(X+Z)+\tau X+\tau(X+Y+Z))$

$=aX+a(X+Y)+a(X+Z)+a(X+Y+Z)$ ∎

d4) $X+\tau X=\tau X$

an instance of A6) ,using A4) twice. ∎

d5) $X+\tau(X+Y)=\tau(X+Y)$

$X+\tau(X+Y)=\tau(X+X+Y)+\tau(X+Y)=\tau(X+Y)$ ∎

d6) $aX+aY=aX+aY+a(X+Y)$

$aX+aY=a(\tau X+\tau Y)=a(X+\tau X+Y+\tau Y)=a(\tau(X+Y)+\tau X+\tau Y)=a(X+Y)+aX+aY$ ∎

II) This section is entirely devoted to the proof of theorem2 and theorem3 ie to the soundness and completeness of system S_0.

I) Soundness

We first prove the following lemma which will be usefull later:

lemma $\forall$ t1,t2 $\in$ C.F.A. s.t.

1) $I(t1) \leqslant I(t2)$

2) $\forall a, (t1 =^{a} \Rightarrow t') \Rightarrow (t2 =^{a} \Rightarrow t')$

then $t1 \underset{\sim}{\subset} t2$.

Proof

1) of course $I(t1) \leqslant I(t2)$

2) * $(t1 =^{s} \Rightarrow) \Leftrightarrow (t1 =^{a} \Rightarrow t' =^{s'} \Rightarrow)$ where $s=as'$

$\Rightarrow (t2 =^{a} \Rightarrow t' =^{s'} \Rightarrow) \Leftrightarrow (t2 =^{s} \Rightarrow)$

* $(t1 \uparrow s) \Leftrightarrow (t' \uparrow s') \Rightarrow t2 \uparrow s$ where $s=as'$

* $(t2 =^{s} \Rightarrow t'2 \Rightarrow t'2 =^{a} \Rightarrow) \Leftrightarrow (t2 =^{b} \Rightarrow t'' =^{s'} \Rightarrow t'2 \Rightarrow t'2 =^{a} \Rightarrow) \Rightarrow$

$((t1 =^{b} \Rightarrow t' =^{s'} \Rightarrow t'1 \Rightarrow t'1 =^{a} \Rightarrow)$ where $s=bs'$ or $(t1 \neq^{sa} \Rightarrow))$ ■

Soundness of axioms A_0-A_4 , A_7-A_{11} and A_{13}-A_{14} comes from the above lemma ,together with definition of relations $=^{a}\Rightarrow$, and operational semantics of C.F.A. operators. Let's show A_{12} ,A_5 and A_6 sound .

A_{12} is sound:

a) $X+\Omega \underset{\sim}{\subset} \tau X+\Omega$ using proposition2 of section 3) chapter III since X and τX accept the same language,and $X \underset{\sim}{\subset} \tau X$.

b) $\tau X+\Omega \underset{\sim}{\subset} X+\Omega$

1) $I(\tau X+\Omega)=I(X+\Omega)=\{\tau\}$.

2) * obiously, $(X+\Omega =^{s} \Rightarrow) \Leftrightarrow (\tau X+\Omega =^{s} \Rightarrow)$

* $\forall s \ (\tau X+\Omega)\uparrow s \Leftrightarrow (X+\Omega)\uparrow s$

* $\forall s, (X+\Omega =^{s} \Rightarrow Y) \Leftrightarrow (\tau X+\Omega =^{s} \Rightarrow Y)$ ■

A_5 is sound:

a) $aX+aY \underset{\sim}{\subset} a(\tau X+\tau Y)$ from the above lemma.

b) $a(\tau X+\tau Y) \underset{\sim}{\subset} aX+aY$ since

1) $I(a(\tau X+\tau Y))=I(aX+aY)=\{a\}$.

