1. Introduction

Consider a language $P$ for the description of programs and an equivalence relation $\equiv$ for their comparison. If $F$ is the language of formulas of a logic used as a specification language for programs of $P$ and $\models$ a satisfaction relation, subset of $P \times F$, this relation induces on $P$ an equivalence $\equiv$ defined by,

$$\forall f \in F \ (p_1 \equiv f \iff p_2 \equiv f)$$

Obviously, a minimal requirement for $F$ to be an appropriate specification language for $(P, \equiv)$ is that

$$\equiv = \equiv, \ \text{i.e., the two relations agree on } P.$$

This requirement, known as the adequacy of $(F, \models)$ for $(P, \equiv)$ has first been considered in [HM] where a modal language adequate for CCS [Mi1] with observational equivalence is given. It sets the appropriate framework for the choice of equivalence relations on programs by considering both behavioral and property preserving criteria as well. It is important that the equivalence relation on $P$ is a congruence [Mi1].

The definition of this relation by ignoring adequacy can lead to inconsistencies. Consider for instance, the CCS terms $p_1 = \alpha.(b.\text{nil} + x.c.\text{nil})$ and $p_2 = \alpha.\text{nil} + \alpha.c.\text{nil} + \alpha.c.\text{nil}$. They are observationally equivalent (by axiom A6.1 in [HM]). However, for any temporal logic adequate for CCS, having an operator $\diamond$ expressing inevitability ($\diamond f$ means that for any execution path there exists a state satisfying $f$), these terms are not logically equivalent. In fact, if such a logic exists, there is a formula $f$ such that $\text{nil} \not\models f$, $c.\text{nil} \not\models f$, $p_2 \not\models f$ but $b.\text{nil} + \alpha.c.\text{nil} \models f$ as $\text{nil}$, $c.\text{nil}$, and $p_2$ are not observationally equivalent to $b.\text{nil} + \alpha.c.\text{nil}$. Then, we have $p_1 \models \diamond f$ and $p_2 \not\models \diamond f$ whatever the underlying model of time is.

A stronger compatibility requirement between $(P, \equiv)$ and $(F, \models)$ is expressivity [Pn1]. $(F, \models)$ is said to be expressive for $(P, \equiv)$, if it is adequate for it, and there exists a function $\phi : P \rightarrow F$ such that:

$$p \equiv p' \iff p' \models \phi(p),$$

i.e., $\phi(p)$ represents the equivalence class of $p$.

Logics expressive for observational equivalence have been defined in [P1][GS1,2].

To obtain a logic expressive for $(P, \equiv)$, two approaches can be considered.

The first, consists in finding a logic allowing to characterize the congruence class of any term (process) as a formula. Then, if necessary, its language of formulas is restricted to formulas representing unions of congruence classes [P1][GS1,2].

The other approach consists in defining a language of formulas $F$ which contains $P$ by consistently extending the operators of $P$ on unions of congruence classes. Thus, terms of $P$ can be given the status of formulas of $F$. This approach has been applied to obtain a $\mu$-calculus à la Kozen [Ko], called Synchronization Tree Logic (STL) [GS3], which is expressive for the algebra of regular behaviors in [Mi2].

The basic modality in STL is prefixing a formula $f$ by a symbol of the action vocabulary $A$ of the process algebra. If $a \in A$ then $af$ is a formula representing the class of the processes whose initial actions are $a$-actions (and only $a$-actions) and such that after their execution $f$ holds.

The definition of such logics for process algebras with silent actions (like CCS) seems to be a non trivial problem. The authors' experience with CCS [GS1,2] shows that the feasibility of the extension of a given processes algebra into a logic depends on the form of the axioms characterizing the equivalence on processes. This operation becomes difficult in presence of certain non linear axioms; for example, the axiom $\tau t + t = \tau t$ of CCS.

The aim of this work is the definition of a process algebra with silent actions which can be integrated into a logic and is such that:

1. The logic thus obtained is compatible with STL, i.e. the language of formulas and its interpretation are the same with the difference that the action vocabulary contains a distinguished action $\tau$ satisfying specific axioms.

