A SURVEY OF RECOGNIZABLE LANGUAGES OF INFINITE TRACES*

Paul GASTIN
Université PARIS 6
LITP, Institut Blaise Pascal
4, place Jussieu
75 252 PARIS CEDEX 05
FRANCE
gastin@litp.ibp.fr

Antoine PETIT
Université PARIS SUD
LRI, URA CNRS 410
Bât. 490
91 405 ORSAY CEDEX
FRANCE
petit@lri.lri.fr

ABSTRACT: A. Mazurkiewicz [Maz77] defined traces in order to represent non-sequential processes. In order to describe non-sequential processes which never terminate, e.g. distributed operating systems, the notion of infinite traces is needed. The aim of this survey is to present in a uniform way the results on recognizable infinite trace languages stated in [Gas91], [GPZ91] and [DGP91]. The proofs of the presented results are not proposed here but can be found in the original papers.

Keywords: Semantics of concurrent processes, Infinite traces, Recognizable languages.

CONTENTS

1. Introduction
2. Preliminaries
3. Recognizable languages
4. Concatenation
5. Connected traces
6. Iterations
7. Canonical form and co-rational languages
8. Conclusion
9. References

1. Introduction

The concept of traces has been introduced by A. Mazurkiewicz [Maz77] as a suitable semantics for non-sequential processes. There are at least two equivalent ways to see a trace. A trace can be considered as the set of all possible sequential observations of a concurrent process. Each sequential observation is described, in a classical way, by a finite word. Hence, a trace is an equivalence class of words. An equivalent point of view is to see a trace as a finite labelled acyclic graph where edges represent dependence between actions. Such a graph is called a dependence graph and the intuitive meaning is that an execution has to respect the induced partial order. Whatever the approach chosen, the definition of the concatenation of two traces is easy and natural and traces form a monoid. Originally, this monoid has been introduced and studied by combinatorists [CF69] for other purposes. But it has particularly been investigated

* This work has been partly supported by the ESPRIT Basic Research Action N° 3148 (DEMON).

as a model for concurrent systems since the pioneering work of Mazurkiewicz. Let us refer, for instance, to surveys [Maz86, AR88, Per89] or to the monograph [Die90] where also extensive bibliographies on the subject are given.

In order to describe non-sequential processes which never terminate, e.g. distributed operating systems, the notion of infinite traces is needed. An infinite (real) trace is an infinite dependence graph where every vertex has only a finite past. This notion has been introduced and studied in distinct frameworks by many authors: Flé and Roucairol [FR85] for problems of infinite serialisabilities, Best and Devillers [BD87] in relation with Petri Nets, Mazurkiewicz [Maz86], Gastin [Gas90a] and Kwiatkowska [Kwi90] to generalize the notion of finite traces. They have all proposed definitions which turn out to be equivalent. Then, the theory of infinite real traces has been developed in several papers. Topological properties of infinite traces [BMP89, Gas90b], PoSet properties [Kwi90, GR91] and links with event structures [GR91] have in particular been studied.

The concatenation of two infinite dependence graphs can naturally be defined. Unfortunately, the restriction of the graph concatenation to real traces is not an internal operation since the concatenation of two real traces may contain vertices with an infinite past. A first obvious solution to this problem is to consider only a partial concatenation on real traces. Another possibility, proposed by Kwiatkowska [Kwi91], is to erase vertices with an infinite past. Acting this way, she obtained a non associative total concatenation on real traces. Independently, and in order to define an associative total operation, Gastin [Gas90a] added a new element, the error, used as a result for all non real concatenations. Hence, he obtained a monoid which is in fact the Rees quotient of the monoid of infinite dependence graphs by an appropriate ideal (the set of infinite dependence graphs which have vertices with infinite past). One of the main drawbacks of this concatenation is that when all actions are pairwise dependent, the trace monoid is not isomorphic to the free monoid of (finite and infinite) words.
Diekert [Die91] proposed a new solution to the problem of concatenation. He defined the notion of complex traces using a suitable quotient of the set of dependence graphs and studied their topological properties. A complex trace is simply a real trace together with a second component which is some alphabetic information. Acting this way, a concatenation between complex traces is defined, which combines both the free case and the pure concurrent one.

Recognizable languages describe the behaviour of finite state systems and hence form one of the basic families of a monoid. The family of recognizable real trace languages has been introduced in [Gas90] and studied in [Gas91, GPZ91, GP92, Ebi92], whereas recognizable complex trace languages have been investigated in [DGP91]. The aim of this survey is to give a uniform presentation of the results on recognizable infinite trace languages.
We will now recall some aspects of the theory of recognizable languages of finite traces and infinite words. Next we present the results of recognizable infinite trace languages.
For finite and infinite words, there exist several equivalent characterizations of the family of recognizable languages. The principal ones are: recognizability by a morphism in a finite monoid, acceptance by a finite automaton, definition by a rational expression (Kleene's theorem) or by a monadic second order logic formula. The reader is referred to [Eil74, Tho90, PP91] for surveys on recognizable (infinite) word languages.

For finite traces, the recognizable languages are defined as the languages recognized by morphisms in finite monoids (as usual in a finitary monoid). These languages are exactly those recognized by "diamond finite automata" or by the more specific asynchronous (cellular) automata introduced by Zielonka [Zie87, Zie89, CMZ89]. A lot of papers are devoted to the study of closure properties of the family of recognizable languages of finite traces. This family is closed under boolean operations and concatenation [Fli74, CP85, Och85] but it turns out that a trace language T may be recognizable whereas T^* is not. Therefore the family of recognizable trace languages is strictly included in the family of rational trace languages i.e. the smallest family of trace languages containing the finite languages and closed under union, concatenation and star iteration. In order to generalize Kleene's theorem to trace languages it was then necessary to find new operations instead of the star iteration. To this purpose, several papers discussed sufficient conditions ensuring the recognizability of T* [CM85, FR85, Sak87, Och90, Roz90, MR91]. But the most interesting sufficient condition has been found independently by Métivier [Mét86] and Ochmanski [Och85]. They showed that if a recognizable language T is connected then the language T* is recognizable too. Moreover Ochmanski defined a concurrent version of the star iteration, the concurrent iteration, and proved the equality of the families of recognizable trace languages and co-rational trace languages. This latter family is obtained as the rational one by simply replacing the star iteration by the concurrent one.
The family of recognizable infinite trace languages is defined using recognizing morphisms in finite monoids. This family is a boolean algebra which is closed under concatenation, left and right quotients. But similarly to the finite case, this family is neither closed under Kleene's iteration nor under infinite iteration. Therefore the same problems as in the finite case arise: find sufficient conditions ensuring the recognizability of T^* and T^ω and new operations to obtain a generalization of Kleene's and Ochmanski's theorems. In fact, as long as the Kleene's iteration of a language of finite traces is recognizable, then so is its infinite iteration. Hence, any sufficient condition ensuring the recognizability of T^* holds also for T^ω. Then, we generalize the notion of connectivity to infinite traces. We show that if a language T containing connected (infinite) traces only is recognizable then so are T^* and T^ω. Then we propose a canonical representation for recognizable trace languages. We define concurrent iterations on trace languages and the family of co-rational trace languages is introduced. This family is equal to the family of recognizable trace languages. Hence we obtain a generalization of Kleene's and Ochmanski's theorems to languages of infinite traces.

