TOWARDS SPECIFICATION AND PROOF
OF ASYNCHRONOUS SYSTEMS

GAMATIE B.
IRISA/INRIA Campus de beaulieu
35042 RENNES CEDEX (FCE)

Abstract
A communication based language is presented. It allows to specify asynchronous systems of processes. Together with this language, we define an equivalence relation which is a congruence, merging both notions of complexity of accepted languages and of nondeterminism. This equivalence is generated by a preorder relation which allows to provide the processes set with a fully defined (and convergent) minimal element. It is also characterized by an axiom set that we prove to be sound and complete.

Key-words: Asynchronous systems, nondeterminism, process interface, equivalence and congruence of processes, least process.

I-INTRODUCTION

Recently, several papers have been published, aiming in defining and formalizing the concept of process: [ARN82, AUS84, DAR83, HOA81, HEN85, HEN83, KAH77, MIL80...]. 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 intuitively be seen as an abstract machine which can perform actions compounded of basic operations concurrently with other processes. More precisely we consider that the only actions a process can perform are communication ones. However the communicated items may be either data values or control signals, and they may be addressed either to another process (the communication is then said to be external) or to some subcomponent of the process (in this case the communication is said to be internal). So, considering that these (communication) actions can be formally represented by relations over the set of processes (two processes t and t' are related by such a relation if t represents the state of the process before the action while t' represents that after the action), we argue that a process is an object whose (operational) semantics can be entirely specified using exclusively communication actions and relations parametrized by these actions. Moreover since operational semantics specifies all possible operations an object can perform, together with all the state changes that result, the
previous point means that process state changes result exclusively from communication action performance.

Now considering a set of processes, it is in general, said to be asynchronous if it is "insensible" to time considerations, in contrast with synchronous ones where time dependencies are a crucial question. And relating these notions to processes semantics usually amounts to neglect all internal communication occurrences while specifying process behaviour. For instance, in models for C.C.S. based languages, this leads to set the equivalence of \( t \) and \( tt \) where \( t \) represents internal actions, \( t \) a process and \( tt \) a process which first performs an internal action and then behaves like \( t \). However, this equivalence is not valid in all contexts, particularly in nondeterministic ones. As for us, we define an asynchronous system as a set of processes where the inactivity of any one among them doesn't induce the inactivity of all the system in contrast with synchronous ones where the "death" of any component induces the "death" of the whole system. And we relate this notion to process states by allowing an asynchronous process to remain idle (in fact to perform a null action which is not the internal one) during a finite amount of time without changing its state. Formally, this implies that our composition operation of actions will admit as neutral element not the internal action but the null one.

Relying upon these ideas, we define structures for expressing asynchronous parallel algorithms (in fact these structures allowed us to define certain class of both synchronous and asynchronous applications; see [GAM84]). In our approach, as stated earlier, a process can change its state exclusively by performing communication actions so as its operational semantics could be always definable using communication relations exclusively. As a consequence, our processes are objects which are fully defined from an operational point of view; so general form of recursivity which allows to specify undefined objects (hint for instance that specified by expression \( \text{rec}x.x \)) is not be considered in our framework. As a counterpart, we introduce a "guarded parallel iteration" operation which allows to define infinite processes using communication actions (which may be either internal or external).

This rather original position quickly leads us to problems of finding appropriate relations between systems of processes that is finding an adequate equivalence relation over the set of processes. In fact this important question has been emphasized by all the proposals of models for processes; moreover, the proposed equivalences have been in general axiomatized leading to processes calculi (hint for instance the well known C.C.S. calculus [MIL80]). Unfortunately, these results do not fit our approach.