ON GUARANTEED RECURSIVELY DEFINED PROCESSES: LIMIT AND LOGICAL DEFINABILITY OF TESTING*

JOHN MULLINS †

Abstract. We extend some negative results on the separability of classes of concurrent processes. This notion of separability has been defined by Darondeau [Dar86]: two processes will be said separable if they can be separated from each other by mean of another test process from the same class i.e., if one can find a test succeeding for one process while failing on the other one. Interpreting the set of infinite visible traces of a process as an ω-language, we obtain this language theoretic definition: given a class $\mathcal{C}$ of ω-languages, we say that L and $L' \in \mathcal{C}$ are separable if there exists $T \in \mathcal{C}$ such that $L \cap T = \emptyset$ while $L' \cap T \neq \emptyset$.

We concentrate on guaranteed processes, i.e processes under a guarantee constraint. Intuitively, a process satisfies a guarantee requirement when that requirement holds at some future point of every computation. A guarantee requirement is a particular case of fairness requirement. We naturally start with some effective classes of guarantee properties proven to form a hierarchy, by Staiger [Sta86], in the case of a finite alphabet. We first prove their non-separability. We then define classes of processes in such a way that non-separability under testing to be induced on by this language theoretic non-separability result. It finally turns out that some processes, having different visible behaviours, are equivalent by test, in the sense of Hennessy [Hen88]. We close the paper on preliminary results concerning the logical complexity of De Nicola and Hennessy's process equivalences when test criteria are ranging over arithmetical hierarchy and on some remarks concerning the relation between separability and full abstraction.

1. Introduction. In [Dar86], it is defined a separation principle for classes of ω-languages. The motivation was neither topological or recursive theoretic where the use of the well known separation principle has been so fruitful, neither, despite the nature of the methods used, language theoretic, but clearly concurrent system theoretic.

His starting point is based on the principle that in the framework of "observational" semantics of a process algebra, an extensional definition of any process can be seen as an ω-language , the one of its infinite sequences of visible actions. Let $|P|$ denote the set of visible behaviours of P. Two processes P and Q in a given class will be separable if we can produce a test set through another process T in the same class such that $|P| \cap |T| = \emptyset$ iff $|Q| \cap |T| \neq \emptyset$.

The following question then arises: which are the separable classes of processes? We answer the question for classes from a particular hierarchy of guarantee properties being themselves a particular class of fairness properties. A process satisfies a guarantee property if a phenomenon of a certain kind is guaranteed to be observable in finite time in any infinite sequence of its visible actions. For example, total correctness is a guarantee property. The phenomenon guaranteed to occur in that case is that the process will eventually reach a final state: the set of infinite sequences satisfying a guarantee property can be expressed in the form UX^ω, X being the set of actions and U, the set of finite behaviours terminating in a guaranteed state.

* This work has been supported by an NSERC grant (Canadian Government).

† Dept. of Computer Science, U. of Ottawa, Ottawa, Canada K1N 6N5. mullins@csi.uottawa.ca.

In a computational framework, it is natural to demand to finite process behaviours to be "effectively" generated by the computational device. That is why we focus to the class of guarantee properties generated when U ranges over the arithmetical hierarchy. Staiger showed that, when X is finite, we get an hierarchy of "effectively generated" guarantee properties not being bounded by some arithmetical class of ω-languages i.e., properties with arbitrary effective structural complexity and more, each class of this hierarchy is closed under finite union and intersection and under projection [Sta86]. Our first result is to prove that, even in the general case, i.e. even when X is infinite, none of these classes can be separated (thm. 1).

As a second step, we define for each above class, a class of non-deterministic recursive automata whose trace sets form exactly this class (thm. 1)and whose computation set always belong to this class. This result seems to be independent on the finiteness of the set of actions as well as on the branching factor of automata but strongly depends on closure of considered classes under projections.

As next step, we define an interaction mechanism between recursive automata as the standard product "à la CCS". It appears that our classes of automata are closed under that product. Having recasted in our framework the Hennessy's notion of test and test equivalences [Hen88] and taking in account guarantee constraint, we prove the non-separability under testing for each class of automata (thm. 2), using of course, previous results but, above all, the capability to remove second order quantifiers we have to deal with (lemma 3).

We finally compute some preliminary results concerning the logical complexity of testing problem (thm. 3). But before doing that we need to define a recursive numbering relative to each class to manage with quantifiers over processes. We close on a discussion about the relationship between *separability* and *full abstraction*.

2. Definitions and background. Let X, an alphabet, X^* and X^ω, the set of respectively finite words and infinite sequences on alphabet X. The empty word is denoted by ϵ. Let $\alpha \in X^* \cup X^\omega$, throughout the paper $\alpha[i]$ will stands for the prefix of length i of α while $\alpha(i)$, for the i-th letter of α and $|\alpha|$ for the length of α. Subsets of X^* and of X^ω will be called languages and ω-languages respectively. They will be denoted indistinctly using capital Roman letters $(U, V, \ldots)$. Let $u\alpha$ to be the concatenation of $u \in X^*$ and $\alpha \in X^\omega$, it can be extended up to a product UV of a language U and an ω-language V in the obvious way. We will denote uV as a shortcut for $\{u\}V$. In the sequel, we will denote words using lowercase Roman letters $(u, v, \ldots)$ and infinite sequences by lowercase Greek letters $(\alpha, \beta, \ldots)$.

It is natural to see X^ω as a topological space and more precisely as the denumerable product of X by itself with X equipped with the discrete topology (every letter of the alphabet X is an open set).

Using Tychonoff's theorem, it is easily seen that X^ω has the compactness property if and only if X is finite. Indeed, discrete topology has this property only in this case.

