INFORMATIQUE THÉORIQUE ET APPLICATIONS

V. GEFFERT

Sublogarithmic $\sum_2$-space is not closed under complement and other separation results

Informatique théorique et applications, tome 27, n° 4 (1993), p. 349-366

<http://www.numdam.org/item?id=ITA_1993__27_4_349_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 27, n° 4, 1993, p. 349 à 366)

SUBLOGARITHMIC Σ_2-SPACE IS NOT CLOSED UNDER COMPLEMENT AND OTHER SEPARATION RESULTS (*)

by V. GEFFERT (1)

Communicated by I. WEGENER

Abstract. – *We shall show, for each* $s(n)$ *between* $\log\log(n)$ *and* $\log(n)$, *that* Σ_2-SPACE$(s(n))$ *is not closed under complement, because* Σ_2-SPACE$(s(n)) - \Pi_2$-SPACE$(s(n)) \neq \emptyset$. *This implies that the alternating hierarchy does not collapse below the level* 3 *and that the first three levels are separated,* i.e., Σ_1-SPACE$(s(n)) \subsetneqq \Sigma_2$-SPACE$(s(n)) \subsetneqq \Sigma_3$-SPACE$(s(n))$

Résumé. – *Nous montrons que pour tout* $s(n)$ *entre* $\log\log(n)$ *et* $\log(n)$, Σ_2-SPACE$(s(n))$ *n'est pas fermé par complémentation car* Σ_2-SPACE$(s(n)) - \Pi_2$-SPACE$(s(n)) \neq \emptyset$. *Ceci implique que la hiérarchie alternante ne s'écroule pas en dessous du niveau* 3 *et que les* 3 *premiers niveaux sont séparés, c'est-à-dire que* Σ_1-SPACE$(s(n)) \subsetneqq \Sigma_2$-SPACE$(s(n)) \subsetneqq \Sigma_3$-SPACE$(s(n))$.

1. INTRODUCTION

During the last few years, there has been some exciting development in the study of space-bounded computations. First, several space bounded hierarchies were shown to be finite ([8], [17], [11]). These results were then superseded by Immerman-Szelepcsényi proofs that nondeterministic space is closed under complement ([7], [15]). This implies that the alternating hierarchy of $s(n)$ space-bounded machines collapses to

$$\text{NSPACE}(s(n)) = \Sigma_1\text{-SPACE}(s(n)),$$

i.e.,

$$\Sigma_k\text{-SPACE}(s(n)) = \Pi_k\text{-SPACE}(s(n)) = \Sigma_1\text{-SPACE}(s(n)),$$

(*) Received May 1992, accepted December 1992.
(1) Department of Computer Science, University of P. J. Šafárik, Jesenná 5, 04154 Košice, Slovakia.

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/93/04 349 18/$3.80/© AFCET-Gauthier-Villars

for each $k \geq 1$, and each $s(n) \geq \log(n)$. Note that all these results were proved under the assumption that $s(n) \geq \log(n)$. Taking this fact into a consideration, a natural question arises; which of the results presented above can be extended to space bounds below $\log(n)$.

The first sign indicating that the alternating hierarchy behaves radically different below $\log(n)$ was the proof [4] that

$$\Pi_2\text{-SPACE}(\log\log(n)) - \Sigma_1\text{-SPACE}(s(n)) \neq \varnothing,$$

for each $s(n)$ between $\log\log(n)$ and $\log(n)$, *i.e.*, $s(n) \geq \log\log(n)$ and $\sup_{n \to \infty} s(n)/\log(n) = 0$. This result was then slightly improved in [16] by showing that $\Pi_2\text{-SPACE}(s(n)) - \Sigma_1\text{-SPACE}(s(n)) \neq \varnothing$ for each $s(n)$ below $\log(n)$, $s(n) \geq l(n)$ for some unbounded fully space constructible $l(n)$. There exist sublogarithmic, fully space constructible functions, but all these functions are necessarily nonmonotone and the corresponding space complexity classes do not contain DSPACE$(\log\log(n))$. That is, Π_2-SPACE$(s(n))$ properly contains Σ_1-SPACE$(s(n))$, and hence the space-bounded alternating hierarchy does not collapse to Σ_1 level for space bounds between $\log\log(n)$ and $\log(n)$. (Machines using less than $\log\log(n)$ space can recognize regular languages only [14].)

However, the inequations $\Pi_2\text{-SPACE}(s(n)) \neq \Sigma_1\text{-SPACE}(s(n))$, or even $\Pi_2\text{-SPACE}(s(n)) \neq \Sigma_2\text{-SPACE}(s(n))$ do not imply, for $s(n)$ below $\log(n)$, that $\Sigma_1\text{-SPACE}(s(n)) \neq \Pi_1\text{-SPACE}(s(n))$.

The situation is much more complicated in space below $\log(n)$ than above, because we do not have enough space to count the number of reachable configurations. In fact, the Immerman-Szelepcsényi algorithm ([7], [15]), for $s(n) \geq \log(n)$, can be used to generate all configurations reachable from *any given configuration* k, not only from the initial configuration. It is this fact that is needed to show that the alternating hierarchy collapses to Σ_1-SPACE$(s(n))$ for $s(n) \geq \log(n)$. This is not possible below $\log(n)$; once the input head has moved too far, the tape position is lost – we need $\log(n)$ bits to remember it – and hence we cannot restart, over and over again, computations beginning in the same configuration.

Moreover, we cannot detect an infinite cycle by counting the number of steps executed, and hence even the proof that *deterministic* space is closed under complement, *i.e.* DSPACE$(s(n)) =$ co-DSPACE$(s(n))$ requires more sophisticated argument [13] for $s(n)$ below $\log(n)$ than the standard cycle-detecting strategy for superlogarithmic case.

