JOURNAL OF COMPUTER AND SYSTEM SCIENCES **24**, 362–372 (1982)

Space Complexity in On-Line Computation

HAJIME MACHIDA AND TAKUMI KASAI

Department of Computer Science,
University of Electro-Communications, Chofu-shi, Tokyo 182 Japan

Received August 7, 1981; revised March 10, 1982

A technique is developed for determining space complexity in on-line computation. It is shown that each of the following functions requires linear space: (i) the conversion of binary numbers into ternary numbers, (ii) the multiplication of integers and (iii) the translation of arithmetic expressions in infix notation into Polish notation.

1. INTRODUCTION

In the past few years much research has been carried out in an attempt to obtain nontrivial lower bounds on the computational complexity of some specific problems. In such investigations off-line Turing machines and on-line Turing machines are the machine models most commonly used. In the case of off-line computation, there are presently large gaps between known upper and lower bounds on the requisite time and space for a number of well-known problems. For example, it is not known that any of the recognition problems of context-free languages, the path-finding problem of digraphs, the multiplication of integers, etc., requires more than $\log n$ space (where n is the length of input). For multiplication, $\log n$ space is sufficient [6]. Techniques known to be applicable to obtaining a lower bound in off-line computation are quite limited: diagonalization argument and crossing sequence argument. On the other hand, in the case of on-line computation, some information-theoretic techniques and other methods are applicable and nontrivial low level lower bounds for some specific problems have successfully been obtained. Hartmanis and Shank [4, 5], for example, studied the recognition problem of prime numbers and showed it requires linear space. Lewis *et al.* [7] proved that the recognition problem of a particular context-free language necessitates linear space, while Gallaire [3] showed that for some other context-free language recognition, the problem requires $n^2/(\log n)$ time.

The purpose of this paper is to develop a technique for determining space complexity in on-line computation. We shall deal mainly with (i) the recognition problem of a set of natural numbers, (ii) the multiplication of integers, and (iii) the problem of translating arithmetic expressions in infix notation into Polish notation. The only restriction imposed is that Turing machines used to compute them must be on-line Turing machines.

From the results of (i), it follows that the space complexity of the recognition problem of a set of natural numbers depends on the positional representation (e.g., binary, ternary, decimal). Specifically, we show the existence of a set A of natural numbers with the following properties: If each member of A is represented as a ternary string, i.e., a ternary number, A is a regular set and hence recognizable in $O(1)$ space, while if each member of A is represented as a binary number the recognition problem of A requires linear space, i.e., cn space for some constant c. The immediate implication of this result is that the conversion of binary numbers into ternary numbers requires linear space.

Note that on-line computation with $S(n)$-space may be considered off-line if $S(n) \geqslant n$.

It is also shown that any on-line Turing machine which computes the multiplication of integers requires linear space. It is easy to see that the multiplication can be carried out in $O(n)$ space. Hence, linear space is both necessary and sufficient for on-line integer multiplication. In passing, we note that Fischer and Stockmeyer [2] showed that integer multiplication can be computed within $O(n(\log n)^2 \log\log n)$ time and $O(n)$ space by an on-line Turing machine.

The existence of a language C over $\{0, 1\}$ is shown which has the following properties: C can be accepted by a $\log n$ space bounded nondeterministic on-line Turing machine, but any deterministic on-line Turing machine which accepts C requires linear space. Hence, in the case of on-line Turing machine, nondeterministic machines are more powerful than deterministic machines with respect to space.

Finally we show that the translation of arithmetic expressions in infix notation into Polish notation requires linear space.

2. Preliminaries

In studying computational complexity one must specify what computing model the discussion will be based on. Throughout this paper we choose the model of an on-line Turing machine defined below.

DEFINITION. An *on-line Turing machine* is a Turing machine with a one-way read-only input tape, a one-way write-only output tape and a finite number of semi-infinite two-way storage tapes.

Our investigation is devoted entirely to the study of space complexity in on-line computation. Throughout we shall always assume that a Turing machine is an on-line Turing machine.

