SIAM J. COMPUT.
Vol. 28, No. 1, pp. 325–340

SUBLOGARITHMIC BOUNDS ON SPACE AND REVERSALS*

VILIAM GEFFERT†, CARLO MEREGHETTI‡, AND GIOVANNI PIGHIZZINI‡

Abstract. The complexity measure under consideration is SPACE × REVERSALS for Turing machines that are able to branch both existentially and universally. We show that, for any function $h(n)$ between $\log\log n$ and $\log n$, Π_1SPACE × REVERSALS$(h(n))$ is separated from Σ_1SPACE × REVERSALS$(h(n))$ as well as from coΣ_1SPACE×REVERSALS$(h(n))$, for *middle*, *accept*, and *weak* modes of this complexity measure. This also separates determinism from the higher levels of the alternating hierarchy. For "well-behaved" functions $h(n)$ between $\log\log n$ and $\log n$, almost all of the above separations can be obtained by using *unary* witness languages.

In addition, the construction of separating languages contributes to the research on minimal resource requirements for computational devices capable of recognizing nonregular languages. For any (arbitrarily slow growing) unbounded monotone recursive function $f(n)$, a nonregular unary language is presented that can be accepted by a *middle* Π_1 alternating Turing machine in $s(n)$ space and $i(n)$ input head reversals, with $s(n) \cdot i(n) \in \mathcal{O}(\log\log n \cdot f(n))$. Thus, there is no exponential gap for the optimal lower bound on the product $s(n) \cdot i(n)$ between unary and general nonregular language acceptance—in sharp contrast with the one-way case.

Key words. alternation, computational complexity, computational lower bounds, formal languages

AMS subject classifications. 68Q05, 68Q15, 68Q68

PII. S0097539796301306

1. Introduction.

In the last few years, some exciting results have been obtained in the field of space bounded computations. First of all, we have a surprisingly short proof that nondeterministic space is closed under complementation [12, 21]. Among others, this result has a fundamental consequence on alternation, namely, the collapse of the alternating space hierarchy to the Σ_1 level.

It is worth noticing that these results were obtained for *strong* space complexity classes with $s(n) \in \Omega(\log n)$. (A Turing machine M works in *strong* space $s(n)$ if no reachable configuration, on any input of length n, uses space above $s(n)$.) The closure under complementation does not hold for *weak* space complexity classes above $\log n$ [23]. (*weak* space $s(n)$: for any accepted input of length n, there exists at least one accepting computation not using more space than $s(n)$.) Below $\log n$, moreover, the above alternating hierarchy is infinite [2, 6, 15]. For the sublogarithmic world, many other results, almost self-evident in the superlogarithmic case, either do not hold or the proofs are quite different and are often highly involved.

In this paper, we continue in this line of research toward the simplest possible nonregular complexity classes by investigating Turing machines having sublogarithmic bounds on the product of *space*—in its different definitions—by the *number of input head reversals*. Turing machines working even within such limited resources can still

*Received by the editors March 29, 1996; accepted for publication (in revised form) by J. Hartmanis February 18, 1997; published electronically June 15, 1998.

http://www.siam.org/journals/sicomp/28-1/30130.html

†Katedra Matematickej Informatiky, Univerzita P.J. Šafárika, Jesenná 5, 04154 Košice, Slovakia (geffert@kosice.upjs.sk). This research was supported by the Slovak Grant Agency for Science (VEGA), under contract "Combinational Structures and Complexity of Algorithms."

‡Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, via Comelico 39, 20135 Milano, Italy (mereghc@dsi.unimi.it, pighizzi@dsi.unimi.it). This research was partially supported by Ministero dell'Università e della Ricerca Scientifica e Tecnologica (MURST).

recognize nonregular languages [3, 4], while the corresponding bound on the product of space by *work head* reversals must be at least linear [9].

Besides *strong* and *weak* space, mentioned above, some intermediate measures have also been proposed. (*accept* space $s(n)$: all accepting computations obey the space bound, *middle* space $s(n)$: all computations obey the space bound for each accepted input of length n. For more precise definitions, see section 2.) Such differences are irrelevant for fully space constructible bounds above $\log n$, but several results (see, e.g., [23, 4, 7]) witness that one must pay special attention to the actual definition when dealing with such limited resources as is sublogarithmic space.

For the product of space by input head reversals, we are able to show several separation results that are still unknown if sublogarithmic bounds *on space only* are considered. Namely, for each of the modes $c \in \{middle, accept, weak\}$ and any function $h(n)$ between $\log\log n$ and $\log n$, c-Π_1SPACE$\times$REVERSALS$(h(n))$ is separated from c-Σ_1SPACE$\times$REVERSALS$(h(n))$ and, somewhat more surprisingly, also from *co-c*-Σ_1SPACE$\times$REVERSALS$(h(n))$. (Here c-XSPACE$\times$REVERSALS$(h(n))$ denotes the class of languages accepted by $X \in \{\Sigma_k, \Pi_k\}$ machines of type $c \in \{strong, middle, accept, weak\}$ in $s(n)$ space and $i(n)$ input head reversals, satisfying $s(n) \cdot i(n) \in \mathcal{O}(h(n))$. We add prefix "*co-*" for complements of such languages, and we use $X = \mathrm{D}$ for deterministic Turing machines, i.e., for $\Sigma_0 = \Pi_0$.)

In other words, for *middle*, *accept*, or *weak* space$\times$input head reversals bounded Turing machines, the class of languages accepted by machines making only universal decisions does not coincide with the class of complements of languages recognizable by machines making only existential decisions. Further, we get that *weak*-DSPACE$\times$REVERSALS$(h(n))$ is properly included in *weak*-Π_1-, *co-weak*-Π_1-, *weak*-Σ_1-, and in *co-weak*-Σ_1SPACE$\times$REVERSALS$(h(n))$, for each $h(n)$ between $\log\log n$ and $\log n$.

The input head motion for machines accepting nonregular languages has been studied in [3, 4]. It turns out that the minimal resource requirements for machines accepting *unary*[1] languages become important: a recognizer, already having too little space to remember an input head position, must also cope with the lack of any structure on the input tape. So the problem arises of whether these results hold even if the corresponding language classes are restricted to unary languages. Using an additional assumption that the function $h(n)$ is "well behaved," we are able to show that the above separations hold even in the case of unary languages, except for the separation of *weak*-Π_1SPACE$\times$REVERSALS$(h(n))$ from *co-weak*-Σ_1SPACE$\times$REVERSALS$(h(n))$, which we leave as an open problem.

Here "well behaved" means $h(n) \in o(\log n)$ with $\frac{h(n)}{\log\log n}$ unbounded and monotone increasing. We only require $h(n)$ to be recursive but do not claim any kind of space constructibility.

The above separations are obtained by exhibiting, for any (arbitrarily slow growing) unbounded monotone recursive function $f(n)$, a nonregular unary language $\mathcal{L}_f$ that can be accepted by a *middle* $s(n)$ space and $i(n)$ input head reversals bounded Π_1 machine with $s(n) \cdot i(n) \in \mathcal{O}(\log\log n \cdot f(n))$. On the other hand, using mainly number theoretical and pumping arguments, we show that, for every $h(n) \in o(\log n)$, $\mathcal{L}_f$ does not belong to *weak*-Σ_1SPACE$\times$REVERSALS$(h(n))$.

The complexity of recognizing $\mathcal{L}_f$ also gives meaningful insights into the study of *minimal resource requirements* for computational devices recognizing nonregular languages, an important research area dating back to works by J. Hartmanis, P. Lewis,

[1] That is, built over a single letter alphabet.

and R. Stearns in 1965. In [14, 19, 10], the authors settled the problem of determining the minimal *strong* space requirement for one–way and two–way, deterministic and nondeterministic Turing machines that accept nonregular languages. Subsequently, the same problem has been widely studied for other and more general paradigms of computation and space notions (for *strong* alternation in [20], for *middle* machines in [22], for *accept* machines in [4], and for *weak* machines in [1, 13]). The analysis of computational lower bounds for nonregular languages is tightly related to the world of sublogarithmic space (see, e.g., [23, 7]) and, more particularly, plays an important role in revealing sharp differences among various space definitions [4].