The topology above described is known to be homeomorphic to Baire space. An ω-language W is an open set if and only if there is a language U such that $W = \{\alpha : \exists_i \alpha[i] \in U\}$ and is a closed set if and only if there is a language U such that

$W = \{\alpha : \forall_i \alpha[i] \in U\}$. So the property to be open or closed for an ω-language may be seen as a "limit" of some finitary language i.e., as the result of a limit operator from the powerset $\mathbf{2}^{X^*}$ to $\mathbf{2}^{X^\omega}$ expressing exactly $\mathbf{\Sigma_1^0}$ and $\mathbf{\Pi_1^0}$ as "limits" of languages and appearing to be dual from each other: let $U \subseteq X^*$ then define

$$\begin{aligned} \lim_{\mathbf{\Sigma}_1^0} U &\stackrel{\text{def}}{=} \{\alpha \in X^\omega : \exists_i \alpha[i] \in U\} \\ \lim_{\mathbf{\Pi}_1^0} U &\stackrel{\text{def}}{=} \{\alpha \in X^\omega : \forall_i \alpha[i] \in U\} \end{aligned}$$

The generalization of the notions of open and closed sets provides the well-suited framework to study the structural properties of ω-languages. Borel ω-languages are the smallest family of ω-languages closed under denumerable unions and intersections and containing the open sets. In terms of predicates defining these sets, it is the smallest class closed under projection and complementation. We will make use of this alternative and maybe more unusual definition which can be found in [Sta86]: a Borel ω-language is an ω-language of the form

$$W = \left\{\alpha \in X^\omega : \mathbf{Q}_{t_1} \mathbf{Q}_{t_2} \cdots \mathbf{Q}_{t_n} \mathbf{P}\left(t_1, t_2, \ldots t_{n-1}, \alpha[t_n]\right)\right\}$$

where $\mathbf{Q}_{t_i}$ are arbitrary quantifiers and $\mathbf{P}$ is a predicate of $n-1$ integer variables and one variable on X^* corresponding to a relation $M_{\mathbf{P}}$ on $\omega^{n-1} \times X^*$ in the following way:

$$\mathbf{P}\left(t_1, t_2, \ldots t_{n-1}, \alpha[t_n]\right) \text{ iff } \left(t_1, t_2, \ldots t_{n-1}, \alpha[t_n]\right) \in M_{\mathbf{P}}.$$

The Borel hierarchy of ω-languages is defined as follows: $W \in \mathbf{\Sigma_n^0}$ if $\mathbf{Q}_{t_1} \equiv \exists_{t_1}$ and $W \in \mathbf{\Pi_n^0}$ if $\mathbf{Q}_{t_1} \equiv \forall_{t_1}$. It turns out that $\mathbf{\Sigma_0^0} = \mathbf{\Pi_0^0} = \{UX^\omega : U \subseteq X^* \text{ finite}\}$ i.e., the family of clopen sets while Σ_1^0 and $\mathbf{\Pi_1^0}$ correspond to open and closed sets respectively.

Arithmetical ω-languages are obtained in the same way but adding the constraint on the predicate $\mathbf{P}$ to be recursive, i.e., corresponding to a recursive relation $M_{\mathbf{P}}$ on $\omega^{n-1} \times X^*$ as above. The arithmetical hierarchy of ω-languages is defined in the same fashion: $W \in \Sigma_n^0$ if $\mathbf{Q}_{t_1} \equiv \exists_{t_1}$ and $W \in \Pi_n^0$ if $\mathbf{Q}_{t_1} \equiv \forall_{t_1}$. Notice the lightface font which distinguishes for example the class Σ_1^0 from the class $\mathbf{\Sigma_1^0}$ of open sets. In this effective setting, $\mathbf{\Sigma_0^0} = \mathbf{\Pi_0^0} = \Sigma_0^0 = \Pi_0^0$ is the family of recursive sets of X^ω so $\mathbf{\Sigma_1^0}$ is the class of recursively enumerable sets of X^ω.

By induction one easily obtains that

$$(1) \qquad \mathbf{\Sigma_n^0} \cup \mathbf{\Pi_n^0} \text{ (resp. } \Sigma_n^0 \cup \Pi_n^0) \subset \mathbf{\Sigma_{n+1}^0} \cap \mathbf{\Pi_{n+1}^0} \text{ (resp. } \Sigma_{n+1}^0 \cap \Pi_{n+1}^0)$$

$$(2) \qquad U \in \mathbf{\Sigma_n^0} \text{ (resp. } \Sigma_n^0) \text{ iff } \overline{U} \in \mathbf{\Pi_n^0} \text{ (resp. } \in \Pi_n^0)$$

The relation between these two hierarchies in upper levels is stated as follow:

$$\Sigma_n^0 \subset \mathbf{\Sigma_n^0} \text{ if } n > 1$$

We assume the reader to be familiar with the basic results on the arithmetical hierarchy and the Tarski-Kuratowski algorithm to rewrite formulas into normal forms [Rog87]

(chapter 15). In particular, the following context-free conversion rules to contract quantifiers and eliminate bounded quantifiers will be extensively used throughout the paper.

$$\begin{aligned} \forall\forall &\longrightarrow \forall \\ \exists\exists &\longrightarrow \exists \\ \exists_{i\leq j} &\longrightarrow \bigvee_{i=0}^{j} \\ \forall_{i\leq j} &\longrightarrow \bigwedge_{i=0}^{j} \end{aligned}$$

3. Effective classes of guarantee properties. In this section we introduce the hierarchy of guarantee properties that we intend to focus to. Recall that a process satisfies a guarantee property if a phenomenon of a certain kind is guaranteed to be observable in finite time in any infinite sequence of its visible actions. Hence, they correspond to open sets in the Baire topology.

Let $W = \lim_{\mathbf{\Sigma}_1^0} U$ to be such a property. From a computational point of view, we could think to U as the set of finite behaviours of the process, meaning that it should be "effectively" generated by some computational device. Hence, let us define now, effective languages.

We define arithmetical languages as the class of languages W of the form

$$W = \left\{ u \in X^* : \mathbf{Q}_{t_1}\mathbf{Q}_{t_2}\cdots\mathbf{Q}_{t_n}\mathbf{P}\left(t_1, t_2, \ldots t_n, u\right)\right\}$$

with all the rest as above. Let Σ_n^* and Π_n^* denote classes of the arithmetical hierarchy of languages. Here again, we get a similar result to equations 1 and 2. Staiger got many interesting results about effective hierarchies that these limit operators generate [Sta86](cor. 4.3 and 5.5):

$$\lim_{\mathbf{\Sigma}_1^0}\Sigma_1^* \text{ (resp. } \lim_{\mathbf{\Pi}_1^0}\Pi_1^*) = \Sigma_1^0 \text{ (resp. } \Pi_1^0) \tag{3}$$

$$\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^* \text{ (resp. } \lim_{\mathbf{\Pi}_1^0}\Pi_n^*) \subset \Sigma_n^0 \text{ (resp. } \Pi_n^0) \tag{4}$$

$$U \in \lim_{\mathbf{\Sigma}_1^0}\Sigma_n^* \text{ iff } \overline{U} \in \lim_{\mathbf{\Pi}_1^0}\Pi_n^* \tag{5}$$

Staiger showed that, when X is finite, we get an hierarchy of "effectively generated" guarantee properties, not being bounded by some arithmetical class of ω-languages [Sta86] (section 6) i.e., properties with arbitrary effective structural complexity and more, each class of this hierarchy is closed under finite union and intersection and under projection [Sta86] (prop. 4.4). Compactness turns to be the main tool for proving this hierarchy result and we do not know yet if this hierarchy result can be extented to the general case, i.e. for infinite alphabet.

