Acta Informatica 6, 95—108 (1976)
© by Springer-Verlag 1976

Transformational Methods and their Application to Complexity Problems

Burkhard Monien

Received February 24, 1975

Summary. The following results are proved by the use of transformabilities.

1. NTAPE $(\log n) = \text{TAPE} (\log n) \Leftrightarrow$ There exists a j such that every language accepted by a nondeterministic one-way one-counter automaton is contained in D_j. (D_j is the family of all languages accepted by deterministic j-head two-way finite automata.)

2. NTAPE $(n) = \text{TAPE} (n) \Leftrightarrow$ There exists a j such that every language $L \subset \{1\}^*$ accepted by a nondeterministic 5-head two-way finite automaton is contained in D_j.

3. $\bigcup_d \text{TIME} (n^d) = \text{TAPE} (\log n) \Leftrightarrow$ There exists a j such that every language accepted by a deterministic 1-head two-way pushdown automaton is contained in D_j.

4. $\bigcup_d \text{TIME} (d^n) = \text{TAPE} (n) \Leftrightarrow$ There exists a j such that every language $L \subset \{1\}^*$ accepted by a deterministic 1-head two-way pushdown automaton is contained in D_j.

5. $D_j \subsetneqq D_{j+1}$ for all $j \in \mathbb{N}$.

1. Introduction

In this paper we study the relationships between deterministic and nondeterministic tape bounded Turing machines and between deterministic time bounded and deterministic tape bounded Turing machines. It is known that TAPE $(f(n)) \subset$ NTAPE $(f(n)) \subset$ TAPE $(f(n)^2)$ and that TIME $(f(n)) \subset$ TAPE $(f(n)) \subset \bigcup_d$ TIME $(d^{f(n)})$. It is an open problem whether in any of these cases equality holds. We show that these well known problems can be reduced to some simple looking problems concerning multihead two-way pushdown automata and multihead two-way finite automata. Especially we show that NTAPE $(\log n) \subset$ TAPE $(f(n))$ holds if and only if each language accepted by a nondeterministic one-way one-counter automaton is contained in TAPE $(f(n))$. Therefore the relationship between nondeterministic and deterministic tape complexities is given by the deterministic tape complexity of this subclass of the context-free languages, and the result of W. J. Savitch [9] follows because all context-free languages can be accepted with deterministic tape bound $(\log n)^2$ [6]. Furthermore we prove a new hierarchy result for deterministic multihead two-way finite automata.

In all these proofs we apply the same method. We use the notion of many-one reducibility as it is defined in recursive function theory (due to D. Knuth [5] we will speak of transformability). We get our results by showing that the classes, which we have to consider, are transformable with respect to restricted transformabilities to some subclasses or that they are closed with respect to these restricted transformabilities, respectively. This method was used implicitely by J. Hartmanis in [3] and explicitely by R. V. Book in [1].

Definition. Let $\mathscr{C}$ be a class of functions (on strings).

(i) Let $f:\Sigma^*\to\Gamma^*$ be a function in $\mathscr{C}$. A set $L_1\subset\Sigma^*$ is *f-transformable* to $L_2\subset\Gamma^*$ if for every $w\in\Sigma^*$, $w\in L_1$ if and only if $f(w)\in L_2$.

(ii) A class $\mathscr{L}_1$ of sets is *$\mathscr{C}$-transformable* to a class $\mathscr{L}_2$ of sets if for every $L_1\in\mathscr{L}_1$ there exist $L_2\in\mathscr{L}_2$ and $f\in\mathscr{C}$ such that L_1 is f-transformable to L_2.

(iii) Let $\mathscr{L}$ be a class of sets. A set L_0 is *$\mathscr{C}$-complete for $\mathscr{L}$* if $\mathscr{L}$ is $\mathscr{C}$-transformable to $\{L_0\}$. (Note that we don't demand $L_0\in\mathscr{L}$)

(iv) A class $\mathscr{L}$ of sets is *closed under $\mathscr{C}$-transformabilities* if for every set L_1, L_1 is $\mathscr{C}$-transformable to some set $L_2\in\mathscr{L}$ implies $L_1\in\mathscr{L}$.

The following lemma can be proved easily.

Lemma 1. Let A_i, B_i, $i\in\mathbb{N}$, be classes of sets such that $A_i\subset A_{i+1}$, $B_i\subset B_{i+1}$ hold for all $i\in\mathbb{N}$ and set $A:=\bigcup_{i\in\mathbb{N}}A_i$, $B:=\bigcup_{i\in\mathbb{N}}B_i$. Let $\mathscr{C}$, $\mathscr{D}$ be classes of functions. Then the following holds.

(i) Let A be $\mathscr{C}$-transformable to A_i for some $i\in\mathbb{N}$ and let B be closed under $\mathscr{C}$-transformabilities. Then $A\subset B$ is equivalent to $A_i\subset B$.

(ii) Let A be $\mathscr{C}$-transformable to A_i for some $i\in\mathbb{N}$ and let A_i be $\mathscr{D}$-transformable to some set $L\in A$. Let B be closed under $\mathscr{C}$-transformabilities and let B_i be closed under $\mathscr{D}$-transformabilities for all $i\in\mathbb{N}$. Then $A\subset B$ is equivalent to $\exists j:A_i\subset B_j$.

Proof. (i) We have to show that $A_i\subset B$ implies $A\subset B$. For each $L\in A$ there exists a set $L_1\in A_i$ such that L is $\mathscr{C}$-transformable to $A_i\subset B$. B is closed under $\mathscr{C}$-transformabilities and therefore $A\subset B$.

(ii) Suppose $A_i\subset B$. Because of (i) this implies $A\subset B$. Since $L\in A$ and $A\subset B$ there exists a j such that $L\in B_j$. A_i is $\mathscr{D}$-transformable to L and B_j is closed under $\mathscr{D}$-transformabilities and therefore $A_i\subset B_j$. q.e.d.

Remark. Lemma 1. (ii) remaines true if we replace the condition "B_i is closed under $\mathscr{D}$-transformabilities for all $i\in\mathbb{N}$" by "For all $i\in\mathbb{N}$ and for every set L_1, L_1 is $\mathscr{D}$-transformable to some set $L_2\in B_i$ implies $L_1\in B_{2i}$".

In the classification of sets according to their complexity dynamic measures (as defined by the computations of multitape Turing machines) are used as well as the accepting power of certain types of automata (depending on the number of input heads and the storage structure).

