Open access • Proceedings Article • DOI:10.1109/SFCS.1983.39

On determinism versus non-determinism and related problems — Source link

Wolfgang J. Paul, Nicholas Pippenger, Endre Szemerédi, William T. Trotter

Institutions: Claremont Colleges, Hungarian Academy of Sciences

Published on: 07 Nov 1983 - Foundations of Computer Science

Topics: Super-recursive algorithm, Time hierarchy theorem, NSPACE, Turing machine and Turing machine examples

1-1-1983

On Determinism Versus Non-Determinism And Related Problems

Wolfgang J. Paul
IBM Research Laboratory

Nicholas Pippenger
Harvey Mudd College

Endre Szemeredi
University of South Carolina

William T. Trotter
University of South Carolina

ON DETERMINISM VERSUS NON-DETERMINISM AND RELATED PROBLEMS

(Preliminary Version)

Wolfgang J. Paul
IBM Research Laboratory
San Jose, CA 95193

Nicholas Pippenger
IBM Research Laboratory
San Jose, CA 95193

Endre Szemerédi
University of South Carolina
Columbia, SC 29208

William T. Trotter
University of South Carolina
Columbia, SC 29208

ABSTRACT: We show that, for multi-tape Turing machines, non-deterministic linear time is more powerful than deterministic linear time. We also discuss the prospects for extending this result to more general Turing machines.

1. Introduction

Our main result in this paper states that, for multi-tape Turing machines, non-deterministic linear time is more powerful than deterministic linear time. More specifically, we show that there is a language recognized by a two-tape non-deterministic Turing machine in linear time, but not recognized by any multi-tape deterministic Turing machine in linear time (both machines receive their input on a read-only input tape). This result (which will be proved in Section 4 below) shows not only that non-determinism adds power, but that this additional power cannot be compensated for by any number of additional tapes.

The first evidence that non-deterministic time-bounded Turing machines are more powerful than deterministic time-bounded Turing machines was provided by Hennie [6], who showed that one-tape deterministic Turing machines (that receive their input on their work tape) require time $\Omega(n^2)$ to recognize non-palindromes of length n (it is easy to see that time $O(n \log n)$ is sufficient for a non-deterministic machine).

Among subsequent attempts to extend this result to multi-tape Turing machines, we may note two results of Kannan. The first (Kannan [9]) shows that non-deterministic two-tape machines are more powerful than deterministic one-tape machines (both machines receive their input on a read-only input tape). The second (Kannan [10]) shows that non-deterministic multi-tape machines are more powerful than deterministic multi-tape machines with an additional space bound (growing strictly more slowly than their time bound). In both of these results, the deterministic machines suffer an additional handicap, and it is not clear that this handicap alone does not account for the observed difference in power.

We should also note a paper of Paul and Reischuk [15], which proved a result similar to ours on the assumption of a certain graph-theoretic hypothesis. This hypothesis was subsequently disproved by

0272-5428/83/0000/0429$01.00

Schnitger [18,19], but the present paper owes both its overall strategy and many of its tactics to the paper of Paul and Reischuk [15].

The overall strategy may be described briefly as follows. If deterministic linear time equals non-deterministic linear time, then deterministic linear time equals alternating linear time for machines making a bounded number of alternations. By a padding argument, this implication extends to non-linear time bounds. This much is as in Paul and Reischuk [15]. The key to the proof is a simulation (which will be given in Section 3) whereby any language recognized by a deterministic Turing machine in non-linear time is recognized faster by an alternating Turing machine making just four alternations. As in Paul and Reischuk [15], a diagonalization completes the proof.

The simulation mentioned above shows that alternating time with four alternations is more powerful than deterministic time. This may be compared with previous results by Paul, Prauss and Reischuk [14] and by Dymond and Tompa [3] showing that alternating time (with no bound on the number of alternations) is more powerful than deterministic time, and by Hopcroft, Paul and Valiant [7] showing that deterministic space is more powerful than deterministic time.

The simulation of Hopcroft, Paul and Valiant [7] involves a certain "pebble game" played on the vertices of an acyclic directed graph that represents (at a certain level of abstraction) the computation of a deterministic Turing machine. Analogously, the simulation of Dymond and Tompa [3], as well as the simualtion of the present paper, involves a certain "two-person pebble game". Additionally, in the present paper we must exploit certain constraints satisfied by the computation graphs of deterministic multi-tape Turing machines. These constraints imply a "segregator theorem" for these graphs, which is proved in Section 2.

