Acta Informatica 8, 371 – 382 (1977)

The LBA-Problem and the Deterministic Tape Complexity of Two-Way One-Counter Languages over a One-Letter Alphabet

Burkhard Monien*

Abteilung Informatik der Universität, Postfach 500500, D-4600 Dortmund 50, Germany (Fed. Rep.)

Summary. It is shown, that NTAPE(n) is equal to TAPE(n) if and only if every language $L \subset \dashv\{1\}^*\vdash$ which is acceptable by a nondeterministic two-way one-counter automaton whose counter length is bounded by the length of its input is contained in TAPE($\log n$).

1. Introduction

The relationships between the classes defined by deterministic and nondeterministic tape complexity (denoted by TAPE($f(n)$) and NTAPE($f(n)$), respectively) have been of considerable interest in the last years. Several authors [3, 4, 7, 8] showed that NTAPE($\log n$) = TAPE($\log n$) holds if and only if certain subclasses of the context free languages (linear languages, one-way one-counter languages) are contained in TAPE($\log n$).

In this paper we consider the original LBA-problem, that is the problem whether NTAPE(n) = TAPE(n). We show that NTAPE(n) is equal to TAPE(n) if and only if each language $L \subset \dashv\{1\}^*\vdash$, which is acceptable by some nondeterministic two-way one-counter automaton whose counter length is bounded by the length of its input, is contained in TAPE($\log n$).

This result gives another possibility to further investigations than the results mentioned above. Since our one-counter automaton operates only on unary input strings it is easy to see (§ 3) that this automaton has the same power as some modified nondeterministic two-register machine with lineary bounded registers. Therefore it is important to gain further knowledge about the functions defined by nondeterministic two-register machines with lineary bounded registers.

I want to thank Dr. Huwig (University of Dortmund) for numerous discussions and for his careful reading of the paper.

* *Permanent address:* Fachbereich Mathematik/Informatik der Gesamthochschule Paderborn, Warburger Str. 100, D-4790 Paderborn, Germany (Fed. Rep.)

2. Proof of the Main Result

We will use the following results proved in [5] and [1, 4], respectively.

Result 1. $\mathrm{NTAPE}(n)=\mathrm{TAPE}(n)$

$$\Leftrightarrow [L \subset \dashv\{1\}^*\vdash \wedge L\in \mathrm{NTAPE}(\log n) \Rightarrow L\in \mathrm{TAPE}(\log n)]$$

Result 2. Let $\mathscr{L}_1$, $\mathscr{L}_2$, $\mathscr{L}_3$ be three families of languages such that $\mathscr{L}_1\subset\mathscr{L}_3$ and $\mathscr{L}_2\subset\mathscr{L}_3$ and $\mathscr{L}_1$ is closed under intersection. If there exists a family T of mappings with the following two properties

1. $\bigwedge\limits_{L\in\mathscr{L}_3} \bigvee\limits_{\tau\in T} \bigvee\limits_{L_1\in\mathscr{L}_1} \bigvee\limits_{L_2\in\mathscr{L}_2} [\tau(L)=L_1\cap L_2]$,
2. $\bigwedge\limits_{L\in\mathscr{L}_3} \bigwedge\limits_{\tau\in T} [\tau(L)\in\mathscr{L}_1 \Rightarrow L\in\mathscr{L}_1]$

then $\mathscr{L}_3\subset\mathscr{L}_1 \Leftrightarrow \mathscr{L}_2\subset\mathscr{L}_1$.

Furthermore we use the notion of a two-way k-counter automaton (see f.e. [2]).

Definition. A two-way k-counter automaton $M=(S, X, \delta, s_0, F)$ consists of a finite memory (S-set of states, $s_0\in S$-starting state, $F\subset S$-set of final states), k counters and an input tape (X-set of input symbols) with a two-way read-only head. δ: $S\times X\times\{0,1\}^k\to 2^{S\times\{-1,0,+1\}^{k+1}}$ is the transition function.

A configuration of M is a $(k+2)$-tuple $(s, w, i, n_1, \ldots, n_k)\in S\times X^*\times\mathbb{N}^{k+1}$. In the usual way δ defines a relation $\to$ on the set of configurations. $\xrightarrow{*}$ is the transitive closure of $\to$. M is called deterministic if each image under δ has at most one element. Otherwise M is called nondeterministic.

We denote by $\mathscr{C}_k$, $k\in\mathbb{N}$, the class of all languages acceptable by nondeterministic two-way k-counter automata whose counter lenghts are bounded by the lenght of the input. More formally,

$L\in\mathscr{C}_k$

$\Leftrightarrow$ There exists a nondeterministic two-way k-counter automaton M with the following properties:

1. $w\in L \Leftrightarrow \exists t\in F$: $i, n_1, \ldots, n_k\in\mathbb{N}$:

$(s_0, w, 1, 0, \ldots, 0) \xrightarrow[M]{*} (t, w, i, n_1, \ldots, n_k)$

2. For all $w\in L$ there exists a sequence of configurations leading to acceptance such that all counter lenghts are bounded by $l(w)$.

The following lemma can be proved with the same method which was used in [2] in order to prove a simular result.

