ON A HIERARCHY OF RECURSIVE PROCESSES: THEORETIC LIMIT AND LOGICAL COMPLEXITY 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 start with a hierarchy of some effective classes of guarantee properties settled down by Staiger [Sta86] and prove their non-separability. Then we define classes of processes in such a way that this hierarchy of properties to be induced on. Thereafter, we conclude the non-separability under testing for each class of this new hierarchy. We close the paper on some results concerning the logical complexity of De Nicola and Hennessy's process equivalences [Hen88] when test criteria are ranging over the hierarchy of properties above defined. An extensive use of the compactness property is made throughout the paper so the assumption of finiteness of alphabet seems to play here a crucial role.

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.

In a computational framework, it is natural to demand to finite process behaviours to be "effectively" generated by the computation device. That is why we focus to the class of guarantee properties generated when U ranges over the arithmetical hierarchy. Staiger showed tha 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 no class in this hierarchy can be separated (thm. 1).

* 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.

As a second step, we define for each class of the above hierarchy, a class of non-deterministic recursive automata recognizing exactly this class (thm. 2). This result strongly depends on the finiteness of the set of actions and on the closure of the class of behaviours under projection. The issue is that the hierarchy over the classes of behaviours induces a hierarchy over the classes of recursive automata.

As last 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], we prove the non-separability under testing for each class of automata in the hierarchy (thm. 3), using the similar result previously derived on corresponding classes of behaviours.

We finally compute the logical complexity of testing problem (thm. 4). For each class of the hierarchy of automata, it appears to be into the "dual" class of the test criteria. But before doing that we need to define a recursive numbering relative to each class to manage with quantifiers over processes. It turns out too, that the elimination of second order quantifiers strongly depends on the finiteness of the set of actions. We close on a discussion about the relationship between *separability* and *full abstraction*.

2. Definitions and background. Let X, a finite 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 roman capital 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 Roman lower case letters $(u, v, \ldots)$ and infinite sequences by Greek lower case 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 since the discrete topology trivially has this property. Since any prefix-closed ω-language is a close set in this topology, it is a compact set and then it has the Bolzano-Weierstass property whose the contraposition is the well-known *König's lemma*. This topology is known to be homeomorphic to the Cantor space. Note that X^ω is compact if and only if X is finite. 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

$$\lim_{\mathbf{\Sigma}_1^0} U \quad \stackrel{\text{def}}{=} \quad \{\alpha \in X^\omega : \exists_i \alpha[i] \in U\}$$

$$\lim_{\mathbf{\Pi}_1^0} U \quad \stackrel{\text{def}}{=} \quad \{\alpha \in X^\omega : \forall_i \alpha[i] \in U\}$$

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

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

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:

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

The Borel hierarchy of ω-languages is defined as follows: $W \in \mathbf{\Sigma^0_n}$ if $\mathbf{Q}_{t_1} \equiv \exists_{t_1}$ and $W \in \mathbf{\Pi^0_n}$ 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^0_1 and $\mathbf{\Pi^0_1}$ 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: $W \in \Sigma^0_n$ if $\mathbf{Q}_{t_1} \equiv \exists_{t_1}$ and $W \in \Pi^0_n$ if $\mathbf{Q}_{t_1} \equiv \forall_{t_1}$. Notice the lightface font which distinguishes for example the class Σ^0_1 from the class $\mathbf{\Sigma^0_1}$ 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 $\mathbf{\Sigma^0_1}$ is the class of recursively enumerable sets of X^ω.

By induction one easily obtains that

$$\mathbf{\Sigma^0_n} \cup \mathbf{\Pi^0_n}\ (\text{resp. } \Sigma^0_n \cup \Pi^0_n) \quad \subset \quad \mathbf{\Sigma^0_{n+1}} \cap \mathbf{\Pi^0_{n+1}}\ (\text{resp. } \Sigma^0_{n+1} \cap \Pi^0_{n+1}) \tag{1}$$

$$U \in \mathbf{\Sigma^0_n}\ (\text{resp. } \Sigma^0_n) \quad \text{iff} \quad \overline{U} \in \mathbf{\Pi^0_n}\ (\text{resp. } \in \Pi^0_n) \tag{2}$$

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

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

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 14). In particular, we assume the rules to contract quantifiers and eliminate bounded quantifiers.

3. Effective hierarchy 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 Cantor topology.

Let $W = \lim_{\mathbf{\Sigma^0_1}} U$ 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 a computation device. Let us define now of course, effective languages.

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

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

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):

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

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

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

Staiger showed that 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).

4. The separability result. The next definition has been widely motivated in the 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 [Dar86].