2. A Segregator Theorem

In this section we lay the graph-theoretic foundation for our simulation.

Let H(N) denote the class of directed graphs with vertices {1, ..., N} in which every edge (i, j) satisfies i<j (so that these graphs are acyclic). We shall say that two edges (i, j) and (i', j') cross if i<i'<j<j'. Let $H_1(N)$ denote the subclass of H(N) comprising those graphs in which every vertex has at most one immediate predecessor and at most one immediate successor, and in which no two edges cross. Let $H_r(N)$ denote the subclass of H(N) comprising those graphs that can be expressed as the union of r graphs belonging to $H_1(N)$. The class $H_r(N)$ is essentially the class of multi-pushdown graphs defined by Pippenger [17].

A set J of vertices of a graph G in H(N) will be called an M-segregator for G if every vertex in G-J has at most M predecessors in G-J. (The name "segregator", coined in analogy to "separator", was suggested by Michael Sipser.) Our main result in this section shows that, for each fixed r, graphs in $H_r(N)$ have M-segregators J with $M=o(N)$ and $|J|=o(N)$.

The proof uses depth-reduction techniques due to Erdős, Graham and Szemerédi [4] and to Valiant [22].

Define $\log^{(0)} x=x$ and, for $\ell\geq 1$, $\log^{(\ell)} x=\log_2 \log^{(\ell-1)} x$. Define

$$\log^* x = \min \{\ell: \log^{(\ell)} x \leq 1\}.$$

Theorem 2.1: Let G be a graph in $H_r(N)$ with $N\geq 16$. There is a set J of at most $15rN/\log^* N$ vertices of G such that every vertex of G-J has at most $6N/\log^* N$ predecessors in G-J.

For the proof we shall need the following lemma.

Lemma 2.2: For every integer $N\geq 16$, there exist integers $k\geq 2$ and $d_0, \ldots, d_k$ such that

(1) $d_0=1$;

(2) for $0\leq\ell\leq k-1$, $d_\ell | d_{\ell+1}$

(3) $d_k\leq 2N$;

(4) For $0\leq\ell\leq k-1$,

$$\exp_k \lceil N/d_{\ell+1} \rceil \leq \lceil N/d_\ell \rceil /k;$$

and

(5) $k\geq(\log^* N)/3$.

Proof: For $k\geq 2$ and $1\leq\ell\leq k-1$, define $e_{k,0}=1$ and $e_{k,\ell+1}=\exp_k (2+2e_{k,\ell})$. Define $\exp^{(0)} x=x$ and, for $\ell\geq 1$, $\exp^{(\ell)} x=\exp_2 \exp^{(\ell-1)} x$. Straightforward estimates show that

$$e_{k,\ell} \leq \exp^{(k+2\ell)} 1.$$

Thus if we choose $k=\lceil(\log^* N)/3\rceil$, then condition (5) is satisfied and

$$1 = e_{k,0} \leq \ldots \leq e_{k,k-1} \leq N.$$

Define $d_0=1$ and, for $1\leq\ell\leq k$, $d_\ell=\exp_2 \lceil \log_2 (N/e_{k,k-\ell})\rceil$. Then conditions (1), (2) and (3) are immediate. To verify (4), observe that

$$\begin{aligned}\exp_k \lceil N/d_{\ell+1} \rceil &\leq \exp_k (2N/d_{\ell+1}) \\ &\leq \exp_k (2e_{k,k-\ell-1}) \\ &= e_{k,k-\ell}/k^2 \\ &\leq 2N/d_\ell k^2 \\ &\leq \lceil N/d_\ell \rceil /k.\end{aligned}$$

This completes the proof. □

Proof of Theorem 2.1: Let k and $d_0, \ldots, d_k$ be as in Lemma 2.2. We shall construct a sequence of partitions $P_0, \ldots, P_k$ of the vertices of G. For $0\leq\ell\leq k$, we construct P_ℓ by taking blocks of consecutive vertices of G, with each block except possibly the last containing d_ℓ vertices, and with the last block containing at most d_ℓ vertices. Clearly, P_0 is the discrete partition and $P_{\ell+1}$ is coarser than P_ℓ for $0\leq\ell\leq k-1$.

Associate with each edge (i, j) of G the coarsest of the partitions $P_0, \ldots, P_{k-1}$ such that i and j appear in different blocks. Since there are at most rN edges, there must be a partition P_ℓ with $0\leq\ell\leq k-1$ that has at most rN/k edges associated with it. Let A denote the set of vertices i such that some edge (i, j) is associated with P_ℓ. Then $|A|\leq rN/k$.

