Computing and Informatics, Vol. 21, 2002, 607–616

REFINEMENT OF THE ALTERNATING SPACE HIERARCHY*

Viliam Geffert, Norbert Popély

Department of Computer Science
P. J. Šafárik University
Jesenná 5, 041 54 Košice, Slovakia
e-mail: {geffert, popely}@kosice.upjs.sk

Manuscript received 2 May 2001; revised 27 June 2002
Communicated by Ladislav Hluchý

Abstract. We refine the alternating space hierarchy by separating the classes Σ_k-SPACE$(s(n))$ and Π_k-SPACE$(s(n))$ from Δ_k-SPACE$(s(n))$ as well as from Δ_{k+1}-SPACE$(s(n))$, for each $s(n) \in \Omega(\log\log n) \cap o(\log n)$, and $k \geq 2$. We also present unary (tally) sets separating Σ_2-SPACE$(s(n))$ and Π_2-SPACE$(s(n))$ from Δ_2-SPACE$(s(n))$ as well as from Δ_3-SPACE$(s(n))$.

Keywords: Computational complexity, sublogarithmic space, alternation

1 INTRODUCTION AND PRELIMINARIES

The problem of alternating space hierarchy has received a lot of attention in the last decade. By inductive counting [8, 11], nondeterministic space classes NSPACE$(s(n))$ are closed under complement, for each $s(n) \in \Omega(\log n)$. This implies that the hierarchy of language classes recognizable by $s(n)$ space bounded machines making a constant number of alternations collapses to the first level, i.e.,

$$\Sigma_k\text{-SPACE}(s(n)) = \Pi_k\text{-SPACE}(s(n)) = \text{NSPACE}(s(n)),$$

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

for each $k \geq 1$ and each $s(n) \in \Omega(\log n)$. Since the technique of inductive counting uses the assumption $s(n) \in \Omega(\log n)$, the above collapse does not extend to sublogarithmic space bounds.

Later, in a series of independent papers [1, 6, 10], it was shown that the above alternating hierarchy is infinite for $s(n) \in \Omega(\log\log n) \cap o(\log n)$, i.e., both Σ_k-SPACE$(s(n))$ and Π_k-SPACE$(s(n))$ are proper subsets of Σ_{k+1}-SPACE$(s(n))$, as well as of Π_{k+1}-SPACE$(s(n))$, for each $k \geq 1$.

For $s(n) \in o(\log\log n)$, the corresponding complexity classes contain only regular languages, by [9].

Here we shall prove a more subtle structure of the alternating space hierarchy for space bounds between $\log\log n$ and $\log n$. For each $k \geq 2$, we shall present a language L_k that can be accepted *both* by Σ_{k+1}-alternating and Π_{k+1}-alternating machines in space $O(\log\log n)$, but *neither* by a Σ_k-alternating nor by a Π_k-alternating machine in space $o(\log n)$. This gives, for each $k \geq 2$ and $s(n) \in \Omega(\log\log n) \cap o(\log n)$, that Δ_k-SPACE$(s(n))$ is a proper subset of both Σ_k-SPACE$(s(n))$ and Π_k-SPACE$(s(n))$, and, in turn, both Σ_k-SPACE$(s(n))$ and Π_k-SPACE$(s(n))$ are proper subsets of Δ_{k+1}-SPACE$(s(n))$.

The situation is not so clear if we restrict the above classes to unary (tally) languages, i.e., to languages over a single letter alphabet. Here we have, by [5],

$$\mathsf{unary}\text{–}\Sigma_1\text{-SPACE}(s(n)) = \mathsf{unary}\text{–}\Pi_1\text{-SPACE}(s(n)) = \mathsf{unary}\text{–}\Delta_1\text{-SPACE}(s(n)),$$

for each $s(n)$, independent of whether $s(n) \in \Omega(\log n)$, together with [4]

$$\mathsf{unary}\text{–}\Sigma_2\text{-SPACE}(s(n)) \neq \mathsf{unary}\text{–}\Pi_2\text{-SPACE}(s(n)),$$

i.e., the collapse at the first level does not imply the collapse at the higher levels. Further, the classes at the second level are incomparable, from which we have that unary–Δ_2-SPACE$(s(n))$ is a proper subset of both unary–Σ_2-SPACE$(s(n))$ and unary–Π_2-SPACE$(s(n))$, and that the unary hierarchy does not collapse below the level two, for $s(n)$ between $\log\log n$ and $\log n$.

