RAIRO. THEORETICAL INFORMATICS AND APPLICATIONS

VILIAM GEFFERT
NORBERT POPÉLY

A space lower bound for acceptance by one-way Π_2-alternating machines

RAIRO. Theoretical Informatics and Applications, tome 34, n° 5 (2000), p. 357-372

<http://www.numdam.org/item?id=ITA_2000__34_5_357_0>

Theoretical Informatics and Applications
Theoret. Informatics Appl. **34** (2000) 357–372

A SPACE LOWER BOUND FOR ACCEPTANCE BY ONE-WAY Π_2-ALTERNATING MACHINES *

VILIAM GEFFERT[1] AND NORBERT POPÉLY[1]

Abstract. We show that one-way Π_2-alternating Turing machines cannot accept unary nonregular languages in $o(\log n)$ space. This holds for an *accept* mode of space complexity measure, defined as the worst cost of any accepting computation. This lower bound should be compared with the corresponding bound for one-way Σ_2-alternating machines, that are able to accept unary nonregular languages in space $O(\log \log n)$. Thus, Σ_2-alternation is more powerful than Π_2-alternation for space bounded one-way machines with unary inputs.

AMS Subject Classification. 68Q17, 68Q45.

1. INTRODUCTION AND PRELIMINARIES

The problem of minimal resource requirements for accepting nonregular languages has been studied since the very beginnings of structural complexity theory. In the early work of 1965 [11], it was shown that either the space used by the given machine is bounded by a constant (hence, the corresponding language is regular), or, for some constant $d > 0$, the space used exceeds $d \cdot \log \log n$ infinitely often. That is, $s(n) \notin o(\log \log n)$. It should be pointed out that the results presented in [11] concern *strong* space (worst cost) for deterministic machines. The $\log \log n$ lower bound has been gradually generalized as follows; first to nondeterministic machines [13], then to nondeterministic *weak* space [1] (best cost of acceptance), and finally to weak space for alternating machines [15]. That is, even if we take

Keywords and phrases: Alternation, lower bounds, nonregular languages, space complexity, Turing machine.

* *This work was supported by the Slovak Grant Agency for Science (VEGA) under contract #1/7465/20 "Combinatorial Structures and Complexity of Algorithms".*

[1] Department of Computer Science, P. J. Šafárik University, Jesenná 5, 04154 Košice, Slovakia; e-mail: geffert,popely@kosice.upjs.sk

into account only accepting computations using minimal space, with the power of alternation, the used space must exceed $\log\log n$ infinitely often.

For machines using such little space, the differences among various modes of space complexity measure become more important. Let us briefly review the different notions of space complexity studied in the literature.

- Worst case cost: the machine $\mathcal{M}$ works in *strong* $s(n)$ space if, for each input of length n, no computation uses more space than $s(n)$ [11–13].
- Worst case cost on accepted inputs: $\mathcal{M}$ works in *middle* $s(n)$ space if, for each accepted input of length n, no computation uses more space than $s(n)$ [21].
- Worst cost of acceptance: $\mathcal{M}$ works in *accept* $s(n)$ space if, for each accepted input of length n, no accepting computation uses more space than $s(n)$ [3,14].
- Best cost of acceptance: $\mathcal{M}$ works in *weak* $s(n)$ space if, for each accepted input of length n, there exists at least one accepting computation using at most $s(n)$ space [1,15,19].

Here the space used by an individual computation of an alternating machine[1] is, by definition, the maximal number of work tape cells used by any configuration in the tree corresponding to that computation. (For nondeterministic machines, the computation tree reduces to a single computation path.)

The above space notions are equivalent for fully space constructible bounds; once the space bound $s(n)$ can be computed in advance, each computation consuming too much space can be aborted. However, this is not the case of unbounded monotone functions below $\log n$ [8].

It is easy to see that the above definitions are given in increasing order of generality. Thus, the lower bound $\log\log n$ of [15] extends from alternating weak space to all "simpler" cases.

Further, this lower bound is optimal for all combinations of the four space complexity notions above with determinism, nondeterminism, or alternation; it cannot be raised to a function growing faster that $\log\log n$, since we have nonregular languages that can be recognized in space $O(\log\log n)$, even by deterministic strongly bounded machines [11]. The optimality of these lower bounds has later been extended to unary (tally) languages, *i.e.*, to languages over a single-letter alphabet, by exhibiting a unary nonregular language that can be recognized deterministically in strong $O(\log\log n)$ space [2].

The situation is more complicated for one-way machines, *i.e.*, for machines that never move the input head to the left. Here the lower bounds for nonregular languages depend on the type of the space bound, the number of used alternations, and may also differ for binary (general) and unary languages.

At the first glance, unary languages look "simpler", they have very little information contents. However, the recognition of a unary language by a machine with

[1]The reader is assumed to be familiar with the notion of an alternating machine. By a Σ_k- or Π_k-alternating machine we denote a machine making at most $k-1$ switches between existential and universal configurations, starting in an existential (universal, respectively) configuration. For a detailed definition, see also *e.g.* [5,6,9].

sublogarithmic space may potentially be more difficult than recognizing a binary language. Such machine, already unable to fix any input tape position, must also cope with the lack of any structure on the input.

Let us now briefly review the known lower bounds for accepting nonregular languages by *one-way* machines, as they were summarized in [4]. (See also Tab. 1.)