DEFINITION. Let Σ_1 and Σ_2 be alphabets. Let f be a function from Σ_1^* to Σ_2^* and S be a function from N to N, where N is the set of nonnegative integers. Then f is *$S(n)$-space computable* if there exists a Turing machine M with Σ_1 and Σ_2 as input and output alphabets, respectively, and for every input $w \in \Sigma_1^*$ of length n, M scans no more than $S(n)$ cells on any storage tape and halts in a finite number of steps with $f(w)$ as its output. Moreover, a subset A of Σ_1^* is *$S(n)$-space recognizable* if the characteristic function c_A of A (i.e., $\Sigma_2 = \{0, 1\}$) is $S(n)$-space computable.

The following are immediate consequences of definitions. A set $A \subseteq \Sigma^*$ is $O(1)$ space recognizable if and only if A is a regular set. Let $A \subseteq \Sigma_1^*$ and $B \subseteq \Sigma_2^*$. Suppose that A is $S_1(n)$ space recognizable and B is $S_2(n)$ space recognizable. Then $A \cup B$ and $A \cap B$ are $(S_1(n) + S_2(n))$-space recognizable.

DEFINITION. Let f and S be as above. Then f is said to *require $S(n)$ space* if for any Turing machine M computing f, there exists a constant $c > 0$ such that M scans more than $cS(n)$ cells for some input of length n for infinitely many n. In other words, f requires $S(n)$ space if and only if f is not $S'(n)$-space computable for any function $S'\colon N \to N$ such that

$$\sup_{n\to\infty} \frac{S'(n)}{S(n)} = 0.$$

Moreover, a set A *requires $S(n)$-space* if its characteristic function c_A requires $S(n)$ space.

The next definition is standard.

DEFINITION. Let Σ_1 and Σ_2 be alphabets. For subsets $A \subseteq \Sigma_1^*$ and $B \subseteq \Sigma_2^*$ and a function $f\colon \Sigma_1^* \to \Sigma_2^*$, A is said to be *reducible* to B with respect to f if for any $w \in \Sigma_1^*$

$$w \in A \qquad \text{iff} \quad f(w) \in B.$$

LEMMA 1. *Let $A \subseteq \Sigma_1^*$ and $B \subseteq \Sigma_2^*$. Suppose that A is reducible to B with respect to f, and $f\colon \Sigma_1^* \to \Sigma_2^*$ is $S_1(n)$-space computable and B is $S_2(n)$-space recognizable. If for some constant c, $|f(w)| \leqslant c\,|w|$, then A is $(S_1(n) + S_2(cn))$-space recognizable.*

Let Γ be an alphabet, and A be a subset of Γ^*. Let the equivalence relation $\approx$ over Γ^* be defined by: $x \approx y$ if and only if, for all z in Γ^*, xz is in A exactly when yz is in A. For each x in Γ^*, let $E(x)$ be the equivalence class of x. Let S_A be the function of N into N defined by

$$S_A(n) = |\{E(x) | \, x \in \Gamma^*, |x| \leqslant n\}|,$$

where, for a set X, $|X|$ denotes the cardinality of X.

LEMMA 2. *Let A be a subset of Γ^*. Then A requires* $\log S_A(n)$ *space.*

Proof. Suppose A is recognizable by a Turing machine M of space complexity $S(n)$ for some $S\colon N \to N$ such that

$$\sup_{n\to\infty} \frac{S(n)}{\log S_A(n)} = 0.$$

Let M have s states and t storage tapes. Let r be the cardinality of the storage tape alphabet. Since M is $S(n)$-space bounded, for any input word of length n there are at most

$$s(S(n)\, r^{S(n)})^t$$

configurations. (By configuration we mean a $(2t+1)$-tuple consisting of a state of the finite control and the head position and content of each storage tape.) Then

$$s(S(n)\, r^{S(n)})^t < S_A(n) \tag{2.1}$$

for sufficiently large n.