Instead of presenting formal proofs, we choose in this survey to illustrate definitions and results by numerous examples. The detailed proofs can be found in the original papers [Gas91, GPZ91, DGP91].

2. Preliminaries

In this section, we recall definitions and results on traces which are relevant in this paper. For more detailed discussions, the reader is referred to [Maz86, AR88, Per89, Die90] as concerns finite traces, to [Gas90b] as concerns real traces and to [Die91] for complex traces.

2.1. Dependence graphs

In the following (X,D) denotes a finite dependence alphabet, i.e., X is a finite alphabet together with a reflexive and symmetric dependence relation $D \subseteq X \times X$. We denote by $I = X \times X \setminus D$ the independence relation which is the complement of D. For a subset $A \subseteq X$, define $D(A) = \{b \in X / \exists\, a \in A, (a,b) \in D\}$ and $I(A) = X \setminus D(A) = \{b \in X / \forall\, a \in A, (a,b) \in I\}$. A **dependence graph** over (X,D) is (an isomorphism class of) a labelled acyclic graph $[V,E,\lambda]$ where V is the (at most) countable set of vertices, $E \subseteq V \times V$ is the set of arcs, $\lambda: V \to X$ is the labelling and which satisfies:

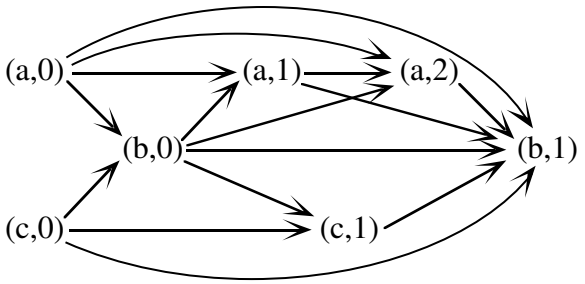
i) the induced partial order $(V, \leq)$ is well-founded: there does not exist an infinite descending chain i.e. an infinite sequence (x_i) such that $(x_{i+1}, x_i) \in E$ for all i.

ii) edges are between dependent vertices:

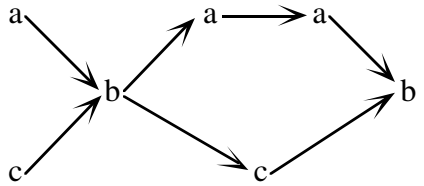
$$\forall\, x,y \in V, \big((\lambda(x),\lambda(y)\big) \in D \Leftrightarrow x = y \text{ or } (x,y) \in E \text{ or } (y,x) \in E$$

Then, for any a in X, the restriction to the set of vertices with label a is well-ordered. This allows us to think of dependence graphs by standard representations where the vertices are pairs (a,i) with $a \in A$ and i is a countable ordinal.

For instance, let (X,D) = a — b — c, a dependence graph is presented in Figure 2.1(a). Since we are only interested in the isomorphism class of the graph, we only write the labelling of vertices. Moreover, in order to simplify the pictures, we will not draw the edges which can be obtained by composition of other edges. Using these remarks, the dependence graph of Figure 2.1(a) can be simply presented as in Figure 2.1(b). As another example, an infinite dependence graph is presented in Figure 2.2.



(a)



(b)

Figure 2.1

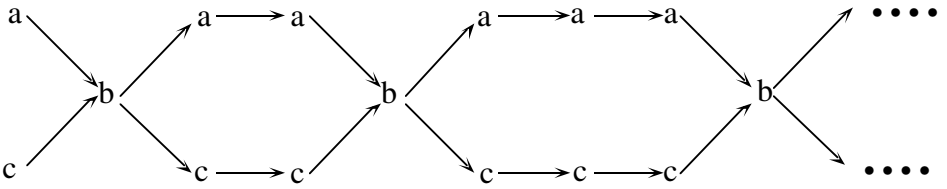


Figure 2.2.

The set of dependence graphs over the dependence alphabet (X,D) is denoted by $\mathbb{G}(X,D)$ or simply by $\mathbb{G}$ when there is no risk of confusion. It is a monoid by the operation $[V_1,E_1,\lambda_1][V_2,E_2,\lambda_2] = [V,E,\lambda]$ where $[V,E,\lambda]$ is the disjoint union of $[V_1,E_1,\lambda_1]$ and $[V_2,E_2,\lambda_2]$ together with new arcs (p_1,p_2) for all vertices $p_1 \in V_1$, $p_2 \in V_2$ such that $(\lambda_1(p_1),\lambda_2(p_2)) \in D$ (see Figure 2.3). The neutral element is the empty graph $1 = [\emptyset,\emptyset,\emptyset]$.

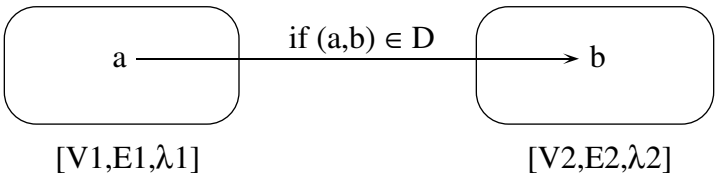


Figure 2.3.

The concatenation is generalized to an infinite product as follows. Let (g_i) be any finite or infinite sequence of dependence graphs. Then $g = g_1g_2\ldots g_i\ldots \in \mathbb{G}$ is defined as the disjoint union of the g_i with the obvious new arcs from g_i to g_j for $i<j$ whenever the vertices have dependent labels. Note that concatenation and infinite product are associative operations.

2.2. Traces

Using these operations, we relate words and dependence graphs. We recall that X^ω denotes the set of infinite words whereas $X^\infty = X^\omega \cup X^*$ denotes the set of all finite and infinite words. We define the mapping $\varphi : X^\infty \to \mathbb{G}(X,D)$ by $\varphi(a) = [\{x\},\emptyset,x \mapsto a]$ for each a in X and $\varphi(a_1a_2\ldots) = \varphi(a_1)\varphi(a_2)\ldots$ for each word $a_1a_2\ldots$ in X^∞. For instance, if (X,D) = a — b — c then, φ(cabacab) and φ(acbaaccbaaacccb…) are the graphs presented in Figures 2.1 and 2.2. Note that φ is not injective, for instance φ(ab) = φ(ba) iff (a,b) ∈ I. We define an equivalence $\sim_I$ (or simply ~ when there is no confusion) on X^∞ by, for u,v in X^∞, u ~ v if φ(u) = φ(v). It turns out that φ is a surjective morphism from the free monoid X^* onto the monoid $\mathbb{M}\mathbb{G}(X,D)$ of finite dependence graphs. The monoid of finite traces introduced by A. Mazurkiewicz [Maz77] is precisely $X^*/\sim = \mathbb{M}\mathbb{G}(X,D)$ which is denoted by $\mathbb{M}(X,D)$ or simply by $\mathbb{M}$ throughout.