Construct a graph G^* by collapsing each block X of P_ℓ into a single "node" X^* and by putting an "arc" from node X^* to node Y^* whenever G-A contains an edge from a vertex in block X to a vertex in block Y. The graph G^* has

$$N^* = \lceil N/d_\ell \rceil$$

nodes.

Every edge (i, j) of G-A either has i and j in the same block of P_ℓ (and thus gives rise to no arc of G^*) or has i and j in different blocks of $P_{\ell+1}$. Since there are just

$$M^* = \lceil N/d_{\ell+1} \rceil$$

block of $P_{\ell+1}$, the longest directed path in G^* has length at most M^*-1.

Let G be the union of $G_1, \ldots, G_r$ in $H_1(N)$. For $1 \le s \le r$ we may construct G_s^* in the obvious way, and G^* is the union of $G_1^*, \ldots, G_r^*$.

For each node X^* of G^*, let $D_s(X^*)$ denote the *number of* immediate predecessors of X^* in G_s^*. We claim

$$\Sigma_{X^*} D_s(X^*) \le 2N^*.$$

To see this, observe that if X^* is an immediate predecessor of Y^* and Z^* in G_s^* with $Y^*<Z^*$, then X^* is the first immediate predecessor of Y^*. For if W^* were an immediate predecessor of Y^* in G_s^* with $W^*<X^*$, then the arcs (W^*, Y^*) and (X^*, Z^*) in G_s^* would cross. The edges of G_s that give rise to these arcs would also cross, contradicting the fact that G_s belongs to $H_1(N)$. Thus, distinct nodes of G_s^* have disjoint sets of immediate predecessors (except possibly for their first immediate predecessors). It follows that

$$\Sigma_{X^*} (D_s(X^*)-1) \le N^*,$$

which proves the claim, since the sum has N^* terms.

Let $D(X^*)$ denote the number of immediate predecessors of X^* in G^*. Then

$$\Sigma_{X^*} D(X^*) \le \Sigma_{1 \le s \le r} \Sigma_{X^*} D_s(X^*) \le 2rN^*.$$

Let us say that a node X^* in G^* is "bad" if $D(X^*)>k$. Let B^* denote the set of bad nodes in G^*. Then $|B^*| \le 2rN^*/k$.

Let B denote the set of vertices of G in blocks X of P_ℓ for which the node X^* belongs to B^*. Since each block of P_ℓ contains at most d_ℓ vertices, and since $d_\ell N^* \le 2N$, we have $|B| \le 4rN/k$.

Since every node of G^*-B^* has at most k immediate predecessors and since every directed path in G^* has length at most M^*-1, every node in G^*-B^* has at most

$$\exp_k M^* \le N^*/k$$

predecessors in G^*-B^*.

Let J be the union of A and B. Then

$$|J| \le 5rN/k \le 15rN/\log^* N.$$

If a vertex in block X of P_ℓ is a predecessor of a vertex in block Y of P_ℓ in G-J, then X^* must be a predecessor of Y^* in G^*-B^*. Since each block of P_ℓ contains at most d_ℓ vertices, every vertex of G-J has at most

$$d_\ell N^*/k \le 2N/k \le 6N/\log^* N$$

predecessors in G-J. □

A set J will be called an M-separator for G if every component of G-J (ignoring the directions of the edges) contains at most M vertices. Separator theorems are known for trees (see Lewis, Stearns and Hartmanis [11]) and for planar graphs (see Lipton and Tarjan [12]). Pippenger [17] conjectured that, for each fixed r, graphs in $H_r(N)$ have M-separators J with $M=o(N)$ and $|J|=o(N)$. As indicated in [17], this would have numerous computational consequences.

The present paper is founded on the fact that, for the purpose of separating (or segregating) determinism from non-determinism, a segregator theorem serves as well as a separator theorem.

3. A Simulation

In this section we shall show how computations by deterministic multi-tape Turing machines can be accelerated by alternating multi-tape Turing machines making just four alternations.

Let Δ_0 denote the class of deterministic multi-tape Turing machines, and let $\Delta_{0,\ell}$ denote the subclass having ℓ work tapes. (Henceforth all machines receive their input on a read-only input tape.) Let Σ_k (respectively, Π_k) denote the corresponding class of alternating machines that start in an existential (respectively, a universal) state and change quantification at most k-1 times, and let $\Sigma_{k,\ell}$ (respectively, $\Pi_{k,\ell}$) denote the subclass having ℓ work tapes.

