A. BERTONI
M. GOLDWURM

On ranking 1-way finitely ambiguous NL languages and $\#P_1$-complete census functions

Informatique théorique et applications, tome 27, nº 2 (1993), p. 135-148
<http://www.numdam.org/item?id=ITA_1993__27_2_135_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 27, n° 2, 1993, p. 135 à 148)

ON RANKING 1-WAY FINITELY AMBIGUOUS NL LANGUAGES AND $\sharp P_1$-COMPLETE CENSUS FUNCTIONS (1) (*)

by A. BERTONI (2) and M. GOLDWURM (2)

Communicated by J. BERSTEL

Abstract. – *We show that the ranking problem for languages accepted by one-way finitely ambiguous nondeterministic log-space bounded Turing machines* (1-*FANL*) *belongs to the class DET and hence lies in* NC^2. *This property is also extended to another class of languages including the boolean closure of* 1-*FANL. The proof is based on the use of Kronecker product and direct sum over matrices which allow to represent the computations of such a model of Turing machine. Moreover we study the counting problem for certain classes of languages whose ranking is known to be* $\sharp P$-*complete and prove that the corresponding census function is complete for the class* $\sharp P_1$.

Résumé. – *Nous étudions le problème du calcul du rang pour les langages reconnus par des machines de Turing à déplacement vers la droite, finement ambigües, et à complexité en place logarithmiquement bornée* (1-*FANL*). *Nous prouvons qu'il appartient à la classe DET, et est donc dans* NC^2.

Nous étendons cette propriété à une autre classe de langages qui contient la fermeture de Boole de la classe 1-*FANL. La démonstration est basée sur le produit de Kronecker et la somme directe des matrices qui permettent de représenter les calculs de ce modèle de machines de Turing. En outre, nous étudions le problème de compter les mots de longueur donnée pour quelques classes de langages pour lesquels le calcul du rang est* $\sharp P$-*complet, et nous prouvons que la fonction de cens correspondante est complète pour la classe* $\sharp P_1$.

1. INTRODUCTION

In this work we study the complexity of counting and ranking problems for certain classes of languages. The ranking problem for formal languages

(*) Received November 1991, accepted June 1992.
(1) This research has been supported by C.E.C. contract n. 3166 (Working Group ASMICS) and by MURST (Ministero dell'Università e della Ricerca Scientifica) Project 40% "Algoritmi, modelli di calcolo e strutture informative".
(2) Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Via Comelico 39, 20135 Milano, Italy.

is defined as follows. Given a finite alphabet Σ and a total order $\leqq$ over Σ, let $\leqq_{\text{lex}} \subseteqq \Sigma^* \times \Sigma^*$ be its lexicographic extension: for every $x, y \in \Sigma^*$, $x \leqq_{\text{lex}} y$ iff either $xz = y$ for some $z \in \Sigma^*$, or $x = waz$, $y = wbz'$, where $a, b \in \Sigma$, $a \neq b$, $a \leqq b$ and $w, z, z' \in \Sigma^*$. The rank function of a language $L \subseteqq \Sigma^*$, is the function $\text{rank}_L : \Sigma^* \to \mathbb{N}$ such that, for every $x \in \Sigma^*$, $\text{rank}_L(x) = \# \{ y \in L / |y| < |x| \text{ or } (|x| = |y|, y \leqq_{\text{lex}} x) \}$, where $\# A$ denotes the cardinality of a set A.

The complexity of computing the rank function has been studied in [8] where it has been considered as a special kind of optimal compression: ranking is in fact related to the more general problem of storing and retrieving strings efficiently. It is easy to see that, for any language L in P, rank_L belongs to the class $\# P$ of functions yielding the number of accepting computations in nondeterministic polynomial time Turing machines [15]. Since such a class contains highly intractable problems a general goal is to determine which languages in P have rank function complete for $\# P$ and, on the other side, which ones are rankable in polynomial time or even in NC^k for some k.

As proved in [9], the ranking problem turns out to be $\# P$-complete for several classes of simple languages contained in P. More precisely for some finitely ambiguous context-free languages L, rank_L is $\# P$-complete and the same holds for languages accepted by the following machines: *a*) nondeterministic log-time bounded Turing machines (TM), *b*) log-space bounded deterministic TM, *c*) 1-way nondeterministic log-space bounded TM, *d*) uniform families of constant depth and polynomial size unbounded fan-in circuits, *e*) CRCW P-RAM working in constant time with a polynomial number of processors, *f*) 2-way deterministic checking stack automata, *g*) 2-way deterministic pushdown automata, *h*) 1-way 2-head deterministic finite state automata.

