Theoretical Computer Science 53 (1987) 307-317
North-Holland

CYCLIC AUTOMATA*

Max GARZON **
Department of Mathematics, University of Illinois at Urbana, IL 61801, U.S.A.

Communicated by M. Nivat
Received February 1986

Abstract. A cyclic graph language is accepted by a Cayley automaton [2] with pebbles if and only if their orders can be recognized in logspace by an ordinary Turing machine. There is an infinite hierarchy of cyclic graph languages and although most decision problems are decidable for 1-pebble cyclic automata, they are recursively unsolvable for 2-pebble automata.

1. Introduction

Turing's idea of a finite control capable of assuming any of a finite set of specified internal states upon reading consecutive symbols of an input string has been one of the ideas truly basic to the theoretical foundations of the theory of computation. The ordinary model of computation by a Turing machine usually involves a doubly infinite tape divided into identical cells on which symbols from a given alphabet can be written or rewritten by the finite control of some computational device. Two fundamental properties of the tape are hidden behind the contents of the string written on it. First, it is possible to map any cell onto any other cell by means of a geometric translation. Second, the possible directions of motion at any cell of the tape exactly correspond to a finite set of possibilities labelled 'left' and 'right' which are perfectly interchangeable. In other words, an observer standing on a blank tape will be unable to distinguish one cell from another on the sole basis of their local appearance or relative position: the tape is homogeneous and isotropic. Therefore, an attempt to generalize the notion of a tape should lead to an object which, while preserving these properties, allows, on the other hand, the flexibility of interpretation to realize very general kinds of inputs. As it turns out, any graph with the above two properties of a tape must in fact be the Cayley graph of a suitable group (cf. [6, p. 63]). Thus it is plausible that a systematic study of sequential computation on more general tapes may lead to a better understanding of some current unsolved problems in the classical theory of computation.

* This is an excerpt of the author's doctoral dissertation at the University Illinois, Urbana, U.S.A.

** Present affiliation: Department of Mathematical Sciences, Memphis State University, Memphis, TN 38152, U.S.A.

In the recognition of ordinary strings the input implicitly defines an 'origin' (the first symbol in the string) and a last cell (the last symbol in the string or an end-marker). Therefore, any computational device on Cayley graphs must be able to detect or create some special marking on at least one vertex of its input for otherwise it will be lost in the homogeneity of its input graph. The most primitive calculation aid is a pebble. Thus one is led to further endow such a device with a finite number of markers or pebbles which it can manipulate on Cayley graphs with a distinguished origin. In [2] we introduced the notion of a Cayley machine, and its finite automaton analogue, the Cayley automaton. A Cayley automaton consists of a finite control capable of assuming any of a finite number of given states while it scans the vertices of an input Cayley graph. Initially, the finite control is poised in a designated initial state over a vertex arbitrarily distinguished on the graph as the identity element of the associated group. Thereafter, the automaton moves about from vertex to vertex, 'sliding' along the labelled edges of its input in successive discrete time steps, switching from one state to another in accordance with the transition system of the automaton. It will accept the current input graph if any of a special subset of final states is ever entered in the course of its computation. The automaton thus defines a collection, or a 'language', consisting precisely of those Cayley graphs which it accepts.

This paper is devoted to questions about the simplest kind of Cayley automata, those on (one-generator) cyclic graphs, here called *cyclic automata.* A characterization is given in Section 2 of cyclic graph languages accepted by 0-pebble cyclic automata. Section 2 also deals with decision problems for 0-pebble automata. In Section 3 arbitrary n pebble cyclic automata are shown to define a strict hierarchy and to recognize exactly sets of cyclic group orders that are recognizable in logspace by an ordinary Turing machine. We conclude with some observations on the equivalence between the problems of separating deterministic and nondeterministic logspace Turing machine and cyclic automata complexity classes.