(1) As pointed out in [4], the $\log\log n$ lower bound of [15] for two-way weakly bounded alternating machines must also hold for all "simpler" machines. Thus, by exhibiting a language that can be accepted by a given type of machines in $O(\log\log n)$ space, we automatically get the optimality of the $\log\log n$ lower bound for this type. This way we obtain [4] that $\log\log n$ is the optimal lower bound for accepting nonregular languages for the following types of one-way machines:
   - middle Π_1-alternating machines [21], hence, for all alternating levels above Π_1 as well, and/or the space modes above middle (*i.e.*, accept, weak), in case of binary languages;
   - weak nondeterministic machines with unary inputs [3,4], hence, also for alternating levels above Σ_1, both for binary and unary languages. (The simpler binary case has been proved earlier in [7];)
   - accept Σ_2-alternating machines recognizing unary inputs [3,4], hence, for accept space bounded machines with alternating level above Σ_2, both for unary and binary languages.

(2) It is easy to present a unary nonregular language that can be recognized in strong $O(\log n)$ space, by a deterministic one-way machine. This $\log n$ "upper" bound must also hold for all "more general" machines [4]. Thus, by showing a $\log n$ lower bound for a given type of machines, we get automatically that $\log n$ is the optimal lower bound for accepting nonregular languages for the following types of one-way machines:
   - strongly space bounded alternating machines [20], hence, for deterministic, nondeterministic, Σ_k- or Π_k-alternating machines as well, both for binary and unary languages. (The simpler cases of strong deterministic or nondeterministic machines were first proved in [11, 12] and [13], respectively;)
   - weakly space bounded deterministic machines [1], hence, also for deterministic machines of all space modes (strong, middle, accept, weak), both in case of binary and unary languages;
   - middle alternating machines with unary inputs [3, 17], hence, for deterministic, nondeterministic, Σ_k- or Π_k-alternating machines as well, recognizing unary nonregular languages. This was the first case where the lower bounds for binary and unary nonregular languages did not coincide;
   - Π_1-alternating machines of all types (strong, middle, accept, weak), in case of unary languages [4];

TABLE 1. Alternation power required by one-way machines to accept nonregular languages in sublogarithmic space. An entry of type "$x - y$" indicates that an x-alternating machine of the given type can recognize a nonregular language in space $O(\log\log n)$, while for y-alternating machines we have the lower bound $s(n) \notin o(\log n)$. "A" stands for unrestricted alternation, "D" for determinism.

	Strong	Middle	Accept	Weak
Binary languages	−A	$\Pi_1-\Sigma_1$	$\Pi_1-\Sigma_1$	Σ_1-D, Π_1-D
Unary languages	−A	−A	$\Sigma_2-\Sigma_1$, $\Sigma_2-\Pi_1$	$\Sigma_1-\Pi_1$

– accept space bounded nondeterministic machines [3,4], both in case of unary and binary languages, hence, for middle nondeterministic machines as well. (The simpler case of middle nondeterminism has been proved first in [21].)

The above results are summarized in Table 1. The only case not covered by this table is the lower bound for acceptance of unary nonregular languages by accept space bounded Π_2-alternating one-way machines. In [4], the authors conjectured that the optimal lower bound for these machines is $\log n$. They supported this conjecture by observation that the space used in the universal phase of any computation must be bounded by a constant, for one-way Π_2-alternating machines with accept $o(\log n)$ space, recognizing unary languages. (More precisely, this holds only until the moment in which the input head gets to the right end marker.)

We shall make some further steps in this direction and show that even the subsequent existential phase of any computation must use only a constant amount of space. As a consequence, the language accepted by such machine must be regular. Thus, the entry of Table 1 for accept space and unary languages can be simplified to "$\Sigma_2-\Pi_2$".

The importance of accept one-way space bounds follows from the following facts. It is not hard to see that a nondeterministic two-way machine simultaneously bounded by $s(n)$ space and $i(n)$ input head reversals (the number of times the machine reverses the direction of input head movement) can be simulated by a one-way machine in $O(s(n)\cdot i(n))$ space. What is less known is the fact that, if the bounds for the original machine are of type strong (also middle or accept), then the resulting one-way machine will be *accept* space bounded by $O(s(n)\cdot i(n))$ [3,4]. Thus, accept space bounded one-way machines play an important role for obtaining combined lower bounds on the product space $\times$ reversals.

2. THE LOWER BOUND

We shall now show that one-way Π_2-alternating Turing machines cannot accept unary nonregular languages in accept $o(\log n)$ space. We first recall some basic definitions.

Definition 2.1. A *memory state* of a Turing machine denotes an ordered triple $q = \langle r, u, j\rangle$, where r is a control state, u is a nonblank content of the work tape, and j is an integer denoting a position of the work tape head. A *configuration* is an ordered pair $k = \langle q, i\rangle$, where q is a memory state and i denotes a position of the input head. The *space used* by a memory state or configuration is the length of the nonblank content of the work tape. By a *machine dependent constant* of a machine $\mathcal{M}$, we denote a constant $c>1$, such that c^s bounds the number of different memory states using up to s space, for each $s>1$.

It should be obvious that, for each machine $\mathcal{M}$, there exists a machine dependent constant c with the desired properties.

In the next two lemmas, we shall recall the observation presented in [4], namely, that a Π_2-alternating machine $\mathcal{M}$, with $o(\log n)$ accept space bound, can reach only a finite number of universal memory states on unary inputs.

Lemma 2.1. *Let $\mathcal{M}$ be a one-way Π_2-machine accepting a unary language $\mathcal{L}$ in accept $s(n) \in o(\log n)$ space. Then there exists an $n_0 \in \mathbb{N}$, such that for each $1^n \in \mathcal{L}(\mathcal{M})$, with $n \geq n_0$, any universal memory state reachable by $\mathcal{M}$ on any unary input, without scanning the right end marker, can also be reached on the input 1^n.*

Proof. Recall that $c^{s(n)}$, where c is a machine dependent constant, bounds the number of memory states using up to $s(n)$ work tape cells. For $s(n) \in o(\log n)$, we have an $n_0 \in \mathbf{N}$, such that $c^{s(n)} < n$, for each $n \geq n_0$. Now take an arbitrary $n \geq n_0$, such that $1^n \in \mathcal{L}(\mathcal{M})$.