In this regard, we shall concentrate on the problem of determining the optimal lower bound on the product space×input head reversals for *middle* alternating Turing machines that accept unary nonregular languages. The reason why we focus on the *middle* mode of acceptance is that *one-way middle* alternating Turing machines exhibit a very interesting behavior having no analogue in any of the other space bounded computational models considered, e.g., in [23, 4]. On the one hand, we have an optimal $\log\log n$ space lower bound for nonregular languages built on binary alphabets (hence, on general alphabets as well) [22]. On the other hand, a tight $\log n$ space lower bound is proved in [16] whenever we restrict our machines to accept *unary* nonregular languages (see also Table 1.1).

A problem left open in [4] asks whether the same gap situation holds for the lower bound on $s(n) \cdot i(n)$ for *middle* space×input head reversals bounded alternating machines accepting nonregular languages. For such machines, a tight $\log\log n$ lower bound on $s(n) \cdot i(n)$ for general nonregular languages is observed in [4]. Should we expect a corresponding exponential gap when accepting unary nonregular languages, as in the one-way case?

Here we provide a negative answer to this open question since, as stated before, for any unbounded monotone recursive function $f(n)$, we have a unary nonregular language $\mathcal{L}_f$, recognizable by a *middle* space×input head reversals bounded alternating Turing machine satisfying $s(n) \cdot i(n) \in \mathcal{O}(\log\log n \cdot f(n))$. With $f(n)$ being *arbitrarily slow growing*, we can approach the optimal $\log\log n$ lower bound for binary languages as much as we like.

Though this does not completely prove the optimality of the lower bound $s(n) \cdot i(n) \notin o(\log\log n)$ for *middle* alternating Turing machines recognizing unary nonregular languages, it shows that such a lower bound cannot be raised to any "well-behaved" function $g(n)$ above $\log\log n$, i.e., to any $g(n)$ that is recursive with $\frac{g(n)}{\log\log n}$ unbounded and monotone increasing. This definitively rules out the possibility of an exponential gap observed on the corresponding one-way devices between the general and the unary cases.

Table 1.1 briefly summarizes the lower bounds for *middle* space×input head reversals bounded Turing machines recognizing nonregular languages (see Theorem 3.1 and [4, 16]).

This paper is organized as follows: section 2 contains basic definitions, in particular, the basic notions of space complexity and the SPACE×REVERSALS resource measure. In section 3, we state our main result: we exhibit, for any (arbitrarily slow growing) unbounded monotone recursive function $f(n)$, a unary nonregular language $\mathcal{L}_f$, mentioned above, and analyze the complexity of recognizing $\mathcal{L}_f$. As a consequence, section 4 proves the above-claimed separation results for the unary case. Finally, in section 5, we improve these separations for the case of general alphabets.

TABLE 1.1

Best lower bounds obtained for $s(n) \cdot i(n)$ on middle *alternating and nondeterministic* (Σ_1) *Turing machines accepting nonregular languages* ($i(n) = 1$ *for one-way machines*). *These bounds are known to be optimal* [4, 16] *except the bound for two-way alternating Turing machines on unary inputs which is "quasi" optimal* (*see Theorem* 3.1).

	Unary languages	General languages
One-way alternating	$\log n$	$\log \log n$
One-way nondeterministic (Σ_1)	$\log n$	$\log n$
Two-way alternating	**lower:** $\log \log n$ **upper:** $\log \log n \cdot f(n)$	$\log \log n$
Two-way nondeterministic (Σ_1)	$\log n$	$\log n$

2. Preliminaries. In this section, we briefly recall some very basic definitions concerning space bounded models of computation. For more details, we refer to [4, 23]. Furthermore, we consider machines having simultaneous bounds on both working space and number of input head reversals, and we introduce SPACE×REVERSALS complexity classes.

Let Σ^* be the set of all strings over an alphabet Σ. Given any language $L \subseteq \Sigma^*$, $L^c = \Sigma^* \setminus L$ denotes the *complement* of L. L is said to be *unary* (or *tally*) whenever it is built on an alphabet consisting of exactly one symbol (usually "1").

The Turing machine model we shall deal with has been presented in [14, 19] to study sublinear space bounded computations. It consists of a finite state control, a two-way read-only input tape (with input enclosed between two end markers), and a separate semi-infinite two-way read-write work tape (initially empty, containing only *blank* symbols). A *memory state* of a Turing machine is an ordered triple $m = (q, u, j)$, where q is a control state, u is a string of work tape symbols (the nonblank content of the work tape), and j is an integer satisfying $1 \leq j \leq |u| + 1$ (the position of the work tape head). A *configuration* is an ordered pair $c = (m, i)$, where m is a memory state and i is an integer denoting the position of the input head.

The reader is assumed to be familiar with the notion of *alternating Turing machine*, introduced in [5], which is, at the same time, a generalization of nondeterminism and parallelism. A Σ_k (Π_k) *machine* is an alternating Turing machine beginning its computation in an existential (universal, respectively) state, and making at most $k - 1$ switches between existential and universal states along each computation path on any input. It can be easily seen that Σ_1 machines are actually nondeterministic machines. It is stipulated that Σ_0 and Π_0 machines are *deterministic* machines.

Let us now review notions of *space complexity* in the literature. In what follows, the space used by a computation of an alternating machine is, by definition, the maximal number of work tape cells used by any configuration in the tree corresponding to that computation. (For deterministic and nondeterministic machines, the computation reduces to a single computation path.) Let M be a deterministic, nondeterministic, or alternating Turing machine. Then the following hold.

- M works in *strong* $s(n)$ space if and only if, for each input of length n, no reachable configuration uses more space than $s(n)$ [14, 19, 10].
- M works in *middle* $s(n)$ space if and only if, for each accepted input of length n, no reachable configuration uses more space than $s(n)$ [22].
- M works in *accept*[2] $s(n)$ space if and only if, for each accepted input of length n, each accepting computation uses at most space $s(n)$ [11, 4, 18].

[2]The designation "*accept*" has been adopted first in [4].

- M works in *weak* $s(n)$ space if and only if, for each accepted input of length n, there exists an accepting computation using at most space $s(n)$ [17, 1, 13].

The above definitions are given in increasing order of generality, as one may easily verify. We remark that, although several differences have been emphasized in the literature [23, 4, 7], the above space notions turn out to be equivalent when considering *fully space constructible*[3] bounds, e.g., "normal" functions above $\log n$: once the space limit $s(n)$ can be computed in advance, each computation consuming too much space may be aborted. Also notice that, for Π_1 machines, *middle*, *accept*, and *weak* notions coincide for arbitrary space complexities.

With a slight abuse of terminology, we will often say, for instance, "a *middle* alternating Turing machine" instead of "an alternating Turing machine working in *middle* space."

The other computational resource we are interested in is the number of *input head reversals*. In general, we say that a Turing machine M is *one-way* if it can never move its input head toward the left; otherwise M is a *two-way* device. To emphasize the role of input head motion, we introduce a bound $i(n)$ on the number of input head reversals. We always compute $i(n)$ by considering those computations by which the space is defined. Thus, for instance, we say that a *middle* alternating Turing machine works *simultaneously* in $s(n)$ space and $i(n)$ input head reversals if, for each accepted input of length n, no reachable configuration uses more than $s(n)$ work tape cells, nor can it be accessed by a path that reverses the input head direction more than $i(n)$ times. For technical reasons, we stipulate $i(n) = 1$ for one-way machines.