We are going to show that the first few levels of alternating hierarchy are distinct and the hierarchy does not collapse below the level three. First, we shall improve the result of Szepietowski [16], who presented a language *LE* such that

$$LE \in \Pi_2\text{-SPACE}(s(n)) - \Sigma_1\text{-SPACE}(s(n)).$$

We shall show that for the language LE presented in [16] we actually have

$$LE \in \Pi_2\text{-SPACE}(s(n)) - \Sigma_2\text{-SPACE}(s(n)),$$

which gives that Σ_2-SPACE$(s(n))$ is not closed under complement and that

$$\Sigma_2\text{-SPACE}(s(n)) \neq \Pi_2\text{-SPACE}(s(n)).$$

This implies that the alternating hierarchy does not collapse below the level Σ_3 and that the first three levels are separated for $s(n)$ between $\log\log(n)$ and $\log(n)$;

$$\Sigma_1\text{-SPACE}(s(n)) \subsetneqq \Sigma_2\text{-SPACE}(s(n)) \subsetneqq \Sigma_3\text{-SPACE}(s(n)).$$

2. PRELIMINARIES

We shall consider the standard Turing machine model having a finite control, a two-way read-only input tape, and a separate semi-infinite two-way read-write worktape.

The reader is assumed to be familiar with the notion of alternating machine, which is at the same time a generalization of nondeterminism and a mechanism to model parallel computations. Alternation was introduced independently in [3] and [9]. See [2] for more exact definition and properties of alternating machines. We shall now introduce this notion less formally.

A memory state of a Turing machine is an ordered triple $q = \langle r, u, j \rangle$, where r is a state of the machine's finite control, u is a string of worktape symbols written down on the worktape, and j is a position of the worktape head.

A configuration is an ordered pair $k = \langle q, i \rangle$, where q is a memory state and i is a position of the input tape head.

An alternating Turing machine is very similar to a standard nondeterministic machine, only the definition of accepting the input has been modified: The behavior of a machine can still be represented as a computation tree, the

branches of which represent all possible computations. For a standard nondeterministic machine such a computation tree is accepting as soon as a single branch is terminated in an accepting configuration. The idea for alternating machines is to equip the finite control states with labels *existential* and *universal*, configuration inherits the label of the state included. Now we assign a quality accept/reject to every node in the computation tree:

1. A leaf node is an accept-node if it corresponds to an accepting configuration.

2. An internal node representing an existential configuration is an accept-node if at least one of its successors is an accept-node.

3. An internal node representing a universal configuration is an accept-node if all of its successors are accept-nodes.

4. Any node which is not determined to be an accept-node by application of the above rules is a reject-node.

This definition is a simplified version of presentation in [1]; we do not distinguish between computation paths terminated in rejecting configurations and infinite cycles. But even nodes having infinite subtrees can be marked as accept-nodes, since an accepting son of an existential node overrides the reject label of another son.

By definition, the input is accepted if the root is determined to be an accept-node.

An alternating machine is $s(n)$ *space bounded*, if all computation paths on all inputs of length n use at most $s(n)$ tape squares on the worktape.

Σ_k-SPACE$(s(n))$ *and* Π_k-SPACE$(s(n))$ denote the classes of languages recognizable by alternating $O(s(n))$ space bounded machines making less than k alternations between universal and existential states, with the initial state existential or universal, respectively.

Clearly, nondeterministic machines equal to Σ_1 machines, *i.e.*,

$$\Sigma_1\text{-SPACE}(s(n)) = \text{NSPACE}(s(n)).$$

The class of languages recognizable by deterministic machines in $O(s(n))$ space is denoted by

$$\Sigma_0\text{-SPACE}(s(n)) = \Pi_0\text{-SPACE}(s(n)) = \text{DSPACE}(s(n)).$$

Finally, we introduce two different notions that should not be confused; it is an infinite never-ending cycle, and an iteration of a loop traversing the input 1^n that must be terminated as soon as the input head hits the left/right

endmarker:

An infinite cycle is a computation that begins and ends in the same configuration $\langle q, i \rangle$.

A loop of length l, for $l \neq 0$, is a computation beginning in configuration $\langle q, i \rangle$ and ending in $\langle q, i+l \rangle$, for some memory state q and tape position i. Moreover, neither of the endmarkers is visited by the input head during this computation.

3. THE $N \to N+N!$ METHOD

It was shown in [14] that a deterministic machine using less than $\log(n)$ space must execute an oblivious loop to move the input head from one endmarker to the other and so it cannot distinguish between inputs 1^n and $1^{n+kn!}$, for each $k \geqq 0$. This "$n \to n+n!$" trick has been then extended to the nondeterministic case [6], *i.e.*, the nondeterministic machine also cannot distinguish inputs 1^n and $1^{n+kn!}$: If L is in Σ_1-SPACE$(s(n))$, for some $s(n)$ satisfying $\sup_{n \to \infty} s(n)/\log(n) = 0$, then there exists $\breve{n}$ such that

$$1^n \in L \quad \text{if and only if} \quad 1^{n+kn!} \in L,$$

for each $n \geqq \breve{n}$, and each $k \geqq 0$. This gives immediately [12, 5, 6] that if $s(n)$ below $\log(n)$ is fully space constructible then, for each $n \geqq \breve{n}$ and each $k \geqq 0$,

$$s(n) = s(n+kn!). \tag{1}$$

A function $s(n)$ *is fully space constructible* if there exists a deterministic Turing machine which for all inputs of length n marks-off exactly $s(n)$ tape squares on its worktape and stops, not having used more than $s(n)$ space.

The proof that Σ_2-SPACE$(s(n)) \neq \Pi_2$-SPACE$(s(n))$ is based on the extension of the $n \to n+n!$ method to Σ_2 and Π_2 alternating machines. However, the extension is not symmetric any more; we shall show that

$$1^n \in L \quad \text{implies} \quad 1^{n+kn!} \in L,$$

for each L in Σ_2-SPACE$(s(n))$, but it is still possible that $1^{n+n!} \in L$ and $1^n \notin L$ for some n. On the other hand, $1^{n+kn!} \in L$ implies $1^n \in L$ for each L in Π_2-SPACE$(s(n))$, but we can have $1^n \in L$ together with $1^{n+n!} \notin L$. This shows that Σ_2 and Π_2 machines behave differently and recognize different classes of languages.