Let m be a universal memory state reachable by $\mathcal{M}$ on an input 1^k, without scanning the right end marker, for some $k \in \mathbf{N}$. Since $\mathcal{M}$ is a Π_2-machine, any computation path leading to m must go through a sequence of universal memory states $m_0, m_1, \ldots, m_\ell$, such that m_0 is the initial memory state, $m_\ell = m$, $\ell < k$, and m_{i+1} is reached from m_i by a computation traversing exactly one input tape position to the right.

If $\ell < n$, then clearly m is reachable on 1^n, since less than n input tape positions are scanned. If, however, $\ell \geq n$, we can show that m is also reachable at the position ℓ', for some $\ell' < \ell$.

Consider the initial segment $m_0, m_1, \ldots, m_n$. Clearly, for each $i \in \{0, \ldots, n\}$, m_i is reached on the input 1^n. Moreover, since $1^n \in \mathcal{L}(\mathcal{M})$, each m_i belongs to some accepting computation tree. Hence, according to the *accept* space notion, it uses at most $s(n)$ space. Since $c^{s(n)} < n$, there must exist some universal memory states $m_i = m_j$, with $1 \leq i < j \leq n$. Thus, the sequence $m_0, \ldots, m_i, m_{j+1}, \ldots, m_\ell$ also leads to m, this time scanning only ℓ' input tape positions, with $\ell' = \ell + i - j < \ell$.

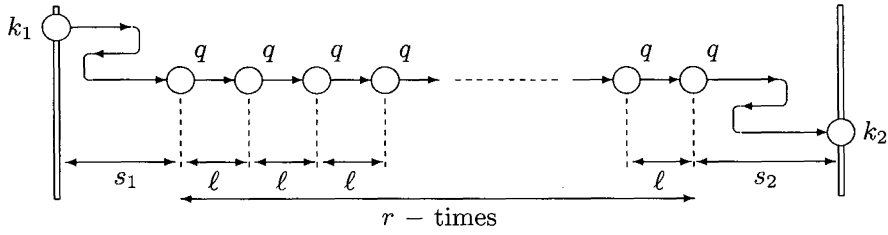

FIGURE 1. Dominant loop.

It is easy to see that, by repeating the above argument, we shall get $\ell' < n$. Thus, m can be reached on the input 1^n. □

Now we can present a simple consequence of the previous lemma:

Lemma 2.2. *Let $\mathcal{M}$ be a one-way Π_2-machine accepting an infinite unary language $\mathcal{L}$ in accept $s(n) \in o(\log n)$ space. Then the set of universal memory states reachable by $\mathcal{M}$, without scanning the right end marker, is finite.*

Proof. Since $\mathcal{L}$ is infinite, we can find a sufficiently large n_0 with $1^{n_0} \in \mathcal{L}(\mathcal{M})$, and hence, by Lemma 2.1, each universal memory state reachable on any unary input without scanning the right end marker must also be reached on the input 1^{n_0}. Since $1^{n_0} \in \mathcal{L}(\mathcal{M})$, by the accept space notion the space used by this memory state must be bounded by $s(n_0)$. The number of such memory states is bounded by $c^{s(n_0)}$, for some constants c and n_0. □

The statement of Lemma 2.2 cannot be extended to universal memory states reachable after the input head hits the right end marker. Consider, for example, a machine that rejects 1^n, for n even, as follows. After some initial universal branching, one computation path traverses the entire input and, for n even, it rejects 1^n by counting from zero to infinity on the work tape.

We shall also need the following technical lemma, proved first in [8] (Th. 1).

Lemma 2.3. *Let $\mathcal{M}$ be a Turing machine with a machine dependent constant c, as introduced in Definition 2.1. Then, for each computation path of $\mathcal{M}$ on the input 1^n,*

- *beginning in a configuration $k_1 = \langle q_1, \mu_1 \rangle$ and ending in $k_2 = \langle q_2, \mu_2 \rangle$, with $0 \leq \mu_1 < \mu_2 \leq n+1$;*
- *using at most space s, with $(c^s)^6 = E^6 < \mu_2 - \mu_1$;*
- *such that the input tape positions μ_1 and μ_2 are visited only in k_1 and k_2,*

there exists an equivalent path connecting k_1 with k_2, such that

- *having traversed s_1 input tape positions to the left;*
- *it gets into a memory state q in which a loop of length ℓ is iterated r times;*
- *and then it traverses the remaining input tape segment of length s_2;*

for some s_1, ℓ, r, s_2, and q, satisfying $E^2+1 \leq s_1 \leq E^4$, $E^2+1 \leq s_2 \leq E^4$, and $1 \leq \ell \leq E$. (See Fig. 1.)

Though the proof in [8] considers nondeterministic machines only, it is easy to see that the argument holds for the alternating machines as well; the statement of the theorem does not say anything about conditions of acceptance, but only about reachability among configurations by computation paths marching across the input tape.

Now we are ready to present the main theorem of the paper:

Theorem 2.1. *Let $\mathcal{M}$ be a one-way Π_2-machine accepting an infinite unary language $\mathcal{L}$ in accept $s(n) \in o(\log n)$ space. Then there exists an $n_0 \in \mathbf{N}$ such that, for each $n \geq n_0$, we have that if $\mathcal{M}$ accepts the input 1^n, using at least h space, for some $h \leq s(n)$, then there exists $n' < n$, such that $\mathcal{M}$ accepts $1^{n'}$, using at least h space.*

Proof. First, we need to introduce some notation:

- c' is the number of $\mathcal{M}$'s machine instructions;
- M is an upper bound for the number of different universal memory states, reachable without scanning the right end marker. This number is finite, due to Lemma 2.2. Thus, M is a machine dependent constant;
- E_n is an upper bound on the number of different memory states not using space above $s(n)$. Clearly, we can take $E_n = d^{s(n)}$, where d is some "new" machine dependent constant, sufficiently large, so that $d \geq 12M!$