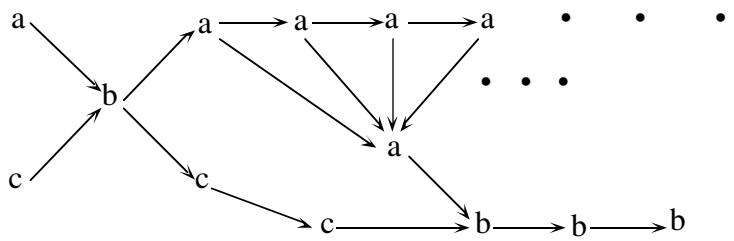


Figure 2.4.

In a similar way, we will define the monoid of infinite traces from the monoid $\mathbb{G}(X,D)$. Unfortunately, $\varphi : X^\infty \to \mathbb{G}(X,D)$ is neither surjective nor a morphism. The set $\varphi(X^\infty) = X^\infty/\sim$ of dependence graphs which can be represented by words is denoted by $\mathbb{R}(X,D)$ or simply by $\mathbb{R}$ when there is no confusion. Its elements are called real dependence graphs or **real traces**. $\mathbb{R}$ is characterized as follows: a dependence graph [V,E,λ] is real if the past $p\downarrow = \{q \in V / q \leq p\}$ is finite for every vertex p of V. For instance, the dependence graphs of Figure 2.1 and 2.2 are real whereas, with the same dependence alphabet, the graph of Figure 2.4 is not a real dependence graph although it can be obtained as the concatenation of the two real dependence graphs $\varphi(acba^\omega)$ and $\varphi(ccab^3)$. Note that, apart from the first one, all occurrences of b have an infinite past (this is not in contradiction with the well-foundedness: there is no infinite descending chain).

More generally, each element $g = [V,E,\lambda] \in \mathbb{G}$ splits into its real part Re(g) and its transfinite part Tr(g) which are the restrictions of the dependence graph g to $\{p \in V / p\downarrow$ is finite$\}$ and $\{p \in V / p\downarrow$ is infinite$\}$ respectively. For instance, let g be the dependence graph of Figure 2.4, we have $Re(g) = \varphi(acbcca^{\omega})$ and $Tr(g) = \varphi(ab^3)$. Obviously, Re(g) and Tr(g) are dependence graphs and, in $\mathbb{G}$, we have the equation: g = Re(g)Tr(g). In fact, $\mathbb{R}$ is the set of dependence graphs with an empty transfinite part.

Two dependence graphs $g,h \in \mathbb{G}$ are said to be **practically distinguishable** if there exists a (finite) real trace s such that gs and hs have different real parts, i.e., if $Re(gs) \neq Re(hs)$. In particular, two traces with different real parts are practically distinguishable. It turns out [Die91] that **practical-indistinguishability** is the coarsest congruence of $\mathbb{G}$ which respects real parts. The quotient of $\mathbb{G}(X,D)$ by this congruence yields the monoid of **complex traces** $\mathbb{C}(X,D)$ (denoted also by $\mathbb{C}$ when there is no risk of confusion). Note that the infinite product is well-defined in $\mathbb{C}$ since practical-indistinguishability is also a "congruence" with respect to the infinite product in $\mathbb{G}$.

2.3. Calculus in the monoid of complex traces.

In order to give an explicit characterization of the congruence "practical-indistinguishability", we introduce the following notations. For a dependence graph $g = [V,E,\lambda]$, let $alph(g) = \lambda(V)$ be the set of letters which occur in g at least once and let $alphinf(g) = \{a \in X / |\lambda^{-1}(a)|$ is infinite$\} \cup alph(Tr(g))$ be the alphabet at infinity of g. Finally, let $Im(g) = D(alphinf(g))$ be the imaginary part of g. For instance, let g be the dependence graph of Figure 2.4, we have alph(g) = {a,b,c}, alphinf(g) = {a,b} and Im(g) = {a,b,c}.
Two dependence graphs $g,h \in \mathbb{G}$ are practically indistinguishable if and only if Re(g) = Re(h) and Im(g) = Im(h) [Die91]. Hence a complex trace is a pair $(r,D(A)) = (Re(g),Im(g))$ for some $g \in \mathbb{G}$.

For instance, let (X,D) = a — b — c — d. The dependence graphs $g = \varphi((ac)^{\omega})$ and $h = \varphi((ac)^{\omega})\cdot\varphi(b)$ are practically indistinguishable since Re(g) = Re(h) = g and alphinf(g) = {a,c}, alphinf(h) = {a,b,c} whence Im(g) = Im(h) = {a,b,c,d}. They represent the complex trace $((ac)^{\omega},D(a,c)) = ((ac)^{\omega},D(a,b,c)) = ((ac)^{\omega},\{a,b,c,d\})$.

For a complex trace x, define $D(x) = D(alph(g))$ and Im(x) = Im(g) for some dependence graph g representing x. Note that, for a complex trace $x = (r,D(A))$, there may exist dependence graphs g and h representing x such that $alph(g) \neq alph(h)$ and $alphinf(g) \neq alphinf(h)$. Hence neither the set A nor the alphabet of x are well-defined. Nevertheless we have Im(x) = D(A) and $D(x) = D(r) \cup D(A)$. For instance, with the practically indistinguishable graphs g and h defined above, we have alph(g) = alphinf(g) = {a,c} whereas alph(h) = alphinf(h) = {a,b,c} but D(g) = D(h) = {a,b,c,d}.
Note that two real traces are practically indistinguishable if and only if they are equal. Thus, $\mathbb{R}(X,D)$ is embedded in $\mathbb{C}(X,D)$. More precisely, a real trace $r \in \mathbb{R}(X,D)$ is identified with its image $(r,Im(r))$ in $\mathbb{C}(X,D)$.

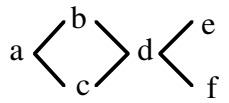
Even if the concatenation in $\mathbb{C}$ is of course defined through the practical-indistinguishability congruence, it will be convenient to have a direct formula of the concatenation of two complex

traces $(r,D(A))$ and $(s,D(B))$. To this purpose we extract from the real part s, the maximal real prefix $\mu_A(s)$ of s containing letters from I(A) only. Formally for any element $g = [V,E,\lambda]$ of $\mathfrak{G}$, $\mu_A(g)$ is the restriction to $\{p \in V / p\downarrow$ is finite & $alph(p\downarrow) \subseteq I(A)\}$ of the dependence graph g. Since $\mathfrak{G}$ is left-cancellative [Die91], there is a unique dependence graph $suff_A(g)$ such that $g = \mu_A(g)suff_A(g)$. With these notations, the concatenation in $\mathfrak{C}$ is simply given by [Die91]:

$$(r,D(A))\cdot(s,D(B)) = \big(r\mu_A(s),D(A) \cup D(B) \cup D\big(alph(suff_A(s))\big)\big).$$

This formula is of first importance when dealing with complex traces and will be used extensively throughout this paper. For instance, let (X,D) = a — b — c — d — e and consider the complex traces $x = (e^\omega,D(d,e))$ and $y = (b^2c^2da^\omega,D(a))$. Then, $\mu_{\{d,e\}}(b^2c^2da^\omega) = b^2a^\omega$, $suff_{\{d,e\}}(b^2c^2da^\omega) = c^2d$ and $x\cdot y = (b^2(ae)^\omega, D(d,e) \cup D(a) \cup D(c,d))$.