Lemma 1. $\mathrm{NTAPE}(\log n)=\bigcup\limits_{k\in\mathbb{N}}\mathscr{C}_k$.

We are only interested in one-letter alphabets.

Definition. $\mathscr{C}_k^1=\{L\in\mathscr{C}_k \mid L\subset\dashv\{1\}^*\vdash\}$.

Let $\mathrm{TAPE}^1(\log n)$, $\mathrm{NTAPE}^1(\log n)$ be defined in the same way. Now we can formulate our main theorem.

Theorem 1. $\mathrm{TAPE}(n) = \mathrm{NTAPE}(n)$

$\Leftrightarrow \mathscr{C}_1^1 \subset \mathrm{TAPE}(\log n)$.

We will prove theorem 1 by means of result 2. The families we have to consider are $\mathscr{L}_1 = \mathrm{TAPE}^1(\log n)$, $\mathscr{L}_2 = \mathscr{C}_1^1$, $\mathscr{L}_3 = \mathrm{NTAPE}^1(\log n)$. Furthermore we have to define a family T of mappings such that property 1 and 2 hold.

For any $f\colon \mathbb{N} \to \mathbb{N}$ we denote by f also the mapping from $2^{\dashv\{1\}^*\vdash}$ into $2^{\dashv\{1\}^*\vdash}$ defined by $f(L) = \{\dashv 1^{f(n)}\vdash \mid n \in \mathbb{N} \wedge \dashv 1^n \vdash \in L\}$.

Let $\beta\colon \mathbb{N} \to \{0,1\}^*$ be the injective mapping which associates to each $n \in \mathbb{N}$ its binary decomposition. Then for any function $f\colon \mathbb{N} \to \mathbb{N}$ let $\tilde{f}\colon \{0,1\}^* \to \{0,1\}^*$ be the mapping defined by

$$\tilde{f}(v) = \begin{cases} \beta f \beta^{-1}(v), & \text{if } v \in \beta(\mathbb{N}) \\ v, & \text{otherwise.} \end{cases}$$

Lemma 2. Let $f\colon \mathbb{N} \to \mathbb{N}$ be an injective function such that $\tilde{f}$ is computable by a deterministic linear bounded automaton. Then for every $L \subset \dashv\{1\}^*\vdash$ $f(L) \in \mathrm{TAPE}(\log n)$ implies $L \in \mathrm{TAPE}(\log n)$.

Proof. Let M be a deterministic $\log n$-tape bounded Turing machine accepting $f(L)$. Then L is accepted by a deterministic Turing machine $\tilde{M}$ with three working tapes which operates on an input string $\dashv 1^n \vdash$, $n \in \mathbb{N}$, in the following way:

1. $\tilde{M}$ writes $\beta(n)$ on its working tape 1.
2. $\tilde{M}$ computes $\tilde{f}(\beta(n))$ on working tape 1.
3. $\tilde{M}$ simulates M step by step. During this simulation M stores on its working tape 2 the head position of M in binary notation and on its working tape 3 the inscription of the working tape of M. In order to simulate one step $\tilde{M}$ examines whether working tape 2 stores 0 or the same inscription as working tape 1.

It is clear that $\tilde{M}$ works with the tape bound $\log n$. □

Note that $\tilde{f}$ is computable by some linear bounded automaton only if f is polynomially bounded.

Lemma 2 will be helpfull in order to prove property 2 of the family T of mappings which we still have to define. In order to verify property 1 we have to show that every two-way k-counter automaton (with bounded counter length) can be simulated by a two-way 1-counter automaton provided the input string has been transformed in a suitable manner. Therefore the main difficulty in proving theorem 1 is to find pairing functions $P_q\colon \mathbb{N}^q \to \mathbb{N}$ which are polynomially bounded and which have the property that given some number $m = P_q(n_1, \ldots, n_q)$ the test "which of the n_v, $v = 1, \ldots, q$, is equal to 0" and the operations $m \to P_q(n_1 + \eta_1, \ldots, n_q + \eta_q)$, $\eta_v \in \{-1, 0, +1\}$, can be performed by a two-way one-counter automaton. Because of technical reasons we will modify the notion of a pairing function a little and will define functions

$$P_q\colon \bigcup_{m \in \mathbb{N}} \{0, \ldots, 2^{l(\beta(m))} - 1\}^q \times \{m\} \to \mathbb{N}$$

in such a way that for all $n_1, \dots, n_q$, m with $n_1, \dots, n_q < 2^{l(\beta(m))}$ the 3-nary decomposition of $P_q\ (n_1, \dots, n_q, m)$ is $2\rho_1 2 \dots 2\rho_q 2$ with $l(\rho_v) = l(\beta(m))$ and $\rho_v = 0 \dots 0\beta(n_v)^R$ for all $v = 1, \dots, q$. (Here l denotes lengths of a 0-1-string and ^{R} its reversal.)