Definition. Let $f:\mathbb{N}\to\mathbb{N}$ be some function.

$$\mathrm{TIME}\,(f(n)) := \left\{L \;\middle|\; \begin{array}{l} L \text{ is accepted by a deterministic Turing machine} \\ \text{which operates with time bound } f(n) \end{array}\right\}$$

$$\mathrm{TAPE}\,(f(n)) := \left\{L \;\middle|\; \begin{array}{l} L \text{ is accepted by a deterministic Turing machine} \\ \text{which operates with tape bound } f(n) \end{array}\right\}$$

$$\mathrm{NTAPE}\,(f(n)) := \left\{L \;\middle|\; \begin{array}{l} L \text{ is accepted by a nondeterministic Turing machine} \\ \text{which operates with tape bound } f(n) \end{array}\right\}$$

On the other hand we consider automata consisting of a finite control and an input tape where k heads may move independently in both directions (k-head two-way finite automata). The input is placed between two endmarkers ($\dashv$ and $\vdash$). The automaton starts in a distinguished starting state with its k heads on the left endmarker. It accepts the input string if it stops in an accepting state. The automaton is called deterministic if its next move function is deterministic, otherwise it is called nondeterministic. Let $D_k(N_k)$, $k \in \mathbb{N}$, be the class of all sets accepted by deterministic (nondeterministic) k-head two-way finite automata.

Furthermore let P_k, $k \in \mathbb{N}$, be the class of all sets accepted by deterministic k-head two-way pushdown automata, where a k-head two-way pushdown automaton consists of a finite control, an input tape where k heads may move independently in both directions and a pushdown tape. Let C be the class of all languages accepted by nondeterministic one-way 1-counter automata. Such an automaton has a counter instead of a pushdown tape and only one head. This head cannot move to the left. It is not difficult to see that $C \subset N_2$, because every string accepted by such a counter automaton can be accepted also by a sequence of moves such that the numbers stored by the counter are always lineary bounded by the length of the input.

Because of S. A. Cook's Theorem [2] and some simple considerations (see for example [3]) the following lemma holds.

Lemma 2.
$$\mathrm{TAPE}\,(\log n) = \bigcup_{k \in \mathbb{N}} D_k,$$
$$\mathrm{NTAPE}\,(\log n) = \bigcup_{k \in \mathbb{N}} N_k,$$
$$\bigcup_{d \in \mathbb{N}} \mathrm{TIME}\,(n^d) = \bigcup_{k \in \mathbb{N}} P_k.$$

In [11] W. J. Savitch uses the idea of encoding an input string in unary notation (he defines a mapping $f: \Sigma^* \to \{1\}^*$ and shows that $L \in \mathrm{TAPE}\,(n)$ is equivalent to $f(L) \in \mathrm{TAPE}\,(\log n) \cap \{1\}^*$) to prove that the LBA-problem (that is the problem whether NTAPE (n) equals TAPE (n)) is equivalent to the question whether every computation of a $\log n$-tape bounded nondeterministic Turing machine on an unary input string can be simulated by a deterministic $\log n$-tape bounded Turing machine.

The same method can be applied to problems concerning time bounded computations. Denoting by $D_k^1(N_k^1, P_k^1)$, $k \in \mathbb{N}$, the class of all subsets of $\{1\}^*$ that are accepted by k-head two-way deterministic finite automata (nondeterministic finite automata, deterministic pushdown automata) we get the following lemma.

Lemma 3.
$$\mathrm{TAPE}\,(n) = \mathrm{NTAPE}\,(n) \Leftrightarrow \bigcup_{k \in \mathbb{N}} D_k^1 = \bigcup_{k \in \mathbb{N}} N_k^1$$
$$\mathrm{TAPE}\,(n) = \bigcup_{d} \mathrm{TIME}\,(d^n) \Leftrightarrow \bigcup_{k \in \mathbb{N}} D_k^1 = \bigcup_{k \in \mathbb{N}} P_k^1.$$

Now we will give a short survey of this paper.

In Section 2 we define some classes of transformabilities, and we prove by means of Lemma 1 (i), Lemma 2 and Lemma 3 that the following holds: NTAPE $(\log n) = \mathrm{TAPE}\,(\log n) \Leftrightarrow C \subset \mathrm{TAPE}\,(\log n)$, NTAPE $(n) = \mathrm{TAPE}\,(n) \Leftrightarrow N_5^1 \subset$ TAPE

$(\log n)$, $\bigcup_d \text{TIME}(n^d) = \text{TAPE}(\log n) \Leftrightarrow P_1 \subset \text{TAPE}(\log n)$, $\bigcup_d \text{TIME}(d^n) = \text{TAPE}$ $(n) \Leftrightarrow P_1^1 \subset \text{TAPE} \log n)$. In Section 3 we improve these results by means of Lemma 1 (ii).

In Section 4 we show that $D_j \subsetneqq D_{j+1}$ holds for all $j \in \mathbb{N}$. This improves a result of O. H. Ibarra [4], stating $D_j \subsetneqq D_{j+2}$. In Section 5 we discuss some further implications of our results.

2. The Application of Transformabilities

In Section 1 we reduced the problems concerning complexity classes to some problems concerning multihead automata. Now we will show that we can formulate further equivalent problems which deal only with a restricted number of heads. The first result in this area was proved by J. Hartmanis [3], who showed that

$$\bigcup_{k \in \mathbb{N}} N_k \subset \bigcup_{k \in \mathbb{N}} D_k \text{ is equivalent to } N_3 \subset \bigcup_{k \in \mathbb{N}} D_k.$$

In order to apply the methods mentioned in section 1 we define some classes of functions.

Definition.

(i) $\Pi = \{f \mid f: \dashv\Sigma^*\vdash \to \dashv(\Sigma \cup \{1\})^*\vdash,\ 1 \notin \Sigma;\ \exists k \in \mathbb{N}: f(\dashv w \vdash) = \dashv w \underbrace{11 \ldots 1}_{l(\dashv w \vdash)^k} \vdash\ \forall w \in \Sigma^*\}$

(ii) $\Pi_1 = \{f \mid f: \dashv\{1\}^*\vdash \to \dashv\{1\}^*\vdash;\ \exists k \in \mathbb{N}: f(\dashv 1^r \vdash) = \dashv 1^{r^k} \vdash\ \forall r \in \mathbb{N} \cup \{0\}\}$

Theorem 1.

(1) $\bigcup_{k \in \mathbb{N}} P_k$ is Π-transformable to P_1

(2) $\bigcup_{k \in \mathbb{N}} P_k^1$ is Π_1-transformable to P_1^1

(3) $\bigcup_{k \in \mathbb{N}} N_k^1$ is Π_1-transformable to N_5^1.