4. A separability result for ω-languages. The next definition has been widely motivated in introduction.

DEFINITION 1. A class $\mathcal{C}$ of ω-languages will be said *separable* if for any pair (U, V) in $\mathcal{C}$ there exists an ω-language $W \in \mathcal{C}$ such that $U \cap W = \emptyset \Leftrightarrow V \cap W \neq \emptyset$.

A first result is immediate: for $n \geq 1, \Sigma_n^0 \cap \Pi_n^0$ and $\lim_{\mathbf{\Sigma_1^0}} \Sigma_n^* \cap \lim_{\mathbf{\Pi_1^0}} \Pi_n^*$ are separable. Indeed, these classes are closed under boolean operators. However, quite surprisingly, there is a majority of non-separable ω-language classes. In particular, it was proved in [Dar86] that ω-language classes obtained from ω-Kleene closure of the language classes from the Chomsky hierarchy, are not separable. Σ_1^0 is one of these classes. The next result extends that using a similar idea to the one used in [Dar86] (theorem in II.4) to prove that Σ_1^1, a class properly containing arithmetical ω-languages, is not separable.

THEOREM 1. *For $n \geq 1, \lim_{\mathbf{\Sigma_1^0}} \Sigma_n^*$ is not separable.*

Proof. Let I a recursive set of integers and $x_0, x_1 \in X$. We define an ω-sequence β such that

$$\beta(i) = \begin{cases} x_0 & \text{if } i \in I \\ x_1 & \text{otherwise.} \end{cases}$$

Clearly $\{\beta\} \in \Pi_1^0$. Indeed, u is a prefix of β iff

$$\forall_{x \in X} \forall_{j \leq |u|} \left(u(j) = x \rightarrow \beta(j) = x\right)$$

so, because the condition $\beta(j) = x$ is, by definition of β, a Π_1^*-predicate. Hence, by the Tarski-Kuratowski algorithm, the predicate *"u is a prefix of β"* is a Π_1^*-predicate.

Let $n \geq 1, u \in X^*$ be any prefix of β and $U = uX^\omega$. Then, $\beta \in U$ and $U \in \lim_{\mathbf{\Sigma_1^0}} \Sigma_n^*$. Define now $V = U \setminus \{\beta\}$. Clearly $\{\beta\} \notin \lim_{\mathbf{\Sigma_1^0}} \Sigma_n^*$ because $\lim_{\mathbf{\Sigma_1^0}} \Sigma_n^*$ contains only infinite ω-languages. By eq. 1 to 5, $V \in \lim_{\mathbf{\Sigma_1^0}} \Sigma_n^*$. If U and V were separable, there would exist W in $\lim_{\mathbf{\Sigma_1^0}} \Sigma_n^*$ such that $U \cap W = \{\beta\}$ and that is impossible: in such case, $\{\beta\}$ would be in $\lim_{\mathbf{\Sigma_1^0}} \Sigma_n^*$ and this ends the proof.

Using equation 3 we obtain immediately:

COROLLARY 1. *Σ_1^0 is not separable.*

So, theorem 1 generalizes this and since from the non-separability of Σ_1^1, one cannot conclude to theorem 1, the above result refines [Dar86] (theorem in II.4).

5. Open questions about separability. As we ever mentioned, we do not know yet if the Staiger's hierarchy result over $\lim_{\mathbf{\Sigma_1^0}} \Sigma_n^*$ classes could be extended to the general case. But even for the restricted case of finite alphabet we point out in this section, some open questions.

To generalize theorem 1 to Σ_n^0 classes using a similar idea, we would need to find an infinite chain of sequences in X^ω such that

$$\forall_n \exists_\beta \{\beta\} \in \lim_{\mathbf{\Pi_1^0}} \Pi_n^* \setminus \Sigma_n^0$$

but unfortunately, even using a remarkable Staiger's result giving an infinite hierarchy of isolated points of ω-languages [Sta86] (corollary 7.2), the best we can conclude is

$$\forall_n \exists_\beta \{\beta\} \in \lim_{\mathbf{\Pi_1^0}} \Pi_n^* \setminus \lim_{\mathbf{\Pi_1^0}} \Pi_{n-1}^*.$$

Combining this with the fact that

$$\lim_{\mathbf{\Pi_1^0}} \Pi_n^* \setminus \Sigma_n^0 \neq \emptyset$$

[Rog87] (theorem 15.X1), we obtain that this hierarchy of singletons is not bounded by any class of the arithmetical hierarchy. Nevertheless, the fact that $\{\beta\}$ could not be in $\Sigma^0_n \cap \Pi^0_n$ does not seem to appear trivially here. So the problem of separability for these classes remains open.

The second and last known result concerning the position of isolated points is a Hilbert and Bernay's result [Rog87] (theorem 15.XII). They constructively define a β such that $\{\beta\} \in \Sigma^0_2 \setminus \bigcup_n \lim_{\mathbf{\Pi^0_1}} \Pi^*_n$. This suggests another important limit operator we did not mention yet:

$$\lim_{\mathbf{\Pi^0_2}} U \stackrel{\text{def}}{=} \{\alpha \in X^\omega : \forall_i \exists_{j>i} \alpha[j] \in U\}$$

It generates exactly $\mathbf{\Pi^0_2}$, the class of recurrence properties (e.g. weak fairness), while U is ranging over 2^{X^*}. The problem of separability for the $\lim_{\mathbf{\Pi^0_2}} \Sigma^*_n$ classes is still open.

Nevertheless, these classes are not interesting from a concurrent theoretic point of view because they are not closed under projection and hence, they cannot be interpreted as traces of some non-deterministic computational device.

More precisely, the finite behaviour of some class of deterministic devices is usually easily described and its infinite behaviour, easily derived as well from its finite one through some limit operator. But the behaviour of a non deterministic device turns out to be the projection of a certain kind of the behaviour of some deterministic device.

Note that projective closure of Σ^0_2 is Σ^1_1. That is why the analytical framework of [Dar86] is the natural general framework but note too that, as we ever said, $\lim_{\mathbf{\Sigma^0_1}} \Sigma^*_n$ classes are closed under projection as well.