Throughout the rest of the paper, we use the following notation for complexity classes. Let $c \in \{strong, middle, accept, weak\}$ and $X \in \{\Sigma_k, \Pi_k\}$. Then, we define

$$c\text{-}X\mathrm{SPACE}\times\mathrm{REVERSALS}(h(n))$$

to be the class of the languages accepted by X machines of type c in $s(n)$ space and $i(n)$ input head reversals satisfying $s(n) \cdot i(n) \in \mathcal{O}(h(n))$. In particular, we let $X = \mathrm{D}$ for $\Sigma_0 = \Pi_0$, i.e., for deterministic Turing machines. By *co-c*-$X\mathrm{SPACE}\times\mathrm{REVERSALS}(h(n))$, we denote the class of the languages L^{c} such that $L \in c\text{-}X\mathrm{SPACE}\times\mathrm{REVERSALS}(h(n))$. Finally, the restriction of these classes to unary languages will be denoted by $c\text{-}X\mathrm{SPACE}\times\mathrm{REVERSALS}^{1^*}(h(n))$ or by *co-c*-$X\mathrm{SPACE}\times\mathrm{REVERSALS}^{1^*}(h(n))$.

3. A family of nonregular unary languages. In this section, we introduce, for each unbounded (arbitrarily slow growing) monotone increasing recursive function $f(n)$, a nonregular unary language $\mathcal{L}_f$ that can be accepted by an alternating Turing machine in *middle* $s(n)$ space with $i(n)$ input head reversals satisfying $s(n) \cdot i(n) \in \mathcal{O}(\log\log n \cdot f(n))$. Hence, the optimal $\log\log n$ lower bound on $s(n) \cdot i(n)$ for general nonregular languages acceptance can be arbitrarily approached by unary languages. Subsequently, we prove that $\mathcal{L}_f$ cannot be accepted by any nondeterministic Turing machine working in *weak* $s(n)$ space and $i(n)$ input head reversals such that $s(n)\cdot i(n) \in o(\log n)$. This implies the nonregularity of $\mathcal{L}_f$ and will later be used to separate several $\mathrm{SPACE}\times\mathrm{REVERSALS}$ complexity classes.

In what follows, p_i denotes the ith prime. Furthermore, $f : \mathbf{N} \to \mathbf{N}$ is assumed to be any (effectively given) unbounded monotone increasing recursive function. That is, we have a deterministic Turing machine A which, for any binary input x, prints

[3] A function $s(n)$ is said to be *fully space constructible* whenever there exists a deterministic Turing machine which, on any input of length n, uses exactly space $s(n)$.

out a binary representation of $f(x)$. (Alternatively, we can avoid any ambiguity by using, in any standard enumeration $A_1, A_2, \ldots$ of Turing machines, the first machine computing f. It will be seen later that the upper and lower bounds proved in this section hold for any choice of A.)

We are now ready for the definition of $\mathcal{L}_f$. For each $n \in \mathbf{N}$, we first define the following statements.

(i) Let p_i be the first prime not dividing n.
(ii) Let x be the smallest integer satisfying $x \geq 2^{p_i}$ and $f(x) \geq p_i$.
(iii) Let $y = \max\{x, 2^{s_1}, 2^{s_2}, \ldots, 2^{s_x}\}$, where s_i denotes the amount of work tape space used by machine A when computing the value of $f(i)$.

Then $1^n \in \mathcal{L}_f$ if and only if the following holds:

(iv) for all prime powers ${p_j}^k \leq y$, with $p_j \leq \sqrt{y}$, $j \neq i$, and $k \geq 1$, we have that ${p_j}^k$ divides n.

In other words, item (iv) states that $1^n \in \mathcal{L}_f$ if and only if all the prime powers ${p_j}^k \leq y$, with $p_j \leq \sqrt{y}$, $j \neq i$, and $k \geq 1$, divide n, $\log y$ denoting the amount of space claimed by A to compute any of the values $f(1), f(2), \ldots, f(x)$ or to represent the integer x (see item (iii)), and $x \geq 2^{p_i}$ being the smallest integer satisfying $f(x) \geq p_i$ as required in (ii). Note that such x must exist: take the first x such that $f(x) > \max\{p_i - 1, f(1), f(2), \ldots, f(2^{p_i} - 1)\}$, using the fact that $f(x)$ is unbounded.

The proof of nonregularity of $\mathcal{L}_f$ will be given later as a consequence of Theorem 3.2. Now we shall study the complexity of $\mathcal{L}_f$ on alternating machines.

THEOREM 3.1. *Let $\mathcal{L}_f$ be a unary language defined as above. Then $\mathcal{L}_f$ can be accepted by a middle alternating Turing machine within $s(n)$ space and $i(n)$ input head reversals satisfying*

$$s(n) \cdot i(n) \in \mathcal{O}\left(\log\log n \cdot f(n)\right) .$$

Proof. First, we shall determine possible values of n so that the string 1^n belongs to $\mathcal{L}_f$. Let p_i, x, and y be defined as stated in items (i) – (iii). Item (iv) in the definition of $\mathcal{L}_f$ requires that possible factorizations of n must be of the following form:

$$n = {p_1}^{\alpha_1+\beta_1} \cdot {p_2}^{\alpha_2+\beta_2} \cdot \ldots \cdot {p_{i-1}}^{\alpha_{i-1}+\beta_{i-1}} \cdot {p_{i+1}}^{\alpha_{i+1}+\beta_{i+1}} \cdot \ldots \cdot {p_s}^{\alpha_s+\beta_s} \cdot \nu \,, \tag{3.1}$$

where p_s represents the largest prime less than or equal to $\sqrt{y}$. The prime p_i does not appear in (3.1) since it does not divide n. Numbers α_j, with $j \in \{1, 2, \ldots, s\} \setminus \{i\}$, are the maximal exponents which the corresponding primes p_j can be raised to in order to have ${p_j}^{\alpha_j} \leq y$. It is easy to see that $\alpha_j = \lfloor \log_{p_j} y \rfloor$ and $\beta_j \geq 0$, where, for any given number z, $\lfloor z \rfloor$ denotes the greatest integer less than or equal to z. Finally, ν contains the (possibly empty) part of the factorization consisting of powers of those primes greater than $\sqrt{y}$.

By (3.1), we are able to obtain a relation between n and y. In fact, we observe that

$$n \geq \prod_{p \leq \sqrt{y},\ p \neq p_i} p^{\lfloor \log_p y \rfloor} \,, \tag{3.2}$$

where the product is taken over all primes not exceeding $\sqrt{y}$ and different from p_i. By noticing that $\log_p y \geq 2$ for $p \leq \sqrt{y}$, we get the following limitation:

$$\sqrt{y} = p^{\frac{1}{2} \cdot \log_p y} < p^{\lfloor \log_p y \rfloor} \,,$$

which, together with (3.2), yields

$$n > \prod_{p \le \sqrt{y},\ p \neq p_i} \sqrt{y} = y^{\frac{1}{2}\cdot(\pi(\sqrt{y})-1)}\,. \tag{3.3}$$

Here $\pi(z)$ denotes the number of primes not exceeding $\lfloor z \rfloor$. A well-known theorem due to P. Čebyšev (see [8, Thm. 7]) states that

$$c_1 \cdot \frac{z}{\log z} \le \pi(z) \le c_2 \cdot \frac{z}{\log z}$$

for some positive constants c_1 and c_2. We can use this in (3.3) and obtain

$$n > y^{\frac{1}{2}\cdot\left(c_1 \cdot \frac{\sqrt{y}}{\log \sqrt{y}} - 1\right)} = 2^{\frac{\log y}{2}\cdot\left(\frac{2c_1\sqrt{y}}{\log y} - 1\right)} > d^{\sqrt{y}} \tag{3.4}$$

for a suitable constant $d > 1$.

With these results in our hands, we are now ready to estimate the amount of space and input head reversals sufficient for a *middle* alternating Turing machine M to accept $\mathcal{L}_f$. On input 1^n, M runs a two-phase algorithm. First, it computes the smallest prime p_i not dividing n. Then, it checks the truth of predicate in item (iv).