Proof. The general strategy is the same in all cases. We restrict ourselves to case (1). Let $L \in \bigcup_{k \in \mathbb{N}} P_k$. Then there is a finite set Σ and a $k \in \mathbb{N}$ such that $L \in P_k \cap \Sigma^*$. Let $f_k \in \Pi$ be the function $f_k(\dashv w \vdash) = \dashv w \underbrace{11 \ldots 1}_{l(\dashv w \vdash)^k} \vdash$. We will show that $\tilde{L} = \{f_k(v) \mid v \in L\}$ is in P_1. Let M_k be a k-head two-way pushdown automaton accepting L. We define a 1-head two-way pushdown automaton M which simulates on the input string $\dashv w \underbrace{11 \ldots 1}_{l(\dashv w \vdash)^k} \vdash$ the moves performed by M_k on the input string $\dashv w \vdash$.

The positions $i_1, \ldots, i_k$ of the k heads of M_k are encoded by the head position i of M in the form $i = i_1 + i_2 n + \ldots + i_k n^{k-1}$, $n = l(\dashv w \vdash)$. We have to show that M is able to change its head position according to a move of M_k.

1. Proof of (1). Let $i = i_1 + i_2 n + \ldots + i_k n^{k-1}$ be the head position of M. By successive subtraction of n (n is given by the position of the rightmost symbol

of Σ) M reaches a configuration where i_1 is the head psition and $i_2+i_3n+\ldots+i_kn^{k-2}$ is written on the top of the pushdown tape. Then M reads the i_1-th input symbol and afterwards it generates the new head position $i=i_2+\ldots+i_kn^{k-2}+i_1n^{k-1}$ by successive addition of n^{k-1} (M has to compute n^{k-1} again each time it wants to add n^{k-1}). In this way, just by rotating $(i_1, i_2, .., i_k)$, M reads all input symbols which are scanned by the heads of M_k. Now M is able to simulate the next move of M_k. The head position of M is changed by computing again $i_1, i_2, .., i_k$ one after the other and changing them according to the next move of M_k.

2. Proof of (2). The situation is a little more complicated for the unary input because the number n is not given directly by the input string which is of the form $\dashv\underbrace{11\ldots1}_{r}\vdash$, $r\in\mathbb{N}$. Therefore M has to decide first whether there is a $n\in\mathbb{N}$ such that $r=n^k$ and has to compute this number n. With the methods which are used to simulate time bounded computations of Turing machines on pushdown automata ([2], [8]) it is easy to prove that for every function $f:\mathbb{N}\to\mathbb{N}$ the following holds:

If f is computable by a deterministic Turing machine in polynomial time where the input number is given in binary notation then f is computable by a deterministic 1-head two-way pushdown automaton operating on the unary notation of the input numbers.

Therefore M is able to compute the function

$$f(r)=\begin{cases} n, & \text{if } r=n^k \\ 0, & \text{if } \nexists n: r=n^k. \end{cases}$$

In order to simulate M_k the automaton M operates in the same way as it is described in 1.

3. Proof of (3). M needs more heads than the pushdown automaton in 2. because it has no additional storage to store intermediate results. Let $\dashv\underbrace{11\ldots1}_{r}\vdash$ be the input string. It is straightforward to show that M can compute the number n such that $r=n^k$ if such a number exists. Afterwards M compute n^{k-1}.

Now let n, n^{k-1}, $i_1+i_2n+\ldots+i_kn^{k-1}$ be the positions of head 1, head 2 and head 3. M has to compute $i_1, i_2, .., i_k$ one after the other. By successive subtraction of n (head 1 and head 5 are used alternately to store n) M puts its head 3 on position $i_2+i_3n+\ldots+i_kn^{k-2}$ and its head 4 on position i_1. Afterwards head 3 reaches the position $i_2+\ldots+i_kn^{k-2}+i_1n^{k-1}$ by successive addition of n^{k-1}. It is obvious how each move of M_k can be simulated by M. $\square$

In 1973 I. H. Sudborough [12] improved the result of Hartmanis and showed that $\bigcup_{k\in\mathbb{N}} N_k \subset \bigcup_{k\in\mathbb{N}} D_k$ is equivalent to $1-N_2 \subset \bigcup_{k\in\mathbb{N}} D_k$, where $1-N_k$, $k\in\mathbb{N}$, is the class of all languages accepted by nondeterministic k-head one-way finite automata (that means, the k heads are only allowed to move from the left to the right between the two endmarkers). We will show in this paper that it is also equivalent to consider the problem whether C is contained in $\bigcup_{k} D_k$.

This result looks similar to Sudborough's result but it seems that this result can't be proved by using his methods, and the fact that C is a subclass of the context-free languages may be useful to get further results. Furthermore we get our results by using only transformational methods whereas Sudborough uses Savitch's language of threadable mazes [10].

We use a class Π_2 of functions which is defined in the following way:

(i) Let Σ be an alphabet, let $\dashv$, $\vdash$ be elements not in Σ and let k be a natural number. Let $f_{\Sigma,k}: \dashv\Sigma^*\vdash \to ((\Sigma\cup\{\dashv,\vdash\}^k)^*$ be the following function. For all $m\in\mathbb{N}$ and all $a_i\in\Sigma$, $i=1,..,m$, $f_{\Sigma,k}(\dashv a_1 \dots a_m\vdash)=\alpha_0\alpha_1\dots\alpha_{n^k-1}$ where $n=m+2$ and $\alpha_j=(a_{i_1}, a_{i_2},\dots,a_{i_k})$ for $j=i_1+i_2n+\dots+i_kn^{k-1}$ with $0\le i_\nu\le n-1$ for all $\nu=1,..,k$. For the sake of simplicity we set $a_0=\dashv$ and $a_{n-1}=\vdash$.

As an example let us consider the case $k=3$ and $n=3$. Then $f_{\Sigma,3}(a_0a_1a_2)=\alpha_0\alpha_1\dots\alpha_{26}$, where

$$\begin{tikzcd}
\alpha_0=(a_0,a_0,a_0) \arrow[r] & \alpha_3=(a_0,a_1,a_0) \arrow[r] & \cdots \arrow[r] & \alpha_9=(a_0,a_0,a_1) \\
\alpha_1=(a_1,a_0,a_0) & \alpha_4=(a_1,a_1,a_0) & \cdots & \alpha_{10}=(a_1,a_0,a_1) \\
\alpha_2=(a_2,a_0,a_0) \arrow[uur] & \alpha_5=(a_2,a_1,a_0) \arrow[uur] & \cdots \arrow[uur] & \alpha_{11}=(a_2,a_0,a_1).
\end{tikzcd}$$

Note that $(\dashv,\dashv,\dots,\dashv)$ and $(\vdash,\vdash,\dots,\vdash)$ enclose the new string and don't occur inside, therefore they can be regarded as endmarkers.

