On Languages Accepted with Simultaneous Complexity Bounds and Their Ranking Problem*

Alberto Bertoni, Carlo Mereghetti, and Giovanni Pighizzini

Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano
via Comelico, 39 – 20135 Milano – Italy

Abstract. In this paper, we consider Turing machines having simultaneous bounds on working space $s(n)$, input head inversions $i(n)$ and ambiguity degree $d(n)$. Besides the standard notion of space complexity (Type 1), we discuss a stronger notion (Type 2). In the deterministic case, we show an optimal $\Omega_\infty(\frac{\log n}{\log\log n})$ bound on input head inversions for recognizing nonregular languages on Type 1 machines that does not hold for Type 2 machines. Subsequently, the parallel complexity of the ranking problem is studied. We prove that any language accepted on Type 2 machines within $s(n), i(n), d(n)$ simultaneous bounds such that $s(n) \cdot i(n) \cdot d(n) = O(\log n)$ can be ranked in DET.

1 Introduction

A well consolidated trend in literature concerns the study of automata having simultaneous bounds on working space, input head inversions and ambiguity degree (see, e.g., [HU69, HM81, CH88, RCH91, Mer92, ÀJ93]). The resulting classes of languages are studied both for structural reasons and for certain combinatorial problems defined on them.

In this work we consider languages accepted by Turing machines within $s(n)$ working space, $i(n)$ input head inversions, $d(n)$ ambiguity degree such that the product $s(n) \cdot i(n) \cdot d(n)$ is $O(\log n)$. This leads to consider sublogarithmic space complexity. In the standard model (Type 1), a machine M, equipped with a read–only input tape and a read–write work tape, works within $s(n)$ space if every accepted word of length n admits an accepting computation taking at most $s(n)$ different work tape squares. As a matter of fact, designing algorithms with sublogarithmic space bounds on such a model seems to be an hard task since, for instance, monotone increasing sublogarithmic functions are not space constructible [Gef91]. In this sense, a more powerful space bounded model of computation, called *demon machine*, has been introduced in [RCH91] (see also

* This work was supported in part by the ESPRIT Basic Research Action No. 6317: "Algebraic and Syntactic Methods in Computer Science (ASMICS 2)" and by the Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST).

[MS82] for related topics). Basically, a demon machine can be obtained by equipping any Turing machine with the function s such that, on any input of length n, $s(n)$ cells on the work tape are automatically marked out before the computation begins, and that the machine never exceeds the marked cells. Such computational devices will be denoted Type 2 machines throughout the rest of the paper.

In Section 3, we briefly discuss some differences between the two models. It is well known a $\Omega_\infty(\log\log n)$ space bound for nonregular languages on Type 1 machines [HU69]. We extend such a result as follows:

1. We prove a $\Omega_\infty(\frac{\log n}{s(n)})$ bound in the deterministic case for the number of input head inversions on Type 1 machines.
2. We prove the optimality of such a bound by exhibiting an algorithm on Type 1 machines accepting the Alt–Mehlhorn nonregular language L_{AM} [AM75] within $O(\log\log n)$ space and $O(\frac{\log n}{\log\log n})$ input head inversions.
3. We show that the bound $\Omega_\infty(\frac{\log n}{\log\log n})$ on input head inversions for recognizing L_{AM} in $O(\log\log n)$ space cannot be improved even when Type 2 model is used. However, it is easy to exhibit nonregular languages accepted on Type 2 machines in sublogarithmic space and without any input head inversion.

Next, we study the parallel complexity of some counting problems on languages namely *counting*, *census* and *ranking* that are defined as follows: given a language $L \subseteq \Sigma^*$, the counting (census) function for L returns, for each positive integer n, the number of the strings in L of length (less than or) equal to n; the rank function for L returns, on each string x in Σ^*, the number of the strings in L less than or equal to x with respect to a lexicographic order on Σ^*. Counting and census functions are related to the enumeration of several combinatorial objects such as polyominoes, rooted planar trees, random walks etc. [Gol78, DV84]. Ranking is studied in [GS85, Hem87, Huy91a] in connection with data compression and with the more general notion of effective language entropy. Further examples of related topics can be found in [JVV86, Huy91b, SF90, RCH91, ABP92].