2. Preliminary definitions

An *alphabet* is any nonempty finite set Σ of *symbols* or *letters* with a distinguished blank element. Refer to [6, 13] for definitions on Cayley graphs. All Cayley graphs are assumed to have a distinguished origin. In general, $\mathscr{C}_n$ will denote the class of n-generator Cayley graphs and π_k will denote the cycle of size k and π_0 the infinite cycle. Unless otherwise qualified, 'automaton' refers to a cyclic Cayley automaton in the remainder of this paper.

Definition 2.1. A *Cayley machine* over m is a 6-tuple

$$M = \langle Q, m, \Sigma_n, \delta, q_0, F \rangle$$

consisting of

(1) a finite nonempty set Q of internal states containing a special *initial state* $q_0 \in Q$ and a subset of *final* (or *accepting*) states $F \subseteq Q$;
(2) an alphabet m of input symbols;
(3) a symmetrized set of generators and their inverses Σ_n;
(4) a transition function (or dynamics)

$$\delta : Q \times m \to 2^{Q \times m \times \Sigma_n},$$

$$(q, i) \mapsto \delta(q, j, x^{\varepsilon}).$$

Only a more restricted type of device than a Cayley machine will be considered here, one not capable of arbitrarily writing on vertices, but which rather is capable of using a more primitive kind of mark for the purpose of keeping track of its calculation on its input graph.

Definition 2.2. A (finite) *Cayley automaton* is a Cayley machine which never rewrites a symbol twice on a vertex of any input graph without first erasing it from any other vertex on which it was previously written.

A more picturesque description is obtained by regarding the input symbols of m as a set of physical *markers* or *pebbles* that the automaton can drop as markers on the vertices of its input graphs in order to recognize a certain property of the input graph (accept or reject). In particular, the distinguished vertex v_0 of input graph π could be regarded as a special pebble attached to a vertex of π which the automaton can never remove and/or replace elsewhere in the course of its computation. The symbol 0 denotes this special pebble, and symbol i, $1 \leq i \leq m$, denotes the ith pebble or marker m_i. $X\mathbf{C}_n\mathbf{A}(m)$ denotes the set of all languages in $\mathscr{C}_n$ accepted by a Cayley automaton of deterministic ($X = \mathrm{D}$) or nondeterministic ($X = \mathrm{N}$) type with m pebbles, and $X\mathbf{C}_n\mathbf{A}$ denotes the set of all languages accepted by an X-Cayley automaton with some number of pebbles.

Under the assumption that $n = 1$ the input Cayley graphs are cyclic groups. Since the order of a cyclic group determines all of its group-theoretic properties, the problem is reduced to determining what languages over a one-letter alphabet $\{b\}$, namely the unary encodings of the corresponding orders, are definable by m-pebble Cayley automata. Consequently, given a set $L \in \mathbf{C}_1\mathbf{A}(m)$, the object is to characterize the set

$$|L| := \{b^k : \pi_k \in L\}.$$

The strings, however, are cyclic strings in case of a finite input, or a doubly infinite ordinary string in case of the infinite cyclic group π_0 with a distinguished origin. The presence of the infinite cycle is an additional complication that is considered next. For that, some simplification needs to be made in an arbitrary function δ.

Also, in case $n = 1$, the transition function assumes a particularly simple form

$$\delta : Q \times m \to 2^{Q \times m \times \Sigma_1}.$$

Further simplifications are also possible. For instance, it is easy to see that, without loss of generality, it can be assumed that the automaton never goes through the origin. Moreover, only some auxiliary states are necessary to remove nondeterminism in the direction of motion of an automaton M at each move for the most general type of transition functions so that δ is simply of the form

$$\delta: Q \times m \to 2^{Q \times m} \times \Sigma_1,$$

without any loss of generality.