(ii) Let d be a natural number and let

$$g_{\Sigma,k,d}: \dashv\Sigma^*\vdash \to \dashv((\Sigma\cup\{\dashv,\vdash\})^k)^*\vdash$$

be defined by

$$g_{\Sigma,k,d}(\dashv w\vdash)=\dashv(f_{\Sigma,k}(\dashv w\vdash)\cdot f_{\Sigma,k}(\dashv w\vdash)^R)^{d\cdot l(\dashv w\vdash)^k}\vdash \quad \forall w\in\Sigma^*.$$

We denote by Π_2 the class of all functions which are defined in (ii).

Theorem 2. $\bigcup\limits_k N_k$ is Π_2-transformable to

$$\{L \mid \exists L_1\in C,\ L_2\in D_3, \text{ such that } L=L_1\cap L_2\}.$$

Proof. Let $L\subset\Sigma^*$ be an arbitrary element of N_k for some k. We will show that there exists a $d\in\mathbb{N}$ such that $g_{\Sigma,k,d}(L)=\tilde{L}\cap g_{\Sigma,k,d}(\dashv\Sigma^*\vdash)$ where $\tilde{L}\in C$. Therefore we first have to show that $g_{\Sigma,k,d}(\dashv\Sigma^*\vdash)\in D_3$. We define a deterministic 3-head automaton M which accepts this language. M needs only two heads in order to test whether the input string has the form $\dashv(\alpha v\beta\beta v^R\alpha)^{d\cdot l(\alpha v\beta)}\vdash$ with $\alpha=(\dashv,\dashv,..,\dashv)$, $\beta=(\vdash,\vdash,..,\vdash)$ and $v\in((\Sigma\cup\{\dashv,\vdash\})^k-\{\alpha,\beta\})^*$. Note that $l(f_{\Sigma,k}(\dashv w\vdash))=l(\dashv w\vdash)^k$. M has to test whether there exists a $w\in\Sigma^*$ such that $f_{\Sigma,k}(\dashv w\vdash)=\alpha v\beta=\alpha_0\dots\alpha_r$ with $\alpha_i\in(\Sigma\cup\{\dashv,\vdash\})^k$. M performs the following operations:

1. M checks whether there is a $n\in\mathbb{N}$ such that the symbols $\alpha_1,\dots,\alpha_{n-2}$ have the form $(a,\dashv,..,\dashv)$ with $a\in\Sigma$ and that $\alpha_{n-1}=(\vdash,\dashv,..,\dashv)$. Let a_i, $i=0,\dots,n-1$, be the first components of α_i.

2. If $\alpha_0,\dots,\alpha_r$ has the desired form then the rest of the string is determined by these n symbols. Let $j=i_1+i_2n+\dots+i_kn^{k-1}$ be some number and let us assume that M has already tested whether the symbols $\alpha_0,..,\alpha_j$ have the correct form. Let furthermore head 1 scan the j-th cell and head 2 scan the i_1-th cell of the input string. It is clear that in this case $\alpha_j=(a_{i_1},..,a_{i_k})$. Now M has to test whether α_{j+1} is the symbol determined by $a_1,..a_{n-2}$ and by j. If $i_1<n-1$, then

α_{j+1} must be the symbol $(a_{i_1+1}, a_{i_2}, .., a_{i_k})$ and this can be tested by means of α_j (the last $k-1$ components of α_j and α_{j+1} must be equal) and by means of head 2 which moves one cell to the right and then reads a_{i_1+1}.

Now consider the case $i_1 = n-1$. Let t be the smallest number such that $i_t \neq n-1$, this is just the smallest number t such that the t-th component of α_j is not equal to $\vdash$.

In this case $j = n-1 + (n-1)n + .. + (n-1)n^{t-2} + i_t n^{t-1} + .. + i_k n^{k-1}$ and therefore $j+1 = (i_t+1)n^{t-1} + i_{t+1}n^t + .. + i_k n^{k-1}$ and M has to compare whether α_{j+1} is the symbol $(\dashv, .., \dashv, a_{i_t+1}, a_{i_{t+1}}, \dots, a_{i_k})$. In order to do this M has to look for a_{i_t+1}. M puts its head 1 and head 2 on the $(j+1)$-st cell. Let us denote by λ the position of head 2. M starts the following algorithm which computes $(i_t+1)n^{t-1}$ with $\lambda = j+1$.

(i) If $\lambda \geqq n^t$ then goto (ii) else goto (iii). In order to test whether $\lambda \geqq n^t$, M moves its head 2 to the left and examines whether there is any symbol of the form $(\underbrace{\dashv, \dashv, .., \dashv}_{t}, a, \dots)$, $a \in \Sigma$, among the $\alpha_0, \dots, \alpha_\lambda$. Head 3 moves to the right each time head 2 moves to the left and therefore the head position λ can be generated again.

(ii) $\lambda = \lambda - n^t$. Head 3 moves to the n^t-th cell, this is the first cell storing a symbol of the form $(\dashv, \dashv, .., \dashv, \dots)$ with $a \in \Sigma$. Afterwards head 2 and head 3 are moving simultaneously to the left. Goto (i).

(iii) STOP. When this algorithm stops, then $\lambda = (i_t+1)n^{t-1}$ and therefore $\alpha_\lambda = (\underbrace{\dashv, .., \dashv}_{t-1}, a_{i_t+1}, \dots)$. Now M is able to test whether α_{j+1} is the correct symbol.

So we have proved that $g_{\Sigma,k,d}(\dashv\Sigma^*\vdash) \in D_3$ and we will now construct a nondeterministic 1-counter automaton $\tilde{M}$ which accepts an input string of the form $g_{\Sigma,k,d}(\dashv w \vdash)$ if and only if $\dashv w \vdash \in L$. We don't care about the behavior of $\tilde{M}$ on input strings which are not of this form.

Let M_k be a nondeterministic k-head two-way finite automaton accepting L. Then there exists a $d \in \mathbb{N}$ such that every computation of M on an input of length n needs at most $2 \cdot d \cdot n^k$ moves. In the following let d be this number. $\tilde{M}$ simulates on the input string $g_{\Sigma,k,d}(\dashv w \vdash)$ all the moves performed by M_k on the input string $\dashv w \vdash$. When $\tilde{M}$ is simulating the t-th step, $1 \leqq t \leqq 2 \cdot d \cdot n^k$, of M_k, then its head is located in the t-th block (a block is a string $f_{\Sigma,k}(\dashv w \vdash)$ or $f_{\Sigma,k}(\dashv w \vdash)^R$, respectively) of $g_{\Sigma,k,d}(\dashv w \vdash)$. Let $i_1, i_2, \dots, i_k$ be the positions of the k heads of M_k before M_k performs its t-th step. Then the head position of M is given by

$$i = i_1 + i_2 n + \dots + i_k n^{k-1} + t n^k \quad \text{if} \quad t \equiv 1 \bmod 2, \text{ and}$$
$$i = (t+1)n^k - (i_1 + i_2 n + \dots + i_k n^{k-1}) \quad \text{if} \quad t \equiv 0 \bmod 2.$$

Note that therefore the i-th symbol of $g_{\Sigma,k,d}(\dashv w \vdash)$ is just $(a_{i_1}, a_{i_2}, \dots, a_{i_k})$, when $\dashv w \vdash = a_0 \dots a_{n-1}$. That means that $\tilde{M}$ reads with its one head just the symbols read by the k heads of M_k and so $\tilde{M}$ has all the information necessary to simulate the next move of M_k.