In order to link real and complex trace languages, we define, for a real trace language R and a subset A of X, the language $inf_A(R) = \{r \in R / alphinf(r) = A\}$. For a complex trace language L and a subset B of X, we define $L_B = \{x \in L / Im(x) = D(B)\}$. A couple (A,B) of subsets of X is said to be **consistent** if $inf_A(Re(\mathfrak{C}_B))$ is not empty i.e. there exists a real trace r such that alphinf(r) = A and $(r,D(B))$ is a complex trace. Consistent couples admit the following decidable characterization: (A,B) is consistent if and only if there exists a sequence of letters $a_1,\ldots,a_k$ such that $a_{i+1} \in D(A \cup \{a_1,\ldots,a_i\})$ for all $i < k$ and $D(B) = D(A \cup \{a_1,\ldots,a_k\})$. Moreover, for every consistent couple (A,B), there exists a finite trace $s_{A,B}$ of length at most |X| (for instance $s_{A,B} = a_1\ldots a_k$) such that $r\cdot s_{A,B} = (r,D(B))$ for all real traces r in $inf_A(\mathbb{R})$.

For instance, let (X,D) = 

. The couple $(\{a\},\{a,d\})$ is consistent since $b \in D(a)$, $d \in D(a,b)$ and D(a,b,d) = D(a,d). On the contrary, one can verify that the couple $(\{a\},\{a,e\})$ is not consistent.

Real trace languages and complex trace languages are closely related. For any complex trace language, it holds:

$$L = \bigcup_{(A,B) \text{ consistent}} inf_A(Re(L_B))\cdot s_{A,B}$$

This formula will be mainly used to infer results on complex trace languages from results on real trace languages.

3. Recognizable languages

Recognizable languages describe the behaviour of finite state systems and hence form one of the basic families of a monoid. For finite traces and finite and infinite words, there exist several equivalent characterizations of the family of recognizable languages. These languages are those which are recognized by some morphism in a finite monoid. Kleene's, Büchi's and Ochmanski's theorems provide characterizations of recognizable languages by means of rational or co-rational expressions. Finite automata which provide the most intuitive idea of finite state machines can also be used to define recognizable languages. Finally the languages defined by some monadic second order logic formula are again the recognizable ones.

In this section, we introduce recognizable languages of complex traces using the first characterization above (recognizing morphisms). In Section 7, we prove that these languages are exactly the languages defined by some co-rational expression which asserts the robustness of our definition. A characterization by means of finite asynchronous (cellular) automata has also been proposed recently in [GP92]. Ebinger [Ebi92] obtained another characterization of these languages using monadic second order formulae

First of all, let us recall the definition of a recognizable word language: a language $U \subseteq X^\infty$ is recognizable if there exists a morphism η from X^* into a finite monoid S such that for any (finite or infinite) sequence $u_1,u_2,u_3\ldots$ of finite words we have:

$$u_1 \cdot u_2 \cdot u_3 \ldots \in U \Rightarrow \eta^{-1}\eta(u_1)\cdot\eta^{-1}\eta(u_2)\cdot\eta^{-1}\eta(u_3)\ldots \subseteq U$$

The family of recognizable word languages is denoted by Rec X^∞.
We can extend this definition to languages of infinite traces by considering a morphism from the monoid of finite traces $\mathbb{M}$ into a finite monoid. But a finite or infinite product of finite traces is always a real trace. Therefore this generalization can be used for real trace languages only.

Definition 3.1. A real trace language R is recognized by a morphism η from $\mathbb{M}$ into a finite monoid S if for any (finite or infinite) sequence $r_1,r_2,r_3\ldots$ of finite traces, we have:

$$r_1 \cdot r_2 \cdot r_3 \ldots \in R \Rightarrow \eta^{-1}\eta(r_1)\cdot\eta^{-1}\eta(r_2)\cdot\eta^{-1}\eta(r_3)\ldots \subseteq R$$

A real trace language is recognizable if it is recognized by some morphism. The family of recognizable real trace languages is denoted by Rec $\mathbb{R}$.

Several characterizations of recognizable real trace languages have been proposed in [Gas91]. In particular a notion of syntactic congruence has been defined and studied. But in this survey we only need the two properties below. The first one, along the lines of [Arn85] and [PP91], gives an extension of the known algebraic formulation of Büchi recognizability for ω-languages. This result will be used in a crucial way in Section 7.

Proposition 3.2 [Gas91]. Let R be a real trace language recognized by some morphism η from $\mathbb{M}$ into a finite monoid S. Then:

$$R = \bigcup_{(s,e) \in P} \eta^{-1}(s).(\eta^{-1}(e))^\omega \text{ with } P = \{(s,e) \in S^2 \mid se=s,\ e^2=e \text{ and } \eta^{-1}(s).(\eta^{-1}(e))^\omega \subseteq R\}.$$

For instance, for any subset A of X, the real trace language $\inf_A(\mathbb{R}) = \{t \in \mathbb{R} \mid \text{alphinf}(t) = A\}$ is recognized by the morphism $\eta : \mathbb{M} \to \mathcal{P}(X)$ defined by $\eta(r) = \text{alph}(r)$ for all r in $\mathbb{M}$. Moreover, the set P associated with $\inf_A(\mathbb{R})$ is simply $P = \{(B,A) \mid A \subseteq B\}$.
The second property shows that the families of recognizable real trace languages and of recognizable word languages are closely related (recall that φ is the canonical surjection from X^∞ onto $\mathbb{R}$).

Proposition 3.3 [Gas91]. Let R be a real trace language. Then:

$$R \in \text{Rec } \mathbb{R} \Leftrightarrow \varphi^{-1}(R) \in \text{Rec } X^\infty$$

From this characterization, we can easily deduce that the family Rec $\mathbb{R}$ is closed under finite union, intersection and complement.

It is possible to define the recognizability of a complex trace language by a morphism using an $(\omega + 1)$-product of finite traces. Here we use the following (equivalent) definition.

Definition 3.4. Let L be a complex trace language. The language L is recognizable if, for every $A \subseteq X$, the set of real traces $Re(L_A) = \{r \in \mathbb{R} / (r,D(A)) \in L\}$ is recognizable. The set of recognizable complex trace languages is denoted by Rec $\mathbb{C}$.

For instance, for any subset B of X, the complex trace language $\mathbb{C}_B = \{x \in \mathbb{C} / Im(x) = D(B)\}$ is recognizable since for any subset A of X, $(\mathbb{C}_B)_A$ is either empty if $D(A) \neq D(B)$ or the whole set $\mathbb{C}_B$ and $Re(\mathbb{C}_B) = \bigcup_{(A,B) \text{ consistent}} \inf_A(\mathbb{R})$.

Although this definition may seem artificial, we will prove its soundness throughout this paper. More precisely we will show that Rec $\mathbb{C}$ is a boolean algebra which is closed under concatenation and quotients (Section 4). In fact it is the unique family of complex trace languages which satisfies 1) and 2) below and which is closed under union, intersection and concatenation (with a finite trace). Moreover, the family of recognizable trace languages so-defined is exactly the family of co-rational trace languages (Section 7).

Remarks.