Without loss of generality, we can assume that

$$s(n) \geq 1, \quad c' \geq 2, \quad M \geq 2, \quad d \geq 12M! \geq 12, \tag{1}$$

since it is easy to bound the above quantities from below. Then we also have

$$E_n = d^{s(n)} > 1, \quad c' \cdot M^2 > 1. \tag{2}$$

Since $\lim_{n\to\infty} s(n)/\log n = 0$, there must exist an $n_0 \in \mathbf{N}$, such that for each $n \geq n_0$, $(d^{s(n)})^{c' \cdot M^2+5} < n$. But then $12M! \cdot (d^{s(n)})^{c' \cdot M^2+4} < n$, using (1), and hence also $2M! \cdot (d^{s(n)})^{c' \cdot M^2+2} + 4(d^{s(n)})^4 + 2M < n$, using (2). This gives

$$\frac{n}{2} > \frac{\frac{n}{2} - 2E_n^4 - M}{E_n} > M! \cdot E_n^{c' \cdot M^2+1}, \tag{3}$$

for each $n \geq n_0$, using (2). It can be easily seen that then also

$$\frac{n}{2} > E_n^6 > E_n > M. \tag{4}$$

Now, let $1^n \in \mathcal{L}$, for arbitrary $n \geq n_0$. Let us consider some fixed accepting computation tree of the machine $\mathcal{M}$ for the input 1^n. We can put each computation path of this computation tree into one of the following two groups. (See Fig. 2.)

(a) Accepting computation paths alternating to some existential configurations earlier than the input head crosses the position M to the right.

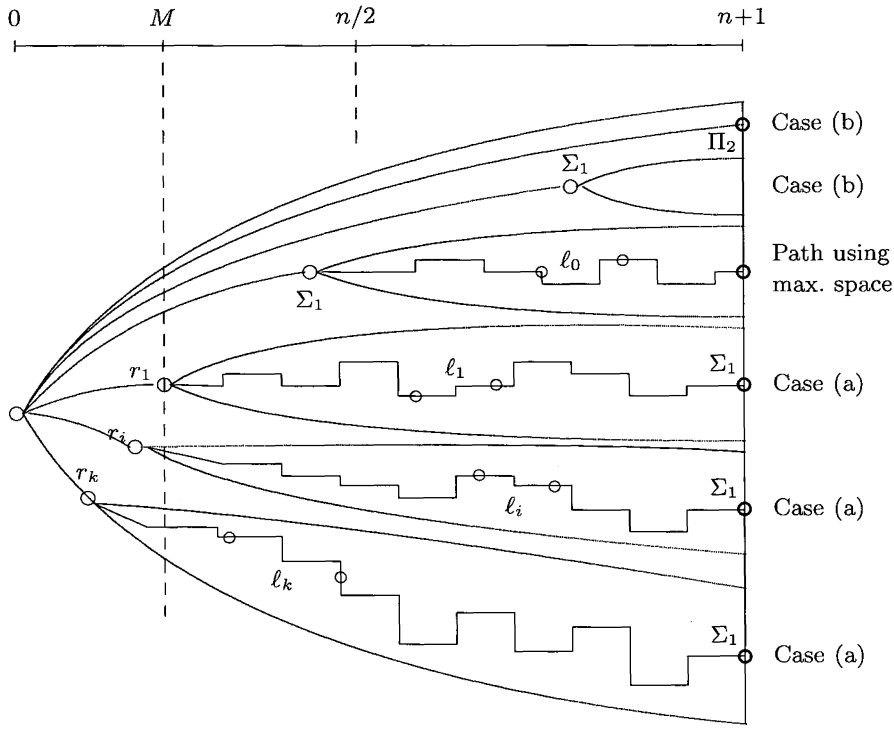

FIGURE 2. Computation paths of the machine $\mathcal{M}$.

(b) Accepting computation paths that do not alternate earlier than the input head crosses the position M. (That is, either the computation path alternates after crossing the position M, or it does not alternate at all.)

First, consider the group (a). Let $\{r_1, \ldots, r_k\}$ be the set of all existential configurations that occur – as the first existential one – along any path of the group (a). Now, for each r_i, we can select one C_i, an accepting computation path from the group (a), so that r_i is the first existential configuration along the path C_i.

The length of the input tape traversed during the existential phase of C_i is at least $n-M > n/2 > E_n$, by (4). Hence, there must exist a memory state along the existential phase of C_i that is repeated. Thus, the existential phase of C_i contains a *cycle* of the length ℓ_i, with

$$\ell_i \leq E_n, \quad \text{for each } i \in \{1, \ldots, k\}. \tag{5}$$

Note that r_i, for each $i \in \{1, \ldots, k\}$, is reachable from some universal memory state *in one computation step*, by executing a single instruction. Since there are at use only c' different machine's instructions and M different universal memory states, that can differ in M distinct input head positions, the number of existential

configurations in $\{r_1, \ldots, r_k\}$ is bounded by

$$k \leq c' \cdot M^2. \tag{6}$$

This also bounds the number of different cycles in $\{\ell_1, \ldots, \ell_k\}$.

Let us also consider the accepting computation path $\mathcal{C}_0$, on which the machine uses maximal space. (We do not care whether $\mathcal{C}_0$ belongs to the group (a) or (b).) We suppose that $\mathcal{C}_0$ uses h space on the input 1^n, for some $h \leq s(n)$. There are two cases:

(c1) the accepting computation path $\mathcal{C}_0$ using maximal space alternates into the existential phase sooner than it reaches the position $n/2$;

(c2) $\mathcal{C}_0$ alternates after crossing the position $n/2$ to the right, or it does not alternate at all.