Let us suppose that $\tilde{M}$ is simulating the t-th step of M_k and let i be the head-position of $\tilde{M}$. Then $\tilde{M}$ has to move its head to the position $i' = t \cdot n^k + (t \cdot n^k - \tilde{i})$, where $\tilde{i} = i + \sum_{j=0}^{k-1} \beta_j n^j$ and $\beta_j \in \{-1, 0, +1\}$, $j = 1, \ldots, k$, are determined by the moves performed by the k heads of M_k in its t-th step.

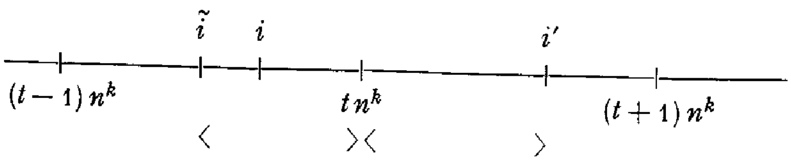

Fig. 1

$\tilde{M}$ performs the following operations (note that the head of $\tilde{M}$ is not allowed to move to the left).

Its head moves to the right and the number of these moves is stored by its counter. During this process $\tilde{M}$ subtracts all numbers n^j such that $\beta_j = +1$. That means, if $\tilde{M}$ reads a symbol of the form $(\underbrace{\dashv, \dashv, \ldots, \dashv}_{m}, a, \ldots)$, $a \in \Sigma$, on the input tape, it looks for the greatest $j \leq m$ such that it still has to subtract n^j. Then $\tilde{M}$ moves to the next cell containing a symbol of the form $(\underbrace{\dashv, \dashv, \ldots, \dashv}_{j}, a, \ldots)$, $a \in \Sigma$. The distance of these two cells is just n^j. During these moves the counter remains unchanged. Therefore the counter stores the number $(t \cdot n^k - i) - \sum_{j, \beta_j = 1} n^j$ when the head reaches the $(t \cdot n^k)$-th cell.

Now $\tilde{M}$ adds to its headposition all numbers n^j such that $\beta_j = -1$ one after the other beginning with the largest one. Again the movement of the head is controlled by symbols of the form $(\dashv, \dashv, \ldots, \dashv, a, \ldots)$, $a \in \Sigma$. During this process the head moves to the position $t \cdot n^k + \sum_{j, \beta_j = -1} n^j$. Now $\tilde{M}$ reaches the head position i' by adding the contents of the counter.

$\tilde{M}$ accepts the input string when it notices that M_k reaches a final state. Therefore $\tilde{M}$ accepts $g_{\Sigma,k,d}(\dashv w \vdash)$ if and only if M_k accepts $\dashv w \vdash$. Let $\tilde{L}$ be the language accepted by $\tilde{M}$, then $g_{\Sigma,k,d}(L) = \tilde{L} \cap g_{\Sigma,k,d}(\dashv \Sigma^* \vdash)$. □

It is not difficult to see that the class TAPE $(\log n)$ is closed under all the transformabilities defined in this section.

Theorem 3. TAPE $(\log n)$ is closed under Π, Π_1, Π_2-transformabilities.

Proof. We have to show that $L \in$ TAPE $(\log n)$ implies

$$f^{-1}(L) = \{w \mid f(w) \in L\} \in \text{TAPE } (\log n) \quad \text{for all } f \in \Pi \cup \Pi_1 \cup \Pi_2.$$

Let M be some deterministic $\log n$-tape bounded Turing machine accepting L. We will define a Turing machine $\tilde{M}$ accepting $f^{-1}(L)$. $\tilde{M}$ simulates on the input w all moves performed by M on the input $f(w)$. There exists a k such that $l(f(w)) = l(w)^k$ and we may assume that $\tilde{M}$ has k read-only heads moving on its input tape. Whenever the position of the input head of M is $i = i_1 + i_2 \cdot l(w) + \ldots + i_k \cdot l(w)^{k-1}$, then the position of the j-th head of $\tilde{M}$, $1 \leq j \leq k$, is i_j. Since f is a function belonging to $\Pi \cup \Pi_1 \cup \Pi_2$, the i-th symbol of $f(w)$ is determined uniquely by the i_1-th, $\ldots$, i_k-th symbols of w. Furthermore $\tilde{M}$ needs not more than $\log l(f(w)) = k \cdot \log l(w)$ cells to store in each step of the simulation the contents of the working tape of M. Therefore $\tilde{M}$ can simulate M step by step and it is clear that $\tilde{M}$ can be simulated by a normal $\log n$-tape bounded machine with only one input head. □

Because of our basic lemma (Lemma 1) the results of this section imply together with Lemma 2 and Lemma 3 the following theorem.

Theorem 4.

$$\begin{aligned}
&\mathrm{NTAPE}\ (\log n) = \mathrm{TAPE}\ (\log n) \Leftrightarrow C \subset \mathrm{TAPE}\ (\log n)\\
&\mathrm{NTAPE}\ (n) = \mathrm{TAPE}\ (n) \Leftrightarrow N_5^1 \subset \mathrm{TAPE}\ (\log n)\\
&\bigcup_{d\in\mathbb{N}} \mathrm{TIME}\ (n^d) = \mathrm{TAPE}\ (\log n) \Leftrightarrow P_1 \subset \mathrm{TAPE}\ (\log n)\\
&\bigcup_{d\in\mathbb{N}} \mathrm{TIME}\ (d^n) = \mathrm{TAPE}\ (n) \Leftrightarrow P_1^1 \subset \mathrm{TAPE}\ (\log n).
\end{aligned}$$

3. Complete Languages

The results of Section 2 might help to find a solution of one of these important problems if equality holds. Now we will prove some results which might help to prove inequality. In order to do this we use the notion of a complete language (see Section 1). We define the following two classes of functions: F_1 is the class of all word-homomorphisms. F_2 is the class of all mappings $f_r: \dashv\{1\}^*\vdash \to \dashv\{1\}^*\vdash$, $r\in\mathbb{N}$, defined by $f_r(\dashv 1^n \vdash) = \dashv 1^{(n+r)^2+n}\vdash\ \forall n\in\mathbb{N}$.