Let us now consider 0-pebble automata. Thus, δ can be decomposed into the cartesian product $\tau \times \omega$ of two functions

$$\tau: Q \times m \to 2^Q \quad \text{(the next-state function)},$$

$$\omega: Q \times m \to \Sigma_n \quad \text{(the next-move function)}$$

since $\delta: Q \times m \to 2^Q \times \Sigma_n$ as M does not deal with any pebbles at all. This decomposition resembles the transition function of a 2-way deterministic finite state machine with output. The 1-way version of this machine is well-known in the literature as a *Mealy machine*, or, in general, *2-way finite-state transducer*. The functions τ and ω can be obviously extended to functions

$$\tau: Q \times m^* \to 2^Q \quad \text{and} \quad \omega: Q \times m^* \to \Sigma_1,$$

which will be used with the same notation throughout the sequel when dealing with 0-pebble automata.

Lemma 2.3. *Let M be an arbitrary cyclic automaton. If $\pi_0 \in L(M)$, then $L(M)$ is cofinite.*

Proof. Since the k-balls of finite cyclic graphs of order greater than k_0 are identical to the k-ball of π_0 and since M's transition function is a local transition function which only depends on the initial states of M and whether M has returned to the origin of its input or not, the same run will be an accepting run on any graph of order k_0+1. Therefore, $L(M)$ contains $\{\pi_k \in \mathscr{C}_1 : k > k_0\}$ and hence has a finite complement in $\mathscr{C}_1$. □

Corollary 2.4. *If $\pi_0 \in L$ and L is an m-pebble automaton language, then $L = L(M)$ for some 0-pebble Cayley automaton M.*

Proof. If L contains π_0, then L is cofinite by Lemma 2.3, say

$$L = \{\pi_{k_1}, \pi_{k_2}, \ldots, \pi_{k_r}\} \cup \{\pi_k : k \geq k_0\}.$$

It is a simple matter to construct a 0-pebble automaton

$$M' = \langle Q', \mathbf{0}, \Sigma_1, \delta', q_0, F' \rangle$$

which accepts $\{\pi_{k_1}, \ldots, \pi_{k_r}\}$ and halts at ν_0 in state q_0'' in case M' does not accept. Likewise, a second 0-pebble automaton $M'' = \langle Q'', \mathbf{0}, \Sigma_1, \delta'', q_0'', F''\rangle$ can be constructed which accepts $\{\pi_k : k \geq k_0\}$. The 0-pebble automaton

$$M = \langle Q' \cup Q'', \mathbf{0}, \Sigma_1, \delta' \cup \delta'', q_0, F''\rangle$$

can be regarded as a series composition of M' and M'' if states q_0' in both of them are identified but otherwise they are disjoint. It is easily seen that

$$L(M) = L(M') \cup L(M''). \qquad \square$$

Lemma 2.3 reduces the problem of characterizing 0-pebble cyclic languages to the similar problem for 2-way finite automata on ordinary strings of form

$$0bbb\ldots bbb0$$

set in between two identical endmarkers 0 over a unary alphabet $\{b\}$. Now, well-known results about 1- and 2-way finite state automata can be used in order to obtain a characterization of 0-pebble cyclic languages.

Lemma 2.5 (Harrison [3, p. 70]). *For each 2-way finite automaton M_0 the string language $L(M_0)$ is a regular set.*

The statement holds independently of endmarkers and is effective, that is, there exists an algorithm which constructs a 1-way automaton M_0' over $\{b\}$, instead of $\{\text{¢}, b, \$\}$ with endmarkers, such that $\text{¢}L(M_0')\$ = L(M_0)$.