It is not known whether the alternating hierarchy is infinite in the unary case, or if it consists of a finite number of distinct levels. Here we shall raise the alternating hierarchy "half a level up", by presenting a language that belongs to both unary–Σ_3-SPACE$(\log\log n)$ and unary–Π_3-SPACE$(\log\log n)$, but that is contained neither in unary–Σ_2-SPACE$(o(\log n))$, nor in unary–Π_2-SPACE$(o(\log n))$. Summing up, we can separate unary–Σ_2- and unary–Π_2-SPACE$(s(n))$ both from unary–Δ_2-SPACE$(s(n))$ and from unary–Δ_3-SPACE$(s(n))$.

We first briefly review some basic definitions and notations used throughout.

We shall consider the standard Turing machine having a finite control, a two-way read-only input tape with the input enclosed in two endmarkers, and a separate semi-infinite two-way read-write worktape, initially empty. The reader is assumed to be familiar with the notion of an alternating Turing machine, which is, at the same time, a generalization of nondeterminism and parallelism [1, 2, 6, 10].

For $s(n): \mathbb{N} \to \mathbb{N}$, we call an alternating machine $s(n)$ *space bounded*, if, for each input of the length n, no reachable configuration uses more than $s(n)$ cells on the worktape.

The class of languages recognizable by machines making at most $k-1$ alternations between existential and universal configurations, starting from the initial configuration that is existential (universal), and space bounded by $O(s(n))$, will be denoted by Σ_k-SPACE$(s(n))$, or Π_k-SPACE$(s(n))$, respectively.

Further, Δ_k-SPACE$(s(n))$ denotes Σ_k-SPACE$(s(n)) \cap \Pi_k$-SPACE$(s(n))$.

By unary–Σ_k-SPACE$(s(n))$, unary–Π_k-SPACE$(s(n))$, and unary–Δ_k-SPACE$(s(n))$, we denote the corresponding space complexity classes restricted to unary (tally) languages, i.e., to languages over a single letter alphabet.

The above definition corresponds to so-called *strongly* space bounded machines (worst case cost). We shall not consider *weakly* space bounded machines here. (Weak definition of space complexity considers the best cost of acceptance; for each accepted input of the length n, there exists at least one accepting computation not using more space than $s(n)$). For more differences, see [12].

2 ALTERNATING HIERARCHY IN BINARY CASE

Now we are ready to state and prove main theorems, refining the alternating space hierarchy. We first introduce some known results presented in [6], separating the complexity classes Σ_k-SPACE$(o(\log n))$ from Π_k-SPACE$(o(\log n))$, for $k \geq 2$. The languages separating these classes have a simple block structure. The structure of the blocks can be described by a sequence of regular languages $R_2, R_3, R_4, \ldots$ Then the separating languages S_k and P_k are defined, by induction on $k \geq 2$.

Definition 1. Let $\{0,1\}$ denote a two-letter alphabet. Then

$$\begin{aligned} R_2 &= 1^+, \\ R_k &= 0(R_{k-1}0)^+, \quad \text{for each } k > 2. \end{aligned}$$

Further, let

$$f(n) = \text{the first number that does not divide } n.$$

Then

$$\begin{aligned} S_2 &= \{1^n \colon f(n) \leq \max\{f(1), \ldots, f(n-1)\}\}, \\ P_2 &= \{1^n \colon f(n) > \max\{f(1), \ldots, f(n-1)\}\}, \\ S_k &= \{w \in R_k \colon w = 0w_1 0w_2 0 \ldots 0w_\ell 0, \\ &\qquad \exists j \in \{1, \ldots, \ell\} : w_j \in P_{k-1} \ \& \ w_1, \ldots, w_\ell \in R_{k-1}\}, \\ P_k &= \{w \in R_k \colon w = 0w_1 0w_2 0 \ldots 0w_\ell 0, \\ &\qquad \forall j \in \{1, \ldots, \ell\} : w_j \in S_{k-1} \ \& \ w_1, \ldots, w_\ell \in R_{k-1}\}, \end{aligned}$$

for each $k > 2$.