On the other hand the ranking problem for regular languages is NC^1-reducible to integer division [2, 10], and hence it can be solved by log-space uniform boolean circuits of $O(\log n \log \log n)$ depth and polynomial size [7]. Moreover, in [8] it is proved that, for any language L accepted by 1-way unambiguous log-space bounded TM, rank_L belongs to the class DET of all functions NC^1-reducible to computing the determinant of an integer matrix [7]. Finally the problem is in NC^2 for all languages accepted in polynomial time by 1-way unambiguous (log-space) auxiliary pushdown automata [9] and hence in particular for unambiguous context-free languages.

Other results concerning the complexity of ranking have been obtained for the uniform version of the problem, *i.e.* when a suitable representation of the language is considered as part of the input [1]. In this case the problem is "difficult" even for regular languages represented by nondeterministic finite automata.

In this work we extend the previous results obtaining an NC^2 algorithm for the rank function of languages accepted by 1-way nondeterministic log-space bounded TM with bounded ambiguity degree (1-FANL). We recall that a nondeterministic Turing machine has bounded ambiguity degree iff it has at most k accepting computations on every input for a suitable $k \in \mathbb{N}$. More precisely we prove that the rank functions of 1-FANL languages belong to the class DET and hence lie in NC^2. To prove this result we develop a sort of abstract calculus of formal expressions based on Kronecker product and direct sum for manipulating integer matrices. The idea of applying such matrix operations to transform finitely ambiguous computations into unambiguous ones has already been used in [11] to study inclusion and equivalence problems for (languages recognized by) n-ambiguous finite automata and the problem of deciding whether a finite automaton has a fixed ambiguity degree.

We also prove that the ranking problem belongs to DET for another class of languages including the boolean closure of 1-FANL. Such a class is defined as the set of all languages L such that there exists a positive integer k, a subset $A \subseteq \{0, 1, \ldots, k\}$ and a 1-FANL Turing machine M of ambiguity degree k, such that $L = \{x / \# \mathrm{acc}_M(x) \in A\}$, where $\# \mathrm{acc}_M(x)$ denotes the number of accepting computations of M on input x. We prove that such a class is closed under boolean operations and hence it appears to be larger than 1-FANL which we conjecture not to be closed under complement.

Other results we present in this work concern the complexity of computing the number of words of given length in a fixed language. In several significant cases, such a problem is related to the enumeration of combinatorial objects and to the algebraic properties of generating functions [4].

Given a language $L \subseteq \Sigma^*$, the counting function of L is the function $f_L : \{1\}^* \to \mathbb{N}$ such that $f_L(1^n) = \# (\Sigma^n \cap L)$ for every $n \in \mathbb{N}$. We note that f_L is NC^1-equivalent to the census function of L defined by $c_L : \{1\}^* \to \mathbb{N}$, such that $c_L(1^n) = \# \{x \in L / |x| \leq n\}$ for every $n \in \mathbb{N}$. It is easy to see that f_L is reducible to rank_L for any language L, and hence rank_L is at least as hard as f_L. On the other hand there exist undecidable languages whose counting function is computable in polynomial time while clearly the corresponding rank function is not recursive.

However, the complexity of these two problems have some analogies. First of all, for any language L in P it is easy to verify that f_L belongs to $\# P_1$ which is the restriction of $\# P$ to the functions having unary inputs. We recall that $\#P_1$ was introduced by Valiant who proposed the class of functions complete for $\#P_1$ as a representative class of difficult enumeration problems [15]. Moreover, it is known that, for some context-free language L of ambiguity degree 2, f_L is complete for $\#P_1$ with respect to a polynomial Turing reducibility with one oracle call ($\#_1$-reducibility) [4]. On the other hand the counting function of any unambiguous context-free language is NC^1-reducible to integer division [3]. This yields a significant example of a class of languages for which counting seems to be easier than ranking.

In this work we extend to the counting function (and hence also to the census function) the completeness results obtained in [9] for the ranking problem. More precisely we prove that f_L is $\#P_1$-complete, with respect to $\#_1$-reducibility, for languages L accepted by the machines

a), *b*), *c*), *d*), *e*), *f*), *g*), *h*)

mentioned above. As in [4] and [9], the proof is based on the idea of codifying the accepting computations of a counting Turing machines by words of equal length in a suitable language. These results provide new examples of enumeration problems complete for $\#P_1$ and seem to suggest that, for most natural classes of languages, a difficult ranking problem implies a difficult counting problem.