1) Let $L \in$ Rec $\mathbb{C}$, then $Re(L) = \bigcup_{A \subseteq X} Re(L_A) \in$ Rec $\mathbb{R}$.
2) Let R be a real trace language, then R is recognizable if and only if its image as complex trace language $L = \{(r,Im(r)) / r \in R\} \subseteq \mathbb{C}$ is recognizable since, for all subsets B of X, $Re(L_B) = \bigcup_{D(A)=D(B)} R \cap \inf_A(\mathbb{R})$

Let us point out that it is not possible to define recognizability via the projection $Re: \mathbb{C} \to \mathbb{R}$ saying that $L \in$ Rec $\mathbb{C}$ if and only if $Re(L) \in$ Rec $\mathbb{R}$. Namely if (X,D) = a — b — c — d and $L = \varphi((ac)^*b^\omega)\cdot\varphi(c) \cup \varphi((a+c)^*b^\omega)$, $Re(L) = \varphi((a+c)^*b^\omega)$ is a recognizable real trace language whereas $L_{\{b,c\}} = L \cap \mathbb{C}_{\{b,c\}}$ is not recognizable since $Re(L_{\{b,c\}}) = \varphi((ac)^*b^\omega)$. Therefore we would obtain a class of languages which were not closed under intersection.

Proposition 3.5 [DGP91]. Rec $\mathbb{C}$ is a boolean algebra, i.e., it is closed under union, intersection and complementation.

4. Concatenation.

In this section, we investigate the closure of the family of recognizable complex trace languages under concatenation, left and right quotients. To this purpose, we have to study the family Rec $\mathbb{R}$ and to introduce new operators on complex trace languages.
First, the family Rec $\mathbb{R}$ is closed under concatenation, left and right quotients.

Proposition 4.1 [Gas91, DGP91]. Let R and S be recognizable real trace languages. It holds:

i) $R \cdot S \in \mathrm{Rec}\,\mathbb{R}$ if $\mathrm{alphinf}(R) \times \mathrm{alph}(S) \subseteq I$

ii) $R \cdot S^{-1} = \{r \in \mathbb{R} / rs \in R \text{ for some } s \in S \text{ s.t. } \mathrm{alphinf}(r) \times \mathrm{alph}(s) \subseteq I\} \in \mathrm{Rec}\,\mathbb{R}$

iii) $R^{-1} \cdot S = \{s \in \mathbb{R} / rs \in S \text{ for some } r \in R \text{ s.t. } \mathrm{alphinf}(r) \times \mathrm{alph}(s) \subseteq I\} \in \mathrm{Rec}\,\mathbb{R}$

Note that the alphabetic conditions ensure that the concatenations remain real traces.

The proof of these results follows from the two following lemmas and the characterization of recognizable real trace languages by means of the canonical morphism φ (Proposition 3.3). First, we introduce the following operators on word languages. Recall that two words $u,v \in X^\infty$ are independent, denoted by $(u,v) \in I$, if $\mathrm{alph}(u) \times \mathrm{alph}(v) \subseteq I$. Let $U,V \subseteq X^\infty$, we define

$$U \sqcup\!\sqcup_I V = \{u_1v_1u_2v_2\ldots \in X^\infty / u_i, v_i \in X^* \text{ for all } i, \; u_1u_2\ldots \in U, v_1v_2\ldots \in V \text{ and } (v_i,u_j) \in I \text{ for all } i<j\}$$

$$U \sqcup\!\sqcup_I V^{-1} = \{u \in X^\infty / (u \sqcup\!\sqcup_I v) \cap U \neq \emptyset \text{ for some } v \in V\}$$

$$U^{-1} \sqcup\!\sqcup_I V = \{v \in X^\infty / (u \sqcup\!\sqcup_I v) \cap V \neq \emptyset \text{ for some } u \in U\}$$

As the following lemma shows , these operators are the word counterparts of the concatenation, left and right quotients in $\mathbb{R}$.

Lemma 4.2. Let R,S be real trace languages. Then

$$\varphi^{-1}(R \cdot S) = \varphi^{-1}(R) \sqcup\!\sqcup_I \varphi^{-1}(S)$$

$$\varphi^{-1}(R \cdot S^{-1}) = \varphi^{-1}(R) \sqcup\!\sqcup_I \varphi^{-1}(S)^{-1}$$

$$\varphi^{-1}(R^{-1} \cdot S) = \varphi^{-1}(R)^{-1} \sqcup\!\sqcup_I \varphi^{-1}(S)$$

Moreover, using classical closure properties of recognizable word languages, it can be proved that these operators preserve the recognizability of word languages:

Lemma 4.3. Let U, V be recognizable word languages. Then, $U \sqcup\!\sqcup_I V$, $U \sqcup\!\sqcup_I V^{-1}$ and $U^{-1} \sqcup\!\sqcup_I V$ are recognizable.

In order to extend these results to complex traces, we introduce the following operators. Let A,B be subsets of X and let L be a complex trace language. Define

$$\mathrm{Shift}_{A,B}(L) = \{y \in L / xy \in \complement_B \text{ for some } x \in \complement_A\}$$

$$\mu_{A,B}(L) = \mu_A\big(\mathrm{Re}(\mathrm{Shift}_{A,B}(L))\big)$$

Intuitively, a trace is in $\mathrm{Shift}_{A,B}$ if it shifts the imaginary part from D(A) to D(B) by right concatenation. Note that, in the definition of $\mathrm{Shift}_{A,B}$, the statement "for some $x \in \complement_A$" can be replaced by "for all $x \in \complement_A$" since the imaginary part of a product xy depends on Im(x) and y only.

For instance, let (X,D) = a — b — c — d, then we have :

$$\mathrm{Shift}_{\{a\},\{a,b\}}(\complement) = \{(r,\emptyset) / r \in \{a,c,d\}^*b\{a,b\}^*\} \cup \{(r,D(a)) / r \in \{a,c,d\}^*b\{a,b\}^*a^\omega\} \cup \{(r,D(a,b)) / r \in \{a,c,d\}^*b\{a,b\}^\omega\}$$

Therefore, $\mu_{\{a\},\{a,b\}}(\complement) = \{c,d\}^*$.

The following lemma states that these operators preserve the recognizability.

Lemma 4.4 [DGP91]. Let A,B be subsets of X and let L be a recognizable complex trace language. Then

i) $\mathrm{Shift}_{A,B}(L) \in \mathrm{Rec}\ \complement$
ii) $\mu_A(\mathrm{Re}(L)) \in \mathrm{Rec}\ \mathbb{R}$
iii) $\mu_{A,B}(L) \in \mathrm{Rec}\ \mathbb{R}$.

With these operators, it can be checked that the real part of the concatenation (resp. of the left and right quotients) of two complex trace languages is expressed in the following way.