Theorem 1 ([6]). $P_k \in \Pi_k$-SPACE$(\log\log n)$ and $S_k \in \Sigma_k$-SPACE$(\log\log n)$, for each $k \geq 2$, but $P_k \notin \Sigma_k$-SPACE$(s(n))$ and $S_k \notin \Pi_k$-SPACE$(s(n))$, for each $k \geq 2$ and for each $s(n) \in o(\log n)$.

This implies that there exists an $O(\log\log n)$ space bounded Σ_k-alternating machine $\mathcal{N}_{S_k}$ recognizing S_k, as well as a Π_k-alternating machine $\mathcal{N}_{P_k}$ recognizing P_k.

Now we can present a language L_k, that separates Δ_{k+1}-SPACE$(\log\log n)$ from the class Π_k-SPACE$(o(\log n)) \cup \Sigma_k$-SPACE$(o(\log n))$.

Definition 2. $L_k = \{w_1\$w_2\colon\ w_1 \in (S_k \cup \{\varepsilon\})\ \&\ w_2 \in (P_k \cup \{\varepsilon\})\}$.

In the next two lemmas, we will show that $L_k \in \Delta_{k+1}$-SPACE$(\log\log n)$.

Lemma 1. For each $k \geq 2$, $L_k \in \Pi_{k+1}$-SPACE$(\log\log n)$.

Proof. We have to show that there exists a Π_{k+1}-alternating machine $\mathcal{M}_{P_{k+1}}$ recognizing L_k in $O(\log\log n)$ space. Initially, the machine $\mathcal{M}_{P_{k+1}}$ branches universally into the following two processes:

1. The first process accepts, if $w_1 = \varepsilon$. Otherwise, it alternates to the existential phase and then it simulates the machine $\mathcal{N}_{S_k}$ on the input w_1, imitating the right endmarker at the position of the special symbol "\$". The first process does not use space above $O(\log\log n)$, because the machine $\mathcal{N}_{S_k}$ works in $O(\log\log n)$ space.
2. The second process moves its head to the right until it finds the symbol "\$", which, from this point forward, represents the left endmarker for it. Then the process checks whether $w_2 = \varepsilon$. If yes, it accepts. Otherwise, without alternation, it simulates the machine $\mathcal{N}_{P_k}$ on the input w_2. Clearly, this does not require more space than $O(\log\log n)$, because the machine $\mathcal{N}_{P_k}$ recognizing the language P_k works in $O(\log\log n)$ space.

$\square$

Lemma 2. For each $k \geq 2$, $L_k \in \Sigma_{k+1}$-SPACE$(\log\log n)$.

Proof. Now we have to present a Σ_{k+1}-alternating machine $\mathcal{M}_{S_{k+1}}$ recognizing L_k. This machine proceeds as follows. Initially, in an existential phase, $\mathcal{M}_{S_{k+1}}$ checks whether $w_1 = \varepsilon$ and whether $w_2 = \varepsilon$.

1. If $w_1 = \varepsilon$ and $w_2 = \varepsilon$, then $\mathcal{M}_{S_{k+1}}$ halts and accepts.
2. If $w_1 = \varepsilon$ but $w_2 \neq \varepsilon$, the machine alternates to the universal phase and then simulates the machine $\mathcal{N}_{P_k}$ on the input w_2.
3. If $w_1 \neq \varepsilon$ and $w_2 = \varepsilon$, the machine, without alternation, simulates $\mathcal{N}_{S_k}$ on the input w_1.

4. If $w_1 \neq \varepsilon$ and $w_2 \neq \varepsilon$, the machine $\mathcal{M}_{S_{k+1}}$ starts to simulate $\mathcal{N}_{S_k}$ on the input w_1 until the machine $\mathcal{N}_{S_k}$ alternates for the first time. (Or until it halts in an accepting configuration, making no alternation at all.) At this moment, the machine $\mathcal{M}_{S_{k+1}}$ branches universally into two parallel processes:
   - The first process will carry on the simulation of $\mathcal{N}_{S_k}$ on the input w_1.
   - The second process moves its input head to the right until it finds the symbol "\$". Then it clears the worktape and starts to simulate the machine $\mathcal{N}_{P_k}$ on the input w_2.