2. The existence of an axiom of the form $\alpha \tau t = \alpha t$ as a minimal requirement.

3. The axioms of the process algebra are chosen so as to make extension simple.

A consequence of requirements 1 and 2 is that for any formula $f$ of this logic $\alpha f = af$. It is easy to verify...
that a sufficient condition for the formulas $\alpha x f$ and $\alpha f$ to represent the same class of processes is an axiom of the form $\alpha (\Sigma_{i \in I} t_i) = \Sigma_{i \in I} \alpha t_i$, where $\alpha$ is an arbitrary action and $\{t_i\}_{i \in I}$ is a set of processes. It is a well-known fact that such an axiom characterizes readiness or failures semantics if $t_1 + t_2$ is interpreted as the internal choice of the processes $t_1$ and $t_2$ [BKBO].

Finally, the choice between readiness and failure semantics has been solved by using the third requirement. The non-linear axiom $\alpha t_1 + (\alpha t_2 + \alpha t_3) = \alpha (t_1 + t_2) + \alpha (t_1 + t_2)$, valid for failure but not for readiness semantics, cannot be consistently extended to formulas (unions of equivalence classes).

A logic, extension of the process algebra defined in this paper, is presented in [GS5].

The paper is organized as follows. In section 2, the algebra of readiness models with silent actions is presented. The properties of terms with silent actions have been chosen so as to make extension into a logical language easy. In section 3, a language for the description of sequential processes and its operational semantics are presented. We recall also some well-known results on complete axiomatization of strong congruence on this language. Sections 3 and 4 deal respectively with readiness semantics of the finitary and recursive terms of this language. A complete axiomatization of readiness equivalence is proposed. Finally, section 5 presents readiness semantics for an algebra of regular communicating processes. Terms of this algebra are either sequential processes or terms obtained by applying to sequential processes the operators of parallel composition, restriction and abstraction.

2. The Algebra of Readiness Models

Let $A_\tau = A \cup \{\tau\}$ be a vocabulary of actions with a distinguished action $\tau$, $\tau \in A$ called internal or silent action. If $X$ is a set, $P(X)$ denotes the set of finite subsets of $X$.

A readiness model on $A_\tau$ is an abstraction of a transition system with action vocabulary $A_\tau$. Readiness models on $A_\tau$ can be represented by deterministic trees labelled on $A$; "deterministic" means that for any node and any action $a \in A$ there exists at most one outgoing edge labelled by $a$. It is obvious that any node of a deterministic tree can be characterized by the unique sequence of actions $s \in A^*$ determined by the path from the root to this node. In the sequel, we identify a node with its associated sequence of actions.

A readiness model on $A_\tau$ is a deterministic tree on $A$ whose nodes are labelled by subsets of $P(A_\tau)$. The label of a node $s$ represents the set of states that can be reached by executing the associated sequence of actions $s \in A^\tau$. Each state is characterized by a set $X \subseteq A_\tau$ of initial actions.

This figure shows a transition system $ST$ and the corresponding readiness model $M$.

Readiness models have been introduced in order to distinguish explicitly internal and external choice, i.e. choice non controllable by the environment of the considered system and choice determined by communication. Internal choice expresses the fact that the execution of the same sequence of $A^*$ may lead to different states. For the readiness model of the figure above, the execution of action $a$ may lead to two states from which the actions $b$ and $c$, $b$ and $d$ respectively, can be executed.

Models of this type have been used in [He] - where the term "acceptance tree" is used - and in [Ol]. We propose in this section a variant of these models with the following main differences. Contrary to acceptance trees, we do not consider open nodes, and the constraint of "saturation" of acceptance sets is relaxed for the above mentioned reasons. Furthermore, we do not consider divergence explicitly as in readiness models of [Ol]. Finally, another difference with the aforementioned models is the use of $\tau$'s in the readiness sets to distinguish models representing a term $t$ without initial $\tau$-actions from the term $tt$. This distinction is necessary for the equivalence induced on the process algebra to be a congruence.