Now, we shall consider a Σ_2-SPACE$(s(n))$ machine A, *i.e.*, a machine that has its initial state existential and is allowed to make a single alternation. A is space bounded by $s(n)$, with $\sup_{n \to \infty} s(n)/\log(n) = 0$.

It is not too hard to show that for each $s(n)$ space bounded machine there exists a constant $c \geqq 6$ such that the number of different memory states for inputs of length n is at most $c^{s(n)+1}$. Similarly, the number of configurations is bounded by $n \cdot c^{s(n)+1}$, for each $n \geqq 1$. Define

$$M = c^{s(n)+1}$$

– an upper bound on the number of reachable memory states for input 1^n.

Further, for each space bound $s(n)$ with $\sup_{n \to \infty} s(n)/\log(n) = 0$, there exists $\breve{n} \geqq 2$ such that

$$2M^6 = 2 \cdot (c^{s(n)+1})^6 < n, \quad \text{for each } n \geqq \breve{n}. \tag{2}$$

The proof is straightforward, since there can be only finitely many n's such that $(s(n)+1)/\log(n) \geqq 1/7 \log(c)$.

Before passing further, we need some technical lemmas. For the detailed proof of the Lemma 1, 2, and 3, the reader is referred to [6]. The proofs are based on the assumption that $M^6 < n$, so we assume that the input is 1^n, for sufficiently large n satisfying (2).

LEMMA 1 [6, Lemma 3]: *If there exists a computation path from the configuration* $\langle q_1, i \rangle$ *to* $\langle q_2, i \rangle$ *such that the input head never visits the right (left) endmarker, then the shortest computation path from* $\langle q_1, i \rangle$ *to* $\langle q_2, i \rangle$ *never moves the input head farther than* M^2 *positions to the right (left) of* i.

I.e., each "U-turn" beginning and ending at the same position of the input tape has a "short-cut" not wider than M^2.

The next lemma states that computation paths on tally inputs are "position independent" provided that they begin and end at least M^2+1 positions away from either endmarker:

LEMMA 2 [6, Lemma 4]: *If the machine* A *can get from configuration* $\langle q_1, i \rangle$ *to* $\langle q_2, i+l \rangle$ *by a computation path visiting neither of the endmarkers, then* A *can get from* $\langle q_1, j \rangle$ *to* $\langle q_2, j+l \rangle$, *for each* j *satisfying*

$$M^2 + 1 \leqq j \leqq (n+1) - (M^2+1),$$
$$M^2 + 1 \leqq j + l \leqq (n+1) - (M^2+1).$$

The proof [6] is based on the Lemma 1.

The following theorem asserts that each machine using less than $\log(n)$ space cannot distinguish, by a single traversal from left to right (or vice versa), between an input tape segment of length x on the input 1^n and segment of length $x+n!$ on the input $1^{n+n!}$, for no $x > M^6$.

THEOREM 1: *The machine A has a computation path from configuration $\langle q_1, i_1 \rangle$ to $\langle q_2, i_2 \rangle$ on the input 1^n if and only if A has a path from $\langle q_1, i_1 \rangle$ to $\langle q_2, i_2 + n! \rangle$ on the input $1^{n+n!}$. This holds for each q_1, q_2 using at most $s(n)$ space, and each i_1, i_2 satisfying $i_2 - i_1 = x > M^6 = (c^{s(n)+1})^6$. A similar statement can be formulated for traversals from right to left.*

Proof: A. $(n \to n+n!)$. Because the number of different memory states is bounded by $M \leqq M^6 < x \leqq n$, each machine traversing the segment of length x must enter some memory state twice, *i.e.*, it executes a loop of length l, with $1 \leqq l \leqq M$. But this loop can be iterated $F = \prod_{\substack{i=1 \\ i \neq l}}^{n} i$ more times, and hence A has also a computation path traversing the segment of length $x + l.F = x + n!$.

B. $(n+n! \to n)$. The converse is not so simple because A is far from repeating regularly any loop it gets in, since it can jump out of the loop by making a nondeterministic decision. Still, from [6, Theorem 1] it follows, for each computation path from $\langle q_1, i_1 \rangle$ to $\langle q_2, i_2 + n! \rangle$ not using more than $s(n)$ space on the input $1^{n+n!}$, that there exists a path from $\langle q_1, i_1 \rangle$ to $\langle q_2, i_2 + n! \rangle$ such that (see *fig.* 1):

(*a*) having traversed s_1 input tape positions to the right,

(*b*) A gets into a loop of length l which is iterated r times,

(*c*) and then it traverses the rest of the segment, which is of length s_2,

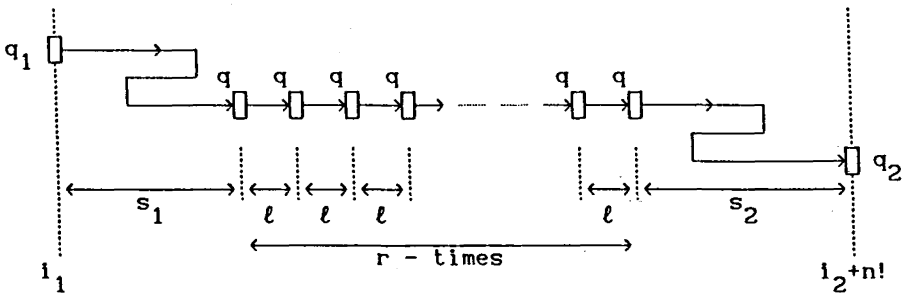

Figure 1

for some s_1, l, r, s_2 satisfying

$$M^2+1 \leqq s_1 \leqq M^4$$
$$M^2+1 \leqq s_2 \leqq M^4$$
$$1 \leqq l \leqq M.$$