6. From languages to traces and computations. We first introduce some classes of non deterministic automata and use the closure of the $\lim_{\mathbf{\Sigma^0_1}} \Sigma^*_n$ under projection to show that the class of accepted ω-languages corresponds to $\lim_{\mathbf{\Sigma^0_1}} \Sigma^*_n$ classes and that computation sets are $\lim_{\mathbf{\Sigma^0_1}} \Sigma^*_n$-sets. It will appear that the extension of our non-separability result to processes, heavily depends on this.

An automaton $\mathcal{A}$ over an alphabet X is a 4-tuple $(Q, Q_0, Q_f, \longrightarrow)$ where Q is a set of states, $Q_0 \subset Q$ is the set of initial states, $Q_f \subset Q$ is the set of final states and $\longrightarrow \subset Q \times X \times Q$ is the transition relation. We will denote $q \xrightarrow{x} q'$ for $(q, x, q') \in \longrightarrow$.

An automaton $\mathcal{A}$ is said n-open provided Q is recursive and $Q_0, \longrightarrow$, r.e. and $Q_f \in \Sigma^*_n$. We denote by $\mathcal{O}_n$ the class of the n-open automata. An automaton $\mathcal{A}$ is said deterministic if Q_0 and for every $x \in X$ and $q \in Q$, $\{q' : q \xrightarrow{x} q'\}$ are singletons. An automaton $\mathcal{A}$ is said finitely, recursively or infinitely branching depending if the set $\{q' : q \xrightarrow{x} q'\}$ is finite, recursive or infinite for every $x \in X$ and $q \in Q$. An automaton is said to be complete if $\{q' : q \xrightarrow{x} q'\} \neq \emptyset$ for every $x \in X$ and $q \in Q$.

Given a word $\alpha \in X^*$ of length n, a sequence on $Q \times X \times Q$ is called a run of $\mathcal{A}$ over α if

$$q_0 \xrightarrow{\alpha(0)} q_1 \xrightarrow{\alpha(1)} \ldots \xrightarrow{\alpha(n-1)} q_n$$

with $q_0 \in Q_0$. It is an accepting run if $q_n \in Q_f$. The set of words $\mathcal{L}(\mathcal{A})$ having an accepting run of $\mathcal{A}$ is called the language accepted by $\mathcal{A}$.

For a sequence $\alpha \in X^\omega$, a run of $\mathcal{A}$ over α is an infinite sequence on $Q \times X$ such that

$$q_0 \xrightarrow{\alpha(0)} q_1 \xrightarrow{\alpha(1)} \dots$$

with $q_0 \in Q_0$. An accepting run for α is then one such that $\exists_i q_i \in Q_f$ i.e., if it goes through Q_f at least once. We extend some notations already used for languages, to computations: reserving lowercase Greek letters χ and ξ to denote computations, $\chi[i]$ stands for the prefix of χ of length i and $\chi(i)$, for the i-th computation step.

We denote $\mathcal{L}_\omega(\mathcal{A})$ the ω-language accepted by $\mathcal{A}$. If $\mathcal{C}$ is a class of automata we denote respectively by $\mathcal{L}(\mathcal{C})$ and $\mathcal{L}_\omega(\mathcal{C})$ the classes of languages and ω-languages accepted by a $\mathcal{C}$-automaton. In the same fashion we denote $\mathcal{C}omp(\mathcal{A})$ and $\mathcal{C}omp_\omega(\mathcal{A})$, $\mathcal{C}omp(\mathcal{C})$ and $\mathcal{C}omp_\omega(\mathcal{C})$, respectively the sets of finite and infinite$\mathcal{A}$-computations and classes of sets of finite and infinite computations of $\mathcal{C}$-automata. To formally establish connection between $\mathcal{L}(\mathcal{O}_n)$ and $\mathcal{L}_\omega(\mathcal{O}_n)$ on one hand and $\mathcal{C}omp(\mathcal{O}_n)$ and $\mathcal{C}omp_\omega(\mathcal{O}_n)$ on the other hand, we will need to define some predicates and, as auxiliary tools, some recursive numberings. A first one to code transitions, a second one, to code finite runs and finally, one to code infinite runs that we will call respectively *t-code*, $*$*-code* and ω*-code*. So, we code the transition $p \xrightarrow{x} q$ on the integer $< p, < x, q >>$, the finite run

$$q_0 \xrightarrow{x_0} q_1 \xrightarrow{x_1} \dots \xrightarrow{x_n} q_n,$$

on the Gödel number encoding the finite chain of t-codes and finally,

$$q_0 \xrightarrow{x_0} q_1 \xrightarrow{x_1} \dots \xrightarrow{x_{i-1}} q_i \xrightarrow{x_i} \dots$$

on the function $f : \omega \to \omega$ defined by $f(i) =< q_i, x_i >$. To access components, we use decoding projections π_1 and π_2, e.g. $\pi_1(< p, < x, q >>>) = p$ (denoting for short, the code of state p) and $\pi_1(\pi_2(< p, < x, q >>>)) = x$. To not overload more than necessary integer and functional codes, we will make use, when it helps to clarity, $< f, g >$ to denote the functional code associated to pairs of computations and $< u, v >$ to denote the integer code associated to pairs of finite ones. Finally, we note that object O is encoded on code c by $O_{<c>}$, e.g. $\chi_{<f>}$ denotes that (infinite) computation χ is encoded on the integer function f. We define:

- Predicate $\mathbf{trans}(f, i, \mathcal{A})$ standing for
  "$\chi_{<f>}(i)$ is a $\mathcal{A}$-transition"
  may be rewritten into:

  (6) $$< \pi_1(f(i)), < \pi_2(f(i)), \pi_1(f(i+1)) >>> \in \longrightarrow_{\mathcal{A}}$$

- Predicate $\mathbf{run}(f, i, \mathcal{A})$ standing for
  "$\chi_{<f>}[i]$ is a $\mathcal{A}$-run prefix"
  may be rewritten into:

  (7) $$\forall_{j \le i} \left(\mathbf{trans}(f, j, \mathcal{A}) \wedge \pi_1(f(0)) \in Q_0^{\mathcal{A}} \right)$$

- Predicate $\mathbf{ack}(f, i, \mathcal{A})$ standing for
  "$\chi_{<f>}[i]$ is an accepting $\mathcal{A}$-run prefix"
  may be rewritten into:

$$\mathbf{run}(f, i, \mathcal{A}) \wedge \pi_1(f(i)) \in Q_F^{\mathcal{A}} \tag{8}$$

We immediately obtain that:

LEMMA 1. *For $n \geq 1$,*

1. $\mathcal{L}_\omega(\mathcal{O}_n) = \lim_{\mathbf{\Sigma}_1^0} \Sigma_n^*$
2. $\mathcal{C}omp_\omega(\mathcal{O}_n) \subseteq \lim_{\mathbf{\Sigma}_1^0} \Sigma_n^*$

Proof.