2) $(a(\tau X+\tau Y) =^{s} \Rightarrow t) \Leftrightarrow (s=as' \wedge (\tau X =^{s'} \Rightarrow t \ \vee \ \tau Y =^{s'} \Rightarrow t)) \Leftrightarrow$

$((aX+aY) =^{s} \Rightarrow t)$ if $s \neq \varepsilon \wedge s \neq a$ otherwise use proposition2,3),III. ■

A_6 is sound:

This proof is quite similar to the above one;it is left to the reader.■

We now turn to the soundness of the rules contained in our system:

soundness of R_0 comes from proposition1,section 1 of chapter IV,

soundness of R_1 and R_2 comes from proposition1,section 3 of chapter III.

soundness of R_3 is obvious since $\simeq$ is generated by $\underset{\sim}{\subset}$.

soundness of R_4 comes from proposition2,section 3 of chapter III.

soundness of R_5 and R_6 comes from propositions1,2 and 3 in section 1 of chapter IV ■

II)Completeness

The proof of completeness relies heavily on the notion of normal-forms(nf)

A)Normal-form

definition1 a set $E=\{X_i : i\in I\}$ of finite subsets is said to be convex-primitive iff it is not empty and if $X\in E \Rightarrow |X / \cap\{X_i : i\in I\}| \leqslant 1$ ■

Example: $\{(a,b),(a,c),(a,d),(a)\}$ and $\{(a),(\tau),(b),\Phi\}$ are convex-primitive; $\{(a,b,c),(a,c)\}$ and $\{(a,b,c),(a,b),(a,c)\}$ are not.

definition2 a C.F.A. term t is said to be under normal-form iff

1) $t=\Sigma_{a\in L} a:t_a$ where $L\subset A_+$ and L finite, and each t_a is under nf
in this case t is said to be under a-nf

2) $t=\Sigma_{L\in\mathcal{L}}\tau\,\Sigma_{a\in L} at_a$ where $\mathcal{L}$ is convex-primitive, and $\mathcal{L}\subseteq A$ and $\forall a$, t_a is under nf; and $t\uparrow \Leftrightarrow \exists L: \tau\in L$.
in this case t is said to be under τ-nf ■

Remark i) nil is under a-nf with $L=\Phi$, and Ω is under τ-nf, with $\mathcal{L}=\{(\tau)\}$.

ii) We do not impose,for τ-nf that the sum be bounded;however an infinite sum denotes a divergent term.

lemma1 if t is under τ-nf then $t=(\Sigma_{b\in L/m}\tau\Sigma_{a\in m\cup(b)} at_a)[+\tau nil]$ where $L\subseteq A$, $m\subseteq L$, $\tau\in L\Leftrightarrow t\uparrow$, $t\neq\tau nil \Leftrightarrow L\neq\Phi$. we assume $t=\tau\Sigma_{a\in m} at_a$ when $L=m\neq\Phi$ (notation [t'] means that t' is an optional summand).

proof: let $L=\cup\{L\in\mathcal{L}\}$, $m=\cap\{L\in\mathcal{L}\}$ and use axioms A_5, A_{14} of system S_0 and derived axiom d_3 ■.

lemma2: $\forall$ t,t' $\in$ C.F.A.,t and t' under nf $\Rightarrow\exists$ n under nf s.t. $n=\tau t+\tau t'$.

proof: by induction on the size of sequences t and t' accept.

Four cases are possible depending on the structure of t and t'.

i) t and t' are under τ-nf then

$\tau t+\tau t'=t+t'$ (from A_{13})

and $t+t'=(\Sigma_{b\in L/m}\tau\Sigma_{a\in m\cup\{b\}}at_a+\Sigma_{b\in L'/m'}\tau\Sigma_{a\in m'\cup\{b\}}at'_a)[+\tau nil]$

using the above lemma1 and axiom A_4 to reduce "τnil" instances to only one whenever they actually exist. Now:

$t+t'=(\Sigma_{b\in L''/m''}\tau\Sigma_{a\in m''\cup\{b\}}at''_a)[+\tau nil]$ where (using A_{14} and d_3)

$L''=L\cup L'$, $m''=m\cap m'$ and

$a\in(L\cap L')\cup(m\cap m')\Rightarrow t''_a=\tau t_a+\tau t'_a$ which by induction,can be put under nf.