Lemma 4.5. Let K and L be complex trace languages. The following formulae hold.

$$\mathrm{Re}((K\cdot L)_B) = \bigcup_{A\subseteq X} \mathrm{Re}(K_A)\cdot\mu_{A,B}(L)$$

$$\mathrm{Re}((K\cdot L^{-1})_A) = \bigcup_{B\subseteq X} (\mathrm{Re}(K_B)\cdot\mu_{A,B}(L)^{-1}) \cap \mathrm{Re}(\complement_A)$$

$$\mathrm{Re}((K^{-1}\cdot L)_C) = \bigcup_{A,B\subseteq X \,\&\, F\subseteq I(E)} \big((\mathrm{Re}(K_A)^{-1}\cdot\mathrm{Re}(L_B) \cap R_{A,E})\cdot S_{A,F}\big) \cap \mathrm{Re}(\mathrm{Shift}_{A,B}(\complement_C))$$

where, $R_{A,E} = \{r \in \mathbb{R} / \mathrm{alph}(r) \subseteq I(A) \,\&\, \mathrm{alphinf}(r) = E\}$ and $S_{A,F} = \overline{\bigcup_{a\in I(A)} a.\mathbb{R}} \cap \mathrm{alph}_F(\mathbb{R})$

is the set of real traces with no minimal vertex in I(A) and with alphabet F.

Using Proposition 4.1 and Lemmas 4.4 and 4.5, we can generalize Proposition 4.1 to complex traces.

Theorem 4.6 [DGP91]. The family Rec $\complement$ of recognizable complex trace languages is closed under concatenation, left and right quotients.

5. Connected traces.

The next problem, as regards the family Rec $\complement$, concerns its closure under iterations. It is not a surprise that, as for finite traces, the family Rec $\complement$ is not closed under iterations. For instance, let (X,D) = a — b — c. The languages $\{\varphi(ac)\}^*$ and $\{\varphi(ac),\varphi(b)\}^\omega$ are not recognizable since $\varphi^{-1}(\{\varphi(ac)\}^*) = \{u \in \{a,c\}^* / |u|_a = |u|_c\}$ and $\varphi^{-1}(\{\varphi(ac),\varphi(b)\}^\omega) \cap \{\varphi(a),\varphi(c)\}^*\varphi(b)^\omega = \{u \in \{a,c\}^* / |u|_a = |u|_c\}b^\omega$ which are not recognizable word languages.

The most widely known condition which ensures the recognizability of the iteration of a recognizable language of finite traces T is the connectivity of the traces in T [Mét86, Och85]. In order to generalize this result to complex traces, we define now the notion of connectivity for complex traces.

Intuitively, a trace is connected if it cannot be split into two non-empty independent traces. Thus we have first to introduce notions of independence. This is clear in $\mathfrak{G}$: two graphs f and g are independent, denoted by $(f,g) \in I$, if $\mathrm{alph}(f) \times \mathrm{alph}(g) \subseteq I$. Unfortunately, we cannot use this definition in $\complement$ since the alphabet of a complex trace is not well-defined (see Section 2.3). Nevertheless, it is easy to verify that the independence relation in $\mathfrak{G}$ factorizes through the congruence "practically indistinguishable". Hence the independence relation in $\complement$ is well-defined and it can be characterized as follows. Two complex traces $(r,D(A))$ and $(s,D(B))$ are independent if and only if $(\mathrm{alph}(r) \cup A) \times (\mathrm{alph}(s) \cup B) \subseteq I$.

For example, let (X,D) = , the complex traces $(gga^{\omega}, D(a))$ and $(d^{\omega},D(c,d))$ are independent whereas the traces $(gga^{\omega}, D(a,b))$ and $(d^{\omega},D(c,d))$ are not. We define now connected traces and connected components of a trace.

Definition 5.1. A complex trace z is said to be connected if $z = xy$ with $(x,y) \in I$ implies $x = 1$ or $y = 1$. A trace x is called a connected component of z if either $x = z = 1$ or x is a connected non-empty trace such that $z = xy$ for some complex trace y with $(x,y) \in I$. The set of connected components of z is denoted by C(z).

For instance, with the dependence alphabet defined above, the trace $x = (gg(ad)^{\omega}, X\backslash\{g\})$ is not a connected trace since $x = (gga^{\omega}, D(a))\cdot(d^{\omega},D(c,d))$ is the product of two independent traces. The connected components of x are $y = (gg,\varnothing)$, $y_1 = (a^{\omega},D(a))$, $z_1 = (d^{\omega},D(c,d)\})$, $y_2 = (a^{\omega},D(a,b))$ and $z_2 = (d^{\omega},D(d))$.

Note that using a direct translation of Definition 5.1 we can define the connectivity notions on dependence graphs as well. The following remarks explain why we chose these formulations for the definitions of connectivity although they may seem different from the usual ones. First, one can check that a finite trace t is connected if and only if the graph (X,D) restricted to alph(t) is connected. But, this *usual* characterization of connectivity for finite traces cannot be used to define the connectivity in $\complement$ since the alphabet of a complex trace is not well-defined. Second, one can verify that a dependence graph is connected if and only if so is its underlying undirected graph. Again, this *usual* definition of connected dependence graphs cannot either be used to define connectivity on $\complement$ since a complex trace may have connected and non-connected representatives. For instance, let (X,D) = a — b — c. The complex trace $((ac)^{\omega},X)$ is represented by the non-connected dependence graph $\varphi((ac)^{\omega})$ and by the connected one $\varphi((ac)^{\omega})\cdot\varphi(b)$.

Every dependence graph in $\mathfrak{G}$ is the disjoint union of its connected components. It follows that every graph g in $\mathfrak{G}$ (in particular any real trace) can uniquely be written as a product of independent connected components (called a decomposition of g). Moreover, if $g = g_1\ldots g_k$ is such a decomposition then $\{g_1,\ldots,g_k\}$ is exactly the set C(g) of connected components of g. This is not true for traces in $\complement$. A complex trace may have several decompositions and some of its connected components may not be independent. Nevertheless, every complex trace admits at least one decomposition.

For instance, the trace x defined above admits two decompositions $x = yy_1z_1 = yy_2z_2$ but neither (y_1,y_2) nor (z_1,z_2) nor (z_1,y_2) are independent. Note that, $\{y,y_1,z_2\}$ is a maximal set of pairwise independent components of x but $yy_1z_2 = (gg(ad)^{\omega}, X\backslash\{e,g\}) \neq x$.

The notation C is extended to complex trace languages in the obvious way. A trace language L is said to be connected if L = C(L). The following inclusions hold: $L^* \subseteq C(L)^*$, $L^{\omega} \subseteq C(L)^{\omega}$. We will see in the next section that if L is recognizable then so are $C(L)^*$ and $C(L)^{\omega}$. For this, we need the following lemma.

Lemma 5.2 [DGP91]. Let L be a recognizable complex trace language. Then the language C(L) is recognizable too.

6. Iterations

The aim of this section is to find sufficient conditions on a recognizable complex trace language L which ensure that L^* and L^ω are recognizable too. We prove that connectivity is such a condition. Again in this section, results on real traces are basic. Hence we first study the infinite iterations of a language of finite traces. As a first step, we will restrict ourselves to languages where all elements have the same connected alphabet. Recall that a subset A of X is connected if the restriction of (X,D) to A is connected.

Proposition 6.1 [GPZ91]. Let A be a connected subset of X and let T be a recognizable language of finite traces such that alph(t) = A for any t in T. Then T^ω is a recognizable real trace language.

