Theoretical Computer Science 41 (1985) 331–339
North-Holland

NOTE

ALTERNATING ON-LINE TURING MACHINES WITH ONLY UNIVERSAL STATES AND SMALL SPACE BOUNDS

Katsushi INOUE

Department of Electronics, Faculty of Engineering, Yamaguchi University, Ube, 755 Japan, and Lehrstuhl C für Informatik, Institut für Theoretische und Praktische Informatik, Technische Universität Braunschweig, 3300 Braunschweig, Fed. Rep. Germany

Itsuo TAKANAMI

Department of Electronics, Faculty of Engineering, Yamaguchi University, Ube, 755 Japan

Roland VOLLMAR

Lehrstuhl C für Informatik, Institut für Theoretische und Praktische Informatik, Technische Universität Braunschweig, 3300 Braunschweig, Fed. Rep. Germany

Communicated by M.A. Harrison
Received August 1983
Revised May 1984

Abstract. Let $\mathscr{L}[\mathrm{AONTM}(L(n))]$ be the class of sets accepted by $L(n)$ space bounded alternating on-line Turing machines, and $\mathscr{L}[\mathrm{UONTM}(L(n))]$ be the class of sets accepted by $L(n)$ space bounded alternating on-line Turing machines with only universal states. This note first shows that, for any $L(n)$ such that $L(n) \geq \log\log n$ and $\lim_{n\to\infty}[L(n)/\log n]=0$, (i) $\mathscr{L}[\mathrm{UONTM}(L(n))] \subsetneq \mathscr{L}[\mathrm{AONTM}(L(n))]$, (ii) $\mathscr{L}[\mathrm{UONTM}(L(n))]$ is not closed under complementation, and (iii) $\mathscr{L}[\mathrm{UONTM}(L(n))]$ is properly contained in the class of sets accepted by $L(n)$ space bounded alternating Turing machines with only universal states. We then show that there exists an infinite hierarchy among $\mathscr{L}[\mathrm{UONTM}(L(n))]$'s with $\log\log n \leq L(n) \leq \log n$.

1. Introduction

Alternating Turing machines were introduced and investigated in [1, 2, 5, 6, 8–14] as a mechanism to model parallel computation. Recently [4, 7], several properties of alternating Turing machines with only universal states have been given. This note continues to investigate some properties of alternating on-line Turing machines with only universal states and with small space bounds. Let $\mathscr{L}[\mathrm{AONTM}(L(n))]$ be the class of sets accepted by $L(n)$ space bounded alternating on-line Turing machines, and $\mathscr{L}[\mathrm{UONTM}(L(n))]$ be the class of sets accepted by $L(n)$ space bounded alternating on-line Turing machines with only universal states. It is shown [4] that for any $L(n)$ such that $L(n) \geq \log n$ and $\lim_{n\to\infty}[L(n)/n]=0$, (i)

$\mathscr{L}[\mathrm{UONTM}(L(n))] \subsetneq \mathscr{L}[\mathrm{AONTM}(L(n))]$, and (ii) $\mathscr{L}[\mathrm{UONTM}(L(n))]$ is not closed under complementation. Section 3 of this note shows that, for any $L(n)$ such that $L(n) \geq \log\log n$ and $\lim_{n\to\infty}[L(n)/\log n] = 0$, (i) $\mathscr{L}[\mathrm{UONTM}(L(n))] \subsetneq \mathscr{L}[\mathrm{AONTM}(L(n))]$, (ii) $\mathscr{L}[\mathrm{UONTM}(L(n))]$ is not closed under complementation, and (iii) $\mathscr{L}[\mathrm{UONTM}(L(n))]$ is properly contained in the class of sets accepted by $L(n)$ space bounded alternating Turing machines with only universal states. Section 3 also shows that there exists an infinite hierarchy among the $\mathscr{L}[\mathrm{UONTM}(L(n))]$'s with $\log\log n \leq L(n) \leq \log n$.

2. Preliminaries

To make this paper self-contained, we first give full definitions of alternating Turing machines.