In what follows, the parallel complexity of problems is evaluated within the framework of the classes NC^k $(k > 0)$ of problems solvable by log–space uniform families of boolean circuits of polynomial size and $O(\log^k n)$ depth together with the notion of NC^1–reduction between problems [Ruz81, Coo85]. Note that, for any language, counting and census are NC^1–equivalent and both NC^1–reducible to ranking so that every bound on ranking complexity holds for counting and census as well.

In [GS85], Goldberg and Sipser prove that the class of languages accepted in logarithmic space by one–way deterministic Turing machines is rankable in polynomial time. As a matter of fact, Huynh [Huy90] observe that the method used in such a proof actually shows that ranking the class of the languages accepted in logarithmic space by one–way unambiguous Turing machines is in DET, the class of problems NC^1–reducible to computing the determinant of integer matrices, a subclass of NC^2. Moreover, Huynh and Vinay [Huy90, Vin91] show that ranking one–way unambiguous pushdown automata with an auxiliary log–space bounded tape and working in polynomial time is in NC^2. Finally, Bertoni and Goldwurm [BG93] take into account also the ambiguity degree and prove that ranking languages accepted in logarithmic space by one–way Turing machines having a constant ambiguity degree is in DET. Such a result can be generalized to an arbitrary constant number of input head inversions [Mer92].

In Section 4, we show that the class of languages accepted by Type 2 machines having a logarithmic bound on the product $s(n) \cdot i(n) \cdot d(n)$ can be efficiently ranked. More precisely, we prove that their counting, census and ranking functions belong to DET. Such a result is, then, extended to more general classes of languages. The proof makes use of a technique introduced in [Kui88], after showing, in Section 3, that any machine working within $s(n)$ space and $i(n)$ input head inversions can be simulated by a one–way machine in $O(s(n) \cdot i(n))$ space and with the same ambiguity degree.

2 Preliminaries

The reader is expected to be familiar with elementary concepts from formal language theory and complexity theory (see, e.g., [HU79, BDG87]). In the following, we briefly recall some basic definitions and results used throughout the paper.

Let Σ^* be the set of strings on a finite alphabet Σ and let $|x|$ denote the length of any element x in Σ^*. Assuming a total ordering $\preceq_\Sigma$ on Σ, a total ordering $\preceq$ on Σ^* may be obtained as follows: for any pair x, y of strings in Σ^*, then $x \preceq y$ if and only if $|x| < |y|$ or $|x| = |y|$ and $x = waz, y = wbz'$ with $w, z, z' \in \Sigma^*$ and $a, b \in \Sigma$ such that $a \preceq_\Sigma b$. Given a language $L \subseteq \Sigma^*$, then notation $L^{\leq n}$ ($L^{=n}$) stands for the set of words of length less than or equal (equal) to n belonging to L while $\#L$ denotes its cardinality. A language defined over a one letter alphabet is called *tally* language. Given an accepting device M, the language recognized by M is denoted by L_M.

In this paper, we are concerned with the *ranking*[2] problem defined, for any language $L \subseteq \Sigma^*$, as the function $rank_L : \Sigma^* \rightarrow \mathbb{N}$ such that $rank_L(x) = \#\{y \in L \mid y \preceq x\}$. Other related enumeration problems on L, namely *counting*, defined as $f_L : \{1\}^* \rightarrow \mathbb{N}$ such that $f_L(1^n) = \#L^{=n}$, and *census*, defined as $c_L : \{1\}^* \rightarrow \mathbb{N}$ such that $c_L(1^n) = \#L^{\leq n}$, are considered as well.

On evaluating problem parallel complexity, we consider log–space uniform families of boolean circuits as presented in [Ruz81, Coo85]. Recall that, for each integer $k \geq 1$, the class NC^k contains those problems solvable by log–space uniform families of boolean circuits having polynomial size and $O(\log^k n)$ depth. We remark that the class $\mathrm{NC} = \bigcup_{k \geq 1} \mathrm{NC}^k$ is usually accepted as the class of problems solvable very rapidly (in polylogarithmic time) by a parallel model of computation with a reasonable (polynomial) number of processors [Coo81, KR90].