It is easy to see that $\mathcal{M}_{S_{k+1}}$ accepts if and only if both $w_1 \in S_k \cup \{\varepsilon\}$ and $w_2 \in P_k \cup \{\varepsilon\}$, and hence $\mathcal{L}(\mathcal{M}_{S_{k+1}}) = L_k$. Because neither $\mathcal{N}_{S_k}$ nor $\mathcal{N}_{P_k}$ use space above $O(\log\log n)$, this much space is sufficient for $\mathcal{M}_{S_{k+1}}$. Note also that, in each of the cases 1–4, the machine $\mathcal{M}_{S_{k+1}}$ does not alternate more than once before is starts to simulate $\mathcal{N}_{P_k}$ and it does not alternate at all before a simulation of $\mathcal{N}_{S_k}$. Hence, it is a Σ_{k+1}-alternating device. □

Theorem 2. For each $k \geq 2$, $L_k \in \Delta_{k+1}$-SPACE$(\log\log n)$.

Thus, we have proved that $L_k \in \Delta_{k+1}$-SPACE$(\log\log n)$, for each $k \geq 2$. We have to prove that L_k can be recognized neither by a Σ_k-SPACE$(o(\log n))$ machine nor by a Π_k-SPACE$(o(\log n))$ machine. First, we need a simple lemma about the functions $s(n)$ and $s(n+1)$, where $s(n) \in o(\log n)$.

Lemma 3. For each function $s(n) \geq 0$, if $s(n) \in o(\log n)$, then $s(n+1) \in o(\log n)$.

Proof. Let $s(n) \in o(\log n)$, which means that $\lim_{n\to\infty} \frac{s(n)}{\log n} = 0$. Then

$$0 \leq \lim_{n\to\infty} \tfrac{s(n+1)}{\log n} = \lim_{n\to\infty} \tfrac{s(n+1)}{\log(n+1)} \cdot \tfrac{\log(n+1)}{\log n} \leq \lim_{n\to\infty} \tfrac{s(n+1)}{\log(n+1)} \cdot \lim_{n\to\infty} \tfrac{1+\log n}{\log n} = 0 \cdot 1 = 0.$$

Hence, $0 \leq \lim_{n\to\infty} \frac{s(n+1)}{\log n} \leq 0$, from which we get that $s(n+1) \in o(\log n)$. □

Lemma 4. For each $k \geq 2$, $L_k \notin \Sigma_k$-SPACE$(o(\log n))$.

Proof. We shall show that there does not exist a Σ_k-alternating machine $\mathcal{M}$ working in $o(\log n)$ space and recognizing the language L_k. Suppose, for contradiction, that such machine does exist, i.e., $L_k \in \Sigma_k$-SPACE$(s(n))$, for some $s(n) \in o(\log n)$. Let $\$P_k = \{\$w\colon\, w \in P_k\} = L_k \cap \$\{0,1\}^+$. Then $\$P_k$ must be in Σ_k-SPACE$(s(n))$. The machine $\mathcal{M}'$ recognizing $\$P_k$ first checks whether the input u is the form $\$\{0,1\}^+$. If not, $\mathcal{M}'$ rejects the input. Otherwise it simulates the machine $\mathcal{M}$ on the input u.

Now it is easy to see that $P_k \in \Sigma_k$-SPACE$(s(n+1))$. The corresponding machine $\mathcal{M}^\star$ for P_k simply simulates $\mathcal{M}'$, pretending that the symbol "\$" is inserted between the left endmarker and the first symbol of the real input. Thus, the language P_k is in Σ_k-SPACE$(s(n+1))$, i.e., $P_k \in \Sigma_k$-SPACE$(s'(n))$, for some $s'(n) \in o(\log n)$, by Lemma 3. But this contradicts Theorem 1. □

Lemma 5. For each $k \geq 2$, $L_k \notin \Pi_k$-SPACE$(o(\log n))$.

Proof. The argument is almost the same as in the proof of Lemma 4. We only have to replace "Σ_k-SPACE" everywhere by "Π_k-SPACE", the language $\$P_k$ by $S_k\$ = \{w\$ \colon\ w \in S_k\}$. Finally, $\mathcal{M}^\star$ pretends that "\$" is inserted between the last symbol of the real input and the right endmarker, instead of the left endmarker. $\square$

Theorem 3. For each $k \geq 2$, $L_k \notin \Pi_k$-SPACE$(o(\log n)) \cup \Sigma_k$-SPACE$(o(\log n))$.