Definition 2.1. An *alternating Turing machine* (ATM) is a seven-tuple $M = (Q, U, \Gamma, \Sigma, \delta, q_0, F)$, where (1) Q is a finite set of *states*, (2) $U \subseteq Q$ is the set of *universal* states, (3) Γ is a finite *storage tape alphabet* ($B \in \Gamma$ is the *blank symbol*), (4) Σ is a finite *input alphabet* ($\mathrm{¢} \notin \Sigma$ is the left endmarker, and $\$ \notin \Sigma$ is the right endmarker), (5) $\delta \subseteq (Q \times (\Sigma \cup \{\mathrm{¢}, \$\}) \times \Gamma) \times (Q \times (\Gamma - \{B\}) \times \{\text{left, no move, right}\}^2)$ is the *next move relation*, (6) $q_0 \in Q$ is the *initial state*, and (7) $F \subseteq Q$ is the set of *accepting* states. A state q in $Q - U$ is said to be *existential*.

The ATM M has a read-only input tape with the left and right end-markers ¢ and \$, and one semi-infinite storage tape, initially blank. A *step* of M consists of reading one symbol from each tape, writing a symbol on the storage tape, moving the input and storage heads in specified directions, and entering a new state, in accordance with the next move relation δ. Note that the machine cannot write the blank symbol.

Definition 2.2. An *instantaneous description* (ID) of an ATM $M = (Q, U, \Gamma, \Sigma, \delta, q_0, F)$ is an element of

$$\Sigma^* \times (N \cup \{0\}) \times S_M,$$

where $S_M = Q \times (\Gamma - \{B\})^* \times N$, and N denotes the set of all positive integers. The first and second components, x and i, of an ID $I = (x, i, (q, \alpha, j))$ represent[1] the input (excluding the left and right endmarkers ¢ and \$) and the input head position, respectively. The third component (q, α, j) $(\in S_M)$ of I represents the state of the finite control, the nonblank contents of the storage tape, and the storage head position. An element of S_M is called a *storage state* of M. If q is the state associated with an ID I, then I is said to be a *universal* (*existential*, *accepting*) ID if q is a

[1] We note that $0 \leq i \leq |x|+1$, and $1 \leq j \leq |\alpha|+1$, where for any string w, $|w|$ denotes the length of w (with $|\lambda| = 0$).

universal (existential, accepting) state. The *initial* ID of M on x is $I_M(x) = (x, 0, (q_0, \lambda, 1))$, where λ is the null word.

Definition 2.3. Given $M = (Q, U, \Gamma, \Sigma, \delta, q_0, F)$, we write $I \vdash I'$ and say I' is a *successor* of I if an ID I' follows from an ID I in one step, according to the transition rules δ. The reflexive transitive closure of $\vdash$ is denoted $\vdash^*$. A *computation path* of M on input x is a sequence $I_0 \vdash I_1 \vdash \cdots \vdash I_n$ $(n \geq 0)$, where $I_0 = I_M(x)$. A *computation tree* of M is a finite, nonempty labeled tree with the following properties:

(1) each node π of the tree is labeled with an ID, $l(\pi)$,

(2) if π is an internal node (a non-leaf) of the tree, $l(\pi)$ is universal and $\{I \mid l(\pi) \vdash I\} = \{I_1, \ldots, I_k\}$, then π has exactly k children $\rho_1, \ldots, \rho_k$ such that $l(\rho_i) = I_i$,

(3) if π is an internal node of the tree and $l(\pi)$ is existential, then π has exactly one child ρ such that $l(\pi) \vdash l(\rho)$.

A *computation tree* of M on input x is a computation tree of M whose root is labeled with $I_M(x)$. An *accepting computation tree* of M on x is a computation tree of M on x whose leaves are all labeled with accepting ID's. We say that M *accepts* x if there is an accepting computation tree of M on x. Define $T(M) = \{x \in \Sigma^* \mid M$ accepts $x\}$.

An *alternating on-line Turing machine* (AONTM) [6] is an ATM whose input head cannot move to the left. In this note, we are interested in an ATM (AONTM) with only universal states, i.e., with no existential state. We denote such an ATM (AONTM) by UTM (UONTM).