PHASE 1. M deterministically computes p_i by means of the following routine:

```
/* input is 1^n */
p := 2
while n mod p = 0 do
    begin
        p := p + 1
        while p not a prime do p := p + 1
    end
/* now p contains p_i */
```

The amount $s_1(n)$ of space used in this phase equals the number of bits needed to represent p_i in binary notation, i.e., $s_1(n) \in \mathcal{O}(\log p_i)$. Furthermore, it is easy to see that, for each value of p, the test "$n \bmod p = 0$" can be accomplished by scanning the input only once. Therefore, the number $i_1(n)$ of input head reversals equals the number of primes not exceeding p_i. By Čebyšev's theorem, we obtain $i_1(n) \in \mathcal{O}(\frac{p_i}{\log p_i})$.

PHASE 2. M deterministically computes the smallest integer $x \ge 2^{p_i}$ such that $f(x) \ge p_i$ (p_i being already stored on the work tape during the former phase). To this purpose, M simply loops as follows:

```
x := 1
while x < 2^{p_i} or f(x) < p_i do
    x := x + 1
```

Note that M computes each of the values $f(1), f(2), \ldots, f(x)$ by simulating A; therefore, it marks off exactly $\max\{\log x, s_1, s_2, \ldots, s_x\} = \log y$ space on the work tape. Recall that s_i denotes the amount of space used by A when computing $f(i)$.

Subsequently, M *universally* generates all the prime powers whose binary representation takes at most $\log y$ bits and checks whether each ${p_j}^k \le y$, with $p_j \le \sqrt{y}$, $j \neq i$, and $k \ge 1$, divides n. Clearly, the space requirement in this phase is bounded by $s_2(n) \in \mathcal{O}(\log y)$.

For input head reversals, we observe that the computations of $f(1), f(2), \ldots, f(x)$ are performed deterministically with the input head parked at one end marker, and that the divisibility of n by each ${p_j}^k$ is tested universally in parallel by one input scan, whence $i_2(n) = 1$.

Let us now evaluate $s(n) \cdot i(n)$ *in case* 1^n *belongs to* $\mathcal{L}_f$. The algorithm uses a total amount of space and input head reversals along each computation path of M to be estimated as

$$s(n) = s_1(n) + s_2(n) \in \mathcal{O}(\log p_i + \log y) = \mathcal{O}(\log y)\,,$$
$$i(n) = i_1(n) + i_2(n) \in \mathcal{O}\left(\frac{p_i}{\log p_i}\right)\,,$$

using $p_i \leq \log x \leq \log y$ by (ii) and (iii) in the definition of $\mathcal{L}_f$. This gives

$$s(n) \cdot i(n) \in \mathcal{O}\left(\log y \cdot \frac{p_i}{\log p_i}\right)\,.$$

Items (ii) and (iii) in the definition of $\mathcal{L}_f$ require that $p_i \leq f(x)$ and $x \leq y$. Further, $\frac{t}{\log t}$ is monotone increasing. Hence, $\frac{f(x)}{\log f(x)}$ is also monotone increasing whenever $f(x)$ is monotone increasing. Therefore, in case of $1^n \in \mathcal{L}_f$, we are able to obtain the following bounds:

$$\log y \cdot \frac{p_i}{\log p_i} \leq \log y \cdot \frac{f(x)}{\log f(x)} \leq \log y \cdot \frac{f(y)}{\log f(y)}\,.$$

Moreover, inequality (3.4) states that $n > d^{\sqrt{y}}$, i.e., $y \leq k \cdot \log^2 n$ for a suitable positive constant k. Hence, for sufficiently large n, we get

$$\begin{aligned}\log y \cdot \frac{f(y)}{\log f(y)} &\leq \log(k \cdot \log^2 n) \cdot \frac{f(k \cdot \log^2 n)}{\log f(k \cdot \log^2 n)} \\ &\leq \log(k \cdot \log^2 n) \cdot \frac{f(n)}{\log f(n)} \;\in\; \mathcal{O}\left(\log\log n \cdot f(n)\right)\,,\end{aligned}$$

which completes the proof. □

As a consequence of Theorem 3.1, it turns out that, using unary languages, we can arbitrarily approach the optimal $\log\log n$ lower bound on $s(n) \cdot i(n)$ holding for general nonregular language acceptance by *middle* space×input head reversals bounded alternating Turing machines (see Table 1.1). Take, for instance, $f(x) = \log^* x = \min\{k \in \mathbf{N} \mid \log^{(k)} x \leq 1\}$, with $\log^{(0)} x = x$ and, for each $k \geq 1$, $\log^{(k)} x = \log^{(k-1)} \log x$.

We now show that the language $\mathcal{L}_f$ is nonregular. Actually, we prove a stronger result that, together with Theorem 3.1, will also allow us to obtain some separations in the case of weakly bounded computations.

THEOREM 3.2. *Let* $\mathcal{L}_f$ *be a unary language defined as above. Then for any* $h(n) \in o(\log n)$, $\mathcal{L}_f$ *is not in weak-*Σ_1SPACE×REVERSALS$^{1^*}(h(n))$.

Proof. Suppose, by contradiction, that $\mathcal{L}_f$ can be accepted by a *weak* nondeterministic machine M in $s(n)$ space and $i(n)$ input head reversals, with $s(n) \cdot i(n) = h(n) \in o(\log n)$. The number of different memory states of M not using more space than $s(n)$ on the work tape can be bounded by $c^{s(n)}$, where c is a constant dependent on the number of work tape symbols and finite-control states of M. Since $\lim_{n\to\infty} \frac{s(n)\cdot i(n)}{\log n} = 0$, there exists $n_0 \in \mathbf{N}$ such that

$$c^{s(n)\cdot i(n)} < \sqrt{n} \quad \text{for each } n \geq n_0\,. \tag{3.5}$$

Now, consider the string $1^{n'}$, where n' is defined in the following way:

(i) First, let $p_i \geq 5$ be a sufficiently large prime such that there is another prime $\tilde{p}$ between n_0 and p_i, i.e., $p_i > \tilde{p} > n_0$.
(ii) Let x be the smallest integer satisfying $x \geq 2^{p_i}$ and $f(x) \geq p_i$.
(iii) Now, let $y = \max\{x, 2^{s_1}, 2^{s_2}, \ldots, 2^{s_x}\}$, where s_i denotes the space used by the machine A to compute $f(i)$.
(iv) Finally, define

$$n' = \prod_{p \leq \sqrt{y},\ p \neq p_i} p^{\lfloor \log_p y \rfloor},$$

where the product is taken over all primes not exceeding $\sqrt{y}$ and different from p_i.

We want to show that $1^{n'} \in \mathcal{L}_f$. First, we prove that p_i is the first prime that does not divide n'. Clearly, p_i does not divide n'. For primes $p < p_i$, we obtain

$$p^{\lfloor \log_p y \rfloor + 1} > y \geq \sqrt{y} \geq \log y \geq \log x \geq p_i > p = p^1,$$

using (iii) and (ii) in the definition of n'. (The inequality $\sqrt{y} \geq \log y$ follows from $\log y \geq p_i \geq 5$ by (i).) Thus, $\lfloor \log_p y \rfloor > 0$ and $p < \sqrt{y}$; i.e., n' is divisible by p for each prime $p < p_i$.

At this point, to conclude that $1^{n'} \in \mathcal{L}_f$, it is enough to prove that each prime power ${p_j}^k \leq y$, with $p_j \leq \sqrt{y}$, $j \neq i$, and $k \geq 1$, divides n'. This can be immediately shown by observing that $k \leq \log_{p_j} y$ and $k \in \mathbf{N}$; hence,

$$k = \lfloor k \rfloor \leq \lfloor \log_{p_j} y \rfloor.$$

This gives that $1^{n'} \in \mathcal{L}_f$.