In the Case (c1), the existential phase of $\mathcal{C}_0$ traverses more than $n/2$ positions along the input, and hence it must contain a cycle of length $\ell_0 \leq E_n < n/2$, by the same argument as was used for $\mathcal{C}_1, \ldots, \mathcal{C}_k$, using (4). In the Case (c2), we can find the cycle of length ℓ_0 with the desired properties in the universal phase of $\mathcal{C}_0$. (Fig. 2 illustrates the Case (c1).) In either case,

$$\ell_0 \leq E_n. \tag{7}$$

Now we shall show that the machine $\mathcal{M}$ accepts $1^{n'}$, where

$$n' = n - M! \cdot \prod_{i=0}^{k} \ell_i. \tag{8}$$

Using (5, 7), and (6), we get that

$$M! \cdot \prod_{i=0}^{k} \ell_i \leq M! \cdot E_n^{c' \cdot M^2 + 1}. \tag{9}$$

Hence, by (3) and (4), $M! \cdot \prod_{i=0}^{k} \ell_i < n/2$, which gives that

$$n' > \frac{n}{2} > M. \tag{10}$$

Similarly as for the input 1^n, the computation paths on $1^{n'}$ can also be arranged into two groups (a) and (b), depending on whether the machine alternates earlier than it crosses the position M. By analyzing computation paths in either of these groups, we shall show that $\mathcal{M}$ successfully accepts the input $1^{n'}$.

Case (a): a computation path $\mathcal{C}$ alternating to some existential configuration earlier than the input head crosses the position M to the right. Since $n' > M$, by (10), such computation path, on the input $1^{n'}$, must share the first existential configuration with an ith computation path $\mathcal{C}_i$ on the input 1^n, for some

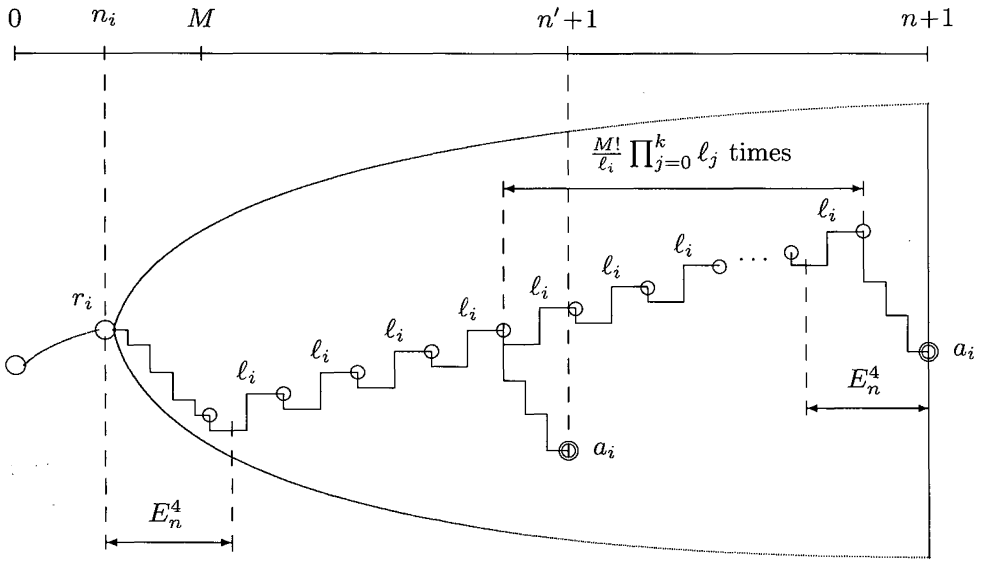

FIGURE 3. Computation path which is switched into an existential configuration in a distance $n_i \leq M$ from the left end marker.

$i \in \{1, \ldots, k\}$. (See Figs. 3 and 2.) Thus, the memory state, in which $\mathcal{C}$ alternates, is r_i. Suppose that $\mathcal{C}_i$ alternates at a position $n_i \leq M$. Recall that $\mathcal{C}_i$ is accepting on 1^n. Therefore, the existential phase of the path $\mathcal{C}_i$ runs from r_i to some accepting memory state a_i, at the position $n+1$. (Without loss of generality, we assume that $\mathcal{M}$ accepts always at the right end marker.) Now we shall show that some existential path, beginning in r_i on the input $1^{n'}$, must end up in an accepting memory state.

First, since $n_i \leq M$, we have that the segment traversed in the existential phase of $\mathcal{C}_i$ is of length $n-n_i \geq n-M \geq n/2 > E_n^6$, using (4). Hence, by Lemma 2.3, we can assume without loss of generality that the cycle of length $\ell_i \leq E_n$ is iterated along the dominant portion of the existential phase of $\mathcal{C}_i$, with the exception of some short segments of lengths at most E_n^4, placed at the beginning and at the end. Thus, this cycle is iterated at least H_i times, where

$$H_i \geq \left\lfloor \frac{n - n_i - 2E_n^4}{\ell_i} \right\rfloor \geq \left\lfloor \frac{\frac{n}{2} - 2E_n^4 - M}{E_n} \right\rfloor \geq \left\lfloor M! \cdot E_n^{c' \cdot M^2 + 1} \right\rfloor,$$

using $n_i \leq M$, equation (5), and equation (3). Note that the last quantity is always an integer, and hence

$$H_i \geq \left\lfloor M! \cdot E_n^{c' \cdot M^2 + 1} \right\rfloor = M! \cdot E_n^{c' \cdot M^2 + 1} \geq M! \cdot \prod_{j=0}^{k} \ell_j \geq \frac{M!}{\ell_i} \cdot \prod_{j=0}^{k} \ell_j,$$

using (9) and $\ell_i \geq 1$. Note also that ℓ_i divides $\prod_{j=0}^{k} \ell_j$.