Note that

$$x+n\,!=(i_2+n\,!)-i_1=s_1+r.l+s_2 \leqq 2M^4+r.l \leqq M^6+r.l<x+r.l,$$

and therefore

$$r.l \geqq n\,!=l.\prod_{\substack{i=1 \\ i \neq l}}^{n} i=l.F,$$

i.e.,

$$r \geqq F,$$

since $l \leqq M \leqq 2M^4 \leqq M^6 < x \leqq n$. Thus, we have a loop that is iterated at least $r \geqq F$ times. The first F iterations of this loop traverse exactly $l.F=n\,!$ positions to the right, beginning and ending in the same memory state. Cutting F iterations out of the computation path, we shall obtain a computation that traverses the segment of length x on the input 1^n.

For a more detailed proof, the reader is referred to [6]. Theorem 2 in [6] is actually a special case of the above argument, for $x=n$. □

We are now ready to present a characterization theorem for Σ_2-SPACE$(s(n))$:

THEOREM 2: *If* L *is in* Σ_2-SPACE$(s(n))$, *for some* $s(n)$ *satisfying* $\sup_{n \to \infty} s(n)/\log(n)=0$, *then there exists* $\breve{n}$ *such that, for each* $n \geqq \breve{n}$ *and each* $k \geqq 0$,

$$1^n \in L \quad \textit{implies} \quad 1^{n+kn\,!} \in L.$$

Proof: Let $1^n \in L$, for some sufficiently large n satisfying $2M^6 < n$. Since L is recognized by a Σ_2-alternating $s(n)$ space bounded machine A, we have that there exists a computation path from the initial configuration $\langle q_I, 0 \rangle$ to some configuration $\langle q_x, x \rangle$ such that

(A) the machine changes its state from existential to universal,

(B) all computation paths from $\langle q_x, x \rangle$ are terminated in accepting configurations,

(C) no configuration reachable from $\langle q_x, x \rangle$ uses more than $s(n)$ space on the worktape.

Because $2M^6 < n$, there are now two cases:

Either $x > M^6$, *i.e.*, A alternates from existential to universal state at least M^6 positions away from the left endmarker,

or $y = n+1-x > M^6$, *i.e.*, A alternates at least M^6 positions away from the right endmarker.

Suppose that $x > M^6$, the argument for $y > M^6$ is very similar, due to symmetry reasons, and therefore it is omitted. Now, consider the input $1^{n+n!}$. It is not too hard to show that our machine cannot distinguish between position x on the input 1^n and position $x+n!$ on the input $1^{n+n!}$.

CLAIM: *The machine A has a computation path from $\langle q_A, i_A \rangle$ to $\langle q_B, i_B \rangle$ on the input 1^n, for $i_A, i_B \in \{0, x, n+1\}$, if and only if A has a computation path from $\langle q_A, i'_A \rangle$ to $\langle q_B, i'_B \rangle$ on the input $1^{n+n!}$, for $i'_A, i'_B \in \{0, x+n!, n+1+n!\}$, respectively. This holds for each q_A, q_B not using space above $s(n)$.*

(*If $y > M^6$, we can formulate a very similar claim for positions $i'_A, i'_B \in \{0, x, n+1+n!\}$.*)

Proof: We shall show, for example, that if A has a computation path from $\langle q_A, x+n! \rangle$ to $\langle q_B, 0 \rangle$ on the input $1^{n+n!}$, then A has a computation path from $\langle q_A, x \rangle$ to $\langle q_B, 0 \rangle$ on the input 1^n. The proofs for all other cases parallel this one and therefore they are omitted.

Let $q_1, q_2, \ldots, q_t$ be the sequence of memory states along the computation path from $\langle q_A, x+n! \rangle$ to $\langle q_B, 0 \rangle$ in which the input head scans the endmarkers or visits the position $x+n!$. (See *fig.* 2.)

(i) If A can get from memory state q_i to q_{i+1} by traversing the tape segment of length $x+n!$ from left to right on the input $1^{n+n!}$, then, by Theorem 1, A can get from q_i to q_{i+1} by traversing the tape segment x on the input 1^n. The same also holds for traversals from right to left.

(ii) Further, if A gets from $\langle q_i, 0 \rangle$ to $\langle q_{i+1}, 0 \rangle$ on the input $1^{n+n!}$, never visiting the position $x+n!$, then, by the Lemma 1, we have a computation path from $\langle q_i, 0 \rangle$ to $\langle q_{i+1}, 0 \rangle$ such that the head is never moved to the right of the position M^2. Because $M^2 \leqq M^6 < x$, we have a room to execute this computation on the input 1^n. The same holds also for computations beginning and ending at the position $x+n!$, taking place to the left of $x+n!$, and never visiting the left endmarker.

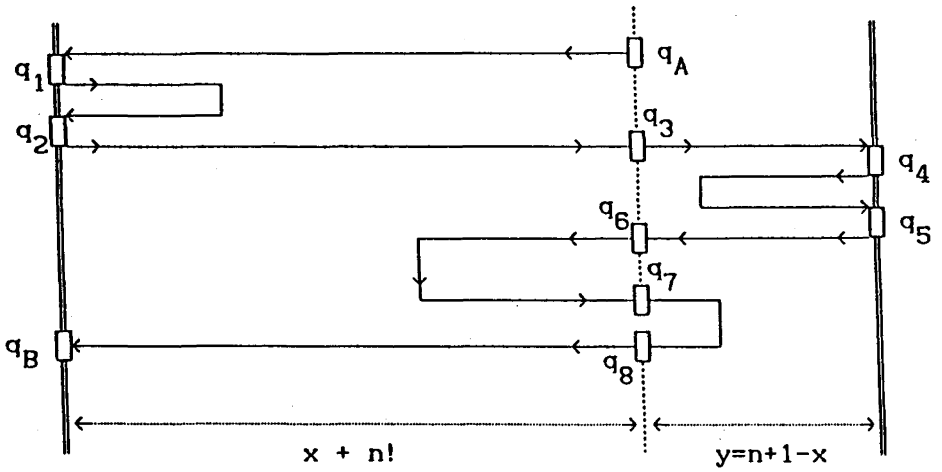

Figure 2

(iii) Because the behavior on segment y placed at the right is exactly the same for inputs 1^n and $1^{n+n!}$, we have, by a straightforward induction on the number of times the head hits the endmarkers or visits the position $x+n!$, that $\langle q_B, 0 \rangle$ is reachable from $\langle q_A, x \rangle$ on the input 1^n.