Mappings b: $\mathbb{N}^2 \to \mathbb{N}$; th, l, L, r_q, τ_q: $\mathbb{N} \to \mathbb{N}$ are defined for every $q \in \mathbb{N}$ in the following way

$$b(i, n) = \begin{cases} 0, & \text{if } \left[\frac{n}{2^i}\right] \text{ even} \\ 1, & \text{otherwise,} \end{cases}$$

$$l(n) = \min\{i \mid 2^i > n\},$$

$$L(n) = \min\{i \mid 3^i > n\},$$

$$th(n) = \sum_{i=0}^{l(n)-1} b(i, n) \cdot 3^{l(n)-1-i}$$

(Note that $\beta(n) = b(0, n) \dots b(l(n)-1, n)$, that $l(n)$ is the length of the binary and $L(n)$ the lenght of the 3-nary decomposition of n and that the 3-nary decomposition of $th(n)$ is just the reversal of the binary decomposition of n.)

$$r_q(n) = q \cdot l(n+1) + q + 1,$$

$$\tau_q(n) = 3^{r_q(n)} \cdot \left(\sum_{v=0}^{q-1} 2 \cdot 3^{v(l(n+1)+1)} + 3^{(q-1)\cdot l(n+1)+q} \cdot th(n+1) + 2 \cdot 3^{q \cdot l(n+1)+q}\right)$$
$$+ 3^{3r_q(n)+1}.$$

Note that $\tau_q(n)$ has the 3-nary decomposition

$$\underbrace{0 \dots 0}_{r_q(n)}\ 2\ \underbrace{\underbrace{0 \dots 0}_{l(n+1)}\ 2 \dots 2\ \underbrace{0 \dots 0}_{l(n+1)}}_{q-1}\ 2\beta(n+1)^R\ 2\ \underbrace{0 \dots 0}_{r_q(n)}\ 1$$

We set $\mathrm{T} = \{\tau_q \mid q \in \mathbb{N}\}$.

Theorem 2. For each $L \subset \dashv\{1\}^*\vdash$, $q \in \mathbb{N}$ $\tau_q(L) \in \mathrm{TAPE}(\log n)$ implies $L \in \mathrm{TAPE}(\log n)$.

Proof. Because of Lemma 2 we have only to define a deterministic linear bounded automaton M_q which computes $\bar{\tau}_q$. Starting with some $v \in \{0, 1\}^*$ M_q tests first whether there exists some $n \in \mathbb{N}$ with $\beta(n) = v$. If this is the case then M_q writes the 3-nary decompositions of $\tau_q(\beta^{-1}(v))$ on its working tape and computes afterwards the binary decomposition. □

Theorem 3. For each $L \in \mathrm{NTAPE}^1(\log n)$ there exist $q \in \mathbb{N}$ and $L_1 \in \mathrm{TAPE}^1(\log n)$, $L_2 \in \mathscr{C}_1^1$ such that $\tau_q(L) = L_1 \cap L_2$.

Proof. Because of Lemma 1 there exists a $p \in \mathbb{N}$ such that $L \in \mathscr{C}_p^1$. Set $q = p + 2$. (We will encode by one number the contents of the p counters and the distances of the head position from the endmarkers.)

1. Set $L_1 = \{\dashv 1^{\tau_q(n)} \vdash | n \in \mathbb{N}\}$.

In order to test whether an arbitrary $m \in \mathbb{N}$ is of the form $\tau_q(n)$, a Turing machine has to compute the 3-nary decomposition of m and has to test whether it is of the form $0^r(20^{l(v)})^{q-1} 2v\, 20^r 1$ with some $v \in \{0\} \cup \{0,1\}^* \{1\}$ and $r = q \cdot l(v) + q + 1$. This can be done using not more than $\log n$ cells.

2. Now we have to define a nondeterministic two-way one-counter automaton $\tilde{M}$ which accepts an input string of the form $\dashv 1^{\tau_q(n)} \vdash$ if and only if $\dashv 1^n \vdash \in L$. We don't have to consider the behaviour of $\tilde{M}$ on input strings which are not of this form.

Let $M = (S, \{\dashv, 1, \vdash\}, \delta, s_0, F)$ be a nondeterministic two-way p-counter automaton accepting L in such a way that its counter lenghts are bounded by the lenght of its input. $\tilde{M}$ will simulate M step by step. Formally we define our "pairing functions" P_k, $k \in \mathbb{N}$, in the following way:

$$P_0(m) = 2,$$
$$P_{k+1}(n_0, n_1, \dots, n_k, m) = 2 + 3^{1 + l(m) - l(n_0)} \cdot th(n_0) + 3^{l(m)+1} P_k(n_1, \dots, n_k, m)$$
for all $m \in \mathbb{N}$ and $n_0, n_1, \dots, n_k < 2^{l(m)}$.

To each configuration $\mathscr{K} = (s, \dashv 1^m \vdash, i, n_1, \dots, n_p)$ of M with $\bigwedge_{j \in \{1,\dots,p\}} [n_j < 2^{l(m+1)}]$ we associate the configuration

$$\kappa(\mathscr{K}) = (s, \dashv 1^{\tau_q(m)} \vdash, P_q(n_1, \dots, n_p, i-1, m+2-i, m+1), 0)$$

of $\tilde{M}$. We will define $\tilde{M}$ in such a way that for any two configurations

$$\mathscr{K}_v = (s_v, \dashv 1^m \vdash, i_v, n_1^v, \dots, n_p^v), \; m \in \mathbb{N} \;\; v = 1, 2,$$

of M with $\bigwedge_{v \in \{1,2\}} \bigwedge_{j \in \{1,\dots,p\}} [n_j^v < 2^{l(m+1)}]$

$$\mathscr{K}_1 \xrightarrow[M]{} \mathscr{K}_2 \quad \text{implies} \quad \kappa(\mathscr{K}_1) \xrightarrow[\tilde{M}]{*} \kappa(\mathscr{K}_2)$$

and

$$\kappa(\mathscr{K}_1) \xrightarrow[\tilde{M}]{*} \kappa(\mathscr{K}_2) \quad \text{implies} \quad \mathscr{K}_1 \xrightarrow[M]{*} \mathscr{K}_2.$$