Theorem 2.6. *Let $L \subseteq \mathscr{C}_1$ be a set of cyclic Cayley graphs, let $|L|$ be the subset of b^* of orders of groups associated with elements of L and let π_0 be the infinite cyclic group.*

(1) *If $\pi_0 \in L$, then L is accepted by a 0-pebble Cayley automaton if and only if $|L|$ is a cofinite set;*

(2) *If $\pi_0 \notin L$, then L is accepted by a 0-pebble Cayley automaton if and only if $|L|$ is a regular subset of b^+, that is, if and only if $|L|$ is a union of a finite set and a finite number of arithmetic progressions.*

Proof. Statement (2) is clear from the remarks preceding the theorem. The condition in (1) is necessary by Lemma 2.3. If now $|L|$ is cofinite, say $|L| = \{\pi_k : k \geq k_0\}$, the deterministic Cayley automaton

$$M_1 = \langle \{q_0, q_1, \ldots, q_{k_0}\}, 0, \Sigma_1, \delta_1, q_0, \{q_{k_0}\}\rangle$$

given by

$$\delta_1(q_0, 0) = \{(q_1, 0, x)\}; \qquad \delta_1(q_j, b) = (q_{j+1}, b, x), \quad 1 \leq j \leq k_0;$$

and

$$\delta_1(q, i) = \emptyset \quad \text{elsewhere,}$$

obviously accepts L. $\square$

Corollary 2.7. *Cyclic languages defined by 0-pebble automata are closed under union and intersection but not under complementation.*

Proof. Closure under union is immediate from nondeterminism. Closure under intersection is clear in case both or none of the languages contain π_0. If L_1 is cofinite and $\pi_0 \in L_1$, then $L_1 \cap L_2$ differs from L_2 in just a finite set and hence the corollary follows from Theorem 2.6. Finally, the set of odd-order cycles plus π_0 is not a Cayley language but it is the complement of the even order cycles, which is a 0-pebble cyclic language. □

In particular, the class of simple cyclic groups is not accepted by any 0-pebble nondeterministic Cayley automaton; in fact, it is not accepted by any nondeterministic 1-pebble Cayley automaton.

Corollary 2.8. *Nondeterministic 0-pebble Cayley automata are equivalent to deterministic Cayley automata, that is,* $\mathbf{NC_1A}(0) = \mathbf{DC_1A}(0)$.

Let us now consider decision problems for 0-pebble cyclic automata. As pointed out before, all the above reductions are effective except possibly for the construction in Lemma 2.3 of a 0-pebble automaton equivalent to a given m-pebble automaton accepting π_0. The next theorem proves that this step is also effective. The membership problem of π_0 for a class $\mathscr{C}$ asks if an arbitrary automaton M in $\mathscr{C}$ accepts the infinite cycle. The problems of emptiness, inclusion and equivalence are defined in the usual way.

Theorem 2.9. *The problem of π_0-membership is decidable for 0-pebble cyclic automata.*

Proof. Let M be an arbitrary 0-pebble cyclic automaton. The idea is to effectively construct a (1-way) pda M_0 satisfying the condition that $\pi_0 \in L(M)$ if and only if $L(M_0) \neq \emptyset$.

The proof is suggested by the observation that, from the finite control of a 0-pebble automaton, the free group π_0 will look like a string of empty cells only interrupted by an occasional occurrence of the symbol 0 which signals the return to vertex v_0. Thus, M_0 can be designed to store, into a pushdown stack, a unary representation of M's current position on π_0, say by pushing on (respectively popping off) the stack symbol b whenever M makes a right (left)-move on the right half of π_0, or vice versa on the left half. Hence, M_0 will reach the bottom of its stack precisely when M returns to v_0 and, therefore, it will accept some string over $\{b\}$ by final state if and only if M eventually accepts π_0. Note that such an M_0 is just a '1-counter automaton', that is, a pda with a one-letter stack alphabet in addition to the bottom-of-stack symbol z_0.