[2] More precisely, such a definition of ranking function corresponds to that of *strong ranking function* as given in [Hem87].

Several typical problems in elementary arithmetic and linear algebra lay within the first two levels of hierarchy $\{NC^k\}$. The class NC^1 contains, for instance, the iterated sum of n–bit integers, the product of two n–bit integers and the boolean and integer matrix multiplication. Here we are mainly interested in a class of problems, called DET, laying between NC^1 and NC^2 and defined by means of NC^1–reducibility notion: given two problems Π and Π', we say that Π is NC^1*–reducible* to Π' if there exists a log–space uniform family of boolean circuits of polynomial size and logarithmic depth solving Π and allowed to have oracle nodes for Π'. Every oracle node having r inputs and s outputs counts as depth $\lceil \log(r+s) \rceil$. Problems Π and Π' are NC^1*–equivalent* if each one is NC^1–reducible to the other. It is easy to prove that, for any integer $k \geq 1$, the class NC^k is closed under NC^1–reducibility. The class DET of problems NC^1–reducible to computing the determinant of square matrices of n–bit integers is considered in literature [Coo85]. Examples of members of such a class are: the transitive closure of square boolean matrices, the iterated product and powering of square integer matrices, computing the number of paths of given length between any two nodes in a digraph, the integer division of two n–bit integers and the iterated product of n–bit integers.

As a final remark, we observe that, for every language $L \subseteq \Sigma^*$, census and counting are NC^1–equivalent and both NC^1–reducible to ranking. It is enough, in fact, to note that, for any integer $n \geq 1$, it holds that $c_L(1^n) = \sum_{i=1}^{n} f_L(1^i)$ and $f_L(1^n) = c_L(1^n) - c_L(1^{n-1})$. Moreover, let w and l be respectively the maximal and the minimal word of length n in Σ^* with respect to the total ordering above defined; hence $f_L(1^n) = rank_L(w) - rank_L(l) + 1$. Thus, every bound on ranking complexity holds for counting and census as well.

3 Computational Models and Their Properties

Let us consider Turing machines, equipped with a read–only input tape and a read–write work tape, having simultaneous bounds on *working space*, *input head inversions* and *ambiguity degree*.

In the standard model (Type 1), a machine works within $s(n)$ *space*, $i(n)$ *inversions* of its input head and $d(n)$ *ambiguity degree* if, on any accepted input of length n, it presents at most $d(n)$ accepting computations each one taking at most $s(n)$ different work tape squares and at most $i(n)$ inversions of its input head direction.[3]

A more powerful space–bounded model of computation (Type 2 or, as introduced in [RCH91], *demon machines*) may be obtained by equipping any Turing machine with the function s such that, on any input of length n, $s(n)$ cells are automatically marked out before the computation begins, and that the Turing machine never exceeds the marked cells. We assume that functions $s(n)$, $i(n)$ and $d(n)$ are computable in logarithmic space.

[3] We recall that a Turing machine M is said to be *one–way* if it can never move its input head toward left. Otherwise, M is a *two–way* machine. For technical reasons, we set $i(n) = 1$ for one–way machines.

First of all, we briefly point out a relationship between space and input head inversions for the two model of computation above introduced:

Theorem 1. *For any Turing machine M working within $s(n)$ space, $i(n)$ input head inversions and $d(n)$ ambiguity degree, there exists a* one–way *Turing machine M' working in $O(s(n) \cdot i(n))$ space, with $d(n)$ ambiguity degree and such that $L_{M'} = L_M$.*

Proof. (outline) The result can be proved by generalizing the simulation of two–way finite state automata with one–way automata (see, e.g., [HU79]). In particular, note that $O(s(n) \cdot i(n))$ squares on the work tape of M' are needed to represent *crossing sequences* [Hen65] of at most $i(n)+1$ configurations of M each one of length $O(s(n))$. □

Hence, previous theorem shows that it is possible to eliminate input head inversions by suitably augmenting working space. Moreover, notice that, since a two–way machine working in constant space accepts a regular language, the converse does not hold.