For each n satisfying inequality (2.1) we can choose a distinct pair of words u, u' such that $u \not\approx u'$, $|u| \leqslant n$, $|u'| \leqslant n$, and when u is supplied to M as an input, M finishes its computation in the same configuration as it does when u' is given as its input. This contradicts the fact $u \not\approx u'$, however, and hence A requires S_A space.

COROLLARY 1. (i) *The set* $A = \{w \# w^R \# \mid w \in \{0, 1\}^*\}$ *requires linear space.* (ii) *The set* $B = \{0^n 1^n \mid n \in N\}$ *requires log space.*

Proof. (i) $S_A(n) \geqslant 2^n$, since for any distinct pair of words u, v in $\{0, 1\}^*$, $u \not\approx v$. (ii) $S_B(n) \geqslant n$, since $0^i \not\approx 0^j$ if $i \neq j$.

Remark. The proof of (i) is also found in Lewis *et al.* [7].

3. Recognition of 3^k in the Binary Notation

Throughout this paper we denote by Σ the set $\{0, 1\}$. For a string $x = a_1 a_2 \cdots a_n$ of Σ^*, $[x]$ denotes the number corresponding to the binary string x with the rightmost letter as the most significant bit, i.e.,

$$[x] = \sum_{i=1}^{n} a_i \cdot 2^{i-1}.$$

This is the reverse of the standard representation of numbers, and we consider numbers only in this reverse representation. It is not clear if the theorems proved in this section hold also for the standard representation.

Let

$$T = \{w \in \Sigma^* 1 \mid [w] = 3^k, k \in N\}^1.$$

In other words, the set T consists of strings each of which represents a power of 3 in the binary notation.

The purpose of this section is to prove the following.

[1] $\Sigma^* 1$ denotes the set of words over Σ ending in 1.

THEOREM 1. *The set T requires linear space.*

On the other hand, a straightforward algorithm can be designed using no more than linear space. Hence linear space is both necessary and sufficient for recognizing the set T.

In order to prove this theorem, we need to develop preliminary results on what may be called initial segments of T, which will play an essential role in the proof of the theorem. For every $n \in N$ and $w \in \Sigma^*$ define

$$\begin{aligned} w^{(n)} &= w0^{n-m}, && \text{if } \; |w| = m \leqslant n, \\ &= u, && \text{if } \; w = uv \; \text{ and } \; |u| = n \quad (u, v \in \Sigma^*). \end{aligned}$$

Thus, when interpreted as a binary number $w^{(n)}$ satisfied the following conditions:

(i) $|w^{(n)}| = n$, and

(ii) $[w^{(n)}] \equiv [w] \pmod{2^n}$.

For each $k \in N$ let $w_k \in \Sigma^*1$ be the word such that $[w_k] = 3^k$. Hence $T = \{w_k | k \in N\}$. Furthermore, let $T^{(n)}$, $n \in N$, denote the set $\{w^{(n)} | w \in T\}$, which equals to $\{w_k^{(n)} | k \in N\}$. $T^{(n)}$ is the collection of initial segments of T with length n. Since

$$[w_{k+1}^{(n)}] \equiv 3[w_k^{(n)}] \qquad \pmod{2^n},$$

it follows that the sequence

$$w_0^{(n)}, w_1^{(n)}, \ldots, w_k^{(n)}, \ldots$$

is a periodic sequence for each $n > 0$. Denote by $\pi(n)$, $n > 0$, the period of this sequence. Thus for any $k \in N$

$$w_{k+\pi(n)}^{(n)} = w_k^{(n)}.$$

It follows by inspection that

$$\pi(1) = 1, \qquad \pi(2) = 2, \qquad \pi(3) = 2, \qquad \pi(4) = 4, \ldots,$$

and in general we have

LEMMA 3. $\pi(n) = 2^{n-2}$ *for every* $n \geqslant 3$.

Proof. Clearly, $\pi(n)$ is the minimal integer $m > 0$ such that $3^m \equiv 1 \pmod{2^n}$. Then it is easy to prove by induction that $m = 2^{n-2}$, i.e.,

(a) $3^{2^{n-2}} = (2k+1)\,2^n + 1$ for some k, and

(b) for $1 \leqslant s < 2^{n-2}$, $3^s \not\equiv 1 \pmod{2^n}$.