It is already known that under the hypotheses of Proposition 6.1, T^* is a recognizable language of finite traces [Mét86, Och85]. In order to extend this result to the infinite iteration, we first reduce to the case where all the traces of T have the same alphabet. For any subset $A \subseteq X$, we denote $\mathrm{alph}_A(T) = \{t \in T \,/\, \mathrm{alph}(t) = A\}$.

Lemma 6.2 [GPZ91]. Let T be a language of finite traces such that $T^* = T$. Then,

$$T^\omega = \bigcup_{A \subseteq X} T \cdot (\mathrm{alph}_A(T))^\omega$$

In order to use Proposition 6.1, we reduce once more to the case where all traces in T have the same *connected* alphabet.

Lemma 6.3 [GPZ91]. Let A be a subset of X, let $A_1,\ldots,A_k$ be the connected components of A and let T be a language of finite traces such that T^* is recognizable and alph(t) = A for all t in T. Then T^ω is a finite union of sets of the form $RS_1^\omega \ldots S_k^\omega$ where $R,S_1,\ldots,S_k$ are in Rec IM and $\mathrm{alph}(t) = A_i$ for all t in S_i.

We are now ready to give the main result on the infinite iteration of languages of finite traces. It claims that any sufficient condition ensuring the recognizability of T^* holds also for T^ω.

Theorem 6.4 [GPZ91]. Let T be a language of finite traces such that T^* is recognizable. Then T^ω is a recognizable real trace language.

From this theorem and the result of Métivier and Ochmanski [Mét86, Och85], we deduce immediately:

Corollary 6.5 [GPZ91]. Let T be a recognizable language of connected finite traces. Then T^ω is a recognizable real trace language.

Now we come back to complex traces. As announced, we will show that L^* and L^ω are recognizable provided L is a recognizable language of connected complex traces.

For a complex trace language L, define $\nu_A(L) = \mu_{A,A}(L)$. Note that each trace in $\nu_A(L)$ is finite since for any real trace r, we have alph(r) $\subseteq$ I(A) and Im(r) $\subseteq$ D(A) imply alphinf(r) = $\emptyset$.

Lemma 6.6 [DGP91]. Let L be a complex trace language, then $L^\dagger$ (with $\dagger$ = * or ω) is a finite union of languages $\nu_{B_0}(L)^*\mathrm{Shift}_{B_0,B_1}(L)\nu_{B_1}(L)^*\ldots\mathrm{Shift}_{B_{k-1},B_k}(L)\nu_{B_k}(L)^\dagger$ where B_0, B_1,…, B_k is a sequence of subalphabets such that $\emptyset = D(B_0) \subsetneq D(B_1) \subsetneq \ldots \subsetneq D(B_k) \subseteq X$.

Hence, from Lemma 4.4, Theorem 6.4 and the closure properties of Rec ¢, we obtain:

Theorem 6.7 [DGP91]. Let L be a recognizable complex trace language such that $\nu_A(L)^*$ is recognizable for all $A \subseteq X$. Then L^* and L^ω are both recognizable.

In order to generalize Corollary 6.5 to complex trace languages, we need the following lemma:

Lemma 6.8 [DGP91]. Let L be a connected complex trace language, then $\nu_A(L)$ is connected for any subset A of X.

Now using this lemma, the result of Métivier and Ochmanski and Theorem 6.7, we get:

Theorem 6.9 [DGP91]. Let L be a recognizable complex trace language. If L contains connected traces only then L^* and L^ω are recognizable too. In particular, $C(L)^*$ and $C(L)^\omega$ are recognizable for all recognizable complex trace language L.

As shown in the first example of Section 5, the finite iteration of a single trace is not necessarily recognizable. For instance, let (X,D) = a — b — c, the trace language $\{ac\}^*$ is not recognizable (see Section 5). However the real trace language $\{ac\}^\omega$ is recognized by the morphism $\eta : \mathbb{IM} \to \mathcal{P}(X)$ defined by $\eta(r)$ = alph(r) for all r in $\mathbb{IM}$. The set P associated with $\{ac\}^\omega$ (see Proposition 3.2) is simply the singleton $P = \{(\{a,c\},\{a,c\})\}$. This surprising result can be generalized to any complex trace, as stated in the following proposition.

Proposition 6.10 [DGP91]. Let x be a complex trace such that the language L = {x} is recognizable. Then the language $L^\omega = \{x^\omega\}$ is recognizable too.

This result fails as soon as the language L contains at least two traces. For instance, if (X,D) = a — b — c, the language $\{ac,b\}^\omega$ is not recognizable (see Section 5).

7. Canonical form and co-rational languages

In this section, we first propose a characterization of recognizable languages of infinite traces in terms of appropriate "rational operations". It extends to traces the Büchi characterization of recognizable word languages by means of a finite union of sets of the form UV^ω where $U,V \in$ Rec X^*. This charaterization provides a "canonical" form for recognizable languages of infinite traces.

Theorem 7.1 [GPZ91, DGP91]. A complex trace language L is recognizable if and only if L is a finite union of languages $RS_1^\omega\ldots S_k^\omega.s$ where s is a finite trace, R, $S_i \in$ Rec $\mathbb{IM}$ are such that all traces in S_i have the same connected alphabet A_i and $A_i \times A_j \subseteq I$ for all $i \neq j$. Moreover, if $L \subseteq \mathbb{IR}$ then we may assume that the finite trace s is empty.

Apart from the family of recognizable sets, the family of rational sets is also a basic family in a monoid since it collects languages which can be generated in a modular way starting from actions and using rational operators. Therefore, the study of the relationships between the recognizable and the rational families is of first importance.
Kleene's theorem states the equality of these families in the free monoid X^*. It has been generalized to infinite words by Büchi and to finite traces by Ochmanski (using co-rationals instead of rationals). Our aim is to suppress the question mark in the diagram below.

	Words	Traces
Finite	Kleene	Ochmanski
Infinite	Büchi	?

As for finite traces [Och85], define the concurrent iterations co-$*$ and co-ω of a trace language L by $L^{co\text{-}*} = C(L)^*$ and $L^{co\text{-}\omega} = C(L)^\omega$.
Let the co-rational family Co-Rat $\mathbb{C}$ be the smallest family of trace languages containing finite sets of finite traces and closed under union, concatenation, co-star and co-omega.
From Proposition 3.5 and Theorems 4.6 and 6.9 for one inclusion and from Theorem 7.1 using Ochmanski's result Rec $\mathbb{M} \subseteq$ co-Rat $\mathbb{M}$ [Och85] for the converse, we obtain:

Theorem 7.2 [DGP91]. Rec $\mathbb{C}$ = co-Rat $\mathbb{C}$.

Finally, note that the family Rec $\mathbb{C}$ is the smallest family of complex trace languages which contains finite sets of finite traces and which is closed under finite union, concatenation and iterations $*$ and ω restricted to languages of finite connected traces.

8. Conclusion

Recognizable languages describe the behaviour of finite state systems and hence form one of the basic families. For infinite traces, we defined this family using morphisms into finite monoids. Its closure properties under rational operations have been investigated. Then we gave a characterization of this family by means of co-rational expressions.
But, the characterization of recognizable languages which provides the most intuitive idea of finite state machines uses finite automata (see [Eil74, Tho90, PP91] for automata on infinite words and [Zie87, Zie89, CMZ89] for asynchronous automata on finite traces). For infinite traces, such a characterization has been recently proposed by means of (non-deterministic) Büchi asynchronous (cellular) automata [GP92]. On the other hand, Ebinger [Ebi92] obtained a characterization of these languages using monadic second order formulae (see also [Tho89] for finite traces and [Tho90] for words).