2. OPERATIONS OVER MATRICES

The operations of direct sum and Kronecker product over matrices are classical notions of linear algebra [12]. Here we recall the definitions with some basic properties and prove further elementary identities used in the following sections. The standard operations over arrays (matrices and vectors), as sum, product and product by a scalar, are assumed to be known.

Let $A=[a_{ij}]$ and $B=[b_{ij}]$ be two matrices of size n and m respectively. The direct sum and the (right) Kronecker product of A and B are defined by

$$A \oplus B=\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}, \qquad A \otimes B=\begin{pmatrix} a_{11}B & a_{12}B\dots & a_{1n}B \\ a_{21}B & a_{22}B\dots & a_{2n}B \\ \dots & \dots & \dots \\ a_{n1}B & a_{n2}B\dots & a_{nn}B \end{pmatrix}$$

It is easily seen that $A \oplus B$ and $A \otimes B$ are square matrices of size $n+m$ and nm respectively. Analogously, given two row-vectors $\pi=(a_1, a_2, \dots, a_n)$,

$\eta=(b_1, b_2, \ldots, b_m)$, we define $\pi \oplus \eta=(a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m)$ and $\pi \otimes \eta=(a_1 \eta, a_2 \eta, \ldots, a_n \eta)$. Similarly, denoting by A^T the transposed of an array A, the direct sum and the Kronecker product of two column-vectors π^T and η^T are defined as the transposed of $\pi \oplus \eta$ and $\pi \otimes \eta$ respectively. Note that the direct sum of two arrays $\pi=(a)$ and $\eta=(b)$, each having one entry, yields different arrays according whether π and η are considered as square matrices, row-vector or column-vector: the operands will be clear from the context.

It is easy to verify that the direct sum and the Kronecker product are associative and hence the Kronecker power of an array A is defined as usual: $A^{(1)}=A$, $A^{(k)}=A \otimes A^{(k-1)}$ for every integer $k>1$. Analogously, for every finite sequence of matrices (row-vectors, column-vectors resp.) $A_1, A_2, \ldots, A_k$, the sum

$$\bigoplus_{i=1}^{k} A_i=A_1 \oplus A_2 \oplus \ldots \oplus A_k$$

is well defined. Moreover it is easy to verify that $\otimes$ is distributive over $\oplus$:

$$(A \oplus B) \otimes C=(A \otimes C) \oplus (B \otimes C),\ C \otimes (A \oplus B)=(C \otimes A) \oplus (C \otimes A).$$

The following proposition relates the direct sum and the Kronecker product to the traditional operations over arrays. The proof follows from the definitions using standard calculation and can be found in [12] (Section 43).

LEMMA: *Let A and B be two $n \times n$ matrices and let π and η be two row-vectors of size n. Analogously let C and D be two $m \times m$ matrices and let λ and μ be two row-vectors of size m. Then the following equalities hold:*

a) $AB \oplus CD=(A \oplus C)(B \oplus D)$;

b) $AB \otimes CD=(A \otimes C)(B \otimes D)$;

c) $\pi A \eta^T+\lambda C \mu^T=(\pi \oplus \lambda)(A \oplus C)(\eta^T \oplus \mu^T)$;

d) $(\pi A \eta^T)(\lambda C \mu^T)=(\pi \otimes \lambda)(A \otimes C)(\eta^T \otimes \mu^T)$.

Moreover, for every scalar a, we have:

$$a(A \oplus C)=(aA) \oplus (aC),$$
$$a(A \otimes C)=(aA) \otimes C=A \otimes (aC).$$

Let $p(z)=a_1 z+a_2 z^2+\ldots+a_k z^k$ be a polynomial with null coefficient of degree 0 and let $\alpha(p) \subseteq \{1, 2, \ldots, k\}$ be the set of indices of the nonnull coefficients of $p: \alpha(p)=\{i \in \mathbb{N}/a_i \neq 0\}$. Then, for any $n \times n$ matrix A, we can define the square matrices $\hat{p}(A)$ and $\tilde{p}(A)$ given by the formal expressions

$$\hat{p}(A) = \bigoplus_{i \in \alpha(p)} a_i A^{(i)}, \qquad \tilde{p}(A) = \bigoplus_{i \in \alpha(p)} A^{(i)}.$$

Note that both these matrices have size $m(p) = \sum_{i \in \alpha(p)} n^i$.

Further, for any row-vector π of size n, the row-vectors $\hat{p}(\pi)$ and $\tilde{p}(\pi)$ are defined in the same way, and similar obvious definitions can be given for any column-vector.