This completes the proof of the claim.

By the Claim, we have that if A can get from $\langle q_I, 0 \rangle$ to $\langle q_x, x \rangle$ on the input 1^n, then A can get from $\langle q_I, 0 \rangle$ to $\langle q_x, x+n! \rangle$ on the input $1^{n+n!}$, *i.e.*, from the initial configuration to the same alternating memory state, but $n!$ positions more to the right. We shall now show that all computation paths beginning in $\langle q_x, x+n! \rangle$ on the input $1^{n+n!}$ are terminated in accepting configurations. Supposing the contrary, we have the following cases to consider:

(*a*) There is a computation path from $\langle q_x, x+n! \rangle$ that terminates in a rejecting configuration, not using more than $s(n)$ space.

(*b*) Some computation path enters an infinite cycle, not using more than $s(n)$ space.

(*c*) The same as (*a*) or (*b*), but the space used exceeds $s(n)$, since the machine can use as much as $s(n+n!)$ space on the input $1^{n+n!}$.

We shall show that each of the above cases leads to a contradiction:

(*a*) Suppose that some computation path beginning in $\langle q_x, x+n! \rangle$ is terminated in a rejecting configuration $\langle q_R, 0 \rangle$. (We may assume, without

loss of generality, that all terminating computation paths stop with the input head positioned at the left endmarker.)

Because the space used does not exceed $s(n)$, we can use the Claim, and hence $\langle q_R, 0 \rangle$ is reachable from $\langle q_x, x \rangle$ on the input 1^n. Thus, we have a path from $\langle q_x, x \rangle$ on 1^n terminated in the rejecting configuration. But this is a contradiction to (B).

(*b*) Now, suppose that some computation path enters an infinite cycle, not using more than $s(n)$ space. There are the following subcases:

(*b* 1) The cycle itself does not visit any of the endmarkers, but the computation path from $\langle q_x, x+n! \rangle$ to the first configuration of the cycle does (see *fig.* 3):

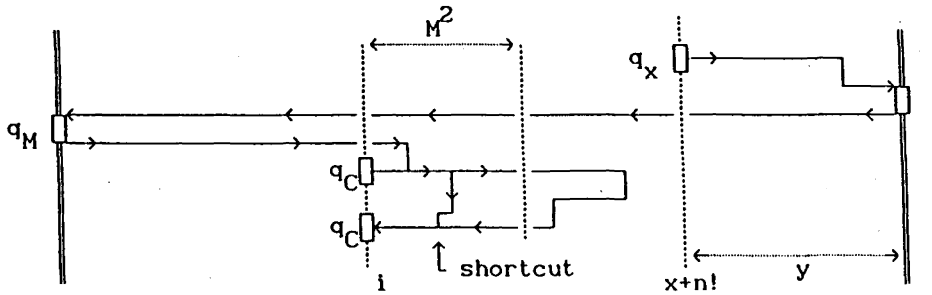

Figure 3

Let q_M be the last memory state along the path from $\langle q_x, x+n! \rangle$ to the cycle such that the input head scanned an endmarker. Assume that q_M was at the left endmarker of the input $1^{n+n!}$. The argument for the right endmarker is very similar and therefore it is omitted. By the Claim, $\langle q_M, 0 \rangle$ is reachable from $\langle q_x, x \rangle$ on the input 1^n.

Now, let $\langle q_C, i \rangle$ be the leftmost configuration of the cycle, *i.e.*, with minimal i. Thus, $\langle q_C, i \rangle$ is reachable from $\langle q_C, i \rangle$ by a computation path neither visiting the right endmarker, nor moving the head to the left of i. But then, by Lemma 1, we have also a cycle from $\langle q_C, i \rangle$ to $\langle q_C, i \rangle$ never moving the head to the right of $i+M^2$.

Further, $\langle q_C, i \rangle$ is reachable from $\langle q_M, 0 \rangle$. But then the memory state q_C is also reachable at a position $i' \leq (M^2+1)+(M+1)$, for, if $i > (M^2+1)+(M+1)$, then we can find a loop between positions M^2+1 and $(M^2+1)+(M+1)$. By Lemma 2, we can remove this loop from the computation path and shift the cycle from $\langle q_C, i \rangle$ to $\langle q_C, i \rangle$ more to the left (see *fig.* 4):

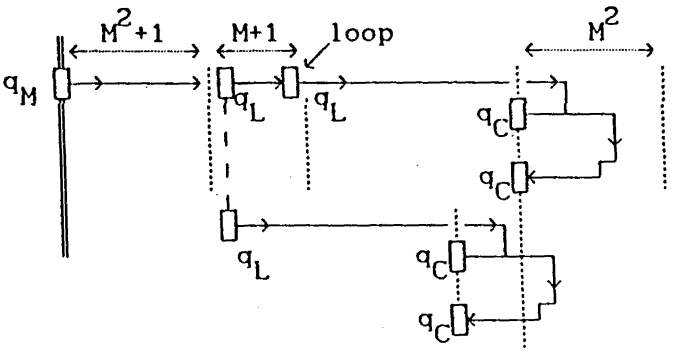

Figure 4

This process can be repeated until we obtain a position

$$i' \leqq (M^2+1)+(M+1).$$

Thus we a have a computation path such that

(i) $\langle q_C, i' \rangle$ is reachable from $\langle q_M, 0 \rangle$, for some $i' \leqq (M^2+1)+(M+1)$,

(ii) $\langle q_C, i' \rangle$ is reachable from $\langle q_C, i' \rangle$ by a path never moving the input head to the right of $i'+M^2$.