Let $\mathscr{K} = (s, \dashv 1^m \vdash, i, n_1, \dots, n_p)$ be some configuration of M. We have to show how $\tilde{M}$ simulates the next move of M. This simulation is done in three steps:

A. $\tilde{M}$ computes all the information which is necessary in order to apply the transition function of M.

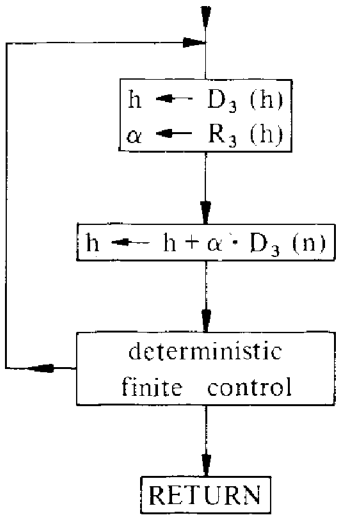
B. $\tilde{M}$ applies the transition function of M.

C. $\tilde{M}$ changes its head-position according to the alternations performed by M.

A. First we show how $\tilde{M}$ starting with some configuration

$$\tilde{\mathscr{K}} = (s, \dashv 1^n \vdash, P_q(z_1, \dots, z_q, z), 0), \; s \in S; \; z_1, \dots, z_q, \; z \in \mathbb{N} \text{ and}$$

$n = 3^{q \cdot l(z)+q+1} \cdot n'$ with some $n' \in \mathbb{N}$, decides which of the $z_1, \ldots, z_q$ are 0. (Remember that $L(P_q(z_1, \ldots, z_q, z)) = q \cdot l(z) + q + 1$.) $\tilde{M}$ performs the following algorithm A1:



Here h denotes the head position of $\tilde{M}$ and D_3, R_3: $\mathbb{N} \to \mathbb{N}$ are the functions $D_3(i) = \max\{k \mid 3k \leqq i\}$, $R_3(i) = i - 3 \cdot D_3(i)$. It is clear that $\tilde{M}$ can compute $D_3(h)$ and $R_3(h)$ simultaneously by using its counter. In order to compute $h + \alpha \cdot D_3(n)$ $\tilde{M}$ stores h on its counter and successively adds α to its counter whenever it moves 3 cells to the right with its head.

The finite control stores how often α has had the value 2 and uses its RETURN-exit when this number is $q+1$. Furthermore the finite control stores for each occuring $\alpha = 2$ (except of the first one) whether the preceding 0-1-string contained a 1. Note that in the 3-nary decomposition of $P_q(z_1, \ldots, z_q, z)$ a substring $\ldots 20^{l(z_v)} 2 \ldots$ occurs if and only if the corresponding z_v is zero.

Starting with the configuration $(s, \dashv 1^n \vdash, P_q(z_1, \ldots, z_q, z), 0)$

$$P_q(z_1, \ldots, z_q, z) = \sum_{v=0}^{r-1} a_v 3^v; \qquad a_v \in \{0, 1, 2\} \ \forall v = 0, \ldots, r-1;$$

$r = L(P_q(z_1, \ldots, z_q, z)) = q \cdot l(z) + q + 1$, $\tilde{M}$ reaches after i, $0 \leqq i \leqq r$, runs of the algorithm the head position

$$h = \sum_{v=i}^{r-1} a_v 3^{v-i} + \sum_{v=0}^{i-1} a_v D_3^{i-v}(n).$$

We will prove this by induction.

$i=0$: $h = P_q(z_1, \ldots, z_q, z)$.

Now suppose

$$h = \sum_{v=i}^{r-1} a_v 3^{v-i} + \sum_{v=0}^{i-1} a_v D_3^{i-v}(n)$$

and $i < r$. Then $\tilde{M}$ will perform another run. Note that $n = 3^r \cdot n'$ with some $n' \in \mathbb{N}$.

$$\Rightarrow h = \sum_{v=i}^{r-1} a_v 3^{v-i} + n' \cdot 3^{r-i} \sum_{v=0}^{i-1} a_v 3^v,$$

$$\Rightarrow \alpha = R_3(h) = a_i \quad \text{and} \quad D_3(h) = \sum_{v=i+1}^{r-1} a_v 3^{v-i-1} + \sum_{v=0}^{i-1} a_v D_3^{i-v-1}(n).$$

Performing $h \leftarrow h + \alpha \cdot D_3(n)$ $\tilde{M}$ reaches

$$h = \sum_{v=i+1}^{r-1} a_v 3^{v-(i+1)} + \sum_{v=0}^{i} a_v D_3^{(i+1)-v}(n).$$