PROPOSITION 1: *Let A be a $n \times n$ matrix, let π be a row-vector of size n and let η be a column vector of the same size. Then for every polynomial $p(z)$ such that $p(0) = 0$ we have $p(\pi A \eta) = \hat{p}(\pi)\tilde{p}(A)\tilde{p}(\eta)$.*

Proof: We prove the proposition by induction on the degree of p. If p has degree one the result is immediate. If p has degree $k > 1$ let a be the coefficient of degree one in p and assume $a \neq 0$. Then $p(z) = az + zq(z)$, where $q(z)$ is a polynomial of degree $k-1$ and $q(0) = 0$. Hence, by induction hypothesis and points *c*), *d*) of the previous lemma, we obtain

$$\begin{aligned} p(\pi A \eta) &= a(\pi A \eta) + (\pi A \eta)(\hat{q}(\pi)\tilde{q}(\mathrm{A})\tilde{q}(\eta)) \\ &= a(\pi A \eta) + (\pi \otimes \hat{q}(\pi))(A \otimes \tilde{q}(A))(\eta \otimes \tilde{q}(\eta)) \\ &= (a\pi \oplus \pi \otimes \hat{q}(\pi))(A \oplus A \otimes \tilde{q}(A))(\eta \oplus \eta \otimes \tilde{q}(\eta)) \\ &= \hat{p}(\pi)\tilde{p}(A)\tilde{p}(\eta). \end{aligned}$$

The case $a = 0$ is similar.

PROPOSITION 2: *Let $p(z)$ be a polynomial with integer coefficients such that $p(0) = 0$ and let A and B be two square matrices of size n. Then*

$$\tilde{p}(AB) = \tilde{p}(A)\tilde{p}(B).$$

Proof: Clearly we may assume that the coefficients of p are in $\{0, 1\}$. We prove the result by induction on the degree of p. If $p(z) = z$ then the property is immediate. Assume that $p(z)$ has degree $k > 1$ and let $p(z) = z + zq(z)$ for a polynomial $q(z)$ of degree $k-1$, with coefficients in $\{0, 1\}$, such that $q(0) = 0$; then by inductive hypothesis and points *a*), *b*) of the previous lemma we obtain

$$\tilde{p}(AB)=AB\oplus(AB)\otimes(\tilde{q}(A)\tilde{q}(B))=AB\oplus(A\otimes\tilde{q}(A))(B\otimes\tilde{q}(B))$$
$$=(A\oplus A\otimes\tilde{q}(A))(B\oplus B\otimes\tilde{q}(B))=\tilde{p}(A)\tilde{p}(B).$$

The case $p(z)=zq(z)$ is similar.

3. RANKING 1-WAY FINITELY AMBIGUOUS NL LANGUAGES

For any $k\in\mathbb{N}$ let 1-NL(k) denote the class of all languages accepted by 1-way nondeterministic log-space bounded Turing machines having ambiguity degree k. We prove that ranking languages in such a class belongs to DET. We recall that DET is the class of all functions NC^1-reducible to computing the determinant of a $n\times n$ integer matrix. This class is widely studied since it contains many problems of matrix calculation, like computing the n-th power of a $n\times n$ matrix with integer entries of n-bits, or computing the iterated product of n matrices of the same king [7].

PROPOSITION 3: *For every $k\in\mathbb{N}$ and every language $L\in$ 1-$NL(k)$, $rank_L$ belongs to DET.*

Proof: We apply the analysis presented in the previous section to suitable expressions representing computations of (one way) finitely ambiguous *NL* machines. A similar reasoning is used in [11] (Sec. 4) to transform finitely ambiguous automata into unambiguous $\mathbb{Z}$-automata.

Let $L\subseteq\Sigma^*$ be a language in 1-NL(k) and let M be a 1-way NL Turing machine of ambiguity k which recognizes L. Since M works in logarithmic space, it has at most $q(n)$ configurations on inputs of size n for a suitable polynomial q. Without loss of generality we may assume that, during any computation on input of size n, M executes at most $q(n)$ consecutive steps without moving the input head one cell right and it stops just after moving the input head one cell right to the n-th input symbol. Hence for every integer n and every $\sigma\in\Sigma$ we define the square matrices $M(\sigma)=[m_{ij}]$, $R(\sigma)=[r_{ij}]$ of size $q(n)$ such that:

$m_{ij}=1$ if M in configuration i, reading σ on the input tape, can move to configuration j in one step shifting the input head one cell right;

$m_{ij}=0$ otherwise;

$r_{ij}=1$ if M in configuration i, reading σ on the input tape, can move to configuration j in one step without shifting the input head one cell right;

$r_{ij}=0$ otherwise.