Corollary 1. For each $k \geq 2$ and each $s(n) \in \Omega(\log\log n) \cap o(\log n)$,

$$\begin{array}{rcl} \Delta_k\text{-SPACE}(s(n)) & \subset & \Sigma_k\text{-SPACE}(s(n)), \\ \Delta_k\text{-SPACE}(s(n)) & \subset & \Pi_k\text{-SPACE}(s(n)), \\ \Sigma_k\text{-SPACE}(s(n)) & \subset & \Delta_{k+1}\text{-SPACE}(s(n)), \\ \Pi_k\text{-SPACE}(s(n)) & \subset & \Delta_{k+1}\text{-SPACE}(s(n)). \end{array}$$

Here "$\subset$" denotes a proper inclusion. The first two inclusions follow from Theorem 1. The separating languages are S_k, P_k, and L_k, introduced in Definitions 1 and 2, respectively. Figure 1 resumes such results.

$$\begin{array}{cccccccccccccc} & & & \overset{?}{\subsetneq} & \Sigma_1 & \overset{?}{\subsetneq} & & \subset & \Sigma_2 & \subset & & \subset & \Sigma_3 & \cdots \\ \Sigma_0 = \Pi_0 = \Delta_0 & \overset{?}{\subseteq} & \Delta_1 & & \nparallel_{?} & & \Delta_2 & & \nparallel & & \Delta_3 & & \nparallel & \cdots \\ & & & \underset{?}{\subsetneq} & \Pi_1 & \underset{?}{\subsetneq} & & \subset & \Pi_2 & \subset & & \subset & \Pi_3 & \cdots \end{array}$$

Fig. 1. Alternating space hierarchy. Here Π_i (Σ_i, Δ_i) represents the complexity class Π_i-SPACE$(s(n))$, for $s(n) \in \Omega(\log\log n) \cap o(\log n)$ (Σ_i-SPACE$(s(n))$, Δ_i-SPACE$(s(n))$, respectively). Proper inclusions are indicated by "$\subset$", incomparable classes by "$\neq$". Finally, "$\overset{?}{\subseteq}$" denotes an inclusion which is not known to be proper.

3 ALTERNATING HIERARCHY IN UNARY CASE

Now we shall show the existence of a unary language L separating the complexity class unary–Δ_3-SPACE$(s(n))$ from the complexity classes unary–Σ_2-SPACE$(s(n))$ and unary–Π_2-SPACE$(s(n))$, for each $s(n) \in \Omega(\log\log n) \cap o(\log n)$.

Definition 3. Let

$$\begin{array}{rcl} L & = & \{1^n \colon\ f(n) = p_i,\ \text{and}, \\ & & \quad \text{if } i \text{ is even, then } f(n) > \max\{f(1), \ldots, f(n-1)\}, \\ & & \quad \text{if } i \text{ is odd, then } f(n) \leq \max\{f(1), \ldots, f(n-1)\}\}. \end{array}$$

Here $f(n)$ denotes the first number not dividing n, introduced in Definition 1, and p_i denotes the i-th prime.

In the next two lemmas, we will show that $L \in \mathsf{unary}\text{–}\Delta_3\text{-SPACE}(\log\log n)$. Recall that the languages $S_2 = \{1^n : f(n) \leq \max\{f(1),\ldots,f(n-1)\}\}$ and its complement $P_2 = \{1^n : f(n) > \max\{f(1),\ldots,f(n-1)\}\}$ are in the classes Σ_2-SPACE$(\log\log n)$ and Π_2-SPACE$(\log\log n)$, respectively, by Theorem 1.

Lemma 6. $L \in \mathsf{unary}\text{–}\Pi_3\text{-SPACE}(\log\log n)$.

Proof. We want to show that there exists $\mathcal{M}_{P_3}$, a unary Π_3-alternating machine working in $O(\log\log n)$ space, such that $\mathcal{L}(\mathcal{M}_{P_3}) = L$. First, the machine $\mathcal{M}_{P_3}$ deterministically computes the value of $f(n)$ and checks whether $f(n) = p_i$ for some prime p_i. It should be clear that testing whether $f(n)$ is a prime can be performed in space $O(\log f(n)) \subseteq O(\log\log n)$, since $\log f(n) \in O(\log\log n)$. See, e.g., [3, 7] or [12] (Lemma 4.1.2.) If, for some i, $f(n) = p_i$, then $\mathcal{M}_{P_3}$ also computes $i \bmod 2$.