Corollary 2 is immediate.

COROLLARY 2. *$|T^{(n)}| = 2^{n-2}$ for every $n \geqslant 3$.*

Proof of Theorem 1. Since T contains at most one word of length n for every $n > 0$, for two distinct words u and v in $T^{(n)}$, $u \not\approx v$. Hence, $S_T(n) \geqslant 2^{n-2}$, $n \geqslant 3$, by Corollary 2. Therefore, by Lemma 2, T requires linear space.

Next we show that the conversion of binary numbers to ternary numbers requires linear space. Let $\Delta = \{0, 1, 2\}$. For each w in Δ^*, $[w]_3$ denotes the ternary number represented by w in a usual manner. (The most significant bit is on the right.) Let $\Phi: \Delta^*\{1, 2\} \to \Sigma^*1$ be the function defined by

$$\Phi(x) = y \qquad \text{iff} \quad [x]_3 = [y].$$

THEOREM 2. *The function Φ requires linear space.*

Proof. Note that the set

$$B = \{w \in \Delta^*\{1, 2\} \mid [w]_3 = 3^k, k \in N\} = 0^*1$$

is regular. Clearly, the set A is reducible to B with respect to Φ. Therefore, by Lemma 1 and Theorem 1, Φ requires linear space.

4. ON-LINE INTEGER MULTIPLICATION

The purpose of this and the following sections is to show that on-line integer multiplication requires linear space.

Let $\Sigma = \{0, 1\}$ and $\Sigma_1 = \Sigma \cup \{\varepsilon\}$, where ε is not in Σ. Shuffle product $\#$ is defined for any $x = a_1 a_2 \cdots a_m$ and $y = b_1 b_2 \cdots b_n$ of Σ^* such that $x \# y$ is $a_1 b_1 a_2 b_2 \cdots a_l b_l$, where l is $\max\{m, n\}$, and where $a_{m+1} = \cdots = a_l = \varepsilon$ if $m < n$ and $b_{n+1} = \cdots = b_l = \varepsilon$ if $n < m$. For a positive integer n, $]n[$ denotes the binary string in Σ^*1 expressing the value n when read as a binary number with the rightmost letter as the most significant bit. (Let $]0[$ correspond to 0.) Hence $[(]n[)] = n$ for every $n \in N$. The multiplication function MULT: $\Sigma_1^* \to \Sigma^*$ is defined below only for strings of the form $x \# y$ for some x and y in Σ^*.

$$\begin{aligned} \text{MULT}(w) &=]([x] * [y])[, \qquad \text{if} \quad w = x \# y \quad \text{for} \quad x \quad \text{and} \quad y \quad \text{in} \quad \Sigma^*, \\ &= \text{undefined}, \qquad \text{otherwise}, \end{aligned}$$

where $*$ is usual integer multiplication. Thus the function MULT can quite naturally be understood to read in two numbers x and y from lower order bits and to output their product $x * y$ also from lower order bits.

Now we shall relate space complexity of this function MULT to that of a squaring function SQR. The function SQR is a function from Σ^* to Σ^* such as

$$\text{SQR}(w) =]([w]^2)[$$

for very string w in Σ^*. This means that given w as an input, SQR outputs a string w' whose value is the square of the value of w, i.e., $[w'] = [w]^2$.

LEMMA 4. *Let $S: N \to N$ be a function.* MULT *is $S(n)$-space computable if and only if* SQR *is $S(n)$-space computable.*

Proof. This follows from the fact that

$$a * b = \{(a+b)^2 - (a-b)^2\}/4$$

for integers a and b.