$a\in L\Rightarrow t''_a=t_a$

$a\in L'\Rightarrow t''_a=t'_a$

ii) t is under τ-nf and t' is under a-nf then

$\tau t+\tau t'=t+\tau t'=(\Sigma_{b\in L/m}\tau\Sigma_{a\in m\cup\{b\}}at_a)[+\tau nil]+\tau\Sigma_{a\in m'}at'_a$

$=(\Sigma_{b\in L''/m''}\tau\Sigma_{a\in m''\cup\{b\}}at''_a)[+\tau nil]$ where (using A_{14} and d_3)

$L''=L\cup m'$, $m''=m\cap m'$ and

$a\in(L\cap m')\cup(m\cap m')\Rightarrow t''_a=\tau t_a+\tau t'_a$ (from A_5),which by induction,can be put under nf.

$a\in L\Rightarrow t''_a=t_a$

$a\in L'\Rightarrow t''_a=t'_a$

iii) t is under a-nf and t' is under τ-nf then

$\tau t+\tau t'=\tau t+t'$ an instance of point ii).

iv) t and t' are under a-nf then

$\tau t+\tau t'=(\tau\Sigma_{a\in m}at_a+\tau\Sigma_{a\in m'}at'_a)=(\tau\Sigma_{a\in m''}at''_a)$ where

$m''=m\cup m'$ and

$a\in(m\cap m')\Rightarrow t''_a=\tau t_a+\tau t$ which by induction,can be put under nf.

$a\in m\Rightarrow t''_a=t_a$

$a\in m'\Rightarrow t''_a=t'_a$ ■

<u>lemma3</u>: $\forall$ t,t'$\in$C.F.A. ,t and t' under nf $\Rightarrow\exists$ n under nf s.t. $n=t+t'$.

<u>proof</u> : by induction on the sizes of sequences t and t' accept.

There are four cases depending upon the structures of t and t'.

ii) $t=\Sigma_{L\in\mathcal{L}}\tau\Sigma_{a\in L}at_a$ and $t'=\Sigma_{a\in L'}at'_a$ then

$t+t'=\tau(\Sigma_{a\in L_1}at_a+t')+t''$ (using A_6,A_4).

where $t''=\Sigma_{L\in\mathcal{L}\setminus L_1}\tau\Sigma_{a\in L}at_a$

by induction $t+t'=\tau n_1+n_2=n$ (using the above lemma2).

iii)t is under a-nf and t' is τ-nf similar to case ii) above .

iv) $t=\Sigma_{a\in L}at_a$,$t'=\Sigma_{a\in L'}at'_a$ then

$t+t'=\Sigma_{a\in L\setminus(L\cap L')}at_a+\Sigma_{a\in L'\setminus(L\cap L')}at'_a+\Sigma_{a\in(L\cap L')}a(\tau t_a+\tau t'_a)$

$=n'''+n'+n''=n$ (by induction and lemma2) ■

corollary : t_i under nf, $\forall i\in I\subset\mathbb{N}$, then

a) I is finite $\Rightarrow \Sigma_{i\in I}t_i=n$ under nf

b) $\Sigma_{i\in I}\tau t_i=n'$ under nf

proof : by induction on the size of I using lemma2 and lemma3 ■

lemma4: $\forall t,t'\in C.F.A.$,s.t. t,t' under nf then $t\underset{\sim}{\subset}t'\Rightarrow$

1)$(I(t)\subset I(t'))\vee(I(t')=\{\tau\})$

2) $\forall a\in\mathcal{A}_+$, $t=^a\Rightarrow t_a\Rightarrow(t'=^a\Rightarrow t'_a\ \wedge\ t_a\underset{\sim}{\subset}t'_a)$

proof : 1) is immediate since t under nf $\Rightarrow \tau\in I(t)\Leftrightarrow I(t)=\{\tau\}$ and $\forall X,Y\subseteq\mathcal{A}$ $\tau\notin X\wedge\tau\notin Y\Rightarrow X\underset{\sim}{\subset}Y\Leftrightarrow X\subseteq Y$.