- If $(i \bmod 2) = 1$, i.e. i is odd, then the machine $\mathcal{M}_{P_3}$, after one alternation, simulates the machine $\mathcal{N}_{S_2}$, which recognizes the unary language S_2.
- If $(i \bmod 2) = 0$, i.e. i is even, then the machine $\mathcal{M}_{P_3}$, without any alternation, simulates the machine $\mathcal{N}_{P_2}$ recognizing the unary language P_2.

Thus $\mathcal{L}(\mathcal{M}_{P_3}) = L$. Clearly, $\mathcal{M}_{P_3}$ does not use space above $O(\log\log n)$, since both $\mathcal{N}_{S_2}$ and $\mathcal{N}_{P_2}$ are $O(\log\log n)$ space bounded. The initial computation of $f(n)$, checking if $f(n) = p_i$ for some prime p_i, and computing $i \bmod 2$ is also bounded by space $O(\log\log n)$. If $f(n)$ is not a prime, $\mathcal{M}_{P_3}$ does not alternate at all. If $f(n) = p_i$, for i odd, $\mathcal{M}_{P_3}$ alternates twice, while for i being even it alternates only once. Since the initial deterministic computation can be viewed as a part of a universal phase, $\mathcal{M}_{P_3}$ is a Π_3-alternating device. □

Lemma 7. $L \in \mathsf{unary}\text{–}\Sigma_3\text{-SPACE}(\log\log n)$.

Proof. The machine $\mathcal{M}_{S_3}$ uses the same algorithm as $\mathcal{M}_{P_3}$ in Lemma 6, but this time the initial deterministic computation of $f(n)$, as well as checking if $f(n)$ is a prime, is performed as a part of the initial *existential* phase. Thus, $\mathcal{M}_{S_3}$ does not alternate at all, if $f(n)$ is not a prime, it alternates twice, if $f(n) = p_i$ with i even, and only once, if i is odd. Summing up, $\mathcal{M}_{S_3}$ is a Σ_3-alternating device. □

Theorem 4. $L \in \mathsf{unary}\text{–}\Delta_3\text{-SPACE}(\log\log n)$.

Now we have to prove that the unary language L can be recognized neither by a unary Σ_2- nor by a Π_k-alternating machine in space $o(\log n)$. This requires to show some basic properties about the least common multiple. In what follows, $\mathrm{lcm}\{1,2,\ldots,p-1\}$ denotes the least common multiple of numbers $1,\ldots,p-1$.

Lemma 8. For each prime $p > 2$,

1. $f(\mathrm{lcm}\{1,2,\ldots,p-1\}) = p$,
2. for each $\ell \in \{1,2,\ldots,\mathrm{lcm}\{1,2,\ldots,p-1\}-1\}$, we have that $f(\ell) < p$.

Proof. Let $k = \mathrm{lcm}\{1, 2, \dots, p-1\}$.

1. If $k = \mathrm{lcm}\{1, 2, \dots, p-1\}$, then k is divisible by each $\ell \leq p-1$, and it is not divisible by the prime p. Thus we have that $f(k) = f(\mathrm{lcm}\{1, 2, \dots, p-1\}) = p$.
2. Because k is the *least* common multiple of the numbers $1, 2, \dots, p-1$, no $\ell \leq k-1$ is a common multiple of $1, 2, \dots, p-1$. Thus, for each $\ell \leq k-1$, we have at least one $n \in \{1, 2, \dots, p-1\}$ that does not divide ℓ. But then $f(\ell) \leq n \leq p-1 < p$. Therefore $f(\ell) < p$ for each $\ell \in \{1, 2, \dots, \mathrm{lcm}\{1, 2, \dots, p-1\}-1\}$.

$\square$

We also need a technical lemma describing some properties of the function $f(n)$.

Lemma 9 ([6])**.** $f(n)$ is unbounded, i.e., for each $h \geq 0$, there exists $n \geq 0$ such that $f(n) \geq h$, and $f(n) = f(n + n!)$, for each $n \geq 2$.

S_2 and P_2 are languages separating the complexity class unary–Σ_2-SPACE$(s(n))$ from unary–Π_2-SPACE$(s(n))$ in [4]. We shall now recall a stronger statement.