This implies that now the entire cycle takes place between positions 0 and $(M^2+1)+(M+1)+M^2 \leqq M^6 < x$. Since $\langle q_M, 0 \rangle$ is reachable from $\langle q_x, x \rangle$ on the input 1^n, we have a computation path from $\langle q_x, x \rangle$ that enters an infinite cycle on 1^n. But this is a contradiction to (B).

(*b* 2) Some computation path enters an infinite cycle, not visiting any of the endmarkers at all (see *fig.* 5):

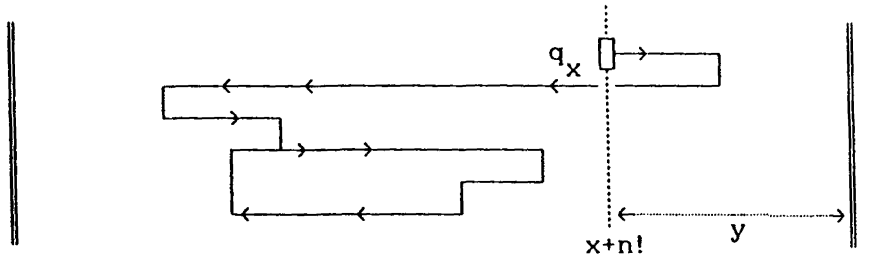

Figure 5

But then, by a reasoning very similar to (*b* 1), we can find a computation path with an infinite cycle not moving the input head farther than $(M^2+1)+(M+1)+M^2 \leqq M^6 < x$ positions to the left of $x+n!$. Thus, we have enough room to run this computation on the input 1^n, which is a contradiction to (B).

(*b* 3) The cycle itself visits the endmarkers: If, for example, the cycle visits the left endmarker in configuration $\langle q_C, 0 \rangle$, then, using the Claim, we can show that

(i) if $\langle q_C, 0 \rangle$ is reachable from $\langle q_x, x+n! \rangle$ on the input $1^{n+n!}$, then so is $\langle q_C, 0 \rangle$ from $\langle q_x, x \rangle$ on the input 1^n.

(ii) Similarly, if $\langle q_C, 0 \rangle$ is reachable from $\langle q_C, 0 \rangle$ on $1^{n+n!}$, then so is $\langle q_C, 0 \rangle$ from $\langle q_C, 0 \rangle$ on 1^n.
Again, we have an infinite cycle on 1^n, which contradicts (B).

(*c*) Suppose that some computation path terminates in a rejecting configuration or enters an infinite cycle, but the space used exceeds $s(n)$: Recall that the machine can potentially use $s(n+n!) > s(n)$ space on the input $1^{n+n!}$. Then the arguments presented in (*a*) or (*b*) do not hold, since they are based on the Lemma 1, 2, and Theorem 1, *i.e.*, on the assumption that the number of different memory states $M = c^{s(n)+1}$ satisfies $M^6 < x$. But it is possible that $(c^{s(n+n!)+1})^6 \geqq x$.

Still, $\langle q_x, x+n! \rangle$ on $1^{n+n!}$ is $s(n)$ space bounded, since $\langle q_x, x \rangle$ is reachable from the initial configuration on 1^n. But if some path from $\langle q_x, x+n! \rangle$ exceeds space $s(n)$, then this path must reach a first configuration $\langle q_S, i \rangle$ such that

(i) $\langle q_S, i \rangle$ uses exactly $s(n)$ space on the worktape,

(ii) the machine is going to use space $s(n)+1$ in the next step. Because the computation between $\langle q_x, x+n! \rangle$ and $\langle q_S, i \rangle$ on $1^{n+n!}$ is $s(n)$ space bounded, we can use the Lemma 1, 2, and Theorem 1. This implies, by the same argument as in (*b* 1) and (*b* 2), that the memory state q_S is also reachable at a position i' such that

(*c* 1) either $\langle q_S, i' \rangle$ is at most $(M^2+1)+(M+1)$ positions away from the left/right endmarker,

(*c* 2) or at most $(M^2+1)+(M+1)$ positions away from $\langle q_x, x+n! \rangle$.

In both cases, we can conclude that q_S is reachable from $\langle q_x, x \rangle$ on the input 1^n. But then we have a computation path from $\langle q_x, x \rangle$ that uses at least $s(n)+1$ space on 1^n (by a single step from q_S), which is a contradiction to (C).

Thus, all computation paths beginning in $\langle q_x, x+n! \rangle$ on the input $1^{n+n!}$ are terminated in accepting configurations, and hence we have an accepting computation subtree for $1^{n+n!}$. In addition, we have shown that if a Σ_2-alternating machine accepts 1^n by an $s(n)$ space bounded subtree, then $1^{n+n!}$ is also accepted by at least one subtree that is $s(n)$ space bounded.

Moreover, all arguments above clearly hold for $1^{n+kn!}$, for each $k \geqq 1$. We only have to replace $n!$ everywhere by $kn!$, which completes the proof of the theorem. □

The converse of Theorem 2 does not hold; it is still possible that $1^{n+n!} \in L$ and $1^n \notin L$, since an accepting computation path from $\langle q_I, 0 \rangle$ on input $1^{n+n!}$ can alternate in a configuration $\langle q_x, x \rangle$ with the input head somewhere in the middle of $1^{n+n!}$ so the segment of length $n!$ is neither to the left, nor to the right of x.