Therefore we have proved our assumption and it is clear that during this process $\tilde{M}$ notices which of the $z_1, \ldots, z_q$ are zero. In r runs $\tilde{M}$ reaches the configuration $((s, b_1, \ldots, b_q), \dashv 1^n \vdash, \sum_{v=0}^{r-1} a_v D_3^{r-v}(n), 0)$ with

$$b_v = \begin{cases} 0, & \text{if } z_v = 0 \\ 1, & \text{otherwise} \end{cases} \qquad \forall v = 1, \ldots, q.$$

B. Now suppose $n = \tau_q(m)$, $z = m+1$, $z_v = n_v$ for $v = 1, \ldots, p$ and $z_{p+1} = i-1$, $z_{p+2} = m+2-i$. Then $\tilde{\mathscr{K}} = \kappa(\mathscr{K})$ and $b_v = 0 \Leftrightarrow n_v = 0$ for all $i = 1, \ldots, p$, $b_{p+1} = 0 \Leftrightarrow i = 1$, $b_{p+2} = 0 \Leftrightarrow i = m+2$. Note that if the configuration of M is $\mathscr{K}$ then M reads the symbol

$$\alpha = \begin{cases} \dashv, & \text{if } b_{p+1} = 0 \\ \vdash, & \text{if } b_{p+2} = 0 \\ 1, & \text{otherwise.} \end{cases}$$

We have to consider the following cases.

a) $\delta(s, \alpha, b_1, \ldots, b_p)$ is not defined. Then the actual computation of $\tilde{M}$ ends without accepting.

b) There exist $s' \in S$ and $\eta_1, \ldots, \eta_q \in \{-1, 0, +1\}$ (possibly not unique) such that $\eta_q = -\eta_{q-1}$ and $(s', \eta_{p+1}, \eta_1, \ldots, \eta_p) \in \delta(s, \alpha, b_1, \ldots, b_p)$. The transition function of $\tilde{M}$ is defined in such a way that in any of these cases and for arbitrary $z_1, z_2 \in \mathbb{N}$

$$((s, b_1, \ldots, b_q), \dashv 1^n \vdash, z_1, z_2) \xrightarrow[\tilde{M}]{} ([s', \eta_1, \ldots, \eta_q], \dashv 1^n \vdash, z_1, z_2)$$

holds if $s' \notin F$ and $\tilde{M}$ accepts $\dashv 1^n \vdash$ otherwise.

C. Remember that so far

$$\kappa(\mathscr{K}) = (s, \dashv 1^{\tau_q(m)} \vdash, P_q(n_1, \ldots, n_p, i-1, m+2-i, m+1), 0)$$
$$\xrightarrow[\tilde{M}]{*} \left([s', \eta_1, \ldots, \eta_p, \eta_{p+1}, -\eta_{p+1}], \dashv 1^{\tau_q(m)} \vdash, \sum_{v=0}^{r-1} a_v D_3^{r-v}(\tau_q(m)), 0\right)$$

where $P_q(n_1, \ldots, n_p, i-1, m+2-i, m+1) = \sum_{v=0}^{r-1} a_v 3^v$; $r = r_q(m)$.

We will now complete this part of the definition of $\tilde{M}$ in such a way that $\tilde{M}$ reaches deterministically the head position

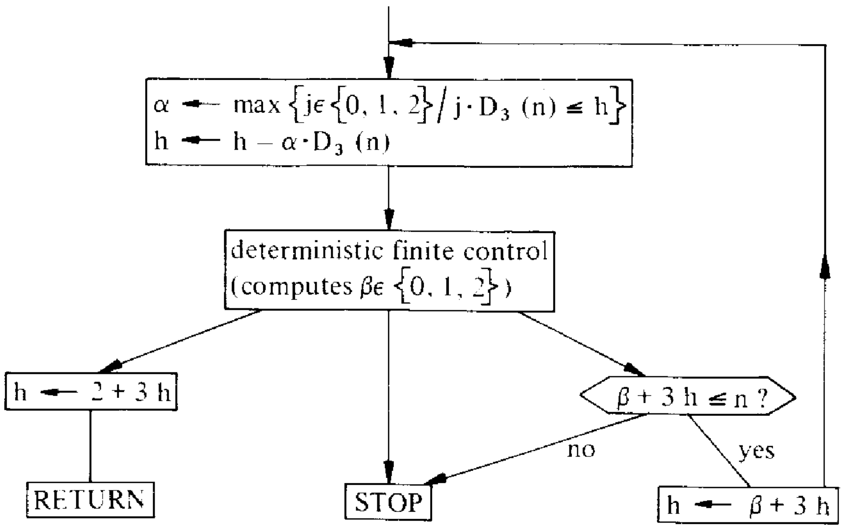
$$h = P_q(n_1+\eta_1, \ldots, n_p+\eta_p, i+\eta_{p+1}-1, m+2-i-\eta_{p+1}, m+1)$$

if $\bigwedge_{j\in\{1,\ldots,p\}} [n_j+\eta_j < 2^{l(m+1)}]$ and stops without acceptance otherwise. Since $\tau_q(m) = z_m \cdot 3^{r_q(m)} + 3^{3r_q(m)+1}$ with some $z_m < 3^{r_q(m)}$ and since $r = r_q(m)$

$$\sum_{v=0}^{r-1} a_v D_3^{r-v}(\tau_q(m)) = (z_m + 3^{2r+1}) \cdot \left(\sum_{v=0}^{r-1} a_v 3^v\right) = z'_m + 3^{2r+1} \sum_{v=0}^{r-1} a_v 3^v$$

with some $z'_m < 3^{2r+1}$.