Theorem 5 ([6])**.** For each $s(n) \in o(\log n)$,

1. if $L \in \Sigma_2$-SPACE$(s(n))$, then there exists $n' > 0$ such that, for each $n \geq n'$, $1^n \in L$ implies $1^{n+n!} \in L$,
2. if $L \in \Pi_2$-SPACE$(s(n))$, then there exists $n' > 0$ such that, for each $n \geq n'$, $1^n \notin L$ implies $1^{n+n!} \notin L$.

Lemma 10. For each $s(n) \in o(\log n)$, $L \notin$ unary–Π_2-SPACE$(s(n))$.

Proof. Suppose, for contradiction, that $L \in$ unary–Π_2-SPACE$(s(n))$. By Theorem 5, we have that there exists $n' \in \mathbb{N}$ such that, for each $n \geq n'$, if $1^n \notin L$ then $1^{n+n!} \notin L$. Let p_i be the i-th prime, with i odd, and $p_i > \max\{n', 4\}$. Since $\mathrm{lcm}\{\ell-1, \ell\} = (\ell-1) \cdot \ell$ for each $\ell > 1$, we have $k = \mathrm{lcm}\{1, \dots, p_i - 1\} \geq \mathrm{lcm}\{p_i - 2, p_i - 1\} = (p_i - 2) \cdot (p_i - 1) > p_i$, for each $p_i > 4$, and hence $k > p_i > n'$. By using Lemma 8, we have $f(k) = p_i$ and $f(k) > \max\{f(1), f(2), \dots, f(k-1)\}$. Thus $1^k \notin L$. But, by Lemma 9, $f(k + k!) = f(k) = p_i$ and hence $f(k + k!) \leq \max\{f(1), \dots, f(k), \dots, f(k + k! - 1)\}$. Therefore $1^{k+k!} \in L$, which contradicts the statement of the Theorem 5, i.e., $1^{k+k!} \notin L$. $\square$

Similarly we can prove:

Lemma 11. For each $s(n) \in o(\log n)$, $L \notin$ unary–Σ_2-SPACE$(s(n))$.

Proof. Here we use the fact that, by Theorem 5, we have that for $L \in$ unary–Σ_2-SPACE$(o(\log n))$ there must exist $n' \in \mathbb{N}$ such that $1^n \in L$ implies $1^{n+n!} \in L$, for each $n \geq n'$. The rest of argument mirrors Lemma 10; choosing $p_i > \max\{n', 4\}$ with i even, we get $1^k \in L$, but $1^{k+k!} \notin L$. $\square$

Theorem 6. $L \notin$ unary–Π_2-SPACE$(o(\log n)) \cup$ unary–Σ_2-SPACE$(o(\log n))$.

Corollary 2. For each $s(n) \in \Omega(\log\log n) \cap o(\log n)$,

$$\begin{array}{rcl}
\mathsf{unary}\text{–}\Delta_2\text{-SPACE}(s(n)) & \subset & \mathsf{unary}\text{–}\Sigma_2\text{-SPACE}(s(n)), \\
\mathsf{unary}\text{–}\Delta_2\text{-SPACE}(s(n)) & \subset & \mathsf{unary}\text{–}\Pi_2\text{-SPACE}(s(n)), \\
\mathsf{unary}\text{–}\Sigma_2\text{-SPACE}(s(n)) & \subset & \mathsf{unary}\text{–}\Delta_3\text{-SPACE}(s(n)), \\
\mathsf{unary}\text{–}\Pi_2\text{-SPACE}(s(n)) & \subset & \mathsf{unary}\text{–}\Delta_3\text{-SPACE}(s(n)).
\end{array}$$

The first two inclusions follow from Theorem 1, using fact that P_2 and S_2 are unary languages. See also Figure 2.

$$\Sigma_0 = \Pi_0 = \Delta_0 \overset{?}{\subseteq} \Sigma_1 = \Pi_1 = \Delta_1 \overset{?}{\subseteq} \Delta_2 \begin{array}{ccccc} \subset & \Sigma_2 & \subset & & \overset{?}{\subsetneq}\ \Sigma_3 \\ & \nparallel & & \Delta_3 & \\ \subset & \Pi_2 & \subset & & \underset{?}{\subsetneq}\ \Pi_3 \end{array}$$

Fig. 2. Alternating space hierarchy in unary case. Here Π_i (Σ_i, Δ_i) represents the complexity class unary–Π_i-SPACE$(s(n))$, for $s(n) \in \Omega(\log\log n) \cap o(\log n)$ (unary–Σ_i-SPACE$(s(n))$, unary–Δ_i-SPACE$(s(n))$, respectively). The equivalence at the first alternating level follows from [5].