Let f be a time-constructable funtion. If Q denotes a class of machines, we shall let Q(f) denote the class of languages recognized by machines in Q in time f(n) on inputs of length n.

Let f be a time-constructable function and let $a(n)=\lceil f(n)^{1/3} \rceil$ and $b(n)=\lceil f(n)^{2/3} \rceil$. Let M be a machine in Δ_0 running in time f. Let the computation of M on an input x of length n be partitioned into "segments" each comprising b(n) consecutive steps (except possibly the last, which comprises at most b(n) steps). Let the tapes of M be partitioned into "blocks" each comprising b(n) contiguous cells. We shall say that M is block-respecting if, during each segment of its computation, each head of M visits one and only one block.

Lemma 3.1: A language recognized in time f by a machine in $\Delta_{0,\ell}$ is also recognized in time f by a block-respecting machine in $\Delta_{0,\ell+1}$.

Proof: See Hopcroft, Paul and Valiant [7]. □

Let M be a block-respecting machine in Δ_0. For every input x of length n, let G(M, x) be the graph with vertices {1, ..., a(n)} (corresponding to the segments of the computation of M) with an edge (i, i+1) for each $1 \le i \le a(n)-1$ and with an edge (i, j) whenever, in the computation of M on input x, some block on some work tape is visited during segment i and revisited during segment j, without being revisited during any intermediate segment.

Lemma 3.2: If M is a block-respecting machine in $\Delta_{0,\ell}$, then for every input x of length n, G(M, x) belongs to $H_{2\ell+1}(a(n))$.

Proof: The edges of G(M, x) of the form (i, i+1) clearly form a graph in $H_1(a(n))$. For each of the ℓ work tapes, revisits of blocks may be partitioned into two classes: revisits from the right and revisits from the left. The revisits in each of these classes also give rise to a graph in $H_1(a(n))$. □

We are now ready to prove the main result of this section.

Theorem 3.3: For every time-constructable f with $f(n) \ge n \log^* n$,

$$\Delta_0(f) \subseteq \Sigma_4(f/\log^* f).$$

Proof of Theorem 3.3: Let M_0 be a machine in $\Delta_{0,\ell}$ that recognizes L in time f. Let $a(n)=\lceil f(n)^{1/3} \rceil$ and $b(n)=\lceil f(n)^{2/3} \rceil$. By Lemma 3.1, there is a block-respecting machine M_1 in $\Delta_{0,\ell+1}$ that also recognizes L in time f. We shall construct a machine M_2 in Σ_4 that recognizes L in time $f/\log^* f$. Let $e(n)=\lceil 15(2\ell+3)a(n)/\log^* a(n) \rceil$. On input x of length n, M_2 deterministically computes f(n), a(n), b(n) and e(n). It then proceeds in four phases as follows.

Phase 1: M_2 existentially guesses the positions P of each of the $\ell+2$ heads of M' at the outset of each of the a(n) segments. From P, it deterministically computes the computation graph G. It existentially guesses a set J of at most e(n) vertices in G. It existentially guesses the computation of M_1 (internal states, head movements, initial inscriptions of blocks visited and final inscriptions of blocks visited) for each of the segments corresponding to the last vertex k in G and to vertices in J. It deterministically checks that the computation for the last segment includes an accepting state. If this check fails, it rejects; otherwise it proceeds to Phase 2.

Phase 2: M_2 universally selects a vertex j in the union of J and {k}. It deterministically computes the set I of predecessors of j in G-J. If there are more that e(n) vertices in I, it rejects; otherwise it proceeds to Phase 3.

Phase 3: M_2 existentially guesses the computation of M_1 for each of the time segments corresponding to vertices in I.

Phase 4: M_2 universally selects a vertex i in the union of I and {j}. It deterministically checks the consistency of all guesses relating to i with those relating to immediate predecessors of i, with the transition function for M_1 and with the input x. If any of these checks fails, it rejects; otherwise it accepts.

It is routine to verify, using Lemma 3.2 and Theorem 2.1, that M_2 runs in time $f/\log^* f$ and recognizes L. □

4. Consequences for Determinism versus Non-Determinism

In this section we show how the simulation of Theorem 3.3 implies that non-deterministic linear time is more powerful than deterministic linear time.

We shall need the folowing "collapsing" lemma.