2) notice first that by definition of normal forms,$\forall t$ under nf,$\forall a\in\mathcal{A}_+$, $t-\tau^*\to_*-^a\to t_a$ then t_a is unique;we then assume t'_a be s.t. $t'-\tau^*\to_*-^a\to t'_a$ t'_a is unique,and there may be various reasons,for t_a not to be less than t'_a

a)$I(t_a)<I(t'_a)$ that is $\exists b\in I(t_a)$ s.t. $\forall b'\in I(t'_a)$,$(b\sqsubset b')$ that is t'_a is under a-nf .Now depending on b,two situations may envisaged:

i) $b\in\mathcal{A}_+$ ie $t=^{ab}\Rightarrow\wedge\ t'\not=^{ab}\Rightarrow$ what is impossible.

ii) $b=\tau\wedge\tau\notin I(t'_a)$ then by definition of $\underset{\sim}{\subset}$ and normal forms, t↑a . this case is also impossible since it would imply $\exists t''_a$ s.t. $t=^a\Rightarrow t''_a\wedge I_+(t'_a)\subseteq I_+(t''_a)$ ($I_+(t)$ denotes $I(t)\setminus\{\tau\}$).

b)$t_a=^s\Rightarrow\wedge\ t'_a\not=^s\Rightarrow$ ie $t=^s\Rightarrow\wedge\ t'\not=^s\Rightarrow$ impossible.

c)$t_a\uparrow s\wedge t'_a\uparrow s$ ie $t\uparrow as\wedge t'\uparrow as$ impossible too.

d)$(t'_a=^s\Rightarrow p\Rightarrow p=^c\Rightarrow)\wedge(t_a=^{sc}\Rightarrow\wedge\ \exists q: t_a=^s\Rightarrow q\wedge q\not=^c\Rightarrow)$ ie: $(t=^{asc}\Rightarrow\wedge\ q\not=^c\Rightarrow)$ what is impossible.

So all case that would imply $t_a\not\underset{\sim}{\subset}t'_a$ are impossible.The result follows ■

lemma5 $\forall t\in C.F.A.,\forall n\in\mathbb{N},\exists t'$ under nf s.t. $t[\triangleright n]=t'$.

proof: first we assume that t[▷n] contains only nil,guard and sum operators.This can be achieved by first applying axiom A_{11} in order to sufficiently unwind terms containing g.p.i. operators,then using axioms A_{7-10} to eliminate renamings and parallel composition operators.Now , using axioms A_{5-6},t[>n] can be put under the two following forms:

i) $t'' = \Sigma_{a\in L} at_a$ where $L=I(t)$ and thus finite

ii) $t''' = \Sigma_{L\in\mathcal{L}} \tau\Sigma_{a\in L} at_a$ and $t''\downarrow \Rightarrow \exists L\in\mathcal{L}$ s.t. $\tau\in L$, relying on axiom A_{12}.

Therefore, by induction on the size of sequences $s\in\mathcal{L}_+$ a term can accept, (nil, τnil and Ω have normal forms) each t_a can be put under nf say t'_a. The result then follows by application of lemmas2-3 and their corollary ∎

lemma6: $\forall\, t \in$ C.F.A., $\forall\, n\in\mathbb{N}, t\simeq_n t[\triangleright n]$.

proof: left to the reader (notice however that validity of part $t[\triangleright n]\subseteq_{\approx n} t$ comes from soundness of rule R_4) ∎

lemma7: let a C.F.A. term t be s.t. $t=\Sigma_{L\in\mathcal{L}} \tau\Sigma_{a\in L} at_a$ then $\forall\, L'\subseteq UL$, if L' is finite then $t+\Sigma_{a\in L'} at_a = t$.

proof: from derived rule d_6, $t=t+\tau\Sigma_{a\in L'} at_a = t'$. By axiom A_{13}, $t'=\tau t'$ so

$t+\Sigma_{a\in L'} at_a = t'+\Sigma_{a\in L'} at_a = \tau(t+\tau\Sigma_{a\in L'} at_a + \Sigma_{a\in L'} at_a) = t$ from d_4 and A_{13} ∎

We now use these results to prove the completeness of system S_0.