$\tilde{M}$ performs the following algorithm A2:



α ← max{j∈{0, 1, 2}/j·D₃(n) ≤ h}
h ← h − α·D₃(n)
deterministic finite control (computes β∈{0, 1, 2})
h ← 2 + 3 h
RETURN
STOP
β + 3 h ≤ n ?
no
yes
h ← β + 3 h

We will show below that for all $i = 1, \ldots, r$ the value of α computed in the i-th run of A2 is a_{r-i}.

The deterministic finite control operates in the following way:

1. It stores how often α has had the value 2.

2. If this number is $q+1$ then it goes to its RETURN-Exit.

3. After reading $\alpha = 2$ for the v-th time it simulates on the 0-1-string following up to the next value $\alpha = 2$ a finite automaton which performs the binary addition of η_v. The value β is the output symbol of this automaton when it reads the symbol α.

4. When it reads $\alpha = 2$ and the carry of the addition performed in 3. is still 1 then it goes to its STOP exit. Otherwise it sets $\beta = 2$.

By induction on i we will prove the following result:

Let $\tilde{M}$ start A2 with the configuration

$$\left([s, \eta_1, \ldots, \eta_p, \eta_{p+1}, -\eta_{p+1}], \dashv 1^{\tau_q(m)} \vdash, \sum_{j=0}^{r-1} a_j D_3^{r-j}(\tau_q(m)), 0\right).$$

Then for all $i=1,\dots,r$ $(i=1+\nu(l(m+1)+1)+\mu,\ 0\leqq\nu\leqq q,\ 0\leqq\mu\leqq l(m+1),\ \nu=q\Rightarrow\mu=0)$ exactly one of the following two cases will occur.

1. A2 uses its STOP exit before the start of the $(i+1)$st run if and only if $n_j+\eta_j\geqq 2^{l(m+1)}$ for some $j\in[q+1-\nu,q]$.

2. After performing the i-th run A2 has computed the head position

$$h=\sum_{j=0}^{\mu-1} b(j,n_{q-\nu}+\eta_{q-\nu})\,3^{\mu-1-j}+3^{\mu}P_\nu(n_{q+1-\nu}+\eta_{q+1-\nu},\dots,n_q+\eta_q,m+1)$$
$$+3^i\sum_{k=0}^{r-1-i} a_k D_3^{r-k}(\tau_q(m)).$$

[In this expression $P_0(n_{q+1}+\eta_{q+1},\dots,n_q+\eta_q,m+1)$ has to be identified with $P_0(m+1)$.]

Proof. 1. $i=1$ $(\mu=\nu=0)$. A2 starts with $h=\sum_{j=0}^{r-1} a_j D_3^{r-j}(\tau_q(m))$. It is clear that $a_{r-1}\cdot D_3(\tau_q(m))\leqq h$. On the other hand we mentioned above that these exists a $z'_m<3^{2r+1}$ such that

$$h=z'_m+3^{2r+1}\sum_{j=0}^{r-1} a_j 3^j=z''_m+a_{r-1}\cdot 3^{3r}$$

with some $z''_m<3^{3r}$. Therefore $a_{r-1}=\max\{j\in\{0,1,2\}\,|\,j\cdot D_3(\tau_q(m))\leqq h\}$ since $a_{r-1}=2$ the finite control computes $\beta=2$ and the instructions $h\leftarrow h-\alpha\cdot D_3(\tau_q(m))$, $h\leftarrow\beta+3h$ yield

$$h=2+3\cdot\sum_{j=0}^{r-2} a_j D_3^{r-j}(\tau_q(m)).$$

2. Now suppose $i\geqq 1$ and $i<r$ and A2 has reached after i runs the head position statet above. Then obviously $a_{r-1-i}D_3(\tau_q(m))\leqq h$. On the other hand

$$P_\nu(n_{q+1-\nu}+\eta_{q+1-\nu},\dots,n_q+\eta_q,m+1)<3^{1+\nu(l(m+1)+1)}$$

and therefore there exists a $z<3^i$ such that

$$h=z+3^i\cdot\sum_{k=0}^{r-1-i} a_k\cdot D_3^{r-k}(\tau_q(m)).$$

$$\Rightarrow h=z+3^i z'+3^{i+2r+1}\sum_{k=0}^{r-1-i} a_k 3^k \quad \text{with some } z'<3^{2r+1}$$

$$\Rightarrow h=z''+a_{r-1-i}3^{3r} \quad \text{with some } z''<3^{3r}$$

$$\Rightarrow a_{r-1-i}=\max\{j\in\{0,1,2\}\,|\,j\cdot D_3(\tau_q(m))<h\}.$$

(Note that $h-a_{r-1-i}D_3(\tau_q(m))<3^{3r}$ and therefore for every $\beta\in\{0,1,2\}$ $\beta+3\cdot(h-a_{r-i-1}D_3(\tau_q(m))<2+3^{3r+1}<\tau_q(m)$.)

