An Optimal Lower Bound for Nonregular Languages∗

Alberto Bertoni Carlo Mereghetti Giovanni Pighizzini

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

Abstract

In this paper we prove a tight $\log n$ lower bound on the product of strong space and input head reversals for any Turing machine (equipped with a read only input tape) accepting a nonregular language.

Keywords: Computational complexity; Formal languages; Lower bounds.

1 Preliminaries

A classical topic in formal language theory and structural complexity concerns the analysis of computational resource requirements for certain classes of problems. A well known result in this field deals with space lower bounds for Turing machines accepting nonregular languages:

Theorem 1 [5] If a language $L$ is accepted by a one–way (two–way, resp.) nondeterministic Turing machine in strong space $o(\log n)$ ($o(\log \log n)$, resp.) then $L$ is regular.

In this work we unify and refine the bounds stated in Theorem 1 by taking into account also the number of input head reversals. More precisely, we prove that, for any nonregular language, the product of strong space and input head reversals must be greater than $\log n$ infinitely often. Furthermore, we show the optimality of such a lower bound by proving that a nonregular language proposed in [1] can be accepted within strong $O(\log \log n)$ space and $O(\frac{\log n}{\log \log n})$ input head reversals simultaneously.

We briefly recall definitions and conventions adopted throughout this work. Unless otherwise stated, Turing machines we consider are intended to be nondeterministic. Any machine $M$ is equipped with a read only input tape and a work tape. The input tape is usually visualized as having end markers which can never be exceeded by the input head during computations. Without loss of generality, we assume that $M$ accepts by entering a

∗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). Presented at “Secondo incontro italiano di Combinatoria Algebrica”, 16–17 October, 1995, Milano – Italy
final state with its input head off the rightmost input symbol. $M$ works within strong $s(n)$ space and $i(n)$ input head reversals (simultaneously) if, on any input of length $n$, each computation takes at most $s(n)$ work tape squares and reverses input head direction at most $i(n)$ times. More generally, $M$ is said to be one–way if, at each step, it can move its input head right one position or keep it stationary. $M$ is a two–way machine if left input head moves are allowed too. For technical reasons, we will assume $i(n) = 1$ for one–way machines.

2 The lower bound

Let us refine space lower bounds for nonregular languages given in [5] by considering also input head reversal. We first show how to remove input head reversals on Turing machines by suitably augmenting work space. Such a result is a generalization of the simulation of two–way finite state automata with one–way corresponding devices as presented, e.g., in [6]. The key idea is the notion of crossing sequence which we now briefly recall.

Let us consider the boundary between two input tape squares and let $c_i$ denote the configuration of a Turing machine $M$ when, during a computation, its input head crosses such a boundary for the $i$th time. The sequence $S = (c_1, c_2, \ldots)$ is called crossing sequence generated in the computation of $M$ at the given boundary [4]. Let us now take an input tape square containing a symbol $\sigma$ and two sequences $S'$ and $S''$ of configurations of $M$. We say that $S'$ matches $S''$ on $\sigma$ if $S'$ and $S''$ can represent two crossing sequences occurring at the left and at the right boundary, respectively, of $\sigma$ in some computation of $M$. It is worth remarking that such a matching can be “locally” tested without knowing input symbols other than $\sigma$ and just taking into account the transition function of $M$ (see, for details, [6]).

By means of such a tool, we are able to prove the following:

**Theorem 2** For any two–way Turing machine $M$ working within strong $s(n)$ space and $i(n)$ input head reversals there exists an equivalent one–way machine $M'$ where each accepting computation takes at most $O(s(n) \cdot i(n))$ work tape cells.

**Proof.** It is not hard to see that the Turing machine $M$ accepts an input string $x$ if and only if there exists a vector $(S_0, S_1, \ldots, S_n)$ of crossing sequences (representing an accepting computation of $M$ on $x$) such that:

1. $S_0$ and $S_n$ are crossing sequences whose only elements are, respectively, the initial and a final configuration of $M$.

2. For each $0 \leq i \leq n$, the crossing sequence $S_i$ has length at most $i(n)$.

3. For each $0 < i \leq n$, the crossing sequence $S_{i-1}$ matches $S_i$ on the $i$th input symbol.

The Turing machine $M'$ simulating $M$ begins by writing $S_0$ on its own work tape. Then, it guesses a crossing sequence $S_1$, verifies matching with $S_0$ on the first input symbol and substitutes $S_0$ with $S_1$. Such a procedure is iterated on each input symbol (see (3)) and $M'$ accepts if and only if, according to (1), a crossing sequence corresponding to a final configuration of $M$ is reached at the end. The fact that $M'$ is one–way follows from local verifiability of matching between crossing sequences above mentioned. Furthermore, by (2), any guessed crossing sequence can be represented within $O(s(n) \cdot i(n))$ space.
In [2], it is proved that Theorem 2 can be extended also to those machines where space is evaluated by just taking into account all the accepting computations (accept mode). Hence, we obtain the following:

**Theorem 3** Let $L$ be a nonregular language accepted by a Turing machine $M$ in strong $s(n)$ space and $i(n)$ input head reversals. Then, for a suitable constant $c > 0$, it holds that $s(n) \cdot i(n) \geq c \cdot \log n$ infinitely often.