Thus, we have a unary language that can be recognized both by Σ_3- and Π_3-alternating machines in space $O(\log\log n)$, but neither by Σ_2- nor by Π_2-alternating machines in space $s(n) \in o(\log n)$.

This shows that, for the alternating sublogarithmic space hierarchy on unary languages, the level "two and a half", i.e., Δ_3-SPACE, is separated from the second level.

An interesting open problem is the position of Δ_2-SPACE$(o(\log n))$. By Theorem 5, if $L \in \Delta_2$-SPACE$(o(\log n))$, then $1^n \in L$ if and only if $1^{n+n!} \in L$, for each sufficiently large n. Thus, the characterization of unary–Δ_2-SPACE$(o(\log n))$ is the same as the characterization of DSPACE, Σ_1-SPACE, or Π_1-SPACE, which makes the separation of these classes extremely difficult. As shown in [7], this question is closely related to the separation of DSPACE$(\log n)$ from NSPACE$(\log n)$, one of the fundamental open problems in complexity theory.

It is also not clear whether, in case of unary languages, the sublogarithmic alternating space hierarchy consists of a finite number of distinct levels, or it is infinite. We were only able to raise this hierarchy "from two to two and a half," despite the fact that, in binary case, we have a complete separation between each two levels [1, 6, 10], including intermediate levels Δ_k-SPACE$(o(\log n))$ [this paper].

REFERENCES

[1] VON BRAUNMÜHL, B.—GENGLER, R.—RETTINGER, R.: The Alternation Hierarchy For Sublogarithmic Space Is Infinite. Comput. Complexity, 1993, No. 3, pp. 207–30.

[2] Chandra, A. K.—Kozen, D. C.—Stockmeyer, L. J.: Alternation. J. Assoc. Comput. Mach., 1981, No. 28, pp. 114–33.

[3] Chang, R.—Hartmanis, J.—Ranjan, D.: Space Bounded Computations: Review and New Separation Results. Theoret. Comput. Sci., 1991, No. 80, pp. 289–302.

[4] Geffert, V.: Sublogarithmic Σ_2–SPACE Is Not Closed Under Complement and Other Separation Results. RAIRO Inform. Théor., 1993, No. 27, pp. 349–66.

[5] Geffert, V.: Tally Versions of the Savitch and Immerman-Szelepcsényi Theorems for Sublogarithmic Space. SIAM J. Comput., 1993, No. 22, pp. 102–13.

[6] Geffert, V.: A Hierarchy That Does Not Collapse: Alternations in low level space. RAIRO Inform. Théor., 1994, No. 28, pp. 465–512.

[7] Geffert, V.: Bridging Across the $\log(n)$ Space Frontier. Inform. & Comput., 1998, No. 142, pp. 127–58.

[8] Immerman, N.: Nondeterministic Space Is Closed Under Complementation. SIAM J. Comput., 1988, No. 17, pp. 935–38.

[9] Iwama, K.: ASPACE($o(\log\log n)$) is Regular. SIAM J. Comput., 1993, No. 22, pp. 136–46.

[10] Liśkiewicz, M.—Reischuk, R.: The Sublogarithmic Alternating Space World. SIAM J. Comput., 1996, No. 25, pp. 828–61.

[11] Szelepcsényi, R.: The Method of Forced Enumeration for Nondeterministic Automata. Acta Inform., 1988, No. 26, pp. 279–84.

[12] Szepietowski, A.: Turing Machines with Sublogarithmic Space. Lect. Notes Comput. Sci., Vol. 843. Springer-Verlag, 1994.

Viliam Geffert was born in 1955. He finished his studies at P. J. Šafárik University, Košice, Slovakia, in 1979. He received his Ph.D. degree in computer science at Comenius University in Bratislava, in 1988. His main research interests are space bounded computations, formal languages and finite automata, and in-place sorting algorithms.

Norbert Popély was born in 1975. He finished his studies at P. J. Šafárik University, Košice, Slovakia, in 1998, and received his Ph.D. degree at the same University in 2001, in the area of alternating computational models.