However, the converse of Theorem 2 does hold for Π_2-SPACE$(s(n))$, with $s(n)$ below $\log(n)$: The statement for Π_2 machines is that if 1^n is rejected, then $1^{n+kn!}$ is also rejected. The argument is very similar to Theorem 2, but instead of analyzing accepting computation paths on input 1^n, we consider a *rejecting* computation path of a Π_2 machine, beginning in universal $\langle q_I, 0 \rangle$ on input 1^n. A rejecting path enters a configuration $\langle q_x, x \rangle$, now changing its state from universal to existential, such that *no computation path* from $\langle q_x, x \rangle$ is terminated in an accepting configuration. Again, we can find a corresponding path from $\langle q_I, 0 \rangle$ to $\langle q_x, x \rangle$ or $\langle q_x, x+n! \rangle$ on input $1^{n+n!}$ (depending on whether $x > M^6$ or not). No computation path from q_x can be terminated in an accepting configuration on $1^{n+n!}$, otherwise we could find an accepting path for q_x on 1^n, which is a contradiction. Hence, $1^{n+n!}$ is also rejected. But there is one more case to consider, namely, a rejecting path from $\langle q_I, 0 \rangle$ on 1^n can enter an infinite cycle making no alternation at all. But then we can easily find a corresponding cycle reachable from $\langle q_I, 0 \rangle$ on $1^{n+n!}$.

4. SEPARATION RESULTS

Using the Theorem 2, we can now separate Σ_2-SPACE$(s(n))$ from Π_2-SPACE$(s(n))$. For any fully space constructible $l(n)$ and any integer $a \geqq 2$, consider the languages

$$GT_{a,l} = \{ 1^n, n > \text{least common multiple of } 1, 2, 3, \ldots, a^{l(n)} \},$$

$$LE_{a,l} = \{ 1^n, n \leqq \text{least common multiple of } 1, 2, 3, \ldots, a^{l(n)} \}.$$

THEOREM 3: *There exist* $a \geqq 2$ *and fully space constructible* $l(n)$ *such that, for each* $s(n)$ *between* $\log\log(n)$ *and* $\log(n)$, i.e., *with* $s(n) \geqq \log\log(n)$ *and* $\sup_{n \to \infty} s(n)/\log(n) = 0$,

$$GT_{a,l} \in \Sigma_2\text{-SPACE}(s(n)) - \Pi_2\text{-SPACE}(s(n)),$$

and

$$LE_{a,l} \in \Pi_2\text{-SPACE}(s(n)) - \Sigma_2\text{-SPACE}(s(n)).$$

Proof: A. It is easy to show [16] that $GT_{a,l}$ is in Σ_2-SPACE$(l(n))$, for each a and each l: The Σ_2-alternating machine first deterministically constructs the value of $l(n)$. Then it traverses along the input tape and nondeterministically finds the tape position $k < n$ such that k can be divided by i, for each $i = 1, 2, 3, \ldots, a^{l(n)}$. Then, branching universally, our machine verifies its guess, *i.e.*, it moves the input head to the left endmarker and checks if i divides k, for each $i \leqq a^{l(n)}$. Note that we have to store $a^{l(n)}$ and i on the worktape, but not k. Hence, our machine is $O(l(n))$ space bounded. Similarly, $LE_{a,l}$ is in Π_2-SPACE$(l(n))$.

B. There exist unbounded fully space constructible functions in $O(\log\log(n))$, for example,

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

(For proof, *see* [4].) By (A), we have that $GT_{a,f}$ is in Σ_2-SPACE$(s(n))$, for each $a \geqq 2$ and each $s(n) \geqq \log\log(n)$.

C. Now we shall show that there exists $a \geqq 2$ such that, for each $s(n)$ below $\log(n)$, $GT_{a,f}$ is not in Π_2-SPACE$(s(n))$ and that $LE_{a,f}$ is not in Σ_2-SPACE$(s(n))$.

So far, it has been shown [16], for each $s(n)$ below $\log(n)$ and each unbounded fully space constructible $l(n)$, that there exists $a \geqq 2$ such that $LE_{a,l}$ is not in Σ_1-SPACE$(s(n))$. Thus we have $\tilde{a} \geqq 2$ such that $LE_{\tilde{a},f}$ is not in Σ_1-SPACE$(\log\log(n))$. Therefore, $LE_{\tilde{a},f}$ is not regular and hence it is infinite.

Suppose that $LE_{\tilde{a},f}$ is in Σ_2-SPACE$(s(n))$, for some $s(n)$ below $\log(n)$. We can now take sufficiently large n so that $1^n \in LE_{\tilde{a},f}$, and hence, by Theorem 2, $1^{n+kn!} \in LE_{\tilde{a},f}$ for each $k \geqq 0$. Thus, we have n satisfying

$$n + kn! \leqq \text{least common multiple of } 1, 2, 3, \ldots, \tilde{a}^{f(n+kn!)},$$

for each $k \geqq 0$. Because $f(n)$ is fully space constructible, we have, by (1), that $f(n+kn!)=f(n)$ for each sufficiently large n and each $k \geqq 0$. Therefore,

$$n+kn! \leqq \text{least common multiple of } 1, 2, 3, \ldots, \tilde{a}^{f(n)},$$

for each $k \geqq 0$. But this is a contradiction for sufficiently large k. Hence, $LE_{\tilde{a}, f} \notin \Sigma_2\text{-SPACE}(s(n))$ or, equivalently, $GT_{\tilde{a}, f} \notin \Pi_2\text{-SPACE}(s(n))$. Combining this with (B), we have

$$GT_{\tilde{a}, f} \in \Sigma_2\text{-SPACE}(s(n)) - \Pi_2\text{-SPACE}(s(n)),$$

and also

$$LE_{\tilde{a}, f} \in \Pi_2\text{-SPACE}(s(n)) - \Sigma_2\text{-SPACE}(s(n)). \quad \square$$