1. The result is a quite straightforward consequence of the closure of

$$\lim_{\mathbf{\Sigma}_1^0} \Sigma_n^*$$

under projections.
We prove that $\mathcal{L}(\mathcal{O}_n) = \Sigma_n^*$.
Let $U = \mathcal{L}(\mathcal{A})$ and $\mathcal{A} \in \mathcal{O}_n$ then U may be expressed as

$$U = \{u \in X^* : \exists_{v:*-code} \exists_i \forall_{j \leq i} \, (i \leq |v| \wedge \mathbf{ack}(v, i, \mathcal{A}) \wedge \pi_2(v(j)) = u(j))\}$$

which is easily seen to be in Σ_n^*, using Tarski-Kuratowski algorithm, while the reverse inclusion is trivial since we have just to define, for a given $U \in \Sigma_n^*$, an $\mathcal{O}_n$-automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = U$ as

$$\begin{array}{rcl} Q^{\mathcal{A}} & = & X^* \\ Q_0^{\mathcal{A}} & = & \epsilon \\ Q_F^{\mathcal{A}} & = & U \\ u \xrightarrow{x}_{\mathcal{A}} v & \Leftrightarrow & v = ux \end{array}$$

2. Let $\mathcal{A} \in \mathcal{O}_n$ then $\mathcal{C}omp_\omega(\mathcal{A})$ may be expressed as the following $\lim_{\mathbf{\Sigma}_1^0} \Sigma_n^*$-set:

$$\mathcal{C}omp_\omega(\mathcal{A}) = \{\chi_{<f>} : \exists_i \mathbf{ack}(f, i, \mathcal{A})\}.$$

This completes the proof.

Proving the reverse inclusion in 2 is more difficult than proving the reverse inclusion of 1 because we have to reflect the branching structure of the computation set into the transition relation of the automaton. But to cope with definition of $\mathcal{O}_n$-automaton, we have to code the Σ_n^*-complexity of this branching structure into the accepting set. We do not know yet how to do this.

7. Testing processes. Let us first sketch a rudimentary and general *interaction product* $\mathcal{A}_1 \parallel \mathcal{A}_2$ of two $\mathcal{O}_n$-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ by synchronizing transitions on a common action. This complete finitely branching automaton over $X \cup \{\tau\}$ (with $\tau \notin X$ denoting "internal" move) corresponds to the so called *experimental system* in [Hen88]. The idea behind the definition of interaction product is to provide a mechanism to test

behaviours of mechanical devices using a device from the same class. Recalling that the main purpose of this construction is to deduce non-separability by testing from non-separability of $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$, we then go back to the root of testing and point out some milestones to make more precise the relationship between the formal language theoretic notion of language separability defined in section 4 and the separability of recursive processes under testing.

Let $\mathcal{A}_1 = (Q_1, Q_{10}, Q_{1f}, \longrightarrow_1)$ and $\mathcal{A}_2 = (Q_2, Q_{20}, Q_{2f}, \longrightarrow_2)$ then

$$\mathcal{A}_1 \parallel \mathcal{A}_2 = (Q, Q_0, Q_f, \longrightarrow)$$

with $Q_0 = Q_{10} \times Q_{20}$, $Q = Q_1 \times Q_2$, $\longrightarrow$ is the relation satisfying

$$(9) \qquad (q_1, q_2) \xrightarrow{x} (q_1', q_2') \text{ iff } \begin{cases} (q_1 \xrightarrow{x} q_1') \wedge q_2 = q_2' & \text{or} \\ (q_2 \xrightarrow{x} q_2') \wedge q_1 = q_1' & \text{or} \\ \exists_{y \in X}(q_1 \xrightarrow{y} q_1') \wedge (q_2 \xrightarrow{y} q_2') \wedge x = \tau. \end{cases}$$

The definition of Q_f has to reflect the intuition that a run will be an accepting run if and only if each component run is an accepting run of the corresponding component automaton. Of course to achieve it properly, it is not sufficient to use $Q_{1f} \times Q_{2f}$ for guaranteed states because, thinking to Φ_i as the guarantee constraint of $\mathcal{A}_i$(i=1,2), $\exists_i\Phi_1 \wedge \exists_i\Phi_2$ may not be collapsed into $\exists_i(\Phi_1 \wedge \Phi_2)$. An elegant way to achieve it could be to "encode" the guarantee constraint preserved through the synchronization into the transition relation of $\mathcal{A}_1 \parallel \mathcal{A}_2$ in the fashion it is usually done for finite automata over finitary languages. Unfortunately, in that case, the transition relation would inherit the complexity of Q_{1f} and Q_{2f} while it has to be kept into Σ_1^*, by definition. This motivates the following definition: given a word $\alpha \in (X \cup \{\tau\})^*$ of length n, and a run of $\mathcal{A}_1 \parallel \mathcal{A}_2$ over α,

$$(q_0, q_0') \xrightarrow{\alpha(0)} (q_1, q_1') \xrightarrow{\alpha(1)} \ldots \xrightarrow{\alpha(n-1)} (q_n, q_n')$$

then $(q_n, q_n') \in Q_f$ if

$$(10) \qquad (q_n \in Q_{1f} \wedge \exists_{i \leq n} q_i' \in Q_{2f}) \vee (q_n' \in Q_{2f} \wedge \exists_{i \leq n} q_i \in Q_{1f}).$$

We establish a first and quite immediate result concerning interaction product: the class $\mathcal{O}_n$ (over $X \cup \{\tau\}$) is closed under interaction product. This simple result will later appear to be quite useful.

LEMMA 2. *For $n \geq 1$ and $\mathcal{A}_1, \mathcal{A}_2 \in \mathcal{O}_n$*

1. $\mathcal{L}_\omega(\mathcal{A}_1 \parallel \mathcal{A}_2) \in \lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$
2. $\mathcal{C}omp_\omega(\mathcal{A}_1 \parallel \mathcal{A}_2) \in \lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$

Proof. By lemma 1, it is sufficient to prove that $\mathcal{A}_1 \parallel \mathcal{A}_2 \in \mathcal{O}_n$.