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

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

so, because X is finite, the quantifiers $\forall_x$ and $\forall_j$ are bounded and the condition $\beta(j) = x$ is, by definition of β, a Π_0^*-predicate. Hence, by the Tarski-Kuratowski algorithm, the predicate *"u is a prefix of β"* is a Π_0^*-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 equations 1 to 4, $V \in \lim_{\mathbf{\Sigma}_1^0} \Sigma_n^*$. 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).

It is in general not possible to bound the size of the actions of a process in the way we bounded the alphabet X. However, in the above result, the finiteness of the alphabet X is not essential. If X is denumerable, the quantifier $\forall_x$ is not bounded and the predicate *"u is a prefix of β"* turns to be a Π_1^*-predicate leaving unchanged all the rest of the above argument.

5. Open questions about separability. 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

$$\lim_{\mathbf{\Pi}^0_1}\Pi^*_n \setminus \Sigma^0_n \neq \emptyset$$

[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 Bernays'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:

$$\lim_{\mathbf{\Pi}^0_2} U \stackrel{\text{def}}{=} \{\alpha \in X^\omega : \stackrel{\infty}{\exists}_i \alpha[i] \in U\}$$

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 he 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 computations. Now we 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 ω-languages accepted corresponds to $\lim_{\mathbf{\Sigma}^0_1}\Sigma^*_n$ classes.

An automaton $\mathcal{A}$ over a finite 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 $q \stackrel{x}{\longrightarrow} q'$ 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 $q \in Q$, $\{q' : q \stackrel{x}{\longrightarrow} q'\}$ are singletons. An automaton $\mathcal{A}$ is said finitely or infinitely branching depending if the set $\{q' : q \stackrel{x}{\longrightarrow} q'\}$ is finite for every $x \in X$ and $q \in Q$ or not. An automaton is said to be complete if $\{q' : q \stackrel{x}{\longrightarrow} q'\} \neq \emptyset$ for every $x \in X$ and $q \in Q$. In this section, unless it is otherwise indicated, all automata are considered to be complete and finitely branching.

Given a word $\alpha \in X^*$ of length n, a sequence $q_0, q_1, \ldots, q_n$ is called a run of $\mathcal{A}$ over α if

$$q_0 \stackrel{\alpha(0)}{\longrightarrow} q_1 \stackrel{\alpha(1)}{\longrightarrow} \cdots \stackrel{\alpha(n-1)}{\longrightarrow} q_n$$

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 $q_0, q_1, \ldots$ such that

$$q_0 \stackrel{\alpha(0)}{\longrightarrow} q_1 \stackrel{\alpha(1)}{\longrightarrow} \cdots$$

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 denote $\mathcal{L}_\omega(\mathcal{A})$ the ω-language accepted by $\mathcal{A}$ in such a way. 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. The following connection between $\mathcal{L}(\mathcal{O}_n)$ and $\mathcal{L}_\omega(\mathcal{O}_n)$ and its corollary is quite immediate and seems to strongly depend both on the finiteness of the alphabet and of the branching as well.

THEOREM 2.

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

COROLLARY 2. *The family $\{\mathcal{O}_n\}$ forms an infinite hierarchy.*

7. From computations to visible behaviour. In this section, we first 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. We apply thm. 1 and thm. 2 to process separability under testing for $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$ classes. We then postpone any further analysis through one among the many recursive process specification formalisms (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 testing equivalence relations on 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].

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 $Q = Q_1 \times Q_2$, $Q_0 = Q_{10} \times Q_{20}$, $Q_f = Q_{1f} \times Q_{2f}$ and $\longrightarrow$ is the relation satisfying

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

Because X is finite, the quantifier $\exists_x$ is bounded. Hence, by definition of $\longrightarrow_1$ and $\longrightarrow_2$ and by the Tarski-Kuratowski algorithm $\longrightarrow \in \Sigma_1^*$ and we obtain that the class $\mathcal{O}_n$ (on $X \cup \{\tau\}$) is closed under the synchronisation product. Here again, note that this last property would hold even if X was denumerable.