With each ATM (AONTM, UTM, or UONTM) M, we associate a *space complexity function* SPACE which takes ID's to natural numbers. That is, for each ID $I = (x, i, (q, \alpha, j))$, let SPACE$(I)$ be the length of α. Let $L: N \to R$ be a function, where R denotes the set of all nonnegative real numbers. We say that M is $L(n)$ *space bounded* if, for each n and for each input x of length n, if x is accepted by M, then there is an accepting computation tree of M on x such that for each node π of the tree, SPACE$(l(\pi)) \leq \lceil L(n) \rceil$.[2] By ATM$(L(n))$ (UTM$(L(m))$, AONTM$(L, (n))$, UONTM$(L(n))$) we denote an $L(n)$ space bounded ATM (UTM, AONTM, UONTM). For each $X \in \{A, U, AON, UON\}$, define

$$\mathcal{L}[\text{XTM}(L(n))] = \{T \mid T = T(M) \text{ for some XTM}(L(n))\ M\}.$$

3. Results

In this section we first show that for any $L(n)$ such that $L(n) \geq \log \log n$ and $\lim_{n \to \infty} [L(n)/\log n] = 0$, (i) $\mathcal{L}[\text{UONTM}(L(n))] \subsetneq \mathcal{L}[\text{AONTM}(L(n))]$, (ii) $\mathcal{L}[\text{UONTM}(L(n))] \subsetneq \mathcal{L}[\text{UTM}(L(n))]$, and (iii) $\mathcal{L}[\text{UONTM}(L(n))]$ is not closed under complementation.

The following is the key lemma.

[2] $\lceil r \rceil$ means the smallest integer greater than or equal to r.

Lemma 3.1. *Let* $L_1 = \{\text{bin}(1)\#\text{bin}(2)\#\cdots\#\text{bin}(n)2wcw' \in \{0, 1, 2, c, \#\}^+ \mid n \geq 2 \,\&\, (w, w' \in \{0, 1\}^+) \,\&\, |w| = |w'| = \lceil \log n \rceil \,\&\, w \neq w'\}$,[3] *where, for each positive integer* $i \geq 1$, $\text{bin}(i)$ *denotes the string in* $\{0, 1\}^+$ *that represents the integer i in binary notation (with no leading zeros). Then:*

(i) $L_1 \in \mathcal{L}[\text{AONTM}(\log\log n)]$,

(ii) $L_1 \in \mathcal{L}[\text{UTM}(\log\log n)]$, *and*

(iii) $L_1 \notin \mathcal{L}[\text{UONTM}(L(n))]$ *for any function* $L: N \to R$ *such that* $\lim_{n\to\infty}[L(n)/\log n] = 0$.

Proof. (i) The set L_1 will be accepted by an AOTNM($\log\log n$) M_1 which acts as follows. Suppose that an input string

$$\text{¢}y_1\#y_2\#\cdots\#y_n2wcw'\$,$$

where $n \geq 2$, and the y_i's, w, and w' are all in $\{0, 1\}^+$, is presented to M_1. (Input strings in different form from the above can easily be rejected by M_1.) In the first phase, M_1 marks off $\log\log n$ storage-tape cells when $y_i = \text{bin}(i)$ for each $1 \leq i \leq n$. This can be done using the successive values bin(1), bin(2), ..., bin(n) that are supposed to occur in the input. That is, M_1 will check that these values actually do occur (i.e., $y_i = \text{bin}(i)$, $1 \leq i \leq n$) and in doing so it will construct the storage-tape space $\log\log n$. By using universal branches only, M_1 can check in a way described below that these values are actually found in successive blocks of the input. M_1 compares the value y_i in the ith block with the value y_{i+1} in the $(i+1)$st block and verifies that y_{i+1} represents in binary notation a number which is one more than that represented by y_i. In doing so, M_1 will compare the jth symbol of these two values, for all appropriate j. Observe that when y_{i+1} is one more than y_i, then $y_{i+1} = x10^m$ and $y_i = x01^m$, where x is a string (starting with 1) over $\{0, 1\}$ and m is some positive integer, or $y_{i+1} = 10^m$ and $y_i = 1^m$, again with m some positive integer. M_1 starts by writing the binary representation of one in its storage-tape space and performing a universal branch. In one branch it compares the first symbol of the two strings. In the other branch it increases the value in the storage-tape space by one and goes on to compare the remaining symbols. In general, let us suppose that M_1 has the value j written in binary notation in its storage-tape space. It performs a universal branch. In one branch it compares the jth symbol of the two strings. (It determines whether they should be identical or not by scanning the remaining symbols of the first of the two strings. If the remaining symbols are all 1, then the symbols should be opposite; otherwise, they should be the same.) In the other branch, it adds one to the storage-tape value and continues to compare the remaining symbols. In this way, M_1 can check the successive values in the input. Also, in comparing the last two strings y_{n-1} and y_n it will create a string in its storage-tape of length $\log\log n$, since the length of the last string y_n is $\log n$ (if $y_n = \text{bin}(n)$) and M_1 needs only remember a position in this string in binary notation. If M_1 successfully completes this first phase, then it checks that $|w| = |w'| = \lceil \log n \rceil$ and $w \neq w'$.