B) COMPLETENESS

1) theorem2

Let t and t' be two C.F.A. terms s.t. $t\subseteq_{\approx n} t'$ the purpose of this section is to show that we can infer $t\subseteq_n t'$.

From lemma6, $t\subseteq_{\approx n} t' \Leftrightarrow t[\triangleright n]\subseteq_{\approx n} t'[\triangleright n]$. Relying on lemma5 we assume $t[\triangleright n]$ and $t'[\triangleright n]$ under normal forms say n_1 and n_2. We now proceed by induction on the size of sequences the terms may accept. Three cases are possible depending on what kind of nf n_1 and n_2 are.

i) $n_1 = \Sigma_{L\in\mathcal{L}} \tau\Sigma_{a\in L} at_a$, $n_2 = \Sigma_{L\in\mathcal{L}'} \tau\Sigma_{a\in L} at'_a$ from lemma4, $\forall a, t_a\subseteq_{\approx n} t$ so, by induction we consider $\forall a$ $t_a\subseteq_n t'_a$; and by definition of $\subseteq_{\approx n}$ obviously $\cap\{L\in\mathcal{L}'\}\cap\cup\{L\in\mathcal{L}\}\subseteq\cap\{L\in\mathcal{L}\}$ so using rule R_4 and axiom A_2, we immediately infer $n_1\subseteq_n n_2$ ie $t\subseteq_n t'$.

ii) $n_1 = \Sigma_{a\in L} at_a$, $n_2 = \Sigma_{L\in\mathcal{L}'} \tau\Sigma_{a\in L} at'_a$. The proof is similar to the above one, noticing that in this case $L\subset\cup\{L' : L'\in\mathcal{L}'\}$.

iii) $n_1 = \Sigma_{a\in L} at_a$, $n_2 = \Sigma_{a\in L'} at'_a$. We again apply the same rule and axioms, since here obviously $L\subset L'$. ∎

2) theorem3

Let t and t' be two C.F.A. terms s.t. $t\subseteq t'$; we prove here, that using system S_0 and rules R_0-R_6 one can infer $t\sqsubseteq t'$. Now this is immediate since $t\subseteq t' \Leftrightarrow \forall n, t\subseteq_{\approx n} t' \Leftrightarrow \forall n, t\subseteq_n t'$, using the above result. So using rule R_6 one obviously infer $t\sqsubseteq t'$ ∎

PI 265 *Automatice design of systolic chips*
Patrice Quinton et Pierrick Gachet – 21 pages ; Septembre 85.

PI 266 *Expression of syntactic and semantic features in logic–based grammars*
Patrick Saint–Dizier – 19 pages ; Septembre 85.

PI 267 Nestor : noyau d'exécutif pour le suivi en temps–réel des applications orientées robotique
Philippe Belmans, Jean–Jacques Borrelly, Maryline Silly, Daniel Simon – 16 pages ; Septembre 85.

PI 268 Etude des méthodes de surveillance du comportement vibratoire des structures en mer : positionnement optimal des capteurs et détection d'anomalies
Michèle Basseville – 70 pages ; Octobre 85.

PI 269 Modélisation et gestion d'univers 3D (une première approche à partir du logiciel PADL–2
Gérard Hégron, Patrick Rives – 60 pages ; Octobre 85.

PI 270 LGRC : Un langage pour la synthèse d'images par lancer de rayons
Bruno Arnaldi, Kadi Bouatouch, Thierry Priol – 30 pages ; Novembre 85.

PI 271 Towards specification and proof of asynchronous systems
Boubakar Gamatié – 30 pages ; Novembre 85.

Boubakar GAMATIÉ

TOWARDS SPECIFICATION AND PROOF OF ASYNCHRONOUS SYSTEMS

Publication interne
n° 271

Novembre 1985