Two cases have to be distinguished

a) $\mu = l(m+1)$. In this case $r-1-i=(q-v-1)\cdot l(m+1)$ and therefore $a_{r-1-i}=2$. If the carry stored in the finite memory is still 1 then $b(l(m+1), n_{q-v}+\eta_{q-v})=1$ and $n_{q-v}+\eta_{q-v}\geqq 2^{l(m+1)}$ and A2 uses its STOP exit. Otherwise the finite memory has $\beta=2$ as its output and the instruction $h\leftarrow h-\alpha\cdot D_3(n)$, $h\leftarrow\beta+3h$ yield

$$h=P_{v+1}(n_{q-v}+\eta_{q-v},\dots,n_q+\eta_q,m+1)+3^{i+1}\sum_{k=0}^{r-1-i} a_k D_3^{r-k}(\tau_q(m)).$$

b) $\mu<l(m+1)$. In this case $a_{r-1-i}\in\{0,1\}$. The finite control computes β as the binary addition of a_{r-1-i} and the carry. Since $a_{r-1-i}=b(\mu,n_{q-v})$ it is clear that $\beta=b(\mu,n_{q-v}+\eta_{q-v})$. It is obvious that $h\leftarrow h-\alpha\cdot D_3(u)$, $h\leftarrow\beta+3h$ generate the correct head position.

Therefore the upper result is proved.

Because of this result M reaches the head position

$$h=P_q(n_1+\eta_1,\dots,n_p+\eta_p,i-1+\eta_{p+1},m+2-i-\eta_{p+1},m+1)$$

if $n_v+\eta_v<2^{l(m+1)}$ for all $v=1,\dots,p$.

Therefore for any two configurations $\mathscr{K}_1,\mathscr{K}_2$ such that $\mathscr{K}_1\xrightarrow[M]{}\mathscr{K}_2$ and $\kappa(\mathscr{K}_1)$ and $\kappa(\mathscr{K}_2)$ are defined $\kappa(\mathscr{K}_1)\xrightarrow[\tilde M]{*}\kappa(\mathscr{K}_2)$ holds. Note that for every $\dashv 1^n\vdash\in L$ there exists a sequence $\mathscr{K}_i$, $i=0,\dots,k$, of configurations such that $\mathscr{K}_0=(s_0,\dashv 1^n\vdash, 1,0,\dots,0)$, $\mathscr{K}_i\xrightarrow[M]{}\mathscr{K}_{i+1}$ and $\kappa(\mathscr{K}_i)$ is defined for all $i=0,\dots,k-1$ and $\mathscr{K}_k=(\mathrm{t},\dots)$ with some $t\in F$. Therefore $\kappa(\mathscr{K}_0)\xrightarrow[\tilde M]{*}\kappa(\mathscr{K}_k)$.

We end this proof by constructing $\tilde M$ in such a way that

$$\begin{aligned}(s_0,\dashv 1^{\tau_q(m)}\vdash,1,0)&\xrightarrow[\tilde M]{*}(s_0,\dashv 1^{\tau_q(m)}\vdash,P_q(0,\dots,0,m+1,m+1),0)\\ &=\kappa(s_0,\dashv 1^m\vdash,1,0,\dots,0)\end{aligned}$$

holds for all $m\in\mathbb{N}$.

Note that $\tau_q(m)=3^{r_q(m)}\cdot P_q(0,\dots,0,m+1,m+1)+3^{3r_q(m)+1}$.

$\tilde M$ operates in the following way (Algorithm A3):

1. $h\leftarrow\max\{3^j\mid 3^j\leqq\tau_q(m)\}$, $h\leftarrow\tau_q(m)-h$ (afterwards $h=3^{r_q(m)}\cdot P_q(0,\dots,0,m+1,m+1)$).
2. While $R_3(h)=0$ do $h\leftarrow D_3(h)$.

After performing these operations $\tilde M$ reaches the head position $h=P_q(0,\dots,0,m+1,m+1)$ and therefore the configuration $\kappa(s_0,\dashv 1^m\vdash,1,0,\dots,0)$.

We have constructed our automaton $\tilde M$ in such a way that for every $m\in\mathbb{N}$ and for every configuration $\mathscr{K}$ of M

$$(s_0,\dashv 1^m\vdash,1,0,\dots,0)\xrightarrow[M]{*}\mathscr{K}\Leftrightarrow(s_0,\dashv 1^{\tau_q(m)}\vdash,1,0)\xrightarrow[\tilde M]{*}\kappa(\mathscr{K}).$$

Therefore $\dashv 1^m\vdash\in L$ if and only if $\dashv 1^{\tau_q(m)}\vdash$ is accepted by $\tilde M$. Let L_2 be the language accepted by $\tilde M$. We still have to show that $L_2\in\mathscr{C}_1^1$.