More precisely, let $M = \langle Q, \mathbb{0}, \Sigma_1, \delta, q_0, F\rangle$ be any 0-pebble automaton. Let

$$M_0 = \langle Q', \{b\}, \Gamma, \delta_0, [q_0, 1], z_0, F \times \{-1, 1\}\rangle$$

be the 1-way 1-counter pda defined as follows:

$$Q' = Q \times \{-1, 1\}, \qquad \Gamma = \{0, b\}, \qquad z_0 \equiv 0$$

and the transition function

$$\delta_0 : Q' \times \{b\} \times \Gamma \to 2^{Q' \times \Gamma}$$

is given by

$$\delta_0([q, \varepsilon], b, b) = \{([p, \varepsilon], b^{(|\mu+\varepsilon|)/2}) : (p, b, x^\mu) \in \delta(q, b)\},$$

$$\delta_0([q, \varepsilon], b, 0) = \{([p, \mu], 0b) : (p, 0, x^\mu) \in \delta(q, 0)\},$$

where the pairs in Q' have been written in square brackets $[q, \varepsilon]$ and b^0 denotes the empty word. It is readily verified that M_0 operates as stated above for the second component of the states allow M_0 to store on which half M is located on its input π_0. An obvious induction on the number of moves will show that after t moves M scans cells $x^{-\varepsilon k}$ on π_0 in state q if and only if M_0 scans the top of a stack of $k\,b$'s in state $[q, \varepsilon]$. Therefore, the result follows from the decidability of the emptiness problem for ordinary pda's. □

Thus, the constructions in Lemma 2.3 and Theorem 2.6 are effective. Its effectiveness brings about decision algorithms available for string machines.

Corollary 2.10. *The decision problems of emptiness, finiteness, inclusion, equivalence and membership are decidable for* 0*-pebble cyclic automata.*

The decidability of equivalence is all the more interesting in view of the fact that, as was mentioned above, a pebble automaton is essentially a generalized sequential machine (or transducer) on a reduced class of inputs, and that Ibarra [5] has shown the undecidability of inclusion and equivalence for arbitrary nondeterministic finite-state transducers with inputs (or outputs) restricted to a unary alphabet.

Corollary 2.11. *There is an effective procedure to find a state-minimal automaton equivalent to a given* 0*-pebble cyclic automaton.*

This section concludes with a result analogous to the well-known equivalence of finite automata with one marker and ordinary finite automata by Meyer (see [1, p. 159]).

Corollary 2.12. *Every* 1*-pebble cyclic automaton is equivalent to a* 0*-pebble cyclic automaton.*

3. Multipebble automata

The picture exhibited by multipebble cyclic automata is in sharp contrast with the situation observed in the previous section. As it turns out, addition of at least

two pebbles increases the computational capability of both varieties of cyclic automata, deterministic and nondeterministic. More interestingly, all decision problems solvable for 1-pebble automata are recursively unsolvable for deterministic 2-pebble cyclic automata since even the simplest kind of decision problems, membership of the free group π_0, then becomes recursively unsolvable.

Some introductory examples will illustrate the computational power of multi-pebble cyclic automata. Of course, these automata are at least as powerful as their pebbleless counterparts.

Example 3.1. The set of 2-groups is a 2-pebble language. A 2-pebble automaton $M(2^n)$ first checks that its input π is finite, say of order k, drops m_2 at $k-1$, returns to v_0, and drops m_1 at cell x. Then $M(2^n)$ successively goes back and forth moving m_1 and m_2 one cell right or left respectively, until it either attempts to drop m_1 on top of m_2, in which case it loops and rejects its input π_k, or else $M(2^n)$ replaces m_1 with m_2, brings m_1 back to cell 1 and repeats the binary subdivision it just performed on the copy of $\pi_{k/2}$ now between v_0 and m_2. $M(2^n)$ accepts π_k if and only if m_2 ever reaches cell 1 in this fashion. Hence,

$$L(2^n) := \{\pi_k : k = 2^n \text{ for some } n \geq 1\}.$$

The description of the automata in the following examples will be omitted since it can be done by similar manipulation of the pebbles and elementary arithmetic facts:

$$L(2^{pn}) = \{\pi_k : k = 2^{pn} \text{ for some } n \geq 1\},$$

for each $p \geq 1$, and more generally, the set of d-power groups is a 3-pebble cyclic language; the set of simple cyclic groups is a 3-pebble language; the set of perfect square order groups is a 3-pebble language; and the set of cyclic groups of order in the range of a diophantine polynomial with nonnegative coefficients is a 6-pebble language.