Let now Σ be some alphabet, $\{0, 1\}\notin\Sigma$, and set $\Sigma' = \Sigma\cup\{0, 1\}$. Let $h: (\Sigma')^*\to\Sigma^*$ be the homomorphism defined by $h(a) = a$ for all $a\in\Sigma$, $h(0) = h(1) = \varepsilon$. Let us consider the following language $L_0 \subset \dashv(\Sigma')^*\vdash$. $\dashv w\vdash \in L_0$ if and only if the following conditions are fulfilled:

1. $\exists\varphi\in\{0, 1\}^*$ such that $w\in(\Sigma\cdot\varphi)^*$
2. φ is the encoding of a nondeterministic one-counter automaton (we choose some fixed method to encode a onecounter machine by a 0-1-string).
3. Let M_φ be the one-counter machine whose encoding is φ. Then M_φ accepts the string $h(w)$.

Essentially L_0 is a universal language for all languages out of C. It can be proved with the usual methods that $L_0\in\mathrm{NTAPE}\ (\log n)$.

It is easy to see that L_0 is a language which is F_1-complete for C. For let $L\in C$, $L\subset\Sigma^*$, $(0, 1\notin\Sigma)$, be an arbitrary language, let M be a nondeterministic one-counter automaton accepting L, let φ be the encoding of M and let h_φ be the homomorphism defined by: $h_\varphi(a) = a\varphi$ for all $a\in\Sigma$. Then $w\in L$ holds if and only if $h_\varphi(w)\in L_0$.

In the same way we define a language $L_0'\in\bigcup_d \mathrm{TIME}\ (n^d)$ which is universal for P_1 and therefore F_1-complete for P_1.

Now we will define in a similar way a universal language L_1 for N_5^1. In order to apply Lemma 1 (ii) L_1 must be an element of $\bigcup_k N_k^1$ and therefore we have to encode each automaton by some natural number. We use the mapping $\alpha: \{1\}\cdot\{0, 1\}^* \to\{1\}^*$, $\alpha(\varphi) = 1^{un(\varphi)}$ for all $\varphi\in\{1\}\cdot\{0, 1\}^*$, where $un(\varphi)$ is the natural number whose binary notation is φ. Let M be any nondeterministic 5-head two-way finite automaton, then we define $gn(M) = \alpha(\varphi_M)$ where φ_M is the o-1-encoding of M.

L_1 is defined as the set of all strings $\dashv 1^m \vdash$ which fulfill the following conditions:

1. $\exists r, n \in \mathbb{N}$ such that $m = (n+r)^2 + n$. (Note that r, n are determined uniquely if they exist.)

2. There exists a nondeterministic 5-head two-way finite automaton M such that $gn(M) = r$.

3. M accepts $\dashv 1^n \vdash$.

L_1 is F_2-complete for N_5^1. For let L be an element of N_5^1, let M be a nondeterministic 5-head two-way finite automaton accepting L and set $r = gn(M)$. Then $\dashv 1^n \vdash \in L$ if and only if $\dashv 1^{(n+r)^2+n} \vdash \in L_1$. Furthermore it is not difficult to see that $L_1 \in \mathrm{TAPE}(\log n)$. The corresponding Turing machine M_1 performs on the input string $\dashv 1^m \vdash$ the following operations: It computes the greatest number p such that $p^2 < m$ and examines whether $n. = m - p^2 < p$. If this is true then $r = p - n$. The binary notations of n and r are stored on the working tape. Afterwards M_1 examines whether r is the encoding of some nondeterministic 5-head two-way finite automaton M and simulates M on the input string $\dashv 1^n \vdash$.

In the same way we define a language L_1' belonging to $\bigcup_k N_k^1$ which is F_2-complete for P_1^1. Now Lemma 1 (ii) leads to the following theorem:

Theorem 5.

$$1.\ \mathrm{NTAPE}(\log n) = \mathrm{TAPE}(\log n) \Leftrightarrow \exists_j : C \subset D_j$$
$$2.\ \mathrm{NTAPE}(n) = \mathrm{TAPE}(n) \Leftrightarrow \exists_j : N_5^1 \subset D_j$$
$$3.\ \bigcup_d \mathrm{TIME}(n^d) = \mathrm{TAPE}(\log n) \Leftrightarrow \exists_j : P_1 \subset D_j$$
$$4.\ \bigcup_d \mathrm{TIME}(d^n) = \mathrm{TAPE}(n) \Leftrightarrow \exists_j : P_1^1 \subset D_j.$$

Proof. Let us first prove the first relation. Let again $L_0 \in \mathrm{NTAPE}(\log n)$ be the language which is F_1-complete for C. It is easy to see that each D_j, $j \in \mathbb{N}$, is closed under F_1-transformabilities (that means, that D_j is closed under inverse homomorphism). Therefore Lemma 1 (ii) implies:

$$\mathrm{NTAPE}(\log n) = \mathrm{TAPE}(\log n) \Leftrightarrow \exists_j : C \subset D_j.$$

In the same way the third relation is proved. The proof of the relations 2 and 4 goes along the same lines. We only have to take into account that the D_j, $j \in \mathbb{N}$, are not closed under F_2-transformabilities but that $F_2^{-1}(D_j) \subset D_{2j}$ holds for all $j \in \mathbb{N}$. (If L is accepted by a j-head automaton then $L_r = \{\dashv 1^n \vdash \mid \dashv 1^{(n+r)^2+n} \vdash \in L\}$ is accepted by a $(2j)$-head automaton. The two heads are needed to simulate on an input of length n a head position of length n^2). Because of the remark to Lemma 1 the relations 2 and 4 follow. □

4. A Hierarchy Result

We will use now the method of transformabilities to prove a new hierarchy result for the D_j, $j \in \mathbb{N}$. O. H. Ibarra showed in [4] (also by using transformabilities) that $D_j \subsetneq D_{j+2}$ for all $j \in \mathbb{N}$.

We will show first that we can get this result by a direct diagonalization argument and that even languages of a special form belong to $D_{j+2}-D_j$:

Lemma 4. Let Σ be some alphabet which contains at least two symbols and let $f_{\Sigma,2}$ be the function defined in Section 2. Then to each $j\in\mathbb{N}$ there exists a language $L_j\subset f_{\Sigma,2}(\dashv\Sigma^*\vdash)$ with $L_j\in D_{j+2}-D_j$.

Proof. Set $\alpha=(\dashv,\dashv)$, $\beta=(\vdash,\vdash)$ and $\Sigma'=(\Sigma\cup\{\dashv,\vdash\})^2-\{\alpha,\beta\}$. Then $f_{\Sigma,2}(\dashv\Sigma^*\vdash)\subset\alpha(\Sigma')^*\beta$. Let $\tilde{h}:\Sigma\to\{0,1\}$ be some fixed surjective mapping and let $h:\Sigma^*\to\{0,1\}^*$ be the monoid homomorphism induced by $\tilde{h}$. We define a $(j+2)$-head automaton M which works on an input string $w\in\alpha(\Sigma')^*\beta$ in the following way:

1. M tests whether $w\in f_{\Sigma,2}(\dashv\Sigma^*\vdash)$. Because of the proof of Theorem 2 this can be done by using only 3 heads. M rejects w if $w\notin f_{\Sigma,2}(\dashv\Sigma^*\vdash)$.