By definition of $\longrightarrow_1$ and $\longrightarrow_2$ and by the Tarski-Kuratowski algorithm $\longrightarrow \in \Sigma_1^*$. In the same way, closing the proof, all quantifiers in eq. 10 being bounded, the eq. 10 is a Σ_n^*-predicate.

In Hennessy's terminology [Hen88], an *experiment or test* on the *process* $\mathcal{P}$ by the tester $\mathcal{E}$, specified here by two $\mathcal{O}_n$-automata, is a sequence of possible interactions

between the experimenter and the process. The *test criterion* is here the Σ_n^*-predicate specifying the final states of $\mathcal{E}$, and of course the guarantee constraint on the process is the Σ_n^*-predicate specifying those of $\mathcal{P}$.

To define the notion of test more precisely, let us introduce some auxiliary definitions. A *closed run* is any run of $\mathcal{P} \parallel \mathcal{E}$ whose both components are accepting runs of component automata and having a τ-prefix that is, a prefix of the form

$$(q_0, q_0') \xrightarrow{\tau} (q_1, q_1') \xrightarrow{\tau} \ldots \xrightarrow{\tau} (q_n, q_n').$$

A closed run is *maximal* if it is a run on τ^ω or if there exists no way to extend its τ-prefix toward some run with an accepting τ-prefix. An *experiment or test* on $\mathcal{P}$ by $\mathcal{E}$ is any run of $\mathcal{P} \parallel \mathcal{E}$ is precisely a maximal closed run. A test is *successful* if it has an accepting τ-prefix.

Let us express all this in a more useful way:

- Predicate τ-**prefix**$(f, i, \mathcal{P})$ standing for
  "$\chi_{<f>}[i]$ is a $\mathcal{P}$-run over τ^i"
  may be rewritten into:

  (11) $$\forall_{j \leq i} \pi_2(f(j)) = \tau$$

- Predicate **success**$(f, i, \mathcal{P})$ standing for
  "$\chi_{<f>}[i]$ is a successful test of length i"
  may be rewritten into:

  (12) $$\mathbf{ack}(f, i, \mathcal{P}) \wedge \tau\text{-}\mathbf{prefix}(f, i, \mathcal{P})$$

- Predicate τ-**ext**$(f, g, i, j, \mathcal{P})$ standing for
  "$\chi_{<g>}[j]$ is an accepting $\mathcal{P}$-computation over τ^j, extending $\xi_{<f>}[i]$"
  So it is defined exactly as predicate **success**$(g, j, \mathcal{P})$ with

  (13) $$\pi_1(g(0)) = \pi_1(f(i+1))$$

  substituted to $\pi_1(g(0)) \in Q_0^{\mathcal{P}}$ in **run**$(g, j, \mathcal{P})$ (eq. 7).
- Predicate τ-**max**$(f, i, \mathcal{P})$ standing for
  "there is no accepting silent $\mathcal{P}$-extension to $\chi_{<f>}[i]$"
  may be rewritten into:

  (14) $$\tau\text{-}\mathbf{prefix}(f, i, \mathcal{P}) \wedge \forall_g \forall_j \neg \tau\text{-}\mathbf{ext}(f, g, i, j, \mathcal{P})$$

- Predicate **failure**$(f, i, \mathcal{P})$ standing for
  "$\chi_{<f>}[i]$ is an unsuccessful test of length i"
  may be rewritten into:

  (15) $$\mathbf{ack}(f, i, \mathcal{P}) \wedge \exists_{j<i} \tau\text{-}\mathbf{max}(f, j, \mathcal{P})$$

Now, we tabulate the possible test results on $\mathcal{P}$ by $\mathcal{E}$ using the symbols $\top$ to denote a successful computation and $\bot$ to denote an unsuccessful one. So let $\mathrm{Result}(\mathcal{P},\mathcal{E})$ be defined by

$$\top \in \mathrm{Result}(\mathcal{P},\mathcal{E}) \quad \text{iff} \quad \exists_{<f,g>}\exists_m \mathbf{success}(<f,g>,m,\mathcal{P} \parallel \mathcal{E}) \tag{16}$$

$$\bot \in \mathrm{Result}(\mathcal{P},\mathcal{E}) \quad \text{iff} \quad \exists_{<f,g>}\exists_m \mathbf{failure}(<f,g>,m,\mathcal{P} \parallel \mathcal{E}) \tag{17}$$

The next result will appear to be the cornerstone to prove non-separability under testing for $\mathcal{O}_n$ classes.

LEMMA 3. *For $n \geq 1$, given an $\mathcal{O}_n$-process $\mathcal{P}$ and an $\mathcal{O}_n$-tester $\mathcal{E}$, the set of $\mathcal{E}$-computations as well as the set of $\mathcal{P}$-computations resulting in a successful (resp. unsuccessful) test on $\mathcal{P}$ are $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$-sets (resp. $\lim_{\mathbf{\Sigma}_1^0}\Sigma_{n+1}^*$-sets).*

Proof. We first want to prove that

$$\{\chi_{<f>} : \exists_g \exists_m \mathbf{success}(<f,g>,m,\mathcal{P} \parallel \mathcal{E})\} \tag{18}$$

is a $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$-set. Eq. 18 involves a quantifier ranging over an infinite object. Here we can simplify because this quantifier is easily eliminated ([Rog87], thm. 16-VII): predicate in eq. 18 can be transformed into

$$\exists_i \exists_{<v,m,n>} n \leq \max\{i,|v|\} \wedge m \leq n \wedge \mathbf{success}(<f[i],v>,m,\mathcal{P} \parallel \mathcal{E})$$

where u, v are $*$-codes.

It is easily demonstrated using lemma 2, Tarski-Kuratowski algorithm and finally using the fact that $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$ is closed under finite intersection.

Estimate now the logical complexity of

$$\{\chi_{<f>} : \exists_g \exists_m \mathbf{failure}(<f,g>,m,\mathcal{P} \parallel \mathcal{E})\} \tag{19}$$

The quantifier on infinite objects appearing in predicate of eq. 19 may easily be eliminated as above and turns into:

$$\exists_i \exists_{<v,m,n>} n \leq \max\{i,|v|\} \wedge m \leq n \wedge \mathbf{failure}(<f[i],v>,m,\mathcal{P} \parallel \mathcal{E}) \tag{20}$$

where u, v are $*$-codes.