THEOREM 3. SQR *and* MULT *require linear space.*

It should be noted that this is not a particular phenomenon due to our encoding scheme of two operands x and y, or the definintion of the multiplication function, but that this result of integer multiplication requiring linear space would hold also for any other *natural* definition of integer multiplication. For example, when the numbers $x = a_1 a_2 \cdots a_n$ and $y = b_1 b_2 \cdots b_n$ are given as

$$\begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \begin{pmatrix} a_2 \\ b_2 \end{pmatrix} \cdots \begin{pmatrix} a_n \\ b_n \end{pmatrix}$$

in $(\Sigma \times \Sigma)^*$, the result still holds because the transformation from the previous encoding to this encoding is computable in $O(1)$ space.

Another remark is that MULT is clearly computable in linear space. Therefore, linear space is the exact space complexity of on-line integer multiplication.

The remainder of this section is devoted to the proof of Theorem 3. By Lemma 4, it is sufficient to consider SQR.

Let A be a subset of Σ^* defined as follows: A word w belongs to A if and only if w is of the form

$$1^{e_1} 0^{f_1} 1^{e_2} \cdots 0^{f_t} 1^{e_{t+1}}$$

where $e_1, \ldots, e_{t+1}, f_1, \ldots, f_t > 0$ $(t \geqslant 3)$ with the following properties: Among $f_1, f_2, \ldots, f_t$ the third largest is unique. Let it be f_s $(1 \leqslant s \leqslant t)$. Then the (f_{s+1})th bit from the leftmost bit of the substring $1^{e_{s+2}} 0^{f_{s+2}} \cdots 0^{f_t} 1^{e_{t+1}}$ does exist and is equal to 1.

LEMMA 5. *A is recognizable in log space.*

Proof. At every intermediate stage in processing an input w, use three storage tapes for special purposes to hold records of lengths of the longest, the second longest, and the third longest sequences of 0's so far encountered and of the contents of the corresponding (f_{s+1})th bit. Each time a new sequence of 0's is read in whose length exceeds the previous third longest length, update the records on those three special storage tapes. When w is completely scanned, the corresponding bit of the third longest sequence of 0's is examined and w is accepted if and only if it is 1.

Let B be the subset of Σ^* defined by

$$B = \{w \in 1\Sigma^*1 \mid \mathrm{SQR}(w) \in A\}.$$

We contend that the recognition of the set B requires linear space.

LEMMA 6. *B requires linear space.*

Proof. Let $B^{(n)}$, $n \in N$, denote the set $\{u \mid u \in 1\Sigma^*1,\ |u| = n$ and $uv \in B$ for some $v \in \Sigma^*\}$. First we show that

$$|B^{(n)}| = 2^{n-2} \qquad \text{for every} \quad n > 1. \tag{4.1}$$

Let u be a string of length n in $1\Sigma^*1$. Consider the square of $[u0^j10^i1]$ for $j \geqslant n-1$ and $i \geqslant j+n-1$. Since

$$\begin{aligned}[u0^j10^i1]^2 &= (2^{i+1+j+n} + 2^{j+n} + [u])^2 \\ &= 2^{2i+2j+2n+2} + 2^{i+2j+2n+2} + 2^{i+j+n+2}[u] + 2^{2j+2n} + 2^{j+n+1}[u] + [u]^2,\end{aligned}$$

the string $\mathrm{SQR}(u0^j10^i1)$ is as shown in Fig. 1. It is easily verified by definition of the set A that, unless $u = 11$ or 101, $\mathrm{SQR}(u0^{2n}10^{3n}1)$ belongs to A and thus $u0^{2n}10^{3n}1$ is in B. Two exceptional cases are similarly handled as $\mathrm{SQR}(11010^61)$ and $\mathrm{SQR}(101010^61)$ are in A. Therefore, (4.1) holds.

Now we show that

$$\text{for any distinct pair } u_1 \text{ and } u_2 \text{ of } B^{(n)},\ u_1 \not\approx u_2. \tag{4.2}$$

(The equivalence relation $\approx$ is defined with respect to B.)