2. Let us assume now that there exists a $v\in\Sigma^*$ such that $w=f_{\Sigma,2}(\dashv v\vdash)$. M tests whether $h(v)$ is the encoding of a deterministic j-head two-way finite automaton. An encoding of such an automaton is a list of 0-1-strings of the form $001^s01^{a_1}01^{a_2}\ldots01^{a_j}01^{s'}01^{\eta_1}0\ldots01^{\eta_j}00$ describing the change of the state and the head positions if the automaton is in the state s reading the symbols $a_1,\ldots,a_j$. $s',\eta_1,\ldots,\eta_k$ must be determined uniquely by $s,a_1,\ldots,a_j$. M needs two heads to perform this test.

3. Let M_v be the j-head automaton whose encoding is $h(v)$. M simulates M_v on the input w. During the simulation the position of the i-th head, $1\leqq i\leqq j$, of M is always equal to the position of the i-th head of M_v. M needs its two additional heads to simulate the next move function of M_v. Let s be the actual state of M_v and let $a_1,\ldots,a_j\in\Sigma'$ be the symbols read by the j heads. Then head $j+1$ of M is in the position s (we can assume that all states are natural numbers) and head $j+2$ looks for a substring of $h(v)$ of the form $001^s01^{a_1}0\ldots01^{a_j}01^{s'}01^{\eta_1}0\ldots01^{\eta_j}00$. If no such substring exists and if s is no final state of M_v then M_v does not accept w. In this case M accepts w. Otherwise the head positions are changed according to $\eta_1,\ldots,\eta_j$, s' and the next move of M_v (whose actual state is now s') is simulated. M accepts w if and only if M_v does not accept w.

Since $j+2\geqq3$ holds for all $j\in\mathbb{N}$. M needs only $j+2$ heads to perform the operations described in 1., 2., 3. and therefore $L_j\in D_{j+2}$. By the definition of L_j it is clear that $L_j\subset f_{\Sigma,2}(\dashv\Sigma^*\vdash)$. Furthermore let $L\in D_j$ be some language and let $v_0\in\{0,1\}^*$ be the encoding of a j-head automaton accepting L. Now consider some $v\in h^{-1}(v_0)$ and set $w=f_{\Sigma,2}(\dashv v\vdash)$. Because of 3. $w\in L$ holds if and only if $w\notin L_j$. Therefore $L\neq L_j$ for every $L\in D_j$ and this implies $L_j\notin D_j$. □

Theorem 6. Let $j\geqq4$ be an even natural number and let L_j, Σ' be defined as in the proof of Lemma 4. Then $f_{\Sigma',2}(L_j)\in D_{j/2+1}$.

Proof. Let Σ, Σ' be the same sets as in the proof of Lemma 4. L_j is accepted by a $(j+2)$-head automaton M which has the following property: Let $\alpha w\beta$, $w\in(\Sigma')^*$, be any input string, then the heads $j+1$, $j+2$ are used only if there exists a $v\in\dashv\Sigma^*\vdash$ such that $\alpha w\beta=f_{\Sigma,2}(\dashv v\vdash)$, and in this case these two heads move only on the initial string of $\alpha w\beta$ of length $l(v)=\sqrt{l(\alpha w\beta)}$. Note that $u\in f_{\Sigma',2}(L_j)$ implies that there exists a $v\in\Sigma^*$ such that $u=f_{\Sigma',2}(f_{\Sigma,2}(\dashv v\vdash))$.

We have to define a $(j/2+1)$-head automaton $\tilde{M}$ accepting $f_{\Sigma',2}(L_j)$. Let u be the input string. Then $\tilde{M}$ tests first whether there exists a $w\in(\Sigma')^*$ such that $u=f_{\Sigma,2}(\alpha w\beta)$. Note that $j/2+1\geqq 3$ and therefore $\tilde{M}$ can perform this test. Afterwards $\tilde{M}$ simulates on the input string $f_{\Sigma',2}(\alpha w\beta)$ all the moves performed by M on the input string $\alpha w\beta$. (Let $\alpha w\beta=a_0\dots a_{n-1}$ with $a_0=\alpha$, $a_{n-1}=\beta$ and $a_\nu\in\Sigma'$ for $1\leqq\nu\leqq n-2$.) Let $i_1,\dots,i_{j+2}$ be the head position of M after a number of steps. $\tilde{M}$ encodes these head positions by its head positions $h_1,\dots,h_{j/2+1}$ in the following way:

$$h_\nu=i_{2\nu-1}+i_{2\nu}\cdot n \quad\text{for}\quad \nu=1,\dots,j/2-1,$$

$$h_{j/2}=i_{j+1}+i_{j+2}\cdot\sqrt{n}+i_{j-1}\cdot n,\qquad h_{j/2+1}=i_j\cdot n.$$

Note that $i_{j+1}, i_{j+2}<\sqrt{n}$.

The symbol read by the ν-th head of $\tilde{M}$ is $(a_{i_{2\nu-1}}, a_{i_{2\nu}})$ if $1\leqq\nu<j/2$ and (α, a_{i_j}) if $\nu=j/2+1$. Furthermore we know that the heads $j+1$, $j+2$ are used only if there exists a $v\in\Sigma^*$ such that $w=f_{\Sigma,2}(\dashv v\vdash)$. Let $\dashv v\vdash$ be $b_0\dots b_{m-1}$, $m=\sqrt{n}$. Then $a_{r_1+\sqrt{n}\cdot r_2}=(b_{r_1}, b_{r_2})$ for all $0\leqq r_1, r_2\leqq m-1$, and therefore $\tilde{M}$ reads the symbol $((b_{i_{j+1}}, b_{i_{j+2}}), a_{i_{j-1}})$ with its $j/2$-th head whenever M reads the symbols $(b_{i_{j+1}},)$, $(b_{i_{j+2}},)$, $a_{i_{j-1}}$ with its heads $j+1$, $j+2$, $j-1$.