[3] From here on, logarithms are base 2.

The check of "$|w| = |w'| = \lceil \log n \rceil$" can deterministically be done using the storage-tape space $\log \log n$ constructed above. The check of "$w \neq w'$" can easily be done by existentially choosing some i $(1 \leq i \leq \lceil \log n \rceil)$ and then checking that $w(i) \neq w'(i)$.[4] This check is also done using the $\log \log n$ storage-tape cells constructed above.

(ii) L_1 is accepted by a UTM($\log \log n$) M_2 which acts as follows. Suppose that an input string described in the proof of (i) is presented to M_2. By using the technique in the proof of (i), M_2 first marks off $\log \log n$ storage-tape cells when $y_i = \mathrm{bin}(i)$ for each $1 \leq i \leq n$. If, in the first phase, M_2 successfully marks off $\log \log n$ storage-tape cells, then M_2 deterministically checks that $|w| = |w'| = \lceil \log n \rceil$ and $w \neq w'$. The check of "$w \neq w'$" can be done by sending its input head back and forth to compare the corresponding symbols in w and w'. Of course, the comparison is done using $\log \log n$ storage-tape space constructed above.

(iii) Suppose that there exists a UONTM($L(n)$) M accepting L_1, where $\lim_{n \to \infty} [L(n)/\log n] = 0$. Let s and k be the numbers of states (of the finite control) and storage-tape symbols of M, respectively. For each $n \geq 2$, let

$$V(n) = \{\mathrm{bin}(1) \# \mathrm{bin}(2) \# \cdots \# \mathrm{bin}(n) 2wcw \mid w \in \{0, 1\}^+ \ \& \ |w| = \lceil \log n \rceil\}.$$

For each $x = \mathrm{bin}(1) \# \mathrm{bin}(2) \# \cdots \# \mathrm{bin}(n) 2wcw$ in $V(n)$, let $S(x)$ and $C(x)$ be sets of storage states of M defined as follows:

$S(x) = \{(q, \alpha, j) \mid$ there exists a computation path $I_M(x) \vdash^* (x, r(n), (q', \alpha', j')) \vdash (x, r(n)+1, (q, \alpha, j))$ of M on x (that is, $(x, r(n)+1, (q, \alpha, j))$ is an ID of M just after the point where the input head left the symbol "c" of $x)\}$,

where $r(n) = |\mathrm{bin}(1) \# \mathrm{bin}(2) \# \cdots \# \mathrm{bin}(n) 2| + \lceil \log n \rceil + 1$;

$C(x) = \{\sigma \in S(x) \mid$ when, starting with the ID $(x, r(n)+1, \sigma)$, M proceeds to read the last segment $w\$$ or $\mathrm{¢}x\$$, there exists a sequence of steps of M in which M never enters an accepting state$\}$.

(Note that, for each x in $V(n)$, $C(x)$ is not empty, since x is not in L_1, and so not accepted by M.) Then the following proposition must hold.

Proposition 3.2. *For any two different strings x, y in $V(n)$, $C(x) \cap C(y) = \emptyset$.*

Proof. For otherwise, suppose that $x = \mathrm{bin}(1) \# \mathrm{bin}(2) \# \cdots \# \mathrm{bin}(n) 2wcw$, $y = \mathrm{bin}(1) \# \mathrm{bin}(2) \# \cdots \# \mathrm{bin}(n) 2w'cw'$, $w \neq w'$, $C(x) \cap C(y) \neq \emptyset$, and $\sigma \in C(x) \cap C(y)$. Let $z = \mathrm{bin}(1) \# \mathrm{bin}(2) \# \cdots \# \mathrm{bin}(n) 2wcw'$. Since $\sigma \in C(x)$, there is a computation path $I_M(z) \vdash^* (z, r(n)+1, \sigma)$. When, starting with the ID $(z, r(n)+1, \sigma)$, M proceeds to read the last segment $w'\$$ of $\mathrm{¢}z\$$, there exists a sequence of steps of M in which M never enters an accepting state since $\sigma \in C(y)$. This means that z is not accepted by M. This contradicts the fact that z is in $L_1 = T(M)$. □

[4] For each string w and each integer i $(1 \leq i \leq |w|)$, $w(i)$ denotes the ith symbol (from the left) of w.