The foregoing examples show that deterministic pebble automata are indeed very powerful devices. Further evidence is provided by the following consequence of the result by Minsky on the equivalence between 2-counter finite-state machines and arbitrary Turing machines. His statement can be phrased in the following form.

Theorem 3.2. *The halting problem on blank tape is recursively unsolvable for deterministic 2-counter finite automata.*

An obvious modification allows the further assumption that a 2-counter machine changes the content of only one counter per move.

The present reduction of π_0-membership to the halting problem of 2-counter finite automata consists in exhibiting an effective procedure which constructs a 2-pebble deterministic cyclic automaton M_T from a given 2-counter machine T such that the following condition holds:

(*) $\pi_0 \in L(M_T)$ if and only if T halts on a blank input tape.

Theorem 3.3. *The problem of π_0-membership is undecidable for deterministic 2-pebble cyclic automata.*

Proof. Let T be a deterministic 2-way 2-counter finite automaton with state set Q and initial state q_0. The state set of a deterministic Cayley automaton M_T contains the set Q plus some auxiliary states and has the same initial state q_0. Let c_1 and c_2 be typical counter contents of T. Initially, c_1 and c_2 are empty. M_T simulates M's manipulations of the counters by the position of its two pebbles m_1 and m_2 from ν_0 as follows:

(1) M_T moves right of π_0 and drops m_1 or m_2 according as T increases c_1 or c_2 by 1 in its first move respectively.

(2) M_T keeps track in its internal control of which direction of motion to follow in order to find m_1 and/or m_2 from the current position of its input head, as well as in which direction it last moved.

(3) M_T halts and rejects if it ever runs into ν_0 while moving in the direction of generator x.

(4) In a typical set of instructions to simulate one move of T, M_T's final control repeatedly moves left or right, depending on the current information stored in its internal state, until it finds m_1 or m_2, picks it up at some vertex g, moves in the direction of x^{ε}, $\varepsilon = \pm 1$, and drops it at node gx^{ε} whenever T would change the contents of counter c_1 (c_2 respectively) by ε. Lastly,

(5) M_T will halt and accept if and only if T halts.

Thus, at any instant of M_T's run on π_0, the distances of the pebbles m_1 and m_2 from the origin ν_0 represent the contents of T's counters c_1 and c_2 respectively. In particular, M_T will be able to detect when T would reach the bottom of either counter by encountering the corresponding pebble at ν_0. Therefore, if M_T does not halt and reject, that is, if M_T does not encounter ν_0 while moving to the right, then M_T will be in one of T's states after executing each set of instructions as in step (4) above. So, if T ever halts on blank input, say after μ moves, then a larger integer μ' can be chosen so that, on input $\pi_{\mu'}$, the automaton M_T will never reach ν_0 while moving right. Hence, it will accept all π_k of order at least μ' and in particular it will accept π_0.

Conversely, the design of M_T allows it to accept π_0 only in the event that T ever enters a halting state. The net result is that condition (*) holds true. Therefore, if there were an algorithm to decide π_0-membership for 2-pebble automata, the effective construction just described would produce an algorithm which would solve the halting problem for 2-counter finite automata, contradicting Theorem 3.2. □

A moment's reflection further shows that in the above proof T and M_T also satisfy the above condition that $\pi_0 \in M_T$ if and only if $L(M) \neq \emptyset$. Therefore, other decision problems are also undecidable.