Let us now briefly discuss some differences between the two models of computation by restricting ourselves to the deterministic case.

It is known that the space complexity of nonregular languages on two–way Type 1 machines is at least $\log \log n$ [HU69]. Here we refine such a lower bound by taking into account also the number of input head inversions.

Theorem 2. *Let L be a* nonregular *language and M be a deterministic* Type 1 *machine accepting L within $s(n)$ space and $i(n)$ input head inversions. Then, for a suitable constant $c > 0$, it holds that $s(n) \cdot i(n) \geq c \cdot \log n$ infinitely often.*

Proof. (outline) By Theorem 1, given $n \geq 0$, it is easy to construct from M a one–way finite automaton A_n with $2^{O(s(n) \cdot i(n))}$ states such that $L^{\leq n} = L_{A_n}^{\leq n}$. On the other hand, by a result proved in [DS90], it turns out that the number of states of the automata A_n is greater than or equal to $\sqrt{n} - 1$ infinitely often. Hence, it holds that $2^{O(s(n) \cdot i(n))} \geq \sqrt{n} - 1$ infinitely often. □

As an immediate consequence, we obtain the following:

Corollary 3. *Let L be a nonregular language accepted by a deterministic* Type 1 *machine within $s(n) = O(\log \log n)$ space and $i(n)$ input head inversions. Then, for a suitable constant $c > 0$, it holds that $i(n) \geq c \cdot \frac{\log n}{\log \log n}$ infinitely often.*

The following example shows that lower bounds given in Theorem 2 and Corollary 3 are optimal even for tally languages.

Example. For any positive integer n, let $q(n)$ denote the smallest number not dividing n. Alt and Mehlhorn [AM75] introduce the language:

$$L_{AM} = \{1^n \mid q(n) \text{ is a power of } 2\}.$$

They prove that L_{AM} is recognizable in space $O(\log\log n)$. However, the algorithm proposed uses $O(\log n)$ input head inversions. In the following, we present a different algorithm that recognizes L_{AM} and is *optimal* with respect to the lower bound stated in Theorem 2. More precisely, we prove the following result:

Theorem 4. *The language L_{AM} can be recognized by a deterministic* Type 1 *machine in $O(\log\log n)$ space and with $O(\frac{\log n}{\log\log n})$ input head inversions.*

Proof. (outline) The algorithm presented in [AM75] accepts L_{AM} by iteratively searching $q(n)$ among all the integers greater than 1. Here, we observe that, for any n, $q(n)$ must be a *power of a prime number*. In fact, let $< n >_j$ denote the rest of the integer division of n by j. Given an integer $k > 2$ such that $< n >_j = 0$, for $j = 2, 3, \ldots, k$, if $k+1$ is not a prime power, then $k+1$ can be expressed as a product of two relatively prime numbers a and b both less than k. So, it holds that $< n >_a = < n >_b = 0$ and hence $< n >_{a \cdot b} = < n >_{k+1} = 0$. This property allows us to search $q(n)$ only among prime powers. Thus, we can recognize L_{AM} by means of the following algorithm:

input(1^n)
$k := 2$
while $< n >_k = 0$ **do**
  begin
    $k := k+1$
    while not$(pp(k))$ **do** $k := k+1$
  end
if k is a power of 2 **then** ACCEPT
            else REJECT

where $pp(x)$ denotes the predicate "x is a power of a prime number".

Let us now estimate space and input head inversions complexity of the previous algorithm. As shown in [AM75], it holds that $q(n) = O(\log n)$. Thus, it is immediate to obtain the $O(\log\log n)$ space requirement. Moreover, we observe that the computation of $< n >_k$ can be done by scanning the input once while counting its length modulo k. So, each iteration of the outermost loop in the above algorithm uses exactly one input head inversion. It turns out that the total number of inversions equals the number of prime powers less than or equal to $q(n)$. Such a number can be shown to be $O(\frac{\log n}{\log\log n})$ [BMP94]. □

The $\Omega_\infty(\log\log n)$ space and $\Omega_\infty(\frac{\log n}{\log\log n})$ input head inversions bounds are no longer valid on Type 2 machines. In fact, it is easy to exhibit nonregular languages accepted by Type 2 machines with space and/or input head inversions complexity bounded by functions growing slower than $\log\log n$ and $\frac{\log n}{\log\log n}$, respectively.