Proof of Lemma 3.1 (*continued*). Clearly, $|V(n)| = 2^{\lceil \log n \rceil}$,[5] and $p(n) \leq sL(r(n)+\lceil \log n \rceil)k^{L(r(n)+\lceil \log n \rceil)}$, where $p(n)$ denotes the number of possible storage states of M just after the point where the input head left the symbol "c" of strings in $V(n)$. Since $\lim_{n\to\infty}[L(n)/\log n] = 0$, it follows that

$$\lim_{n\to\infty} [L(r(n)+\lceil \log n \rceil)/\log(r(n)+\lceil \log n \rceil)] = 0. \tag{1}$$

It is easily seen that, for some constant $c' \geq 0$, $r(n)+\lceil \log n \rceil \leq c' n \log n$, and thus $\log(r(n)+\lceil \log n \rceil) \leq \log n + \log\log n + \log c'$. From this and equation (1), we have

$$\lim_{n\to\infty} [L(r(n)+\lceil \log n \rceil)/(\log n + \log\log n + \log c')] = 0.$$

From this, it follows that $\lim_{n\to\infty}[L(r(n)+\lceil \log n \rceil)/\log n] = 0$. Therefore, we have $|V(n)| > p(n)$ for large n, and so it follows that for large n there must be two different strings x, y in V(n) such that $C(x) \cap C(y) \neq \emptyset$. This contradicts Proposition 3.2, and completes the proof of part (iii) of the lemma. □

From Lemma 3.1, we can derive the following theorem.

Theorem 3.3. *Let* $L: N \to R$ *be any function such that* $L(n) \geq \log\log n$ *and* $\lim_{n\to\infty}[L(n)/\log n] = 0$. *Then,* (i) $\mathscr{L}[\mathrm{UONTM}(L(n))] \subsetneq \mathscr{L}[\mathrm{AONTM}(L(n))]$, and (ii) $\mathscr{L}[\mathrm{UONTM}(L(n))] \subsetneq \mathscr{L}[\mathrm{UTM}(L(n))]$.

Theorem 3.4. $\mathscr{L}[\mathrm{UONTM}(L(n))]$ *is not closed under complementation for any function* $L: N \to R$ *such that* $\lim_{n\to\infty}[L(n)/\log n] = 0$ *and* $L(n) \geq \log\log n$.

Proof. Let $L_2 = \{\mathrm{bin}(1)\#\mathrm{bin}(2)\#\cdots\#\mathrm{bin}(n)2wcw \in \{0, 1, 2, c, \#\}^+ \mid n \geq 2 \,\&\, w \in \{0, 1\}^+ \,\&\, |w| = \lceil \log n \rceil\}$. The set L_2 is accepted by a UONTM(log log n) M which acts as follows. Suppose that an input string

$$\mathrm{¢}y_1\#y_2\#\cdots\#y_n2wcw'\$,$$

where $n \geq 2$, and the y_i's, w, and w' are all in $\{0, 1\}^+$, is presented to M. By using the technique in the proof of Lemma 3.1(i), M first marks off log log n storage-tape cells when $y_i = \mathrm{bin}(i)$ for each $1 \leq i \leq n$. If in the first phase M successfully marks off log log n storage-tape cells, then M checks by using universal branches that $|w| = |w'| = \lceil \log n \rceil$ and $w(i) = w'(i)$ for each $1 \leq i \leq \lceil \log n \rceil$. Of course, in this phase, M makes use of log log n storage-tape cells (constructed above) to count a number between 1 and $\lceil \log n \rceil$. It will be obvious that $T(M) = L_2$. On the other hand, by using the same technique as in the proof of Lemma 3.1(iii), we can show that $\bar{L}_2$, the complement of L_2, is not accepted by any UONTM($L(n)$) such that $\lim_{n\to\infty}[L(n)/\log n] = 0$. (The proof is left to the reader as an exercise.) This completes the proof of the theorem. □

[5] For any set S, $|S|$ denotes the number of elements of S.