Moreover, for any $\sigma \in \Sigma$, let the matrix $A(\sigma)$ be defined by

$$A(\sigma) = \left(\sum_{i=0}^{q(n)} (R(\sigma))^i\right) M(\sigma)$$

and let π be the characteristic row-vector of the initial configuration of M on inputs of size n. Analogously let η be the characteristic column-vector of the accepting configurations. Since M is 1-way, by the previous definitions $\pi A(x_1) A(x_2) \ldots A(x_n) \eta$ is the number of accepting computations of M on input $x = x_1 x_2 \ldots x_n \in \Sigma^*$. Now, let $p(z)$ be the polynomial such that $p(0)=0$ and $p(i)=1$ for every $i=1, 2, \ldots, k$: $p(z) = 1 - \prod_{i=1}^{k} (1-(z/i))$. Now, applying both Propositions 1 and 2, for every $x = x_1 x_2 \ldots x_n \in \Sigma^*$ we have

$$p(\pi A(x_1) A(x_2) \ldots A(x_n) \eta) = \hat{p}(\pi) \prod_{i=1}^{n} \tilde{p}(A(x_i)) \tilde{p}(\eta).$$

Hence, by the definition of p, we obtain

$$\begin{aligned} \# \{x \in L / |x| = n\} &= \sum_{\sigma_1, \ldots, \sigma_n \in \Sigma} p(\pi A(\sigma_1) A(\sigma_2) \ldots A(\sigma_n) \eta) \\ &= \sum_{\sigma_1, \ldots, \sigma_n \in \Sigma} \hat{p}(\pi) \prod_{i=1}^{n} \tilde{p}(A(\sigma_i)) \tilde{p}(\eta) = \hat{p}(\pi) \left(\sum_{\sigma \in \Sigma} \tilde{p}(A(\sigma))\right)^n \tilde{p}(\eta). \end{aligned}$$

Therefore, in order to compute $f_L(1^n)$, we first determine the arrays $\underline{a} = \hat{p}(\pi)$, $\underline{b} = \tilde{p}(\eta)$ and the matrices $\tilde{p}(A(\sigma))$ for every $\sigma \in \Sigma$. Then we compute the sum B of all matrices $\tilde{p}(A(\sigma))$ for $\sigma \in \Sigma$ and the product $\underline{a} B^n \underline{b}$. Such an algorithm clearly shows that f_L is NC^1-reducible to computing the power of an integer matrix of polynomial size and hence it belongs to DET.

As regards rank$_L$ we note that the previous algorithm can be used to compute $\# \{y \in L / |y| \leq n\}$ in DET. Moreover, for every $x = x_1 x_2 \ldots x_n \in \Sigma^*$ we have

$$\begin{aligned} &\# \{y \in L / |y| = n, y <_{\text{lex}} x\} \\ &\quad = \sum_{i=1}^{n-1} \sum_{\sigma < x_i} \sum_{\sigma_1, \ldots, \sigma_{n-i-1} \in \Sigma} p(\pi A(x_1) \ldots A(x_{i-1}) A(\sigma) A(\sigma_1) \ldots A(\sigma_{n-i-1}) \eta) \\ &\quad = \sum_{i=1}^{n-1} \underline{a} \left(\prod_{j=1}^{i-1} \tilde{p}(A(x_j))\right) \left(\sum_{\sigma < x_i} \tilde{p}(A(\sigma))\right) B^{n-i-1} \underline{b}, \end{aligned}$$

where $\underline{a}$, $\underline{b}$ and B are defined as before. The last formula requires a linear number of iterated products of matrices of size $O(q(n)^k)$ which can be computed in parallel showing that the overall problem lies in DET.

The property proved in Proposition 3 can be painless extended to more general classes of languages. To this end, for any $k \in \mathbb{N}$ and every set $A \subseteq \{0, 1, 2, \ldots, k\}$, let $\mathscr{L}(A, k)$ be the class of languages L such that, for a suitable 1-NL(k) Turing machine M, L is the set $\{x / \#\mathrm{acc}_M(x) \in A\}$. We improperly say that L is recognized by M. Note that the class $\mathscr{L}(\{1, 2, \ldots, k\}, k)$ coincides with the class 1-way NL(k). Moreover, let $\mathscr{A}$ be the class of languages defined by

$$\mathscr{A} = \bigcup_{(A, k) \in \Delta} \mathscr{L}(A, k),$$

where Δ is the set of all pairs (A, k) such that $k \in \mathbb{N}$ and $A \subseteq \{0, 1, 2, \ldots, k\}$.