Example. Consider the language:

$$L = \{1^n \mid < n >_{\lceil \log\log\log n \rceil} = 0\}.$$

It is immediate to show that L is nonregular. Moreover, it is not difficult to devise an algorithm for a one–way Type 2 machine accepting L within $\lceil \log\log\log n \rceil$ space.

Although Type 2 machines are generally more powerful than Type 1 ones, the algorithm presented in Theorem 4 and accepting L_{AM} is optimal even on Type 2 machines.

Theorem 5. *Let M be a Type 2 machine accepting the language L_{AM} within $s(n)$ space and $i(n)$ input head inversions. Then, there exists a constant $c > 0$ such that $s(n) \cdot i(n) \geq c \cdot \log n$ holds infinitely often.*

Proof. (outline) Let $n_\alpha = \min\{x \mid q(x) = 2^\alpha\}$. It is easy to show that $n_\alpha = d \cdot 2^{\alpha-1}$ for a suitable odd integer $d > 1$. Moreover, for any integer g whose factors are all greater than 2^α and any odd integer $m < d$, it holds that $1^{m \cdot g \cdot 2^{\alpha-1}} \notin L_{AM}$ while $1^{d \cdot g \cdot 2^{\alpha-1}} \in L_{AM}$.

Now, we have $n = 2^{2^j} < n_\alpha < 2^{2^{j+1}} = n^2$ for suitable integers n, j. Using the same construction of Theorem 2, we can obtain from M an automaton A_n with $2^{O(s(n) \cdot i(n))}$ states, such that $L_{A_n} \cap \{1^k \mid n < k < n^2\} = L_{AM} \cap \{1^k \mid n < k < n^2\}$.

Consider an integer g such that $n^{\frac{3}{2}} < g \cdot n_\alpha < n^2$ and either $g = 1$ or each one of its factor is greater than 2^α. For any pair of odd integers m_1, m_2 such that $m_2 < m_1 < d$ and $n < m_2 \cdot g \cdot 2^{\alpha-1}$, it turns out that the automaton A_n has to distinguish the strings $1^{m_1 \cdot g \cdot 2^{\alpha-1}}$ and $1^{m_2 \cdot g \cdot 2^{\alpha-1}}$. In fact, for $z = g \cdot (d - m_1) \cdot 2^{\alpha-1}$, it holds that:

- $\gamma_1 = 1^{m_1 \cdot g \cdot 2^{\alpha-1}} \cdot 1^z = 1^{d \cdot g \cdot 2^{\alpha-1}} \in L_{AM}$,
- $\gamma_2 = 1^{m_2 \cdot g \cdot 2^{\alpha-1}} \cdot 1^z = 1^{(d - m_1 + m_2) \cdot g \cdot 2^{\alpha-1}} \notin L_{AM}$,

and both strings γ_1, γ_2 have length laying between n and n^2. Thus, A_n must have at least $\frac{n^{\frac{3}{2}} - n}{g \cdot 2^\alpha}$ states. Recalling that $q(n_\alpha) = 2^\alpha < \log n$ and observing that $g < n$ we can conclude that $2^{O(s(n) \cdot i(n))} > \frac{n^{\frac{1}{2}} - 1}{\log n}$ holds infinitely often. □

4 Ranking Languages with Nonconstant Ambiguity Degree

Several results in literature witness that ambiguity degree is a critical parameter in efficient ranking evaluation. Recall, for instance, [Huy90] where it is proved that languages accepted in logarithmic space by one–way nondeterministic Turing machines are rankable in polynomial time if and only if $\mathrm{P} = \mathrm{P}^{\#\mathrm{P}}$. On the other hand, if we impose constant ambiguity bounds on such devices, then ranking turns out to be feasible [BG93].