The idea behind this definition is to provide a mechanism to test behaviours of mechanical devices using a device from the same class. So if we want to deduce non separability by testing from non separability of $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$ that is the converse which would be useful that is for any ω-language $W \in \lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$ over $X \cup \{\tau\}$, there are $\mathcal{O}_n$-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ over X such that $W = \mathcal{L}_\omega(\mathcal{A}_1 \parallel \mathcal{A}_2)$. Unfortunately, it is not true in general. From this point, let us go back to the root of testing. 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. It is represented by a run of $\mathcal{P} \parallel \mathcal{E}$. A *test criterion* is here a 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 the final states of $\mathcal{P}$. The test is binary. An infinite run is *successful* if it is an accepting run from $\mathcal{P} \parallel \mathcal{E}$ of the form

$$(q_0, q_0') \xrightarrow{\tau} (q_1, q_1') \xrightarrow{\tau} \dots$$

which is "maximal" with respect to guarantee constraint and test criterion simultaneously i.e. if

$$\exists_{x \in X} \exists_j (q_0, q_0') \xrightarrow{\tau} (q_1, q_1') \xrightarrow{\tau} \dots (q_j, q_j') \xrightarrow{x} \dots$$

then

$$\exists_{i\leq j}(q_i, q'_i) \in Q_f \times Q'_f$$

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

$$\top \in \text{Result}(\mathcal{P}, \mathcal{E}) \quad \Leftrightarrow \quad \mathcal{L}_\omega(\mathcal{P}) \cap \mathcal{L}_\omega(\mathcal{E}) \neq \emptyset \tag{6}$$

$$\bot \in \text{Result}(\mathcal{P}, \mathcal{E}) \quad \Leftrightarrow \quad \mathcal{L}_\omega(\mathcal{E}) \not\subseteq \mathcal{L}_\omega(\mathcal{P}). \tag{7}$$

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]:

$$\mathcal{P} \textit{ may } \mathcal{E} \quad \Leftrightarrow \quad \top \in \text{Result}(\mathcal{P}, \mathcal{E})$$
$$\mathcal{P} \textit{ must } \mathcal{E} \quad \Leftrightarrow \quad \bot \notin \text{Result}(\mathcal{P}, \mathcal{E})$$

This enables us to define equivalence relations:

$$\mathcal{P} \simeq_{MAY} \mathcal{Q} \quad \Leftrightarrow \quad \forall_{\mathcal{E}\in\mathcal{O}_n}(\mathcal{P} \textit{ may } \mathcal{E} \Leftrightarrow \mathcal{Q} \textit{ may } \mathcal{E})$$
$$\mathcal{P} \simeq_{MUST} \mathcal{Q} \quad \Leftrightarrow \quad \forall_{\mathcal{E}\in\mathcal{O}_n}(\mathcal{P} \textit{ must } \mathcal{E} \Leftrightarrow \mathcal{Q} \textit{ must } \mathcal{E})$$

We immediately have [Hen88]:

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

THEOREM 3. *For each n, $\mathcal{O}_n$ 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 thm. 2 there exist $\mathcal{P}, \mathcal{Q} \in \mathcal{O}_n$ such that $U = \mathcal{L}_\omega(\mathcal{P})$ and $V = \mathcal{L}_\omega(\mathcal{Q})$. Noting that $\mathcal{P} \simeq_{MAY} \mathcal{Q}$ iff $\mathcal{L}_\omega(\mathcal{P})$ and $\mathcal{L}_\omega(\mathcal{Q})$ are not separable, we get

$$\mathcal{P} \simeq_{MAY} \mathcal{Q}$$

For the *must* part, since $\mathcal{Q} \subseteq \mathcal{P}$, we have

$$\forall_{\mathcal{E}\in\mathcal{O}_n}(\mathcal{Q} \textit{ must } \mathcal{E} \Rightarrow \mathcal{P} \textit{ must } \mathcal{E})$$

Suppose, toward a contradiction, that there is an $\mathcal{E} \in \mathcal{O}_n$ such that

$$\mathcal{P} \textit{ must } \mathcal{E} \not\Rightarrow \mathcal{Q} \textit{ must } \mathcal{E} \tag{8}$$

then using again thm. 2, there is an $\lim_{\mathbf{\Sigma}^0_1}\Sigma^*_n$-language W such that $W = \mathcal{L}_\omega(\mathcal{E})$ and, by eq. 8,

$$W \subseteq U \text{ and } W \not\subseteq V.$$

This implies that $W = U$ so, $\mathcal{E}$ testing β on $\mathcal{Q}$ must result in a failure. But any accepting computation of $\mathcal{Q}$ has an accepting prefix of length $|u|$ computing u and hence will result in a success when testing with β and this contradicts the assumption.