COROLLARY 1: *For each* $s(n)$ *between* $\log\log(n)$ *and* $\log(n)$,

$$\Sigma_2\text{-SPACE}(s(n)) - \Pi_2\text{-SPACE}(s(n)) \neq \varnothing,$$

and also

$$\Pi_2\text{-SPACE}(s(n)) - \Sigma_2\text{-SPACE}(s(n)) \neq \varnothing.$$

Moreover, it is obvious that $\Sigma_i\text{-SPACE}(s(n)) \subseteq \Pi_{i+1}\text{-SPACE}(s(n))$ and that $\Pi_i\text{-SPACE}(s(n)) \subseteq \Sigma_{i+1}\text{-SPACE}(s(n))$, for each $i \geqq 1$. From this we have:

COROLLARY 2: *For each* $s(n)$ *between* $\log\log(n)$ *and* $\log(n)$,

$$GT_{\tilde{a}, f} \in \Sigma_2\text{-SPACE}(s(n)) - \Sigma_1\text{-SPACE}(s(n)),$$
$$LE_{\tilde{a}, f} \in \Pi_2\text{-SPACE}(s(n)) - \Pi_1\text{-SPACE}(s(n)),$$
$$LE_{\tilde{a}, f} \in \Sigma_3\text{-SPACE}(s(n)) - \Sigma_2\text{-SPACE}(s(n)),$$
$$GT_{\tilde{a}, f} \in \Pi_3\text{-SPACE}(s(n)) - \Pi_2\text{-SPACE}(s(n)).$$

That is, the alternating space hierarchy does not collapse below the level 3*:*

$$\Sigma_1\text{-SPACE}(s(n)) \subsetneqq \Sigma_2\text{-SPACE}(s(n)) \subsetneqq \Sigma_3\text{-SPACE}(s(n)),$$

$$\Pi_1\text{-SPACE}(s(n)) \subsetneqq \Pi_2\text{-SPACE}(s(n)) \subsetneqq \Pi_3\text{-SPACE}(s(n)).$$

The assumption that $s(n) \geqq \log\log(n)$ can be replaced everywhere by a slightly more general $s(n) \geqq l(n)$, for some unbounded fully space constructible $l(n)$.

However, the two most important problems remain open; it is not known if

$$\text{DSPACE}(s(n)) = \text{NSPACE}(s(n)),$$

or

$$\text{NSPACE}(s(n)) = \text{co-NSPACE}(s(n)),$$

i.e., if

$$\Sigma_0\text{-SPACE}(s(n)) = \Sigma_1\text{-SPACE}(s(n)),$$

or

$$\Sigma_1\text{-SPACE}(s(n)) = \Pi_1\text{-SPACE}(s(n)), \text{ for } s(n) \text{ below } \log(n).$$

REFERENCES

1. P. van Emde Boas, Machine models and simulations, I.T.L.I. Prepublication Series for Computation and Complexity Theory, CT-89-02, to appear in: J. van Leeuwen (ed): *Handbook of Theoretical Computer Science*, North-Holland.
2. A. K. Chandra, D. Kozen, L. J. Stockmeyer, Alternation, *J. Assoc. Comput. Mach.*, 1981, *28*, pp. 114-133.
3. A. K. Chandra and L. J. Stockmeyer, Alternation, *Proc. of 17th I.E.E.E.-F.O.C.S.*, 1976, pp. 98-108.
4. J. H. Chang, O. H. Ibarra, B. Ravikumar, and L. Berman, Some observations concerning alternating Turing machines using small space, *Inform. Process. Lett.*, 1987, *25*, pp. 1-9 (Erratum: *27*(1988), 53).
5. A. R. Freedman and R. E. Ladner, Space bounds for processing counterless inputs, *J. Comput. Syst. Sciences*, 1975, *11*, pp. 118-128.
6. V. Geffert, Nondeterministic computations in sublogarithmic space and space constructibility, *S.I.A.M. J. Comput.*, 1991, *20*, No. 3, pp. 484-498.
7. N. Immerman, Nondeterministic space is closed under complement, *S.I.A.M. J. Comput.*, 1988, *17*, pp. 935-938.
8. B. Jenner, B. Kirsig, and K. Lange, The logarithmic alternation hierarchy collapses: $A\Sigma_2^L = A\Pi_2^L$, Information and Computation, 1989, *80*, pp. 269-288.
9. D. Kozen, On parallelism in Turing machines, *17th I.E.E.E.-F.O.C.S.*, 1976, pp. 89-97.
10. D. Ranjan, R. Chang, and J. Hartmanis, Space bounded computations: Review and new separation results, *Theoret. Comp. Sci.*, 1991, *80*, pp. 289-302.
11. U. Schoning and K. Wagner, Collapsing oracle hierarchies, census functions, and logarithmically many queries, *Proc. of STACS'88*, Springer-Verlag, LNCS 294, 1988, pp. 91-97.
12. J. Seiferas, A note on notions of tape constructibility, Technical Report CSD-TR-187, Pennsylvania State University, 1976.

13. M. SIPSER, Halting space bounded computations, *Theoret. Comp. Sci.*, 1980, *10*, pp. 335-338.
14. R. E. STEARNS, J. HARTMANIS, and P. M. LEWIS II, Hierarchies of memory limited computations, In 1965 *I.E.E.E. Conference Record on Switching Circuit Theory and Logical Design*, 1965, pp. 179-190.
15. R. SZELEPCSÉNYI, The method of forced enumeration for nondeterministic automata, *Acta Informatica*, 1988, *26*, pp. 279-284.
16. A. SZEPIETOWSKI, Some remarks on the alternating hierarchy and closure under complement for sublogarithmic space, *Information Processing Letters*, 1989, *33*, pp. 73-78.
17. S. TODA, Σ_2-SPACE(n) is closed under complement, *J. Comput. Syst. Sciences*, 1987, *35*, pp. 145-152.