In this section, we extend such a feasibility result by considering nonconstant ambiguity bounds. At this purpose, we first introduce the class $1\text{–}C(s(n), d(n))$ of languages accepted in $s(n)$ space by one–way Type 2 machines having $d(n)$ ambiguity degree. We prove that ranking $1\text{–}C(s(n), d(n))$ is efficiently parallelizable if the product $s(n) \cdot d(n)$ is logarithmic. Our proof is based on a technique introduced in [Kui88] and reconsidered in [BG93].

Theorem 6. *Ranking 1–$C(s(n), d(n))$ with $s(n) \cdot d(n) = O(\log n)$ is in* DET.

Proof. Let M be a Type 2 machine accepting the language $L_M \subseteq \Sigma^*$ within theorem bounds. For any input string $x \in \Sigma^*$ of length n, it is not hard to construct a finite automaton A_x accepting the language $L_{A_x} = \{y \in \Sigma^* \mid y \preceq x\}$. Moreover, A_x can be easily shown to be one–way, unambiguous and with $O(n)$ states.

Next, let us construct the machine M_x obtained by simulating in parallel the original machine M and the automaton A_x. Notice that M_x recognizes exactly the language $L_{M_x} = L_M \cap L_{A_x} = \{y \in L \mid y \preceq x\}$. So, computing $rank_{L_M}(x)$ reduces to evaluating the cardinality of L_{M_x}. We assume that M_x presents no stationary moves; otherwise they can be eliminated by computing the transitive closure of suitable transition matrices representing stationary moves of M_x. Computations of M_x on a string $\sigma_1\sigma_2\ldots\sigma_k$, with $k \leq n$, may be depicted by the matrix product $H_x(\sigma_1) \cdot H_x(\sigma_2) \cdot \ldots \cdot H_x(\sigma_k)$ where, for each $\sigma \in \Sigma$ and each pair i, j of positive integers denoting configurations of M_x on inputs of length less than or equal to n, the (i, j)th element of $H_x(\sigma)$ is 1 if M_x moves from the ith to jth configuration by reading σ; 0 otherwise. By letting π (η, respectively) be the characteristic vector of the initial (final, respectively) configuration(s), we get that $\pi \cdot H_x(\sigma_1) \cdot H_x(\sigma_2) \cdot \ldots \cdot H_x(\sigma_k) \cdot \eta^T \leq d(n)$ is the number of M_x's accepting computations on input $\sigma_1\sigma_2\ldots\sigma_k$. Let us now introduce the integer polynomial:

$$p(z) = 1 - \prod_{i=1}^{d(n)} (1 - \frac{z}{i}) \tag{1}$$

such that $p(0) = 0$ and $p(z) = 1$ for each $1 \leq z \leq d(n)$. Thus, it is not hard to see that:

$$rank_{L_M}(x) = \sum_{i=1}^{n} \sum_{\sigma_1,\ldots,\sigma_i \in \Sigma} p(\pi \cdot H_x(\sigma_1) \cdot \ldots \cdot H_x(\sigma_i) \cdot \eta^T). \tag{2}$$

Let us now explain how to efficiently evaluate polynomial $p(z)$ in (2). At this purpose, we briefly recall some notions from linear algebra.

Given two square matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ of dimension, respectively, $m \times m$ and $n \times n$, their *direct sum* $A \oplus B$ and *Kronecker product* $A \otimes B$ are defined, respectively, as:

$$A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}, \quad A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ a_{21}B & a_{22}B & \cdots & a_{2m}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mm}B \end{pmatrix}.$$

Matrices $A \oplus B$ and $A \otimes B$ have dimension $(m + n) \times (m + n)$ and $(m \cdot n) \times (m \cdot n)$ respectively. Analogous definitions hold for vectors as well. Direct sum and Kronecker product are easily seen to be associative; moreover, $\otimes$ distributes over $\oplus$. Given a polynomial $p(z) = \sum_{i=1}^{k} a_i z^i$ (with $p(0) = 0$), let $\alpha(p) = \{i \in \{1, 2, \ldots, k\} \mid a_i \neq 0\}$. For each $n \times n$ matrix A, we define the two square matrices $\hat{p}(A)$ and $\tilde{p}(A)$ as:

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

where $\bigoplus$ denotes the iterated sum and $A^{(i)}$ the ith Kronecker power. Note that matrices $\hat{p}(A)$ and $\tilde{p}(A)$ have dimension $\sum_{i \in \alpha(p)} n^i \times \sum_{i \in \alpha(p)} n^i$. Previous definitions may be suitably extended to vectors. The following lemma [BG93] helps us in evaluating an integer polynomial on a vector by matrix by vector product:

Lemma 7. *Let A and B be two $n \times n$ matrices and let π and η be two $1 \times n$ vectors. For any integer polynomial $p(z)$ such that $p(0) = 0$, it holds that:*

- $p(\pi \cdot A \cdot \eta^T) = \hat{p}(\pi) \cdot \tilde{p}(A) \cdot \tilde{p}(\eta^T)$,
- $\tilde{p}(A \cdot B) = \tilde{p}(A) \cdot \tilde{p}(B)$.

Let us go back to computing $rank_{L_M}(x)$ according to (2). By Lemma 7, we get:

$$rank_{L_M}(x) = \sum_{i=1}^{n} \hat{p}(\pi) \cdot \left(\sum_{\sigma \in \Sigma} \tilde{p}(H_x(\sigma)) \right)^i \cdot \tilde{p}(\eta^T). \qquad (3)$$

Observe that the more expensive operation in the above outlined algorithm is the computation of the power of an integer matrix of dimension $2^{O(s(n) \cdot d(n))}$ and hence polynomial. Recalling that the power of an integer matrix of polynomial size is NC^1–reducible to computing determinant (see Section 2), we conclude that $rank_{L_M}$ is in DET. $\square$

By Theorem 1, we obtain the main result of this section:

Theorem 8. *If L is recognized by a* Type 2 *machine within $s(n)$ space, $i(n)$ input head inversions and $d(n)$ ambiguity degree such that $s(n) \cdot i(n) \cdot d(n) = O(\log n)$, then $rank_L$ is in* DET.

The arguments used in the proof of Theorem 6 work for other classes of languages. For instance, given a Type 2 machine M, we can consider the language L containing all the strings for which there is a number of accepting computations of M exceeding a certain threshold. If M is one–way working within $s(n) \cdot d(n) = O(\log n)$ bound and the threshold function is easily computable (i.e., in DET), then L can be efficiently ranked as well. More generally, given an ambiguity bound $d(n)$, we can consider a subset $A(n)$ of $\{0, \ldots, d(n)\}$ and the resulting class of languages containing all the strings whose number of accepting computations belongs to $A(n)$. So, as a generalization of 1–$C(s(n), d(n))$, the class 1–$C(s(n), A(n), d(n))$ can be introduced.

Theorem 9. *Ranking* 1–$C(s(n), A(n), d(n))$ *with* $A(n) \in \mathrm{DET}$ *and* $s(n) \cdot d(n) = O(\log n)$ *is in* DET.

Proof. If $0 \notin A(n)$, then it is enough to substitute polynomial (1) in Theorem 6 with the following obtained by standard Lagrange's interpolation technique:

$$p(z) = \sum_{a \in A(n)} \prod_{\substack{b=0 \\ b \neq a}}^{d(n)} \frac{z-b}{a-b}. \tag{4}$$

If $0 \in A(n)$, then polynomial $p(z)$ is computed using the set $A(n) - \{0\}$ instead of simply $A(n)$ in (4). The proof can, then, be easily completed by observing that for any language L, $rank_{L^c}$ is NC^1–reducible to $rank_L$ (L^c denotes the complement of L). □

Hence, Theorem 8 can be extended with respect to the parameter $A(n)$ in the obvious way. Moreover, by the final remark in Section 2, all the results obtained in this section apply to counting and census as well.

Acknowledgment. The authors wish to thank anonymous referees for some helpful comments.

References

[ABP92] E. Allender, D. Bruschi, and G. Pighizzini. The complexity of computing maximal word functions. Technical Report 92–15, DIMACS, 1992. To appear in *Computational Complexity*.

[ÀJ93] C. Àlvarez and B. Jenner. A very hard log–space counting class. *Theoretical Computer Science*, 107:3–30, 1993.