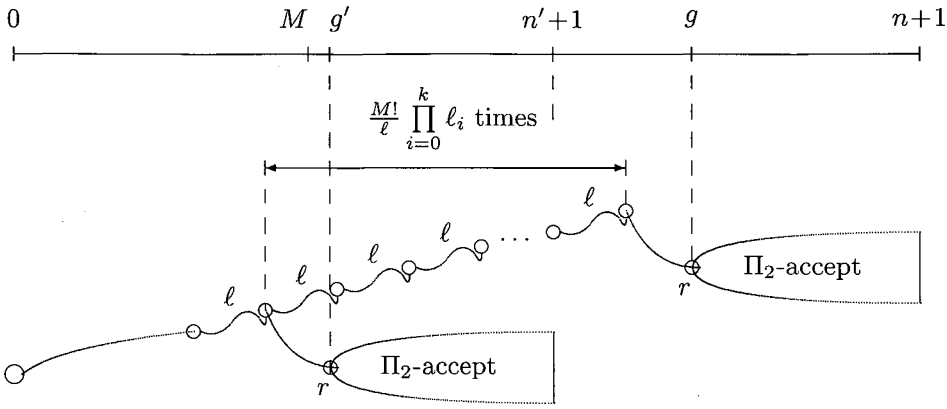

FIGURE 4. Computation paths not alternating earlier than the input head crosses the position M.

Therefore, if we omit the last $M!/\ell_i \cdot \prod_{j=0}^{k} \ell_j$ iterations of this cycle of length ℓ_i, the memory state a_i will be reached exactly at the position $n - M! \cdot \prod_{j=0}^{k} \ell_j = n'$. Recall that a_i was an accepting memory state and hence we are done.

Thus, any computation path alternating to some existential configuration earlier than it crosses the position M must end up successfully for the input $1^{n'}$.

Case (b): computation paths that do not alternate earlier than the input head crosses the position M. Let r denote an arbitrary universal memory state at the position $g' = M+1$, reachable by $\mathcal{M}$ from the initial configuration on the input $1^{n'}$. (See Fig. 4.)

We call r Π_2-*accepting* for the input $1^{n'}$, if each computation path beginning in r, after a finite number of steps, either halts in an accepting configuration or alternates to some existential configuration, from which we have at least one computation path ending in an accepting configuration. By definition, r is Π_2-*rejecting*, if it is not Π_2-accepting.

If all universal memory states reachable by $\mathcal{M}$ at the position $g' = M+1$ are Π_2-accepting for $1^{n'}$, then we are done; $\mathcal{M}$ accepts the input $1^{n'}$, since the paths alternating before getting to the position g' have already been discussed in Case (a). So assume, for contradiction, that some universal memory state r, reachable at the position $g' = M+1$, is Π_2-rejecting for the input $1^{n'}$.

By (10) and (8), $g' < n' < n$, and hence r is reachable on the input 1^n as well. Since $g' > M$ and there are at most M different universal memory states not scanning the right end marker, there must exist a cycle of length $\ell \leq M$ along the way from the initial configuration to r. Note that ℓ, the length of the cycle, divides $M!$. Thus, by repeating this cycle $M!/\ell \cdot \prod_{i=0}^{k} \ell_i$ more times, we shall get to the same memory state r, placed at the position $g = g' + M! \cdot \prod_{i=0}^{k} \ell_i$ of the input 1^n.

Note that $n - g = n' - g'$. Therefore, if r is Π_2-rejecting for the input $1^{n'}$, then it must be Π_2-rejecting for the input 1^n, since it is placed at the same distance from

the *right* end markers of both inputs. But this is a contradiction, since a universal Π_2-rejecting configuration cannot be reached from the initial configuration on the input $1^n \in \mathcal{L}(\mathcal{M})$.

Thus, all universal memory states reachable by $\mathcal{M}$ at the position $g' = M+1$ must be Π_2-accepting for $1^{n'}$.

As a consequence of Cases (a) and (b), the machine $\mathcal{M}$ accepts the input $1^{n'}$. We only have to show that the space used by $\mathcal{M}$ on the input $1^{n'}$ is at least h, the size of the space used on the longer input 1^n.

Recall that $\mathcal{M}$ uses the maximal space h along the computation path C_0, and that there are two cases; (c1) either the path C_0 alternates into the existential phase sooner than it reaches the position $n/2$ on the input 1^n, (c2) or C_0 alternates after crossing the position $n/2$ to the right, or it does not alternate at all. In either case, C_0 is an accepting computation path. In addition, the "longer" of the phases (universal or existential, depending on which one traverses more than $n/2$ input tape positions) contains a cycle of length $\ell_0 \leq E_n < n/2$.

Case (c1): C_0 alternates sooner than it reaches the position $n/2$. Let a_0 denote the halting accepting memory state of C_0, placed at the right end marker. The space used by a_0 is h. Let r_0 denote the first existential memory state along C_0. By assumption, r_0 is placed at the position $p \leq n/2$.

Using (4), we get that $n-p \geq n/2 > E_n^6$. Hence, by Lemma 2.3, we can assume without loss of generality that the iteration of the cycle of length $\ell_0 \leq E_n$ dominates the existential phase of C_0, except for some short segments placed at the beginning and at the end, of lengths at most E_n^4. Thus, this cycle is iterated at least H_0 times, where

$$H_0 \geq \left\lfloor \frac{n-p-2E_n^4}{\ell_0} \right\rfloor \geq \left\lfloor \frac{\frac{n}{2}-2E_n^4-M}{E_n} \right\rfloor \geq \left\lfloor M! \cdot E_n^{c' \cdot M^2+1} \right\rfloor,$$

using $p \leq n/2$, equation (7), and equation (3). Then, by the argument similar to Case (a),

$$H_0 \geq M! \cdot \prod_{j=0}^{k} \ell_j \geq \frac{M!}{\ell_0} \cdot \prod_{j=0}^{k} \ell_j.$$

Therefore, if we omit the last $M!/\ell_0 \cdot \prod_{j=0}^{k} \ell_j$ iterations of the cycle of length ℓ_0, the memory state a_0 will be reached at the position $n - M! \cdot \prod_{j=0}^{k} \ell_j = n'$. Since the space used by the accepting memory state a_0 is h, the machine $\mathcal{M}$ must be using at least that much space on the input $1^{n'}$.