As a consequence, there must exist an accepting computation path $\mathcal{C}$ of M on the input $1^{n'}$, using at most $s(n')$ space and $i(n')$ input head reversals. By (i) in the definition of n', we have a prime $\tilde{p}$ satisfying $p_i > \tilde{p} > n_0$. Since p_i is the first prime not dividing n', $\tilde{p}$ divides n', and hence $n' \geq \tilde{p} > n_0$. Therefore, by (3.5), the bounds $s(n')$ and $i(n')$ must satisfy

$$c^{s(n')} \leq c^{s(n') \cdot i(n')} < \sqrt{n'} \leq n'. \tag{3.6}$$

Along the path $\mathcal{C}$, we can consider $r_1, r_2, \ldots, r_A$, the sequence of all configurations in which the input head scans either of the end markers. Let $b_1, e_1, b_2, e_2, \ldots, b_B, e_B$ be the subsequence of $r_1, r_2, \ldots, r_A$ such that, for each $j = 1, 2, \ldots, B$, machine M begins with the input head positioned at the left or right end marker in b_j, traverses across the entire input $1^{n'}$, and ends in e_j positioned at the opposite end marker, without visiting either of the end markers in the meantime. The segments of computation between e_j and b_{j+1} always return the input head back to the same end marker (or, possibly, $e_j = b_{j+1}$).

Since, by (3.6), the number of different memory states is bounded by $c^{s(n')} < n'$, the machine must enter *a loop* when traversing the entire input $1^{n'}$ from b_j to e_j; i.e., it enters some memory state twice in some configurations (q_j, d_j) and $(q_j, d_j + \ell_j)$ for $\ell_j \neq 0$. To avoid any ambiguity, we take the first loop the machine gets into, i.e., the first pair of configurations having the same memory state, along each input traversal. Observe that

$$\begin{aligned} \ell_j &\leq c^{s(n')} \quad \text{for each } j = 1, 2, \ldots, B, \\ B &\leq i(n'), \end{aligned} \tag{3.7}$$

since M is simultaneously $s(n)$ space and $i(n)$ input head reversals bounded.

Now, define

$$\ell = \prod_{j=1}^{B} \ell_j \,. \tag{3.8}$$

It is not too hard to see that the machine M must also accept the inputs $1^{n'+\mu\cdot\ell}$ for each $\mu \in \mathbf{N}$. In fact, we can replace the accepting computation path $\mathcal{C}$ for the input $1^{n'}$ by a new computation path $\mathcal{C}_\mu$ that visits the end markers in the same sequence of configurations $r_1, r_2, \ldots, r_A$. The path $\mathcal{C}_\mu$ is obtained from $\mathcal{C}$ by iterating, $\frac{\mu\cdot\ell}{\ell_j}$ more times, the loop of length ℓ_j in the segment of computation connecting b_j and e_j, for each $j = 1, 2, \ldots, B$. Note that ℓ is a common multiple of $\ell_1, \ell_2, \ldots, \ell_B$ and hence $\frac{\mu\cdot\ell}{\ell_j} \in \mathbf{N}$; i.e., the new segments of computation traverse exactly $n' + \frac{\mu\cdot\ell}{\ell_j} \cdot \ell_j = n' + \mu\cdot\ell$ positions, beginning and ending in the same configurations b_j and e_j, respectively, for each $j = 1, 2, \ldots, B$. The segments between e_j and b_{j+1} (always returning back to the same end marker) are left unchanged. Hence, for each $\mu \in \mathbf{N}$, $\mathcal{C}_\mu$ is a valid accepting computation path on the input $1^{n'+\mu\cdot\ell}$.

To complete the proof, we are now going to show that $1^{n'+\mu'\cdot\ell} \notin \mathcal{L}_f$, where μ' is defined by

$$\mu' = \prod_{p \le p_i} p \,, \tag{3.9}$$

which is a contradiction. The product is taken over all primes not exceeding p_i; hence, $(n' + \mu'\cdot\ell) \bmod p = n' \bmod p$ for each prime $p \le p_i$. This gives that n' and $n' + \mu'\cdot\ell$ share the same "least prime nondivisor" p_i. Therefore, to show that $1^{n'+\mu'\cdot\ell} \notin \mathcal{L}_f$, there only remains to exhibit a prime power ${p_j}^k \le y$, with $p_j \le \sqrt{y}$, $j \ne i$, and $k \ge 1$, such that ${p_j}^k$ does not divide $n' + \mu'\cdot\ell$.

Since we have shown that $1^{n'} \in \mathcal{L}_f$, each such prime power divides n', and hence

$$(n' + \mu'\cdot\ell) \bmod {p_j}^k = (\mu' \cdot \ell) \bmod {p_j}^k \,. \tag{3.10}$$

Therefore, the problem reduces to exhibit ${p_j}^k$ not dividing $\mu' \cdot \ell$.

To this aim, observe that ℓ, as defined by (3.8), is bounded by

$$\ell = \prod_{j=1}^{B} \ell_j \le \prod_{j=1}^{B} c^{s(n')} = \left(c^{s(n')}\right)^B \le c^{s(n')\cdot i(n')} < \sqrt{n'} \,, \tag{3.11}$$

using (3.7) and (3.6). On the other hand, the factorization of ℓ must be of the form

$$\ell = \nu \cdot \prod_{p \le \sqrt{y},\ p \ne p_i} p^{\alpha_p} \,, \tag{3.12}$$

where the product is taken over all primes not exceeding $\sqrt{y}$ and different from p_i, while ν contains the (possibly empty) part of the factorization consisting of powers of those primes p greater than $\sqrt{y}$ or $p = p_i$.

Let us now show that there exists a prime $p' \le \sqrt{y}$, $p' \ne p_i$, such that $\alpha_{p'} < \lfloor \log_{p'} y \rfloor - 1$. Suppose, by contradiction, that $\alpha_p \ge \lfloor \log_p y \rfloor - 1$ for each prime $p \le \sqrt{y}$, $p \ne p_i$. Then $\log_p y \ge 2$ and hence

$$\lfloor \log_p y \rfloor - 1 \ge \frac{1}{2} \cdot \lfloor \log_p y \rfloor \,.$$

This gives

$$\ell = \nu \cdot \prod_{p \le \sqrt{y},\, p \ne p_i} p^{\alpha_p} \ge \prod_{p \le \sqrt{y},\, p \ne p_i} p^{\lfloor \log_p y \rfloor - 1} \ge \prod_{p \le \sqrt{y},\, p \ne p_i} p^{\frac{1}{2} \cdot \lfloor \log_p y \rfloor} = \sqrt{n'}\,,$$

i.e., $\ell \ge \sqrt{n'}$, which contradicts (3.11). Hence, there exists a prime $p' \le \sqrt{y}$, $p' \ne p_i$ such that $\alpha_{p'} < \lfloor \log_{p'} y \rfloor - 1$.

By (3.12), this implies that ℓ is not an integer multiple of $p'^{\lfloor \log_{p'} y \rfloor - 1}$ and therefore, by (3.9), $\mu' \cdot \ell$ is not an integer multiple of $p'^{\lfloor \log_{p'} y \rfloor}$. But then, we have a prime power $p'^{\lfloor \log_{p'} y \rfloor} \le y$, with $p' \le \sqrt{y}$ and $p' \ne p_i$, that does not divide $n' + \mu' \cdot \ell$, using (3.10). Therefore, $1^{n' + \mu' \cdot \ell} \notin \mathcal{L}_f$, which is a contradiction and completes the proof of the theorem. □

4. Separation results in the unary case. We now draw some important structural consequences from the previous results. Note that the alternating algorithm provided for $\mathcal{L}_f$ in Theorem 3.1 consists of a deterministic phase followed by a single universal branching. Hence, it can be run on a Π_1 machine. Moreover, a *middle* space×input head reversals bounded machine can be viewed as an *accept* device, which in turn is a special case of a *weak* machine. Thus we have the following corollary.

COROLLARY 4.1. *Let $\mathcal{L}_f$ be a unary language defined as above for any unbounded monotone increasing recursive function $f(n)$. Then $\mathcal{L}_f \in$ middle-Π_1SPACE$\times$REVERSALS$^{1^*}(\log\log n \cdot f(n))$. Hence, for $c \in \{middle, accept, weak\}$, $\mathcal{L}_f \in c$-Π_1SPACE$\times$REVERSALS$^{1^*}(\log\log n \cdot f(n))$.*

On the other hand, from Theorem 3.2 we get Corollary 4.2.