Acknowledgement: *We thank the four anonymous referees for many comments and suggestions about the presentation of this survey.*

9. References

[Arn85] A. ARNOLD, “A syntactic congruence for rational ω-languages”, Theoretical Computer Science 39, p. 333-335, 1985.

[AR88] I.J. AALBERSBERG and G. ROZENBERG, “Theory of traces”, Theoretical Computer Science 60, p. 1-82, 1988.

[BD87] E. BEST and R. DEVILLERS, “Sequential and concurrent behaviour in Petri Net theory”, Theoretical Computer Science 55, p. 87-136, 1987.

[BMP89] P. BONIZZONI, G. MAURI and G. PIGHIZZINI, “About infinite traces”, Proceedings of the ASMICS Workshop on Partially Commutative Monoids, Tech. Rep. TUM-I 9002, Technische Universität München, p. 1-10, 1989.

[CF69] P. CARTIER and D. FOATA, “Problèmes combinatoires de commutation et réarrangements”, Lecture Notes in Mathematics 85, 1969.

[CM85] R. CORI and Y. METIVIER, “Recognizable subsets of some partially abelian monoids”, Theoretical Computer Science 35, p. 179-189, 1985.

[CMZ89] R. CORI, Y. METIVIER and W. ZIELONKA, “Asynchronous mappings and asynchronous cellular automata”, Tech. Rep. 89-97, LaBRI, Université de Bordeaux, France, 1989.

[CP85] R. CORI and D. PERRIN, “Automates et commutations partielles”, RAIRO Theoretical Informatics and Applications 19, p. 21-32, 1985.

[DGP91] V. DIEKERT, P. GASTIN, A. PETIT, “Recognizable complex trace languages”, MFCS’91, Lecture Notes in Computer Science 520, p. 131-140, 1991.
Also available as Technical Report , LRI, Université de Paris-Sud, France.

[Die90] V. DIEKERT, “Combinatorics on traces”, Lecture Notes in Computer Science 454, 1990.

[Die91] V. DIEKERT, “On the concatenation of infinite traces”, STACS’91, Lecture Notes in Computer Science 480, p. 105-117, 1991.

[Ebi92] W. EBINGER, “On logical definability of infinite trace languages”, Proceedings of the ASMICS Workshop on Infinite Traces, to appear as Tech. Rep., Universität Stuttgart, 1992.

[Eil74] S. EILENBERG, “Automata, Languages and Machines”, Academic Press, New York, 1974.

[Fli74] M. FLIESS, “Matrices de Hankel”, J. Math. pures et appl. 53, p. 197-224, 1974.

[FR85] M.P. FLE and G. ROUCAIROL, “Maximal serializability of iterated transactions”, Theoretical Computer Science 38, p. 1-16, 1985.

[Gas90a] P. GASTIN, “Un modèle asynchrone pour les systèmes distribués”, Theoretical Computer Science 74, p. 121-162, 1990.

[Gas90b] P. GASTIN, “Infinite traces”, Proceedings of the Spring School of Theoretical Computer Science on “Semantics of concurrency”, Lecture Notes in Computer Science 469, 1990.

[Gas91] P. GASTIN, “Recognizable and rational languages of finite and infinite traces”, STACS’91, Lecture Notes in Computer Science 480, p. 89-104, 1991.

[GPZ91] P. GASTIN, A. PETIT, W. ZIELONKA, “A Kleene theorem for infinite trace languages”, ICALP’91, Lecture Notes in Computer Science 510, p. 254-266,1991.
Also available as Technical Report , LRI, Université de Paris-Sud, France.

[GP92] P. GASTIN and A. PETIT, “Asynchronous cellular automata for infinite traces”, ICALP’92, Proceedings to appear in Lecture Notes in Computer Science. Also available as Tech. Rep. 91-68, LITP, Université Paris 6, France, 1991.

[Gra81] R. L. GRAHAM, “Rudiments of Ramsey theory ”, Regional conference series in mathematics 45, 1981.

[GR91] P. GASTIN and B.ROZOY, “The Poset of infinitary traces”, to appear in Theoretical Computer Science, also available as Technical Report 91-07, LITP, Université Paris 6, France, 1991.

[Kwi90] M.Z. KWIATKOWSKA, “A metric for traces”, Information and Processing Letters 35, p. 129-135, 1990.

[Kwi91] M.Z. KWIATKOWSKA, “On the domain of traces and sequential composition”, CAAP'91, Lecture Notes in Computer Science 493, p. 42-56,1991.

[Maz77] A. MAZURKIEWICZ, “Concurrent program Schemes and their interpretations”, Aarhus University, DAIMI Rep. PB 78, 1977.

[Maz86] A. MAZURKIEWICZ, “Trace theory”, Advanced Course on Petri Nets, Lecture Notes in Computer Science 255, p. 279-324, 1986.

[Mét86] Y. METIVIER, “On recognizable subsets in free partially commutative monoids”, ICALP’86, Lecture Notes in Computer Science 226, p. 254-264, 1986.

[MR91] Y. METIVIER and B. ROZOY, “On the star operation in free partially commutative monoid”, to appear in International Journal of Foundations of Computer Science, also available as Technical Report, Université de Paris-Sud, France, January 1991.

[Och85] E. OCHMANSKI, “Regular behaviour of concurrent systems”, Bulletin of EATCS 27, p. 56-67, October 1985.

[Och90] E. OCHMANSKI, “Notes on a star mystery”, Bulletin of EATCS 40, p. 252-257, February 1990.

[Per89] D. PERRIN, “Partial commutations”, ICALP’89, Lecture Notes in Computer Science 372, p. 637-651, 1989.

[PP91] D.PERRIN and J.E. PIN, “Mots Infinis”, Tech. Rep. 91-06, LITP, Université Paris 6, France, 1991. Book to appear.

[Roz90] B. ROZOY, “On Traces, Partial Order Sets and Recognizability”, Proceedings of ISCIS V, Cappadocia, Turkey, 1990.

[Sak87] J. SAKAROVITCH, “On regular trace languages”, Theoretical Computer Science 52, p. 59-75, 1987.

[Tho89] W. THOMAS, “On logical definability of trace languages”, Proceedings of the ASMICS Workshop on Partially Commutative Monoids, Tech. Rep. TUM-I 9002, Technische Universität München, p. 172-182, 1989.

[Tho90] W. THOMAS, “Automata on infinite objects”, Handbook of Theoretical Computer Science (J.V. Leeuwen, Ed.), Elsevier, Amsterdam, Vol. B, p. 135-191, 1990.

[Zie87] W. ZIELONKA, “Notes on finite asynchronous automata and trace languages”, RAIRO Theoretical Informatics and Applications 21, p. 99-135, 1987.

[Zie89] W. ZIELONKA, “Safe execution of recognizable trace languages by asynchronous automata”, Lecture Notes in Computer Science 363, p. 278-289, 1989.