To prove this, we again consult Fig. 1. By assumption u_1 and u_2 are distinct. Suppose the mth bit of u_1 and that of u_2 is different ($1 \leqslant m \leqslant n$). For $n > 3$, let $j = 2n$ and $i = 3n + m - 1$. Then it is clear from Fig. 1 that only one of $u_10^j10^i1$ or $u_20^j10^i1$ belongs to B and the other does not. Thus $u_1 \not\approx u_2$. For $n = 3$, if we let $j = 2$ and $i = 8$ we see that 1110^210^81 is in A, whereas 1010^210^81 is not, and $101 \not\approx 111$. Therefore, (4.2) is proved.

Recall the definition of $S_B(n)$, $n \in N$, as follows:

$$S_B(n) = |\{E(x) \mid x \in \Sigma^*, |x| \geqslant n\}|,$$

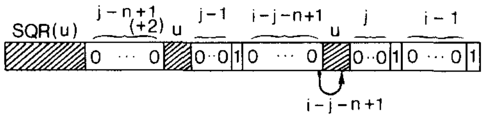

FIGURE 1

where $E(x)$ is the equivalence class of x with respect to $\approx$. It follows from (4.1) and (4.2) that

$$S_B(n) \geqslant 2^{n-2} \qquad (n > 1).$$

Therefore, by Lemma 2, B requires linear space.

5. Some Properties of On-Line Computation

In this section two results are shown, one concerning reversal operation and another concerning nondeterminism.

Let ON-LINE DLOG (resp. NLOG) SPACE denote the class of languages recognizable by on-line deterministic (resp. nondeterministic) Turing machine of $O(\log n)$ space complexity.

Lemma 7. *Let $C \subseteq \Sigma^*$ be the set defined by*

$$C = \{w \mid w = u10^i1, |u| \geqslant i \textit{ and } i\text{th } \textit{bit of } u$$
$$\textit{from its right end is } 1\}$$

Then, (i) *C requires linear space,* (ii) *C^R(= reverse of C) is in* ON-LINE DLOG SPACE, *and* (iii) *C is in* ON-LINE NLOG SPACE.

Proof. It should be clear that the set C is log-space recognizable in nondeterministic manner, and the reversed set C^R is also log-space recognizable (deterministically). In the essentially same line of argument as in the proof of Corollary 1(i) it can be shown that C requires linear space.

The following results are immediate consequences of Lemma 7:

Corollary 3. *The class of languages recognizable on-line in log space is not closed under reversal.*

Corollary 4. ON-LINE DLOG SPACE$\subsetneq$ ON-LINE NLOG SPACE.

6. Object Code Generator

In this section we consider a typical problem on compiler construction—the problem of translating arithmetic expressions in infix notation into Polish, or postfix, notation, and show that this problem requires linear space.

Definition. Let $\mathscr{E}$ and $\mathscr{O}$ be alphabets such that $\mathscr{E} \cap \mathscr{O} = \phi$. Let $\Omega = \mathscr{E} \cup \mathscr{O}$ and $\Delta = \Omega \cup \{(\ ,\ )\}$. The set AEXP of *arithmetic expressions* over Ω is the subset of Δ^* defined as follows:

(i) $\mathscr{E} \subseteq$ AEXP,

(ii) if x, $y \in$ AEXP and $\sigma \in \mathscr{O}$, then $(x\sigma y) \in$ AEXP. The set POSTEXP of *postfix expressions* over Ω is the subset of Ω^* defined as follows:

(i) $\mathscr{E} \subseteq$ POSTEXP

(ii) if $x, y \in$ POSTEXP and $\sigma \in \mathscr{O}$, then $xy\sigma \in$ POSTEXP. An *object code generator* is a function $\Psi: \Delta^* \to \Omega^*$ which satisfies the following conditions:

(i) $\Psi(a) = a$ if $a \in \mathscr{E}$,

(ii) $\Psi((x\sigma y)) = \Psi(x)\, \Psi(y)\, \sigma$, if $x, y \in$ AEXP and $\sigma \in \mathscr{O}$,

(iii) if $x \notin$ AEXP, then $\Psi(x) \notin$ POSTEXP.

We shall show that Ψ requires linear space.

THEOREM 4. *Log space is necessary and sufficient for recognizing the set* POSTEXP.