Proposition 4: *The class $\mathscr{A}$ is closed under boolean operations.*

Proof: It suffices to prove the closure under complement and intersection. Clearly $\mathscr{A}$ is closed under complement since for every $(A, k) \in \Delta$ and every $L \in \mathscr{L}(A, k)$, the set L^c belongs to $\mathscr{L}(\{0, 1, \ldots, k\} \setminus A, k)$. As regards intersection let L_1, L_2 be two languages in $\mathscr{L}(A, k_1)$ and $\mathscr{L}(B, k_2)$ respectively. First note that, for every $u \in \mathbb{N}$, there exists a 1-NL$(k_1 + u)$ machine $M_1(u)$ such that, for every input x, $u \leqq \#\mathrm{acc}_{M_1(u)}(x) \leqq k_1 + u$, and $x \in L_1$ iff $\#\mathrm{acc}_{M_1(u)}(x) \in A + u$, where $A + u = \{h + u / h \in A\}$. Analogously, for any $t \in \mathbb{N}$, let $M_2(t)$ be defined similarly with respect to L_2. Now, for a fixed $u \in \mathbb{N}$, consider the one-way NL Turing machine M which simulates $M_1(u)$ and $M_2(1)$ in parallel and accepts if both machines accept. Clearly, for every input x, $u \leqq \#\mathrm{acc}_M(x) \leqq (k_1 + u)(k_2 + 1)$. Denoting by R the set $\{(a+u)(b+1) / 0 \leqq a \leqq k_1,\ 0 \leqq b \leqq k_2\}$, it is easy to see that, for u large enough, R is unambiguously defined in the sense that for every $n \in R$ there exists only one pair (a, b) such that $0 \leqq a \leqq k_1$, $0 \leqq b \leqq k_2$ and $n = (a+u)(b+1)$. This implies that also the set $C = \{(a+u)(b+1) / a \in A,\ b \in B\}$ is unambiguous and hence $L_1 \cap L_2$ belongs to $\mathscr{L}(C, (k_1 + u)(k_2 + 1))$.

The previous proposition implies that $\mathscr{A}$ contains the boolean closure of the class 1-FANL and hence also the boolean closure of the set of 1-way unambiguous NL languages.

Proposition 5: *For any language $L \in \mathscr{A}$, rank_L belongs to DET.*

Proof: It is easy to verify that the proof of Proposition 3 holds for every language in $\mathcal{L}(A, k)$ with $(A, k) \in \Delta$. The only difference is that the polynomial $p(z)$ is now defined so that $p(a)=1$ for every $a \in A$, and $p(b)=0$ for every $0 \leq b \leq k$ and $b \notin A$, that is

$$p(z)=\sum_{a \in A}\left(\prod_{b \in B-A} \frac{z-b}{a-b} \prod_{\alpha \in A-\{a\}} \frac{z-\alpha}{a-\alpha}\right),$$

where $B=\{0, 1, \ldots, k\} \setminus A$.

Note that $\hat{p}$ may not be defined since $p(0)$ could be different from 0. In this case the proposition is proved just considering the polynomial $q=p(z)-p(0)$ and modifying the reasoning in a suitable way.

4. DIFFICULT COUNTING PROBLEMS

In this section we present some completeness results concerning the counting function of languages belonging to certain subclasses of P. We consider the following notion of reducibility: given two functions f, g in $\# P_1$ we say that f is $\#_1$-reducible to g if there exists an oracle Turing machine working in polynomial time which computes f and calls an oracle for g only once. For sake of brevity we say that a class of languages C has $\# P_1$-complete counting function if there exists $L \in C$ such that f_L is complete for $\# P_1$ with respect to $\#_1$-reducibility.

The first class we consider is $\Sigma_1^{\text{logtime}}$ which is the set of languages recognized by nondeterministic Turing machines in $O(\log n)$ time. Such a class is the first level of the logarithmic time hierarchy $\{\Sigma_k^{\text{logtime}}\}$ studied by Sipser in [13]: for every positive integer k, $\Sigma_k^{\text{logtime}}$ is the set of languages accepted by an alternating Turing machine working in logarithmic time and having at most $k-1$ alternations beginning with an existential state. As usual Π_k^{logtime} denotes the class of complements of languages in $\Sigma_k^{\text{logtime}}$. We recall that, in order to obtain sublinear time computations, the Turing machine has a random access input tape ruled by a special address tape [5].