### 3 Optimality proof

In this section, we prove that the lower bound obtained in Theorem 3 is optimal. Actually, we shall provide a stronger statement by showing that such a lower bound is tight also for deterministic Turing machines accepting nonregular languages over a single letter alphabet.

For any positive integer $n$, let $q(n)$ denote the smallest integer not dividing $n$. Alt and Mehlhorn [1] introduced the nonregular language:

$$L = \{1^n \mid q(n) \text{ is a power of } 2\},$$

and showed how to accept it in $O(\log \log n)$ space: given an input string $1^n$, the rest of the integer division of $n$ by $k$ ($< n >_k$, for short) is computed for each integer $k \geq 2$ until $q(n)$ is reached. Then, $L$ membership is decided by testing whether $q(n)$ is a power of 2. The space bound follows from the observation that $q(n) = O(\log n)$ [1]. Moreover, for each $2 \leq k \leq q(n)$, the computation of $< n >_k$ can be accomplished by scanning the input string once while counting its length modulo $k$. Thus, the number of input head reversals equals the number of iterations on $k$ and hence is $O(\log n)$.

Next lemma allows to reduce such an iteration number in order to obtain an optimal algorithm.

**Lemma** For any positive integer $n$, $q(n)$ is a power of a prime number.

**Proof.** Let $k > 2$ be an integer such that $< n >_j = 0$ for each $2 \leq j \leq k + 1$. If $k + 1$ is not a prime power, then there exists a pair $a, b$ of relatively prime numbers less than $k$ such that $k + 1 = a \cdot b$. So, it holds that $< n >_a = < n >_b = 0$ and hence $< n >_{a \cdot b} = < n >_{k+1} = 0$. Thus, $q(n)$ must be a prime power.

By previous lemma, we are able to obtain the following algorithm for recognizing $L$:

**Algorithm 1**

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

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

3
Now, we can prove the optimality of the lower bound stated in Theorem 3 by estimating the complexity of Algorithm 1:

**Theorem 4** The language \( \mathcal{L} \) can be recognized by a deterministic Turing machine in strong \( O(\log \log n) \) space with \( O(\frac{\log n}{\log \log n}) \) input head reversals.

**Proof.** The correctness of Algorithm 1 is a consequence of previous discussions. Let us now sketch how to evaluate predicate \( pp(x) \) for any positive integer \( x \) expressed in binary notation: for \( 2 \leq j \leq x \), test \( j \)'s primality by simply verifying that no positive integer less than \( j \) divides it. If that is the case, then checks whether \( j \) equals \( x \) by considering integers \( i \) such that \( j^i \leq x \). It is not hard to show that all this can be worked out in \( O(|x|) \) space without moving input head. Therefore, by recalling that \( q(n) = O(\log n) \), we can estimate the space complexity of whole Algorithm 1 as being \( O(\log \log n) \).

The number of input head reversals equals the number of iterations of the outermost loop, i.e., it coincides with the number of prime powers not exceeding \( q(n) \).

**Claim** The number of prime powers not exceeding \( k \) is asymptotic to \( \frac{k}{\log k} \) \((k \to \infty)\).

**Proof.** Let \( p_1, p_2, \ldots, p_s \) denote the primes not exceeding \( k \) and, for each \( 1 \leq i \leq s \), let \( N_i \) be the number of powers of \( p_i \) not exceeding \( k \). Thus, \( k \geq p_i^{N_i} \) and hence \( N_i \leq \frac{\log k}{\log p_i} \). By the Prime Number Theorem \([3]\), it holds that the number \( s \) of primes not exceeding \( k \) is asymptotic to \( \frac{k}{\log k} \) \((k \to \infty)\). Hence we obtain the following asymptotic estimation of the number of prime powers not exceeding \( k \):

\[
\frac{k}{\log k} \sim s \leq N_1 + N_2 + \ldots + N_s \leq \sum_{i=2}^{s} \frac{\log k}{\log p_i} \leq \sum_{i=2}^{s} \frac{\log k}{\log i} = \log k \cdot \sum_{i=2}^{s} \frac{1}{\log i} \sim \log k \cdot \int_{2}^{s} \frac{dx}{\log x} \sim \log k \cdot \frac{s}{\log s} \sim \log k \cdot \frac{k}{\log k}.
\]

In conclusion, by observing that \( q(n) = O(\log n) \) and that the function \( \frac{k}{\log k} \) is monotone non decreasing, we obtain the result.

**References**