[AM75] H. Alt and K. Mehlhorn. A language over a one symbol alphabet requiring only $O(\log \log n)$ space. *SIGACT news*, 7:31–33, 1975.

[BDG87] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer Verlag, 1987.

[BG93] A. Bertoni and M. Goldwurm. On ranking 1–way finitely ambiguous NL languages and $\sharp P_1$–complete census functions. *RAIRO Inf. Theor.*, 27:135–148, 1993.

[BMP94] A. Bertoni, C. Mereghetti, and G. Pighizzini. An optimal lower bound for nonregular languages. 1994. To appear in *Information Processing Letters*.

[CH88] S. Cho and D. Huynh. On a complexity hierarchy between L and NL. *Information Processing Letters*, 29:177–182, 1988.

[Coo81] S. Cook. Towards a complexity theory of synchronous parallel computations. *Enseign. Math.*, 27:99–124, 1981.

[Coo85] S. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

[DS90] C. Dwork and L. Stockmeyer. A time complexity gap for two–way probabilistic finite–state automata. *SIAM J. Computing*, 19:1011–1023, 1990.

[DV84] M. Delest and G. Viennot. Algebraic languages and polyominoes enumeration. *Theoretical Computer Science*, 34:169–206, 1984.

[Gef91] V. Geffert. Nondeterministic computations in sublogarithmic space and space constructibility. *SIAM J. Computing*, 20:484–498, 1991.

[Gol78] J. Goldman. Formal languages and enumeration. *Journal of Combinatorial Theory*, 24:318–338, 1978.

[GS85] A. Goldberg and M. Sipser. Compression and ranking. In *Proc. 17th ACM Symposium on Theory of Computing*, pages 440–448. World Scientific, 1985.

[Hem87] L. Hemachandra. On ranking. In *Proc. 2th Structure in Complexity Theory*, pages 103–117, 1987.

[Hen65] F. Hennie. One–tape, off–line Turing machine computations. *Information and Control*, 8:553–578, 1965.

[HM81] J. Hartmanis and S. Mahaney. Languages simultaneously complete for one–way and two–way log–tape automata. *SIAM J. Computing*, 10:383–391, 1981.

[HU69] J. Hopcroft and J. Ullman. Some results on tape–bounded Turing machines. *Journal of the ACM*, 16:168–177, 1969.

[HU79] J. Hopcroft and J. Ullman. *Introduction to automata theory, languages, and computations*. Addison–Wesley, Reading, MA, 1979.

[Huy90] D. Huynh. The complexity of ranking simple languages. *Mathematical Systems Theory*, 23:1–19, 1990.

[Huy91a] D. Huynh. Effective entropies and data compression. *Information and Computation*, 90:67–85, 1991.

[Huy91b] D. Huynh. Efficient detectors and constructors for simple languages. *International Journal of Foundations of Computer Science*, 2:183–205, 1991.

[JVV86] M. Jerrum, G. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[KR90] R. Karp and V. Ramachandran. A survey of parallel algorithms for shared–memory machines. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, pages 869–941. North–Holland, 1990.

[Kui88] W. Kuich. Finite automata and ambiguity. Technical Report 253, Institüt für Informationsverarbeitung – Technische Universität Graz, 1988.

[Mer92] C. Mereghetti. On space bounded Turing machines with a constant number of input head inversions. In *Theoretical Computer Science – Proceedings of the Fourth Italian Conference*, pages 269–277. World Scientific, 1992.

[MS82] B. Monien and I. Sudborough. On eliminating nondeterminism from Turing machines which use less than logarithmic worktape space. *Theoretical Computer Science*, 21:237–253, 1982.

[RCH91] D. Ranjan, R. Chang, and J. Hartmanis. Space bounded computations: review and new separation results. *Theoretical Computer Science*, 80:289–302, 1991.

[Ruz81] W. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1981.

[SF90] L. Sanchis and M. Fulk. On efficient generation of languages instances. *SIAM J. Computing*, 19:281–295, 1990.

[Vin91] V. Vinaj. Counting auxiliary pushdown automata and semi–unbounded arithmetic circuits. In *Proc. 6th Structure in Complexity Theory*, pages 270–284, 1991.