Remarks. Together with the result in [7], Theorems 3.1 and 3.2 imply that, for any function $L: N \to R$ such that $L(n) \geq \log\log n$ and $\lim_{n\to\infty}[L(n)/n] = 0$, we have (i) $\mathscr{L}[\mathrm{UONTM}(L(n))] \subsetneq \mathscr{L}[\mathrm{AONTM}(L(n))]$, (ii) $\mathscr{L}[\mathrm{UONTM}(L(n))] \subsetneq \mathscr{L}[\mathrm{UTM}(L(n))]$, and (iii) $\mathscr{L}[\mathrm{UONTM}(L(n))]$ is not closed under complementation. It is obvious that $\mathscr{L}[\mathrm{UONTM}(L(n))]$ is closed under intersection for any L. It is, however, unknown whether or not $\mathscr{L}[\mathrm{UONTM}(L(n))]$ is closed under union for any L. It is also unknown whether or not $\mathscr{L}[\mathrm{AONTM}(L(n))] \subsetneq \mathscr{L}[\mathrm{ATM}(L(n))]$ for any L such that $L(n) \geq \log\log n$ and $\lim_{n\to\infty}[L(n)/\log n] = 0$. Note the fact [9] that $\mathscr{L}[\mathrm{ATM}(L(n))] = \mathscr{L}[\mathrm{AONTM}(L(n))]$ for any $L(n) \geq \log n$. Also note that $\mathscr{L}[\mathrm{ATM}(L(n))]$ is equal to the class of regular sets for any L such that $\lim_{n\to\infty}[L(n)/\log\log n] = 0$, which is reported in [12, 13].

We conclude this section by showing that there exists an infinite hierarchy among the $\mathscr{L}[\mathrm{UONTM}(L(n))]$'s with $\log\log n \leq L(n) \leq \log n$.

Definition 3.5. A function $L: N \to R$ is *fully constructible* [3] if there exists a deterministic $L(n)$ space bounded Turing machine M such that, for all inputs of length n, M eventually will halt having marked exactly $\lceil L(n) \rceil$ storage-tape cells.

Theorem 3.6. *Let $f: N \to R$ be a fully constructible function such that $1 \leq \log\log n \leq f(\lceil \log\log n \rceil) \leq \log n$ for all $n \geq n_0$ (where n_0 is some constant), and $g: N \to R$ be a nondecreasing function such that $\lim_{n\to\infty}[g(2n)/f(n)] = 0$. Further, for each function $h: N \to R$, let $L_h: N \to R$ be the function such that $L_h(n) = h(\lceil \log\log n \rceil)$, $n \geq 1$. Then there exists a set in $\mathscr{L}[\mathrm{UONTM}(L_f(n))]$, but not in $\mathscr{L}[\mathrm{UONTM}(L_g(n))]$.*

Proof. Let $S(f)$ be the following set depending on the function f in the theorem:

$$S(f) = \{\mathrm{bin}(1) \# \mathrm{bin}(2) \# \cdots \# \mathrm{bin}(n) 2wcw' \in \{0, 1, 2, c, \#\}^+ \mid n \geq n_0 \ \& \ (w, w' \in \{0, 1\}^+) \ \& \ |w| = |w'| = \lceil L_f(n) \rceil \ \& \ w \neq w'\}$$

(where n_0 is the constant in the theorem).