COROLLARY 4.2. *Let $\mathcal{L}_f$ be a unary language defined as above. Then $\mathcal{L}_f \notin$ weak-Σ_1SPACE $\times$ REVERSALS$(h(n))$ for any $h(n) \in o(\log n)$. Hence, for $c \in \{middle, accept, weak\}$ and any $h(n) \in o(\log n)$, $\mathcal{L}_f \notin c$-Σ_1SPACE$\times$REVERSALS $(h(n))$.*

Combining the above two corollaries, we have the first separation for "well-behaved" functions $h(n)$ between $\log\log n$ and $\log n$ in Theorem 4.3.

THEOREM 4.3. *Let $h(n) \in o(\log n)$ be any recursive function such that $\frac{h(n)}{\log\log n}$ is unbounded and monotone increasing. Then, for $c \in \{middle, accept, weak\}$,*

$$c\text{-}\Pi_1\text{SPACE}\times\text{REVERSALS}^{1^*}(h(n)) \ne c\text{-}\Sigma_1\text{SPACE}\times\text{REVERSALS}^{1^*}(h(n))\,.$$

Proof. Clearly, if $h(n)$ is recursive, then so is $f(n) = \frac{h(n)}{\log\log n}$. Then, by Corollary 4.1, $\mathcal{L}_f$ belongs to c-Π_1SPACE $\times$ REVERSALS$^{1^*}(h(n))$ for each $c \in \{middle, accept, weak\}$. On the other hand, by Corollary 4.2, $\mathcal{L}_f$ is not in c-Σ_1SPACE $\times$ REVERSALS$^{1^*}(h(n))$, since $h(n) \in o(\log n)$. □

For further separations we need the fact that the class of regular languages is closed under complementation and that regular languages are accepted by one-way Turing machines working in constant space. Hence, from Theorem 3.2, we get Corollary 4.4.

COROLLARY 4.4. *Let $\mathcal{L}_f$ be a unary language defined as above. Then neither $\mathcal{L}_f$ nor ${\mathcal{L}_f}^{\mathrm{c}}$ are regular.*

As recalled in the Introduction, apart from *middle* alternation, for any other combination of determinism, nondeterminism, or alternation with the space notions defined in section 2, computational lower bounds for nonregular unary and general languages coincide and are optimal [4]. Thus, for *accept* alternating machines, Table 1.1 can be shrunk as shown in Table 4.1.

TABLE 4.1
Optimal lower bounds on $s(n) \cdot i(n)$ for accept *alternating and nondeterministic* (Σ_1) *Turing machines recognizing nonregular languages* [4].

	Unary and general languages
One-way alternating	$\log\log n$
One-way nondeterministic (Σ_1)	$\log n$
Two-way alternating	$\log\log n$
Two-way nondeterministic (Σ_1)	$\log n$

Now, from Table 4.1 (for proof, see [4, Thm. 6]), we get the following corollary.

COROLLARY 4.5. *Let L be a nonregular language. Then $L \notin$ accept-Σ_1SPACE$\times$REVERSALS$(h(n))$ for any $h(n) \in o(\log n)$. Hence, for $c \in \{middle, accept\}$ and any $h(n) \in o(\log n)$, $L \notin c$-Σ_1SPACE$\times$REVERSALS$(h(n))$.*

This allows us to present some further separations for *middle* and *accept* complexity classes.

THEOREM 4.6. *Let $h(n) \in o(\log n)$ be any recursive function such that $\frac{h(n)}{\log\log n}$ is unbounded and monotone increasing. Then, for $c \in \{middle, accept\}$,*

$$c\text{-}\Pi_1\text{SPACE}\times\text{REVERSALS}^{1^*}(h(n)) \neq co\text{-}c\text{-}\Sigma_1\text{SPACE}\times\text{REVERSALS}^{1^*}(h(n))\,,$$
$$co\text{-}c\text{-}\Pi_1\text{SPACE}\times\text{REVERSALS}^{1^*}(h(n)) \neq c\text{-}\Sigma_1\text{SPACE}\times\text{REVERSALS}^{1^*}(h(n))\,.$$

Proof. Note that, by Corollary 4.5, the classes c-Σ_1SPACE$\times$REVERSALS$^{1^*}(h(n))$ and co-c-Σ_1SPACE$\times$REVERSALS$^{1^*}(h(n))$ coincide with the class of regular languages. On the other hand, for $f(n) = \frac{h(n)}{\log\log n}$, neither $\mathcal{L}_f$ nor ${\mathcal{L}_f}^{\mathrm{c}}$ are regular, by Corollary 4.4, and they belong, respectively, to c-Π_1SPACE$\times$REVERSALS$^{1^*}(h(n))$ and to co-c-Π_1SPACE$\times$REVERSALS$^{1^*}(h(n))$, by Corollary 4.1. □

Theorem 4.6 shows, for *middle* or *accept* SPACE$\times$REVERSALS complexity measure, that the class of the languages accepted by machines making only universal decisions does not coincide with the class of the complements of languages recognizable by machines making only existential decisions.

At this point, it is quite natural to investigate whether or not this separation can be extended even to the *weak* case. The argument of Theorem 4.6 cannot be applied here, since *weak*-Σ_1SPACE$\times$REVERSALS$(\log\log n)$ does contain unary nonregular languages [4]. We conjecture that even ${\mathcal{L}_f}^{\mathrm{c}}$ does not belong to *weak*-Σ_1SPACE$\times$REVERSALS$^{1^*}(h(n))$ for a suitable function $h(n) \in o(\log n)$. Nevertheless, the "pumping" argument used to prove Theorem 3.2 does not work in the case of ${\mathcal{L}_f}^{\mathrm{c}}$. So, we leave the separation of *weak*-Π_1SPACE$\times$REVERSALS$^{1^*}(h(n))$ from *co-weak*-Σ_1SPACE$\times$REVERSALS$^{1^*}(h(n))$ as an open problem. However, in the next section, we will show how to solve this problem by using witness languages defined over a binary alphabet.

Finally, the separation of the higher levels of the alternating hierarchy from determinism can be obtained by recalling the following result in [4, Thm. 6].

THEOREM 4.7. *Let M be a* weak *deterministic (or nondeterministic) Turing machine recognizing a nonregular language within $s(n)$ space and $i(n)$ input head reversals. Then $s(n) \cdot i(n) \notin o(\log n)$ (or $s(n) \cdot i(n) \notin o(\log\log n)$, respectively). These bounds are optimal for both the unary and general cases.*

Using this result, we can easily get Theorem 4.8.

THEOREM 4.8. *Let $h(n)$ be a function satisfying $h(n) \in o(\log n) \cap \Omega(\log\log n)$.*

Then

weak-DSPACE×REVERSALS$^{1^*}(h(n))$ *is properly included in*
weak-Σ_1SPACE×REVERSALS$^{1^*}(h(n))$ *and in*
co-weak-Σ_1SPACE×REVERSALS$^{1^*}(h(n))$.

If, moreover, $h(n)$ *is recursive, with* $\frac{h(n)}{\log\log n}$ *unbounded and monotone increasing, then*

weak-DSPACE×REVERSALS$^{1^*}(h(n))$ *is properly included even in*
weak-Π_1SPACE×REVERSALS$^{1^*}(h(n))$ *and in*
co-weak-Π_1SPACE×REVERSALS$^{1^*}(h(n))$.

Proof. Just note that, by Theorem 4.7, *weak*-DSPACE × REVERSALS$(h(n))$ coincides with the class of regular languages for each $h(n) \in o(\log n)$. On the other hand, again by Theorem 4.7, *weak*-Σ_1SPACE × REVERSALS$^{1^*}(\log\log n)$ contains a nonregular unary language. Moreover, by Corollary 4.1, even *weak*-Π_1SPACE × REVERSALS$^{1^*}(h(n))$ contains unary nonregular languages whenever $h(n)$ satisfies the conditions of the theorem.