But there is still another quantifier on infinite object appearing in predicate **failure**: the one appearing in the definition of predicate τ-**max** (eq. 14) by successive substitutions of eq. 15 and eq. 14 in eq. 20. In that case again, it is easily seen ([Rog87], thm. 16-VII) that the right hand side of eq. 14 is equivalent to

$$\forall_w \forall_j (j \leq |w|) \wedge \neg\tau\text{-}\mathbf{ext}(<f[i],v>,w,m,j,\mathcal{P} \parallel \mathcal{E})$$

where v, w are $*$-codes.

As above, using eq. 13, lemma 2 and Tarski-Kuratowski algorithm, we obtain that eq. 19 defines a $\lim_{\mathbf{\Sigma}_1^0}\Delta_{n+1}^*$-set. The final result follows from the fact that

$$\Delta_{n+1}^* \subseteq \Sigma_{n+1}^*$$

and this closes the proof.

8. The separability result for processes. A natural way to define equivalence between processes is of course the following one: given $\mathcal{P}, \mathcal{Q} \in \mathcal{O}_n$, $\mathcal{P}$ and $\mathcal{Q}$ are *equivalent by test* (noted $\mathcal{P} \simeq \mathcal{Q}$) if for every $\mathcal{O}_n$-process $\mathcal{E}$,

$$\text{Result}(\mathcal{P}, \mathcal{E}) = \text{Result}(\mathcal{Q}, \mathcal{E})$$

We are now ready to state the exact notion of separability for processes:

DEFINITION 2. A class $\mathcal{C}$ of processes will be said *separable by test* if for any pair $(\mathcal{P}, \mathcal{Q})$ in $\mathcal{C}$ there exists a process $\mathcal{E} \in \mathcal{C}$ such that

$$\text{Result}(\mathcal{P}, \mathcal{E}) \neq \text{Result}(\mathcal{Q}, \mathcal{E})$$

To answer the question of separability of $\mathcal{O}_n$, we shall express $\simeq$ in terms of the more primitive relations *may* and *must* [Hen88]:

$$\begin{aligned} \mathcal{P} \ may \ \mathcal{E} &\Leftrightarrow \top \in \text{Result}(\mathcal{P}, \mathcal{E}) \\ \mathcal{P} \ must \ \mathcal{E} &\Leftrightarrow \bot \notin \text{Result}(\mathcal{P}, \mathcal{E}) \end{aligned}$$

This enables us to define preorder relations

$$\begin{aligned} \mathcal{P} \sqsubseteq_{MAY} \mathcal{Q} &\Leftrightarrow \forall_{\mathcal{E} \in \mathcal{O}_n} (\mathcal{P} \ may \ \mathcal{E} \Rightarrow \mathcal{Q} \ may \ \mathcal{E}) \\ \mathcal{P} \sqsubseteq_{MUST} \mathcal{Q} &\Leftrightarrow \forall_{\mathcal{E} \in \mathcal{O}_n} (\mathcal{P} \ must \ \mathcal{E} \Rightarrow \mathcal{Q} \ must \ \mathcal{E}) \end{aligned}$$

inducing equivalence relations $\simeq_{MAY}, \simeq_{MUST}$ in the isual way.

We immediately have [Hen88]:

$$\mathcal{P} \simeq \mathcal{Q} \text{ iff } \mathcal{P} \simeq_{MAY} \mathcal{Q} \text{ and } \mathcal{P} \simeq_{MUST} \mathcal{Q}$$

We now apply previous results to establish that some $\mathcal{O}_n$- processes discriminated throughout their set of traces (i.e. throughout some finite or infinite sequences of visible behaviours) are nevertheless not distinguishable by test.

THEOREM 2. *For $n \geq 1$, $\mathcal{O}_n$ is not separable by test.*

Proof. By thm 1, we know that, for each n, U and V are not separable in $\lim_{\mathbf{\Sigma}^0_1} \Sigma^*_n$ for $U = uX^\omega$, $u \in X^*$ being any prefix of β and $V = U \setminus \{\beta\}$. By lemma 1, there exist deterministic $\mathcal{P}, \mathcal{Q} \in \mathcal{O}_n$ such that $\mathcal{L}_\omega(\mathcal{P}) = U$, $\mathcal{L}_\omega(\mathcal{Q}) = V$. Let $P = \mathcal{C}omp_\omega(\mathcal{P})$ and $Q = \mathcal{C}omp_\omega(\mathcal{Q})$ then $P, Q \in \lim_{\mathbf{\Sigma}^0_1} \Sigma^*_n$ (by lemma 1), and not separable (by thm. 1) since $pr(P) = U$ and $pr(Q) = V$ with $pr(\chi) = \pi_2(\chi(0))\pi_2(\chi(1))\ldots$ More precisely, $Q = P \setminus \{\chi\}$ with $pr(\chi) = \beta$.

Since $\mathcal{C}omp_\omega(\mathcal{Q}) \subseteq \mathcal{C}omp_\omega(\mathcal{P})$, we immediately have

$$\mathcal{Q} \sqsubseteq_{MAY} \mathcal{P} \text{ and } \mathcal{P} \sqsubseteq_{MUST} \mathcal{Q}.$$

Suppose now, toward a contradiction, that

$$\mathcal{P} \not\sqsubseteq_{MAY} \mathcal{Q} \text{ or } \mathcal{Q} \not\sqsubseteq_{MUST} \mathcal{P} \tag{21}$$

Left hand side of eq. 21 implies that there exists an $\mathcal{O}_n$-tester whose, the only $\mathcal{P}$-computation resulting in a successful test is the run over β, contradicting lemma 3 since there is no singleton in $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$.

In a similar way, right hand side of eq. 21 means that for some $\mathcal{O}_n$-tester the only $\mathcal{P}$-computation resulting in an unsuccessful test is the run over β contradicting lemma 3 again since there is no more singleton in $\lim_{\mathbf{\Sigma}_1^0}\Sigma_{n+1}^*$ and this completes the proof.

9. Toward a hierarchy of logical complexity of testing. We now postpone any further analysis through one among the many recursive process specification formalisms (e.g. see [Dar86] for such an analysis applied to Milner's CCS) and take rather advantage of our framework to establish some results concerning the logical complexity of various notions of test equivalence relations on $\mathcal{O}_n$.