Corollary 3.4. *Emptiness, inclusion and equivalence are recursively unsolvable for 2-pebble deterministic cyclic automata.*

Proof. It is a simple matter to construct a 2-pebble Cayley automata M_1 such that $L(M_1) = \emptyset$. Now all statements follow at once from the observation that the following statements:

(1) $L(M) = \emptyset$,
(2) $L(M) \subseteq L(M_1)$,
(3) $L(M) = L(M_1)$,

are equivalent assertions for *all* cyclic automata M. □

Let us now consider the relative power of multipebble automata. Since 0-pebble automata are equivalent to deterministic 0-pebble automata, it is natural to pose the question of whether or not it is possible to simulate an arbitrary m-pebble nondeterministic cyclic automaton by a deterministic automaton with a sufficiently larger number of pebbles. As it turns out, however, any attempt to gauge the relative power of nondeterminism in cyclic automata leads directly into some of the 'classical and embarrasing' (see [4]) open questions of the theory of formal language theory on context-sensitive languages, namely, the LBA problem on the equivalence of nondeterministic and deterministic linear-bounded Turing machines. At this point it is fairly easy to see how these results follow from similar results over a one-letter alphabet by Monien [8, 9] and Meinhardt and Wagner [7].

From a different perspective, the previous theorem showed how 2-pebble cyclic automata have the ability to simulate an arbitrary Turing machine on the infinite cyclic group. Hence, it is conceivable that there might exist an absolute upper bound on the number of pebbles necessary to accept an arbitrary cyclic pebble language of either kind. Nevertheless, it is easy to see that, with the above reductions, multipebble cyclic automata are basically equivalent to multihead finite-state automata over a one-letter alphabet. Thus it follows from results by Monien [8, 9], Sudborough [12] and Seiferas [10] that there exists an infinite hierarchy of cyclic graph languages, even if only groups of order a power of 2 are considered and, moreover, from results by Hartmanis [4] and Seiferas [10], that these unary languages are precisely those that can be recognized by a Turing machine in logspace (deterministically or nondeterministically, respectively).

References

[1] M. Blum and C. Hewitt, Automata on a 2-dimensional tape, in: *Proc. 8th IEEE Conf. on SWAT* (1967) 155-160.

[2] M. Garzon, Cayley automata, submitted.
[3] M.Harrison, *Introduction to Formal Language Theory* (Addison-Wesley, Reading, MA, 1978).
[4] J. Hartmanis, On non-determinacy in simple computing devices, *Acta Inform.* **1** (14) (1972) 336-344.
[5] O.H. Ibarra, The unsolvability of the equivalence problem for ε-free NGSM's with unary input (output) alphabet and applications, *SIAM J. Comput.* **7** (4) (1978) 524-532.
[6] W. Magnus, A. Karrass and D. Solitar, *Combinatorial Group Theory* (Dover, New York, 1976).
[7] V.D. Meinhardt and K. Wagner, Eine Bemrekung zu einer Arbeit von Monien über Kopfzahl Hierarchien für Zweiweg-Automaten, *Elektron. Informationsverarb. Kybernet.* **18** (1982) 69-74.
[8] B. Monien, Transformational methods and their applications to complexity problems, *Acta Inform.* **6** (1976) 95-118. Corrigenda, *ibid.* **8** (1977) 383-384.
[9] B. Monien, Two-way multihead automata over a one-letter alphabet, *RAIRO Inform. Théor.* **14** (1980) 67-82.
[10] J.I. Seiferas, Techniques for separating complexity classes, *J. Comput. System Sci.* **14** (1977) 77-99.
[11] J.I. Seiferas, Relating refined space complexity, *J. Comput. System Sci.* **14** (1977) 100-129.
[12] I.H. Sudborough, Separating tape-bounded auxiliary pushdown automata classes, in: *Proc. ACM Symp. Theory of Computing* **9** (1977) 208-217.
[13] A.T. White, *Graphs, Groups and Surfaces* (North-Holland, Amsterdam, 1984).