Therefore $\tilde{M}$ has all the information necessary to simulate the next move of M. In order to do so $\tilde{M}$ must be able to add (or subtract, respectively) n or $\sqrt{n}$. $\tilde{M}$ uses its head $(j/2+1)$ as a counter. Note that the position of head $(j/2+1)$ is always divisible by n. Therefore $\tilde{M}$ is able to count up to n ($\sqrt{n}$, respectively) by moving its head $(j/2+1)$ to the nearest cell containing a symbol of the form (α, a) with $a\in\Sigma'$ (or $((\dashv, b), a)$ with $b\in\Sigma$, $a\in\Sigma'$, respectively). Afterwards head $(j/2+1)$ can move back to its original position and therefore we can change all head positions one after the other. □

Theorem 7. Let Σ be an alphabet and $j\in\mathbb{N}$. Let $L\subset\dashv\Sigma^*\vdash$ be some language. Then $f_{\Sigma,2}(L)\in D_j$ implies $L\in D_{2j}$.

Proof. Let M be a j-head automaton accepting $f_{\Sigma,2}(L)$. Our $2j$-head automaton $\tilde{M}$ which accepts L encodes each position of M by the position of two of its heads.

Let $f_{\Sigma,2}(\dashv w\vdash)$, $\dashv w\vdash=a_0\dots a_{n-1}$, $a_\nu\in\Sigma$ for all $\nu=1,\dots,n-2$ be the input string of M and let $i=i_1+i_2\cdot n$, $0\leqq i_1, i_2\leqq n-1$, be the position of one of the heads of M during its computation. Then the position of the two corresponding heads of $\tilde{M}$ are i_1 and i_2. Therefore $\tilde{M}$ reads the symbols a_{i_1} and a_{i_2} whenever M reads (a_{i_1}, a_{i_2}) and so $\tilde{M}$ can simulate each move of M. □

Theorem 8. $D_j\subsetneqq D_{j+1}$ for all $j\in\mathbb{N}$.

Proof. D_1 is the class of the regular sets and therefore $\{a^nb^n\mid n\in\mathbb{N}\}\notin D_1$. It is obvious that $\{a^nb^n\mid n\in\mathbb{N}\}\in D_2$ and this implies $D_1\subsetneqq D_2$.

Now let us assume that $j\geqq 2$ and $D_j=D_{j+1}$. Let L_{2j} be the language defined in Lemma 4. Then $f_{\Sigma',2}(L_{2j})\in D_{j+1}=D_j$ holds because of Theorem 6 and therefore $L_{2j}\in D_{2j}$ because of Theorem 7. This is a contradiction to Lemma 4 stating that $L_{2j}\notin D_{2j}$. Thus we have proved that $D_j\neq D_{j+1}$. □

5. Related Results

In Section 2 we proved that NTAPE $(\log n) =$ TAPE $(\log n)$ is equivalent to $C \subset$ TAPE $(\log n)$. We got this result by showing that NTAPE $(\log n)$ is transformable to $\{L_1 \cap L_2 | L_1 \in C, L_2 \in D_3\}$ under our class Π_2 of transformabilities and that TAPE $(\log n)$ is closed under Π_2-transformabilities. Therefore it is clear that NTAPE $(\log n) \subset H$ is equivalent to $C \subset H$ whenever H is closed under Π_2-transformabilities and $D_3 \subset H$. We get the following theorem:

Theorem 9. Let α be some rational number. Then NTAPE $(\log n) \subset$ TAPE $((\log n)^\alpha)$ is equivalent to $C \subset$ TAPE $((\log n)^\alpha)$.

From this result we see that Savitch's theorem [9] stating NTAPE $(\log n) \subset$ TAPE $((\log n)^2)$ is an evident consequence of the theorem given by P. M. Lewis, R. E. Stearns and J. Hartmanis [6] stating that the class of context-free languages is contained in TAPE $((\log n)^2)$. Furthermore if we could improve this result we would get a better bound for the simulation of deterministic Turing machines by nondeterministic ones.

On the other hand even if the upper result is optimal there might be a better bound for the simulation of nondeterministic machines by deterministic ones because we have to consider only the deterministic tape complexity of the one-counter languages and this class is only a subclass of the context-free languages.

We will show further that the following result of the author [7] can be proved using only these simple transformability methods.

Theorem 10. Let $p \in \mathbb{N}$ and let L be any language accepted by some nondeterministic two-way multihead pushdown automaton which performs on every input of length n not more than n^p steps. Then $L \in$ TAPE $((\log n)^2)$.

We have to realize only that the language $\tilde{L} = \{(f_{\Sigma,k}(w) \cdot f_{\Sigma,k}(w)^R)^{l(w)^p} | w \in L\}$, where $L \subset \Sigma^*$ is accepted by a k-head automaton, is accepted by a nondeterministic one-way 1-head pushdown automaton. This can be seen as in Theorem 2. We need no additional counter in this case because the automaton has a pushdown tape which can operate like a counter. $\tilde{L}$ is a context-free language and therefore Theorem 10 follows because of the result of Lewis, Stearns and Hartmanis [6].

References

1. Book, R. V.: On the structure of complexity classes. In: Nivat, M. (ed.): 2nd Colloquium on Automata, Languages and Programming 1974, p. 437–445
2. Cook, S. A.: Characterizations of pushdown machines in terms of time bounded computers. J. ACM **18**, 4–18 (1971)
3. Hartmanis, J.: On Non-determinacy in simple computing devices. Acta Informatica **1**, 336–344 (1972)
4. Ibarra, O. H.: On two-way multihead automata. J. Computer and System Sciences **7**, 28–37 (1973)
5. Knuth, D. E.: Postscript about NP-hard problems. SIGACT News 6: **2**, 15–16 (1974)
6. Lewis, P. M., Stearns, R. E., Hartmanis, J.: Memory bounds for recognition of context-free and context-sensitive languages. IEEE Conf. Rec. 7-th Ann. Symp. Switch. Cir. Th. Log. Des. 1965, p. 191–202
7. Monien, B.: Relationship between pushdown automata and tape—bounded Turing machines. 1st Colloquium on Automata, Languages and Programming 1972, p. 575–583

8. Monien, B.: Characterizations of time-bounded computations by limited primitive recursion. 2nd Colloquium, Automata, Languages and Programming 1974, p. 280–293
9. Savitch, W. J.: Relationships between nondeterministic and deterministic tape complexity. J. Computer and System Sciences **4**, 177–192 (1970)
10. Savitch, W. J.: Maze recognizing automata and nondeterministic tape complexity. J. Computer and System Sciences **7**, 389–403 (1973)
11. Savitch, W. J.: A note on multihead automata and context-sensitive languages. Acta Informatica **2**, 249–252 (1973)
12. Sudborough, I. H.: On tape-bounded complexity classes and multi-head finite automata. IEEE Conf. Rec. 14-th Ann. Symp. Switch. Aut. Th. 1973, p. 138–144

Prof. Dr. Burkhard Monien
Universität Dortmund
Abteilung Informatik
D-4600 Dortmund-Hombruch
Postfach 500
Bundesrepublik Deutschland