The remaining proper inclusions follow by observing that the class of nonregular languages is closed under complement. □

5. Separation results for general languages. In this section, we study separations in the case of languages defined over alphabets of at least two symbols. In particular, we show that all separations in section 4, proved in the unary case for "well-behaved" $h(n)$ between $\log\log n$ and $\log n$, hold in the general case for *any function* $h(n) \in o(\log n) \cap \Omega(\log\log n)$ and in particular for $\log\log n$.

Further, we get that, for general languages, *weak*-Π_1SPACE×REVERSALS$(h(n))$ is separated from *co-weak*-Σ_1SPACE×REVERSALS$(h(n))$. Symmetrically, we have *weak*-Σ_1SPACE×REVERSALS$(h(n)) \neq$ *co-weak*-Π_1SPACE×REVERSALS$(h(n))$ for any function $h(n) \in o(\log n) \cap \Omega(\log\log n)$.

In order to state these results, we consider the language $\mathcal{L}_1$ defined by

$$\mathcal{L}_1 = \{a^k b^{k+m} : m > 0 \text{ is a common multiple of all } r \leq k\}.$$

THEOREM 5.1. *Let* $\mathcal{L}_1$ *be the language defined as above. Then* $\mathcal{L}_1$ *can be accepted by a one-way* Π_1 *machine in* middle $\mathcal{O}(\log\log n)$ *space.*

Proof. The proof is similar to that of Theorem 3.2 in [22]. On input $x = a^k b^t$ of length $n = t + k$, the Π_1 machine M accepting $\mathcal{L}_1$ first counts, on the work tape, the number k of a's at the beginning of the input. Next, it moves the input head k positions to the right, rejecting if the right end marker is reached, i.e., if $t \leq k$. Finally, M universally generates all integers r less than or equal to k, and, by counting the length of the remaining part of the input modulo r, it checks whether each r divides $t - k$.

Clearly, M is a one-way machine. The space used by M on the input $x = a^k b^t$ is proportional to the space needed to represent k in binary notation, i.e., $\mathcal{O}(\log k)$. Moreover, if $x \in \mathcal{L}_1$, then $t - k$ is a common multiple of all $r \leq k$. Since the least common multiple of $\{1, 2, \ldots, k\}$ is bounded from below by d^k, for some constant $d > 1$ (for proof, see [23, Lem. 4.1.2]), we get $k \leq \log_d(t - k) \leq \log_d(t + k) = \log_d n$. Hence, $s(n) \in \mathcal{O}(\log\log n)$. □

COROLLARY 5.2. $\mathcal{L}_1 \in c$-Π_1SPACE × REVERSALS$(\log\log n)$ *for* $c \in \{$*middle*, *accept*, *weak*$\}$.

We now state a lower bound on the product space×input head reversals for *weak* nondeterministic Turing machines accepting $\mathcal{L}_1$.

THEOREM 5.3. *Let $\mathcal{L}_1$ be the language defined as above. Then, for any $h(n) \in o(\log n)$, $\mathcal{L}_1$ is not in weak-Σ_1SPACE$\times$REVERSALS$(h(n))$.*

Proof. Suppose, by contradiction, that $\mathcal{L}_1$ can be accepted by some nondeterministic machine in *weak* $s(n)$ space and $i(n)$ reversals, with $s(n) \cdot i(n) = h(n) \in o(\log n)$. Then $\mathcal{L}_1$ can even be accepted by a *one-way* nondeterministic machine M in *weak* $h(n)$ space. (The simulating machine nondeterministically guesses, at each input tape position, the crossing sequence of memory states that corresponds to an accepting computation path and checks the compatibility of the neighboring crossing sequences. Since the original *weak* machine is simultaneously $s(n)$ space and $i(n)$ input head reversals bounded, then *weak* $\mathcal{O}(s(n) \cdot i(n))$ space suffices for processing the input from left to right. For details, see [4, Lem. 5].)

The number of different memory states of M using no more than $h(n)$ work tape cells can be bounded by $c^{h(n)}$ for a suitable constant $c > 1$. Since $h(n) \in o(\log n)$, there exists $n_0 \in \mathbf{N}$ (cf. (3.5) in Theorem 3.2), such that

$$c^{h(n)} < \sqrt{n} \quad \text{for each } n \geq n_0\,. \tag{5.1}$$

Now, consider an input $x = a^k b^{k+m}$ of length $n = 2k + m \geq n_0$, such that m is the least common multiple of all $r \leq k$. Clearly, the string $x = a^k b^{k+m}$ belongs to $\mathcal{L}_1$. In addition, we choose $k > 0$ sufficiently large, so that $2k \leq d^k \leq m$. Here $d > 1$ is the constant used in the proof of Theorem 5.1; i.e., d^k is a lower bound for m, the least common multiple of $\{1, 2, \ldots, k\}$. Hence, b^m is a "dominant" portion of the input; i.e., we have

$$m \geq \frac{n}{2} > \sqrt{n}\,.$$

On input $x = a^k b^{k+m}$, M must have an accepting computation path $\mathcal{C}$ that uses at most $h(n)$ space. Since $m > \sqrt{n}$ and, by (5.1), the number of different memory states is bounded by $\sqrt{n}$, M must enter some memory state twice while traversing the segment b^m on the input. That is, $\mathcal{C}$ enters some configurations (q, d) and $(q, d+\ell)$, with $2k \leq d < d+\ell \leq n$.

But then M must also accept the input $a^k b^{k+m-\ell}$ by means of the accepting computation path $\mathcal{C}'$ obtained from $\mathcal{C}$ by "skipping" the segment between the configurations (q, d) and $(q, d+\ell)$. Since m is the *least* common multiple of all $r \leq k$ and $\ell > 0$, it turns out that $m - \ell$ is not a multiple of all $r \leq k$. Hence, M accepts $a^k b^{k+m-\ell} \notin \mathcal{L}_1$, which is a contradiction. □

Note that, as a consequence of Theorem 5.3, the language $\mathcal{L}_1$ is not regular. Using Corollary 5.2 and Theorem 5.3, we are now able to rewrite Theorem 4.3 for languages defined over general alphabets.

COROLLARY 5.4. *For each $h(n) \in o(\log n) \cap \Omega(\log \log n)$ and $c \in \{$middle, accept, weak$\}$,*

$$c\text{-}\Pi_1\text{SPACE}\times\text{REVERSALS}(h(n)) \neq c\text{-}\Sigma_1\text{SPACE}\times\text{REVERSALS}(h(n))\,.$$

Even Theorem 4.8 can easily be rewritten in the case of general alphabets as follows.

COROLLARY 5.5. *Let $h(n)$ be a function satisfying $h(n) \in o(\log n) \cap \Omega(\log \log n)$. Then weak-*DSPACE$\times$REVERSALS$(h(n))$ *is properly included in the following classes:*

weak-Σ_1SPACE$\times$REVERSALS$(h(n))$,

co-weak-Σ_1SPACE$\times$REVERSALS$(h(n))$,

weak-Π_1SPACE$\times$REVERSALS$(h(n))$,
co-weak-Π_1SPACE$\times$REVERSALS$(h(n))$.

Let us now compare the classes defined by machines making universal decisions with the classes of complements of languages accepted by machines making existential decisions. For sublogarithmic bounds in the unary case, these classes have been separated for the *middle* and *accept* modes (Theorem 4.6), while the separation is an open problem for the *weak* mode.

Here we prove this separation using witness languages defined over alphabets containing at least two symbols. To this purpose, we extend the lower bound of Theorem 5.3 to the language ${\mathcal{L}_1}^{\mathrm{c}}$. First, consider the following language:

$$\mathcal{L}_2 = \{a^k b^t : t \leq k \text{ or } \exists r \leq k \text{ which does not divide } t-k\}\,.$$

Note that $\mathcal{L}_2 = {\mathcal{L}_1}^{\mathrm{c}} \cap \{a\}^*\{b\}^*$.