In order to see this consider an arbitrary number $n\in\mathbb{N}$ and an arbitrary configuration $\mathscr{K}=(s,\dashv 1^n\vdash,z_1,z_2)$ of $\tilde M$ with $(s_0,\dashv 1^n\vdash,1,0)\xrightarrow[\tilde M]{*}\mathscr{K}$. Then $z_1,z_2\leqq n$. This is realized in the following way:

a) It is clear that during the execution of algorithm A3 the head position h is bounded by n and therefore the same holds for the position of the counter (the counter is used during the whole computation of $\tilde{M}$ only in order to store intermediate results).

b) Consider one run of A1 and suppose that at the beginning of this run $h \leqq n$. Then

$$D_3(h)+R_3(h)\cdot D_3(n) \leqq \frac{n}{3}+2\cdot\frac{n}{3} \leqq n.$$

c) During the computation of A2 it is examined in every run whether the head position exceeds n. (Remember that this can hold only if $n \notin \tau_q(\mathbb{N})$.) □

3. Register Machines

Definition. A nondeterministic k-register machine is a finite program consisting of instructions of the type $X_i \leftarrow X_i+1$, $X_i \leftarrow X_i \dot{-} 1$, Goto m if $X_i=0$, Goto m_1 or Goto m_2, where $i,j \in \{1,\dots,k\}$ and $m, m_1, m_2 \in \mathbb{N}$.

A k-register machine M defines in a natural way (see f.e. [6]) a partial relation ρ_M on $\mathbb{N}$. M is called deterministic if an instruction of the type "Goto m_1 or Goto m_2" does not occur in the program. In this case ρ_M can be identified with a function $f_M\colon \mathbb{N} \to \mathbb{N}$.

Definition. Let M be a k-register machine, let $L \subset \dashv\{1\}^*\vdash$ and let $\lambda\colon \mathbb{N} \to \mathbb{N}$ be some function.

M accepts L iff

$$\dashv 1^n \vdash \in L \Leftrightarrow \exists m_2,\dots,m_k \in \mathbb{N}\colon (n,0,\dots,0)\,\rho_M(0,m_2,\dots,m_k)$$

M accepts L with register length $\lambda(n)$ iff

a) M accepts L.

b) For every $\dashv 1^n \vdash \in L$ there exists a computation of M starting with $(n,0,\dots,0)$ and ending with some $(0,m_2,\dots,m_k)$, $m_2,\dots,m_k \in \mathbb{N}$, such that during the whole computation the contents of the registers are bounded by $\lambda(n)$.

Definition. k-REG $(\lambda(n))$ and k-NREG $(\lambda(n))$ are the classes of one-letter languages acceptable with the register length $\lambda(n)$ by deterministic and nondeterministic k-register machines.

Because of Lemma 2 it is clear that

$$\mathrm{TAPE}^1(\log n)=\bigcup_{k\in\mathbb{N}} k\text{-REG}(n), \qquad \mathrm{NTAPE}^1(\log n)=\bigcup_{k\in\mathbb{N}} k\text{-NREG}(n).$$

We will show that for every $k \in \mathbb{N}$

$$(k+1)\text{-NREG}(n) \subset \mathscr{C}_k^1 \subset (k+2)\text{-NREG}(n).$$

It is clear that $(k+1)\text{-NREG}(n) \subset \mathscr{C}_k^1$. In order to simulate a two-way counter automaton by a register machine we have to consider the fact that a register machine treats all its registers as counters whereas a counter automaton has the possibility to ask whether the position of its input head is equal to the input number during the whole computation. Obviously this test can be performed also by a register machine which stores on an additional register the distance between head position and right endmarker.

Theorem 1 states $\mathrm{NTAPE}(n)=\mathrm{TAPE}(n)\Leftrightarrow\mathscr{C}_1^1\subset\mathrm{TAPE}(\log n)$.

In this context it is very interesting to know how much additional power can be gained in the case of a nondeterministic 2-register machine if the equality between the content of a register and the input number can be tested. Theorem 1 implies $\mathrm{NTAPE}(n)=\mathrm{TAPE}(n)\Leftrightarrow 3\text{-}\mathrm{NREG}(n)\subset\mathrm{TAPE}(\log n)$. It is an open problem whether an analogous result holds for 2-register machines.

References

1. Book, R.V.: On the structure of complexity classes. Automata, languages and programming, 2nd Colloquium, pp. 437–445 (1974)
2. Fischer, P.C., Meyer, A.R., Rosenberg, A.L.: Counter machines and counter languages. Math. Systems Theory **3**, 265–283 (1968)
3. Galil, Z.: Two way deterministic pushdown automaton languages and some open problems in the theory of computation. 15th Ann. Symp. Switch. Autom. Theory, pp. 170–177, 1974
4. Monien, B.: Transformational methods and their application to complexity problems. Acta Informat. **6**, 95–108 (1976)
5. Savitch, W.J.: A note on multihead automata and context-sensitive languages. Acta Informat. **2**, 249–252 (1973)
6. Schnorr, C.P.: Rekursive Funktionen und ihre Komplexität. Stuttgart: Teubner 1974
7. Sudborough, I.H.: A note on tape-bounded complexity classes and linear context-free languages. J. Assoc. Comput. Mach. **22**, 499–500 (1975)
8. Sudborough, I.H.: On tape-bounded complexity classes and multihead finite automata. J. Comput. System Sci. **10**, 62–76 (1975)

Received August 20, 1976