PROPOSITION 6: *The class $\Sigma_1^{\text{logtime}}$ has $\# P_1$-complete counting function.*

Proof: We show that for every function $f \in \# P_1$ there exists a language $L \in \Sigma_1^{\text{logtime}}$ such that f is $\#_1$-reducible to f_L. Since there exist $\# P_1$-complete functions [15, 4], this fact implies that for some language $L \in \Sigma_1^{\text{logtime}}$ f_L is $\# P_1$-complete. Let M be the counting Turing machine computing f and, without loss of generality, assume that for any $n \in \mathbb{N}$ all accepting computations of M on input 1^n have the length $p(n)$ for a suitable polynomial p. In the following we denote by $\operatorname{bin}(n)$ the binary representation of the integer n. Given $c_1 \in \mathbb{N}$ such that $2^{c_1 \lceil \log n \rceil} > p(n)$ for every $n \in \mathbb{N}$, let $t(n) = 2^{c_1 \lceil \log n \rceil}$; given the morphism $\operatorname{cod}: \{0, 1\}^* \to \{0, 1\}^*$ such that $\operatorname{cod}(0) = 01$ and $\operatorname{cod}(1) = 10$, let $c_2 \in \mathbb{N}$ be such that the integer $q(n)$, whose binary representation is $\operatorname{cod}(\operatorname{bin}(n))\, 0^{c_2 |\operatorname{bin}(n)|}$, is greater than $(p(n)+1)(t(n)+1)$ for every $n \in \mathbb{N}$. Let $C = (a_0, a_1, \ldots, a_{p(n)})$ be an accepting computation of M on input 1^n, where each a_i is an instantaneous description of M of the form uqv: q is the current state, uv represents the first $t(n)$ cells of the tape and the tape head reads the first symbol of v. Such a computation can be represented by the padded string I_C of length $q(n)$ such that $I_C = a_0 ¢ a_1 ¢ \ldots ¢ a_{p(n)} ¢ y$, where $y = (a_{p(n)} ¢)^r z$, z being a prefix of $a_{p(n)} ¢$ and r a suitable integer. Then we define the language

$$L = \{ I_C / C \text{ is an accepting computation of } M \}.$$

It is clear that $f(1^n) = f_L(1^{q(n)})$ for every integer n. Moreover we prove that L belongs to Π_1^{logtime}. To this end we show that L can be recognized in logarithmic time by an alternating Turing machine M' (with special address tape) which only has universal states. Denoting by Γ the working alphabet and by Q the set of states of M, the machine M' on input $w = w_1 w_2 \ldots w_{q(n)}$, $w_i \in (\Gamma \cup Q \cup \{¢\})$ for each i, executes the following steps:

1) using a binary search it computes $\operatorname{bin}(|w|)$ in $O(\log |w|)$ deterministic time;

2) recalling that if $w \in L$ then $\operatorname{bin}(|w|) = \operatorname{cod}(\operatorname{bin}(n))\, 0^{c_2 |\operatorname{bin}(n)|}$ for a suitable $n \in \mathbb{N}$, M' considers the longest prefix y of $\operatorname{bin}(|w|)$ belonging to $\{01, 10\}^*$ and computes $n = \operatorname{cod}^{-1}(y)$;

3) using universal states M' verifies whether $q_0 1^n \beta^{t(n)-n-1} ¢$ is a prefix of w (here q_0 is the initial state of M and β represents the blank symbol);

4) using universal states, M' checks whether for every

$$j = 1, 2, \ldots, q(n) - t(n) - 3$$

the pair $w_j w_{j+1} w_{j+2}$ and $w_{j+t(n)+1} w_{j+t(n)+2} w_{j+t(n)+3}$ agrees with the transition relation of M. Note that, since $t(n)$ is a power of 2, each sum $j + t(n)$ requires logarithmic deterministic time;

5) using universal states, M' checks whether there exists a final state among the last $t(n)+1$ symbols of the input.

It is clear that M' works in logarithmic time and it accepts w if and only if all possible computations on input w are accepting. This means that L

belongs to Π_1^{logtime}. Since f_L is $\#_1$-reducible to f_{L^c}, the proposition is proved recalling that L^c belongs to $\Sigma_1^{\text{logtime}}$.

Using standard arguments the previous proofs can be extended to other classes of languages.