THEOREM 5.6. *Let $\mathcal{L}_2$ be the language defined as above. Then, for any $h(n) \in o(\log n)$, $\mathcal{L}_2$ is not in weak-Σ_1SPACE$\times$REVERSALS$(h(n))$.*

Proof. As in Theorem 5.3, suppose, by contradiction, that $\mathcal{L}_2$ can be accepted by some nondeterministic machine in *weak* $s(n)$ space and $i(n)$ input head reversals, with $s(n) \cdot i(n) = h(n) \in o(\log n)$. Then $\mathcal{L}_2$ can even be accepted by a *one-way* nondeterministic machine M in *weak* $h(n)$ space [4, Lem. 5]. The number of different memory states of M using at most $h(n)$ work tape cells is bounded by $c^{h(n)}$ for a suitable constant $c > 1$. Since $h(n) \in o(\log n)$, there exists $n_0 \in \mathbf{N}$, such that

$$c^{h(n)} < \frac{n}{2} \quad \text{for each } n \geq n_0\,. \tag{5.2}$$

Now, for some even $n \geq n_0$, consider the string $x = a^{\frac{n}{2}} b^{\frac{n}{2}}$, easily seen to be in $\mathcal{L}_2$. Hence, M must have an accepting computation path on the input x, consisting of memory states that use at most space $h(n)$. Since, by (5.2), the number of different memory states is bounded by $\frac{n}{2}$, this computation path enters a *loop* while traversing the suffix $b^{\frac{n}{2}}$. More precisely, M enters some memory state twice, in some configurations (q, d) and $(q, d+\ell)$, with $d > \frac{n}{2}$ and $\ell > 0$. Thus, it is not hard to see that M must also accept the strings of the form

$$a^{\frac{n}{2}} b^{\frac{n}{2}+\mu\cdot\ell} \in \mathcal{L}_2 \quad \text{for each } \mu \in \mathbf{N}\,. \tag{5.3}$$

Now, let $\mu' = \left(\frac{n}{2}\right)!$. It is easy to see that $\mu' \cdot \ell > 0$ and that each integer $r \leq \frac{n}{2}$ divides $\mu' \cdot \ell$. That is, the string $a^{\frac{n}{2}} b^{\frac{n}{2}+\mu'\cdot\ell}$ does not belong to $\mathcal{L}_2$, which contradicts (5.3). □

Using Theorem 5.6, we are now able to show Corollary 5.7.

COROLLARY 5.7. *Let $\mathcal{L}_1$ be the language defined as above. Then the complement of $\mathcal{L}_1$ is not in weak-Σ_1SPACE$\times$REVERSALS$(h(n))$ for any $h(n) \in o(\log n)$.*

Proof. Should ${\mathcal{L}_1}^{\mathrm{c}}$ be in *weak*-Σ_1SPACE$\times$REVERSALS$(h(n))$, for some $h(n) \in o(\log n)$, then so would $\mathcal{L}_2 = {\mathcal{L}_1}^{\mathrm{c}} \cap \{a\}^*\{b\}^*$, which contradicts Theorem 5.6. □

Finally, by combining Corollaries 5.2 and 5.7, we get Theorem 5.8.

THEOREM 5.8. *Let $h(n)$ be a function satisfying $h(n) \in o(\log n) \cap \Omega(\log\log n)$. Then, for $c \in \{middle, accept, weak\}$,*

$$c\text{-}\Pi_1\text{SPACE}\times\text{REVERSALS}(h(n)) \neq \textit{co-}c\text{-}\Sigma_1\text{SPACE}\times\text{REVERSALS}(h(n))\,,$$
$$c\text{-}\Sigma_1\text{SPACE}\times\text{REVERSALS}(h(n)) \neq \textit{co-}c\text{-}\Pi_1\text{SPACE}\times\text{REVERSALS}(h(n))\,.$$

Acknowledgments. The authors wish to thank an anonymous referee for helpful comments and remarks.

REFERENCES

[1] M. Alberts, *Space complexity of alternating Turing machines*, in Fundamentals of Computation Theory, Proceedings, Lecture Notes in Computer Science 199, Springer-Verlag, Berlin, New York, 1985, pp. 1–7.

[2] B. von Braunmühl, R. Gengler, and R. Rettinger. *The alternation hierarchy for machines with sublogarithmic space is infinite*, in Symposium on Theoretical Aspects of Computer Science 1994, Proceedings, Lecture Notes in Computer Science 775, Springer-Verlag, Berlin, New York, 1994, pp. 85–96.

[3] A. Bertoni, C. Mereghetti, and G. Pighizzini, *An optimal lower bound for nonregular languages*, Inform. Process. Lett., 50 (1994), pp. 289–292; *Corrigendum*, 52 (1994), p. 339.

[4] A. Bertoni, C. Mereghetti, and G. Pighizzini, *Strong optimal lower bounds for Turing machines that accept nonregular languages*, in Mathematical Foundations of Computer Science 1995, Proceedings, Lecture Notes in Computer Science 969, Springer-Verlag, Berlin, New York, 1995, pp. 309–318.

[5] A. Chandra, D. Kozen, and L. Stockmeyer, *Alternation*, J. Assoc. Comput. Mach., 28 (1981), pp. 114–133.

[6] V. Geffert, *A hierarchy that does not collapse: Alternations in low level space*, RAIRO Inform. Théor. Appl., 28 (1994), pp. 465–512.

[7] V. Geffert, *Bridging across the* $\log(n)$ *space frontier*, in Mathematical Foundations of Computer Science 1995, Proceedings, Lecture Notes in Computer Science 969, Springer-Verlag, Berlin, New York, 1995, pp. 50–65.

[8] G. Hardy and E. Wright. *An Introduction to the Theory of Numbers*, 5th ed., Oxford University Press, 1979.

[9] J.-W. Hong, *A tradeoff theorem for space and reversal*, Theoret. Comput. Sci., 32 (1984), pp. 221–224.

[10] J. Hopcroft and J. Ullman, *Some results on tape–bounded Turing machines*, J. Assoc. Comput. Mach., 16 (1969), pp. 168–177.

[11] J. Hromkovič, B. Rovan, and A. Slobodová, *Deterministic versus nondeterministic space in terms of synchronized alternating machines*, Theoret. Comput. Sci., 132 (1994), pp. 319–336.

[12] N. Immerman, *Nondeterministic space is closed under complement*, SIAM J. Comput., 17 (1988), pp. 935–938.

[13] K. Iwama, $ASPACE(o(\log\log n))$ *is regular*, SIAM J. Comput., 22 (1993), pp. 136–146.

[14] P. Lewis, R. Stearns, and J. Hartmanis, *Memory bounds for the recognition of context free and context sensitive languages*, in IEEE Conference Record on Switching Circuit Theory and Logical Design, 1965, pp. 191–202.

[15] M. Liśkiewicz and R. Reischuk, *The sublogarithmic alternating space world*, SIAM J. Comput., 25 (1996), pp. 828–861.

[16] C. Mereghetti and G. Pighizzini, *A remark on middle space bounded alternating Turing machines*, Inform. Process. Lett., 56 (1995), pp. 229–232.

[17] W. Savitch, *Relationships between nondeterministic and deterministic tape complexities*, J. Comput. System Sci., 4 (1970), pp. 177–192.

[18] A. Slobodová, *On the power of one–way globally deterministic synchronized alternating Turing machines and multihead automata*, Internat. J. Found. Comput. Sci., 6 (1995), pp. 431–446,

[19] R. Stearns, J. Hartmanis, and P. Lewis, *Hierarchies of memory limited computations*, in IEEE Conference Record on Switching Circuit Theory and Logical Design, 1965, pp. 179–190.

[20] I. Sudborough, *Efficient algorithms for path system problems and applications to alternating and time–space complexity classes*, in 21st IEEE Symposium on Foundations of Computer Science, Proceedings, 1980, pp. 62–73.

[21] R. Szelepcsényi, *The method of forced enumeration for nondeterministic automata*, Acta Inform., 26 (1988), pp. 279–284.

[22] A. Szepietowski, *Remarks on languages acceptable in* $\log\log n$ *space*, Inform. Process. Lett., 27 (1988), pp. 201–203.

[23] A. Szepietowski, *Turing Machines with Sublogarithmic Space*, Lecture Notes in Computer Science 843, Springer-Verlag, Berlin, New York, 1994.