Case (c2): C_0 does not alternate sooner than it reaches the position $n/2$. This time the universal phase traverses the distance greater than $n/2$. Further, the universal phase contains also the above cycle of length ℓ_0. Thus, now we can use

the argument of (c1) for the universal phase of C_0, to omit the last $M!/\ell_0 \cdot \prod_{j=0}^{k} \ell_j$ iterations of the cycle. Again, this shifts the accepting memory state a_0 with maximal space to the position n'.

This completes the proof of theorem. □

Now we can state the following lower bound:

Theorem 2.2. *Let $\mathcal{M}$ be a one-way Π_2-machine accepting a unary language $\mathcal{L}$ in accept $s(n) \in o(\log n)$ space. Then the language $\mathcal{L}$ is regular.*

Proof. If $\mathcal{L}$ is finite, then it is clearly regular. Now assume that $\mathcal{L}$ is infinite.

Then, by Theorem 2.1, there exists a constant n_0 such that, for each $n \geq n_0$ satisfying $1^n \in \mathcal{L}$, there exists $n' < n$ satisfying $1^{n'} \in \mathcal{L}$, such that the space used by $\mathcal{M}$ on the input $1^{n'}$ is at least h, the size of the space used on the longer input 1^n.

From this follows that no accepting computation of $\mathcal{M}$, on any input, uses more space than

$$\tilde{h} = \max\{s(0), s(1), \ldots, s(n_0)\}. \tag{11}$$

Otherwise, take the minimal $n \in \mathbf{N}$ such that $1^n \in \mathcal{L}$, for which the space used by some accepting computation of $\mathcal{M}$ is $h > \tilde{h}$. If $n \leq n_0$, we have a contradiction with $h > \tilde{h} = \max\{s(0), s(1), \ldots, s(n_0)\}$, since, by the accept space notion, $h \leq s(n)$. If $n > n_0$, then, by Theorem 2.1, there exists $n' < n$ with the property that $\mathcal{M}$ accepts $1^{n'}$ and uses space of size at least $h > \tilde{h}$, which is a contradiction with the minimal length of n.

Thus, no accepting computation of $\mathcal{M}$ will ever use more space than $\tilde{h}$, which is a fixed constant. This upper bound does not concern *rejecting* computations. (*Cf.* remark below the proof of Lem. 2.2.)

Nevertheless, for the given machine $\mathcal{M}$ and any given constant $\tilde{h}$, we can construct a new machine $\mathcal{M}'$ having, instead of a standard semi-infinite work tape, a read-write work tape of a finite length, consisting only of $\tilde{h}$ work tape cells (initially empty, containing blank symbols), enclosed between two work tape end markers. The simulation of $\mathcal{M}$ by $\mathcal{M}'$ is straightforward, until the head visits the right end marker on the work tape. If this marker is detected, $\mathcal{M}'$ rejects.

It is easy to see that for properly chosen $\tilde{h}$, $\mathcal{M}$ and $\mathcal{M}'$ will recognize the same language $\mathcal{L}$, since accepting computations will never try to use more space than $\tilde{h}$, and hence will never visit the work tape end marker.

Note that $\mathcal{M}'$ has a work tape of a fixed finite length, and therefore it can be kept in a finite-state control. Thus, we can replace $\mathcal{M}'$ by an equivalent one-way Π_2-alternating finite-state automaton. This implies that $\mathcal{L}$ is regular. (For an optimal simulation of a one-way alternating automaton by a deterministic automaton, we refer the reader to [18].) □

3. Concluding remarks

It should be observed that the statement of Theorem 2.2 does not give, automatically, a finite-state automaton for any given one-way Π_2-machine. This is possible only if the value of $\tilde{h}$, defined by (11), can be *computed effectively*.

For example, consider any standard enumeration of Turing machines $T_0, T_1, T_2, \ldots$, and, for each $\nu \in \mathbf{N}$, define a language

$$\mathcal{L}_\nu = \{1^n;\ T_\nu \text{ halts on the input } \nu \text{ in less than } n \text{ computation steps}\}.$$

Clearly, for a fixed $\nu \in \mathbf{N}$, we can construct a deterministic one-way machine $\mathcal{M}_\nu$ that simulates T_ν on the input ν and, after each simulated step, it moves one input tape position to the right. The acceptance depends on whether the simulation of T_ν terminates sooner than the end of the input tape is reached.

If T_ν halts on the input ν, the space used by $\mathcal{M}_\nu$ is determined by the space used by T_ν on ν, *i.e.*, a constant independent from n. If T_ν does not halt, then no input is accepted by $\mathcal{M}_\nu$, and hence, by the accept, middle, or weak space notion, the space used by any computation of $\mathcal{M}_\nu$ does not count. In either case, $\mathcal{M}_\nu$ works in $O(1)$ space. However, a construction of an equivalent finite automaton requires to decide whether T_ν halts on the input ν, which is an undecidable problem.

Nevertheless, the construction of Theorem 2.2 does guarantee, for a sufficiently large $\tilde{h}$, that the original machine $\mathcal{M}$ and the resulting finite-state automaton are equivalent, and hence the corresponding language must be regular.