Proof. It should be clear that the set POSTEXP is $\log(n)$-space computable. Let $0 \in \mathscr{E}$ and $1 \in \mathscr{O}$. Then

$$\text{POSTEXP} \cap 0^*1^* = \{0^{n+1}1^n \mid n \in N\}.$$

Since 0^*1^* is regular, it suffices to show that $\{0^{n+1}1^n \mid n \in N\}$ requires $\log(n)$ space. By an argument similar to Corollary 1(ii), $\{0^{n+1}1^n \mid n \in N\}$ requires $\log(n)$ space.

THEOREM 5. *Linear space is necessary and sufficient for recognizing the set* AEXP.

Proof. Clearly, AEXP is $O(n)$ space computable. Let a be an element of $\mathscr{E}$ and $+$ be an element of $\mathscr{O}$. Let

$$h_1: \{0, 1\}^* \to \{a, +, (\,,\,)\}^* \qquad \text{and} \qquad h_2: \{0, 1\}^* \to \{a, +, (\,,\,)\}^*$$

be homomorphisms defined by

$$h_1(0) = (a+, \qquad h_1(1) = (\,, \qquad h_2(0) =), \qquad h_2(1) = +a).$$

Let f: $\{0, 1\}^* \# \{0, 1\}^* \# \to \Delta^*$ be the function defined by

$$f(u \# v \#) = h_1(u)\, a h_2(v), \qquad u, v \in \{0, 1\}^*.$$

For example,

$$f(01 \# 10\#) = h_1(0)\, h_1(1)\, a h_2(1)\, h_2(0) = (a + (a + a)).$$

Let A be the set in Corollary 1(i). Then, for each element x of $\{0, 1\}^* \# \{0, 1\}^* \#$,

$$x \in A \qquad \text{iff} \quad f(x) \in \text{AEXP}.$$

The function f can be extended naturally to a function f: $\{0, 1, \#\}^* \rightarrow \{a, +, (\, , \,)\}^*$ such that

(i) for all $x \in \{0, 1, \#\}^*$,

$$x \in A \qquad \text{iff} \quad f(x) \in \text{AEXP},$$

(ii) f is $O(1)$ space computable.

Then, by Lemma 1 and Corollary 1(i), AEXP requires linear space.

From Theorems 4 and 5 and Lemma 1, we finally have

THEOREM 6. *The object code generator Ψ requires linear space.*

ACKNOWLEDGMENT

The authors would like to express their gratitude to a referee who simplified some of the proofs and helped to make the presentation clearer.

REFERENCES

1. A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass., 1974.
2. M. J. FISCHER AND L. J. STOCKMEYER, Fast on-line integer multiplication, *J. Comput. System Sci.* **9** (1974), 317–331.
3. H. GALLAIRE, Recognition time of context-free languages by on-line Turing machines, *Inform. and Control* **15** (1969), 288–295.
4. J. HARTMANIS AND H. SHANK, On the recognition of primes by automata, *J. Assoc. Comput. Mach.* **15** (1968), 382–389.
5. J. HARTMANIS AND H. SHANK, Two memory bounds for the recognition of primes by automata, *Math. Systems Theory* **3** (1969), 125–129.
6. J. JA'JA' AND J. SIMON, Some space-efficient algorithms, *in* "Proceedings, Seventeenth Annual Allerton Conference on Communication, Control, and Computing," 1979.
7. P. M. LEWIS II, R. E. STEARNS, AND J. HARTMANIS, Memory bounds for recognition of context-free and context-sensitive languages, *in* "IEEE Conference Record of Sixth Annual Symposium on Switching Circuit Thory and Logic Design," pp. 191–202, 1965.
8. M. S. PATERSON, M. J. FISCHER, AND A. R. MEYER, An improved overlap argument for on-line multiplication, *in* "Complexity of Computation (SIAM-AMS Proceedings, Vol. 7)," American Mathematical Society, Providence, R. I., pp. 97–111, 1974.
9. M. P. SCHUTZENBERGER, A remark on acceptable sets of numbers, *J. Assoc. Comput. Mach.* **15** (1968), 300–303.