We will prove the theorem by showing that the set $S(f)$ is accepted by some $\mathrm{UONTM}(L_f(n))$, but not accepted by any $\mathrm{UONTM}(L_g(n))$. We consider a $\mathrm{UONTM}(L_f(n))$ M which acts as follows. Suppose that an input string $\mathrm{¢}y_1 \# y_2 \# \cdots \# y_n 2wcw'\$$, where $n \geq n_0$, and the y_i's, w, and w' are all in $\{0, 1\}^+$, is presented to M. By using the technique in the proof of Lemma 3.1(i), M first marks off $\lceil \log\log n \rceil$ storage-tape cells when $y_i = \mathrm{bin}(i)$ for each $1 \leq i \leq n$. If in this phase M successfully marks off $\lceil \log\log n \rceil$ storage-tape cells, then, by making use of these $\lceil \log\log n \rceil$ storage-tape cells, M deterministically marks off $\lceil L_f(n) \rceil = \lceil f(\lceil \log\log n \rceil) \rceil$ storage-tape cells. This action is possible because the function f is fully constructible. M then checks by using the $\lceil L_f(n) \rceil$ storage-tape cells that $|w| = |w'| = \lceil L_f(n) \rceil$ and $w \neq w'$, and M accepts the input only if this check is successful. It will be obvious that $T(M) = S(f)$. On the other hand, by using the same technique as in the proof of Lemma 3.1(iii), we can show that $S(f)$ is not

accepted by any UONTM($L_g(n)$), where g is a nondecreasing function such that $\lim_{n\to\infty}[g(2n)/f(n)]=0$. This completes the proof of the theorem. □

4. Discussion

One reads two distinct definitions for measuring space complexity in the literature. These are: (i) an ATM (UTM, AONTM, UONTM) M is $L(n)$ space bounded if for each n and for each input x (of length n) accepted there is an accepting computation tree of M on x such that M uses at most $L(n)$ cells of its storage tape on each computation path of the tree, and (ii) an ATM (UTM, AONTM, UONTM) M is $L(n)$ space bounded if for each n and for each input x of length n each computation path of M on x uses at most $L(n)$ cells of its storage tape.

In Sections 2 and 3 of this note we adopted (i) above. We now briefly investigate what happens if we adopt (ii) above as the definition for measuring space complexity. Let UONTM($L(n)$) denote a UONTM which is $L(n)$ space bounded in the sense of (ii) above. We can easily show that if $\lim_{n\to\infty} L(n)/\log n=0$, then the class of languages accepted by UONTM($L(n)$)'s is equal to the class of regular languages. The proof is left to the reader as an easy exercise.

Acknowledgment

The authors thank the anonymous referee for his (or her) useful comments.

References

[1] A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation, Res. Rept. RC7489, IBM T.J. Watson Research Center, Yorktown Heights, NY, 1978.

[2] A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation, *J.ACM* **28** (1) (1981) 114–133.

[3] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Reading, MA, 1979).

[4] A. Ito, K. Inoue, I. Takanami and H. Taniguchi, Two-dimensional alternating Turing machines with only universal states, *Inform. and Control* **55** (1–3) (1982) 193–221.

[5] K. Inoue, I. Takanami and H. Taniguchi, Two-dimensional alternating Turing machines, *Proc. 14th Ann. ACM Symp. on Theory of Computing* (1982) 37–46.

[6] K. Inoue, I. Takanami and H. Taniguchi, A note on alternating on-line Turing machines, *Inform. Process. Lett.* **15** (4) (1982) 164–168.

[7] K. Inoue, I. Takanami, H. Taniguchi and A. Ito, A note on alternating on-line Turing machines with only universal states, *Trans. IECE Japan* **E-66** (6) (1983) 395–396.

[8] K.N. King, Measures of parallelism in alternating computation trees, *Proc. 13th Ann. ACM Symp. on Theory of Computing* (1981) 189–201.

[9] R.E. Ladner, R.J. Lipton and L.J. Stockmeyer, Alternating pushdown automata, *Proc. 19th IEEE Symp. on Foundations of Computer Science*, Ann Arbor, MI (1978) 92–106.

[10] W.J. Paul, E.J. Prauss and R. Reischuk, On alternation, *Acta Inform.* **14** (1980) 243–255.

[11] W.L. Ruzzo, Tree-size bounded alternation, *J. Comput. System Sci.* **21** (1980) 218–235.

[12] I.H. Sudborough, Efficient algorithms for path system problems and applications to alternating and time-space complexity classes, *Proc. 21st Ann. Symp. on Foundations of Computer Science* (1980) 62–72.

[13] I.H. Sudborough, Bandwidth constraints on problems complete for polynomial time, *Theoret. Comput. Sci.* **26** (1983) 25–52.

[14] M. Tompa, An extension of Savitch's theorem to small space bounds, *Inform. Process. Lett.* **12** (2) (1981) 106–108.