PROPOSITION 7: *The classes of languages accepted by the following devices have $\# P_1$-complete counting function:*

a) *uniform families of constant depth, polynomial size, unbounded fan in circuits;*

b) *CRCW P-RAM working in constant time and using a polynomial number of processors;*

c) *log-space bounded deterministic Turing machines;*

d) 1-*way nondeterministic, log-space bounded Turing machines.*

Proof: We observe that assertions *a*) and *b*) are easy consequences of the previous proposition. In fact it is known that the languages in $\Sigma_1^{\text{logtime}}$ can be recognized by uniform families of constant depth, polynomial size, unbounded fan in circuits [13] which, on the other hand, can be simulated by CRCW P-RAM working in constant time and using a polynomial number of processors [14]. It is also easy to verify that any language in $\Sigma_1^{\text{logtime}}$ is recognizable in deterministic logarithmic space. As regards the last machine we observe that the complement of the language L, defined in the proof of Proposition 4, can be recognized by a 1-way NL Turing machine. A machine of this kind can execute the algorithm described in Proposition 6 in the following way: first it executes steps 1) and 2) by guessing the length of the input string; then it executes the other three steps using existential states rather than universal ones. A computation accepts if and only if at least one condition checked by the algorithm is not verified. The computation clearly requires logarithmic space.

We conclude considering the counting function of languages accepted by simple kinds of automata. We recall that a 2-way deterministic checking stack automaton (2-DCSA) is defined as a 2-way deterministic pushdown automaton (2-DPDA) except that once a symbol is written on the stack it cannot be erased. However, entering a special reading mode, the machine may read all symbols stored into the stack with the restriction that, once such a reading mode is entered, the machine cannot go back to the writing mode [6].

PROPOSITION 8: *The classes of languages accepted by 2-DCSA, 2-DPDA, 1-way 2-head deterministic finite state automata have $\# P_1$-complete counting function.*

Proof: Let f be a function in $\# P_1$ and let M be the counting Turing machine computing f in polynomial time. Without loss of generality we assume that for any integer n, all accepting computations of M on input 1^n have length $p(n)$ for a suitable strictly increasing polynomial p. An accepting computation C of M on input 1^n is represented by the string $I_C = a_0 ¢ a_1 ¢ \ldots ¢ a_{p(n)} ¢$, where a_0 is the initial configuration, every a_i has length $n+i+1$ and is a configuration of the form xqy with standard meaning. Let us define the language L of all strings I_C such that C is accepting computation of M. It is clear that $f(1^n) = f_L(1^{q(n)})$ for a suitable strictly increasing function $q(n)$. It is easy to see that L is recognized by a 2-DCSA which first writes the input on the stack and then, entering the reading phase, checks that consecutive configurations agree with the transition relation of M. Since a similar reasoning holds for 2-DPDA and 1-way 2-head DFA, the proposition is proved.

REFERENCES

1. C. ALVAREZ, B. JENNER, A very hard log space counting problem, Proceedings 5th Conference on Structure in Complexity Theory, 1990, pp. 154-168.
2. A. BERTONI, D. BRUSCHI and M. GOLDWURM, Ranking and formal power series, *Theoretical Computer Science*, *79*, 1991, pp. 25-35.
3. A. BERTONI, M. GOLDWURM and P. MASSAZZA, Counting problems and formal series in noncommuting variables, *Inform. Process. Lett.*, *34*, 1990, pp. 117-121.
4. A. BERTONI, M. GOLDWURM and N. SABADINI, The complexity of computing the number of strings of given length in context-free languages, *Theoretical Computer Science*, *86*, 1991, pp. 325-342.
5. A. CHANDRA, D. KOZEN and L. STOCKMEYER, Alternation, *J. Assoc. Comput. Mach.*, *28*, 1981, p. 114-133.
6. J. H. CHANG, O. H. IBARRA, M. A. PALIS and B. RAVIKUMAR, On pebble automata, *Theoretical Computer Science*, *44*, 1986, pp. 111-121.
7. S. COOK, A taxonomy of problems with fast parallel algorithms, *Information and Control*, *64*, 1985, pp. 2-22.
8. A. V. GOLDBERG and M. SIPSER, Compression and ranking, Proceedings 17th ACM Symposium on Theory of Computing, 1985, pp. 59-68.
9. D. T. HUYNH, The complexity of ranking simple languages, *Math. Systems Theory*, *23*, 1990, pp. 1-20.
10. D. T. HUYNH, Effective entropies and data compression, *Information and Computation*, *90*, 1991, pp. 67-85.

11. W. Kuich, Finite automata and ambiguity, Report 253, Institut für Informationsverarbeitung, Technische Universität Graz, June 1988.
12. C. C. MacDuffee, The theory of Matrices, Chelsea Pub. Comp., New York 1946.
13. M. Sipser, Borel sets and circuits complexity, Proceedings 15th ACM Symposium on Theory of Computing, 1983, pp. 61-69.
14. L. Stockmeyer and U. Vishkin, Simulation of random access machines by circuits, *SIAM J. Comput*, *13*, 1984, pp. 409-422.
15. L. G. Valiant, The complexity of enumeration and reliability problems, *SIAM J. Comput*, *8*, 1979, pp. 410-420.