Assuming a Gödel numbering of all recursively enumerable languages over an alphabet X. This yields recursive numberings for each class Σ_n^* and each class Π_n^* called respectively Σ_n^*-index and Π_n^*-index [Rog87] (section 14.2) and of course, recursive numberings for each class $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$ and each class $\lim_{\mathbf{\Pi}_1^0}\Pi_n^*$. From that numbering, we could easily imagine recursive numberings for each class of $\mathcal{O}_n$-processes as well. We refer the reader to [Mul92] where such indices are built up for more general classes of recursive automata. Let us call such index an *n-index*. Writing $\mathcal{E}_{<i>}$ to mean that i is the n-index of $\mathcal{E}$ (for n fixed), the above equivalences are encoded into the following predicates

$$\mathcal{P} \simeq_{MAY} \mathcal{Q} \quad \Leftrightarrow \quad \forall_i(\mathcal{P}\ may\ \mathcal{E}_{<i>} \Leftrightarrow \mathcal{Q}\ may\ \mathcal{E}_{<i>}) \tag{22}$$

$$\mathcal{P} \simeq_{MUST} \mathcal{Q} \quad \Leftrightarrow \quad \forall_i(\mathcal{P}\ must\ \mathcal{E}_{<i>} \Leftrightarrow \mathcal{Q}\ must\ \mathcal{E}_{<i>}) \tag{23}$$

Using n-indices, we complete now lemma 3 to obtain a first complexity result of testing equivalence for $\mathcal{O}_n$:

THEOREM 3. *For $n \geq 1$, if $\mathcal{P}, \mathcal{Q} \in \mathcal{O}_n$ then $\mathcal{P} \simeq_{TEST} \mathcal{Q}$ is a Π_{n+1}^*-predicate for $TEST \in \{MAY, MUST\}$.*

Proof. Note immediately that predicates in equations 18 and 19 are almost the same then those on right hand side of equations 16 and 17 so, in a fashion similar to the one involved in the proof of lemma 3, second order quantifiers appearing on right hand side of equations 16 and 17 may be collapsed. It turs out that predicate $\mathcal{P}\ may\ \mathcal{E}$ is in Σ_n^* while predicate $\mathcal{P}\ must\ \mathcal{E}$ is in Δ_{n+1}^*.

The MAY part of thm. then follows from eq. 22 and the MUST part, from eq. 23.

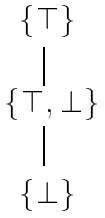
To close this section, note that our *must* definition seems to be a little bit less general than the Hennessy's one in the following sense: Hennessy's *must* gets capabilities to manage with some divergent computations. We could extend in several ways the above notion to cope with divergent computations as well. We could, for example, add a second $\mathcal{O}_n$-automaton $\mathcal{E}'$ to specify some divergent computations and one more constraint to definition 17 asserting that all successful tests under this rejection test criterion are prohibited. It is not difficult to see that the logical complexity of such an extension remains the same.

10. Discussion: separability and full abstraction. Of course, automata are convenient to model the step-by-step operational semantics of a process while the set

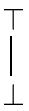
Result$(\mathcal{P}, \mathcal{E})$ is well-suited to define its denotational semantics. For example, the natural equivalence described above enables us to define a preorder between processes:

$$\mathcal{P} \sqsubseteq \mathcal{Q} \text{ if } \text{Result}(\mathcal{P}, \mathcal{E}) \subseteq \text{Result}(\mathcal{Q}, \mathcal{E})$$

This preorder arises naturally from the Egli-Milner powerdomain



of the two-point lattice



The property according to which denotations always discriminate distinct process behaviours is called *full abstraction* [Mil77]. Our separability results show in particular that every even fully abstract semantics according to some testing equivalence for $\mathcal{O}_n$ would be unable to discriminate some processes nevertheless distinguished by their infinite sequences of visible actions. This remark strengthens a similar remark in [Dar86] about CCS processes. This apparent paradox just enlights the fact that our test criteria specify only finite prefixes of computations and hence, tests never take into account any infinite extension of these behaviours. Moreover, it stresses that the notion of full abstraction turns out to be itself a finitary notion.

11. Conclusion and future works. The approach developed in the paper intend to reduce the gap between formal language theory and algebraic theory of processes. In the last section, we even try to connect with semantics of concurrency. We feel that a much more precise connection could be done and we postpone it to further investigations.

Concerning the logical complexity study of testing equivalences, we conjecture a hierarchy result. We point out that even if we know that $\{\Pi^*_n\}$ forms a hierarchy and that

$$\text{TEST}_n \subseteq \Pi^*_{n+1}$$

with

$$\text{TEST}_n = \{< i, j >: \mathcal{P}_{<i>} \simeq_{TEST} \mathcal{Q}_{<j>} \text{ and } n\text{-codes } i, j\},$$

it is not sufficient to prove that $\{\text{TEST}_n\}$ forms a hierarchy as well. But it would, if TEST_n was complete under Turing reduction, for Π^*_{n+1}.

On the other hand, the connection which has been achieved between non-separability of $\lim_{\mathbf{\Sigma}^0_1}\Sigma^*_n$ and non-separability by test of $\mathcal{O}_n$ seems, by its elegance and simplicity, to offer a very promising methodology to search for similar results starting from more general hierarchy results in recursion theory and to serve as a natural framework to achieve a fine classification of the complexity of various process equivalence relations.

Acknowledgements. The author would like to thank André Arnold for proofreading and his comments on this work and more particularly, to have, the first, pointed out that thm. 2 strongly depends on condition 10.

REFERENCES

[Arn83] A. Arnold, Topological characterizations of infinite behaviours of transition systems, in: Automata, Languages and Programming (ed. J. Diaz), LNCS **154**, Springer, Berlin (1983), 28-38.

[Dar86] Ph. Darondeau, Une critique de la notion de test de processus fondée sur la non séparabilité de certaines classes de langages, *Informatique Théorique et Applications* **20**(3) (1986), 291-317.

[Hen88] M. Hennessy, *Algebraic theory of processes*, MIT Press (1988).

[Mil77] R. Milner, Fully abstract models of typed λ-calculi, *Theoretical computer Science* **4** (1977), 1-23.

[Mul92] J. Mullins, Kleene automata and recursion theory, *Logical Foundations of Computer Science - Tver'92*, Lecture Notes in Computer Science **620**, Springer, Berlin (1992), 336-346.

[Mul95] J. Mullins, On a hierarchy of recursive processes: theoretic limit and complexity of testing, Technical Report TR-95-04, University of Ottawa (1995).

[Rog87] H. Rogers, *Theory of recursive functions and effective computability*, MIT Press (1987).

[Sta86] L. Staiger, Hierarchies of recursive of ω-languages, *J. Inform. Process. Cybernet. EIK* **22** (1986), 219-241.