8. Toward a hierarchy of logical complexity of testing. Having recasted various definitions of testing in a common framework, we can now take advantage of it to compare the logical complexity of the induced equivalence relations on $\mathcal{O}_n$ from the theorem 2. We first assume a Gödel numbering of all recursively enumerable languages over an alphabet X. This numbering 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, by the way in the case of ω-languages, 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 of course) the above equivalences are encoded into the following predicates

$$\mathcal{P} \simeq_{MAY} \mathcal{Q} \quad \Leftrightarrow \quad \forall_i(\mathcal{P} \textit{ may } \mathcal{E}_{<i>} \Leftrightarrow \mathcal{Q} \textit{ may } \mathcal{E}_{<i>}) \tag{9}$$

$$\mathcal{P} \simeq_{MUST} \mathcal{Q} \quad \Leftrightarrow \quad \forall_i(\mathcal{P} \textit{ must } \mathcal{E}_{<i>} \Leftrightarrow \mathcal{Q} \textit{ must } \mathcal{E}_{<i>}) \tag{10}$$

These results can be collected into:

THEOREM 4. *If $\mathcal{P}, \mathcal{Q} \in \mathcal{O}_n$ then $\mathcal{P} \simeq_{TEST} \mathcal{Q}$ is a $\lim_{\mathbf{\Pi}_1^0}\Pi_{n+1}^*$-predicate for TEST $\in$ {MAY, MUST}.*

Proof. In terms of predicate logic, the right hand side of the equation 6 may be rewritten as follows:

$$\exists_{\alpha \in X^\omega} (\alpha \in \mathcal{L}_\omega(\mathcal{P}) \wedge \mathcal{L}_\omega(\mathcal{E})) \tag{11}$$

This involves a quantifier ranging over an infinite object. Here we can simplify because this quantifier is easily eliminated using theorem 2. Let $P, T \in \Sigma_n^*$ such that $\mathcal{L}_\omega(\mathcal{P}) = \lim_{\mathbf{\Sigma}_1^0} P$ and $\mathcal{L}_\omega(\mathcal{E}) = \lim_{\mathbf{\Sigma}_1^0} T$. Equation 11 becomes:

$$\exists_{\alpha \in X^\omega} \exists_i \exists_j (\alpha[i] \in P \wedge \alpha[j] \in T)$$

Finally, this last equation can be transformed into

$$\exists_n \exists_{u \in X^*} \exists_{i \leq n} \exists_{j \leq n} (|u| = n \wedge u[i] \in P \wedge u[j] \in T)$$

which is in $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$. It is easily demonstrated using the Tarski-Kuratowski algorithm, eliminating bounded quantifiers and finally using the fact that Σ_n^* is closed under finite intersection. That proves the predicate $\mathcal{P}$ *may* $\mathcal{E}$ to be in $\lim_{\mathbf{\Sigma}_1^0}\Sigma_n^*$. The MAY result then follows from eq. 9 and from the fact that $\Sigma_n^* \subseteq \Pi_{n+1}^*$. Note that we would get the last result even if the alphabet X was denumerable.

Estimate now the logical complexity of the predicate $\mathcal{P} \simeq_{MUST} \mathcal{Q}$. The right hand side of equation 7 may be rewritten into

$$\exists_{\alpha X^\omega} (\exists_i \alpha[i] \in T \wedge \forall_j \alpha[j] \in (X^* \setminus P)) \tag{12}$$

Now, distributing the quantifier on infinite object over the conjunction and eliminating it in its left hand side as above, we get

$$\exists_m \exists_{v \in X^*} (|v| = m \wedge v \in T \wedge Q(m, v))$$

where the predicate $Q(m, v)$ stands for

$$\exists_{\alpha X^\omega} \forall_j (\alpha[m] = v \wedge \alpha[j] \in (X^* \setminus P))$$

But to remove the quantifier on infinite object in $Q(m,v)$ we need the compactness property of X^ω. By König's infinity lemma the condition involved is equivalent to

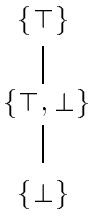
$$\forall_{n\geq m}\exists_{u\in X^*}\,(|u|=n\wedge u[m]=v\wedge\forall_j(j\leq n\rightarrow u[j]\in(X^*\setminus P)))$$

As above, using the Tarski-Kuratowski algorithm, eliminating bounded quantifiers and finally using eq. 5, we obtain that the predicate $\mathcal{P}$ *must* $\mathcal{E}$ is in $\lim_{\mathbf{\Sigma}^0_1}\Sigma^*_n$. The MUST result follows from eq. 10 and from the fact that $\Sigma^*_n\subseteq\Pi^*_{n+1}$.

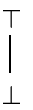
9. Discussion: separability and full abstraction. Of course, automata are convenient to model the step-by-step operational semantics of a process and $\mathrm{Result}(\mathcal{P},\mathcal{E})$ its denotational semantics. For example, the natural equivalence described above enables us to define a preorder between processes:

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

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.

10. 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 as a corollary of the Staiger's hierarchy result [Sta86] (section 6).

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.

REFERENCES

[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).

[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.