The lower bound presented in Theorem 2.2 does not hold for binary languages, cannot be raised above $\log n$, nor can it be used for higher alternation levels. For example, the language

$$\mathcal{L} = \{1^n;\ \text{the first number not dividing } n \text{ is a power of } 2\}$$

can be accepted by a one-way Σ_2-machine in accept $O(\log\log n)$ space. (For proof and other counterexamples, see also [3, 4].) Thus, the entry of Table 1 for the accept space and unary languages has been simplified to "$\Sigma_2 - \Pi_2$". This shows that Σ_2-alternation is more powerful than Π_2-alternation for space bounded one-way machines with unary inputs.

The above observation concerns recognizing unary nonregular languages with sublogarithmic *accept* space, *i.e.*, the worst cost of acceptance. For the *best cost* of acceptance (weak space mode), the corresponding alternation powers are shifted one level down, to "$\Sigma_1 - \Pi_1$".

A similar phenomenon has been observed for pushdown automata with an auxiliary work tape of size $\log n$; with Π_2-alternating machines corresponding to co–NP, while Σ_2-alternation is more powerful, it corresponds to PSPACE [16].

Finally, Theorem 2.2 cannot be extended from one-way machines to two-way devices; the corresponding space lower bound drops from $\log n$ to $\log\log n$. We are able to accept unary nonregular languages with $O(\log\log n)$ space even by

Π_1-alternating machines, in middle, accept, or weak space mode. In addition, the input head movement can be restricted to so-called "sweeping" (the input head movement reverses the direction only at the end markers), with the number of input head reversals bounded by $f(n)$, where $f(n)$ represents any (arbitrarily slow growing) unbounded recursive function [10].

Though $f(n)$, the number of input head reversals, may be arbitrarily slow growing, while keeping the upper bound for space at the level $O(\log \log n)$ [10], it is not clear whether the space lower bound for a unary nonregular language acceptance is $\log n$ or $\log \log n$, if $f(n)$ is bounded by a constant. This problem is open for Π_2-alternating accept space bounded machines, Π_1-alternating middle, accept, or weak space, and also for the entire alternating hierarchy of middle space bounded machines.

REFERENCES

[1] M. Alberts, Space complexity of alternating Turing machines, in *Proc. Fund. Comput. Theory.* Springer-Verlag, *Lecture Notes in Comput. Sci.* **199** (1985) 1-7.

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

[3] A. Bertoni, C. Mereghetti and G. Pighizzini, Strong optimal lower bounds for Turing machines that accept nonregular languages, in *Proc. Math. Found. Comput. Sci.* Springer-Verlag, *Lecture Notes in Comput. Sci.* **969** (1995) 309-18.

[4] A. Bertoni, C. Mereghetti and G. Pighizzini, *Space and reversal complexity of nonregular languages.* Technical Report, University of Milano (1998).

[5] A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation. *J. Assoc. Comput. Math.* **28** (1981) 114-33.

[6] P. van Emde Boas, *Machine models and simulations*, edited by J. van Leeuwen, *Handbook of Theoretical Computer Science.* Elsevier Science (1989).

[7] R. Freivalds, On the work time of deterministic and nondeterministic Turing machines. *Latv. Mat. Ezhegodnik* **23** (1979) 158-65 (in Russian).

[8] V. Geffert, Nondeterministic computations in sublogarithmic space and space constructibility. *SIAM J. Comput.* **20** (1991) 484-98.

[9] V. Geffert, A hierarchy that does not collapse: Alternations in low level space. *RAIRO: Theoret. Informatics Appl.* **28** (1994) 465-512.

[10] V. Geffert, C. Mereghetti and G. Pighizzini, Sublogarithmic bounds on space and reversals. *SIAM J. Comput.* **28** (1999) 325-40.

[11] J. Hartmanis, P.M. Lewis II and R.E. Stearns, Hierarchies of memory limited computations, in *IEEE Conf. Record on Switching Circuit Theory and Logical Design* (1965) 179-90.

[12] J. Hartmanis, P.M. Lewis II and R.E. Stearns, Memory bounds for recognition of context-free and context-sensitive languages, in *IEEE Conf. Record on Switching Circuit Theory and Logical Design* (1965) 191-202.

[13] J.E. Hopcroft and J.D. Ullman, Some results on tape-bounded Turing machines. *J. Assoc. Comput. Mach.* **16** (1969) 168-77.

[14] J. Hromkovič, B. Rovan and A. Slobodová, Deterministic *versus* nondeterministic space in terms of synchronized alternating machines. *Theoret. Comput. Sci.* **132** (1994) 319-36.

[15] K. Iwama, ASPACE($o(\log \log n)$) is regular. *SIAM J. Comput.* **22** (1993) 136-46.

[16] B.E. Ladner, R.J. Lipton and L.R. Stockmeyer, Alternation bounded auxiliary pushdown automata. *Inform. and Control* **62** (1984) 93-108.

[17] C. Mereghetti and G. Pighizzini, A remark on middle space bounded alternating Turing machines. *Inform. Process. Lett.* **56** (1995) 229-32.

[18] C. Mereghetti and G. Pighizzini, Optimal simulations between unary automata, in *Proc. Symp. Theoret. Aspects Comput. Sci.* Springer-Verlag, *Lecture Notes in Comput. Sci.* **1373** (1998) 139-149 (to appear in *SIAM J. Comput.*).

[19] W.J. Savitch, Relationships between nondeterministic and deterministic tape complexities. *J. Comput. System Sci.* **4** (1970) 177-92.

[20] I.H. Sudborough, Efficient algorithms for path system problems and applications to alternating and time-space complexity classes, in *Proc. IEEE Symp. Found. of Comput. Sci.* (1980) 62-73.

[21] A. Szepietowski, Remarks on languages acceptable in $\log \log n$ space. *Inform. Process Lett.* **27** (1988) 201-3.

Communicated by J. Hromkovič.
Received June 30, 2000. Accepted November 28, 2000.