Lemma 4.1: If $\Sigma_1(n)=\Delta_0(n)$ (or, equivalently, if $\Pi_1(n)=\Delta_0(n)$), then, for every k and every time-constructable f, $\Sigma_k(f)=\Delta_0(f)$ (and therefore $\Pi_k(f)=\Delta_0(f)$).

Proof: (See Paul and Reischuk [15] for details.) From the hypothesis, a straightforward induction on k yields $\Sigma_k(n)=\Pi_k(n)=\Delta_0(n)$. The conclusion follows by a padding argument. □

We shall also need the following "hierarchy" lemma.

Lemma 4.2: For every $k\geq 1$, and every

time-constructable f and g with $f(n)=w(g(n))$, then $\Pi_k(f)-\Sigma_k(g)$ (and therefore also $\Sigma_k(f)-\Pi_k(g)$) is non-empty.

Proof: (See Paul and Reischuk [15] for details.) By the tape reduction theorem of Book, Greibach and Wegbreit [1], $\Sigma_1(g)=\Sigma_{1,2}(g)$ (and, by taking complements, $\Pi_1(g)=\Pi_{1,2}(g)$). A straightforward induction on k shows that $\Sigma_k(g)=\Sigma_{k,2}(g)$ (and therefore $\Pi_k(g)=\Pi_{k,2}(g)$). A machine in $\Pi_{k,3}$ running in time f can diagonalize over all machines in $\Sigma_{k,2}$ running in time g. The resulting language is in $\Pi_{k,3}(f)-\Sigma_{k,2}(g)=\Pi_k(f)-\Sigma_k(g)$. □

We are now ready to prove our main result.

Theorem 4.3:

$$\Delta_0(n)<\Sigma_{1,2}(n).$$

Proof: By the tape reduction theorem of Book, Greibach and Wegbreit [1], $\Delta_0(n)\leq\Sigma_1(n)=\Sigma_{1,2}(n)$, so it will suffice to show that the inclusion is strict. To do this, we assume $\Delta_0(n)=\Sigma_1(n)$ and derive a contradiction.

By Lemma 4.1, this assumption implies $\Pi_4(n \log^* n)=\Delta_0(n \log^* n)$. By Theorem 3.3, $\Delta_0(n \log^* n)\leq\Sigma_4(n)$, and so $\Pi_4(n \log^* n)\leq\Sigma_4(n)$. But this contradicts Lemma 4.2. □

The argument of this section can be elaborated to show somewhat more than we have done. If Q is a class of machines and f is time-constructable, let Q(o(f)) denote the union of Q(g) over all time-constructable g with $g(n)=o(f(n))$. Then $\Sigma_{1,2}(n)-\Delta_0(o((n \log^* n)^{1/4}))$ is non-empty.

5. Related Problems

In this section we shall reconsider the graph-theoretic foundation of our work, exposing some of its limitations and examining the prospects for transcending these limitations. To do this we shall use the two-person pebble game, introduced by Tompa [20] (see also Dymond and Tompa [3]). We shall give only hints of proofs.

As its name suggests, the two-person pebble game is played between two players, called the Challenger and the Pebbler, on the vertices of an acyclic directed graph. The Challenger begins by placing his token, called the challenge, on some vertex of the graph. The Pebbler responds by placing some of his tokens, called pebbles, on some set of vertices of the graph. In each succeeding round, the Challenger may leave the challenge where it is or may move it to a vertex pebbled by the Pebbler in the immediately preceding round. If all the immediate predecessors of the challenged vertex are pebbled, the Pebbler wins. Otherwise, he continues by placing pebbles on another set of vertices of the graph. We say that the Pebbler wins in R rounds and time T if he has a strategy that ensures that he wins after making at most R moves and placing a total of T pebbles on the graph.

Theorem 2.1 ensures that for any graph in $H_r(N)$, the Pebbler can win in two rounds and time $T=O(N/\log^* N)$. Let $H^r(N)$ denote the subclass of H(N) comprising those graphs in which every vertex has at most r immediate predecessors and at most r immediate successors. Schnitger [18,19] has shown

that $H^2(N)$ contains "grates" (a notion introduced by Valiant [22]). These are graphs G for which, for every set J of vertices of G, either $|J|=\Omega(N)$ or G-J contains $\Omega(N^2)$ pairs (i, j) for which i is a predecessor of j in G-J (and thus contains some vertex with $\Omega(N)$ predecessors in G-J). For such graphs the Pebbler can win in two rounds only in time $T=\Omega(N)$. Let us consider to what extent the factor "$\log^* N$" in Theorem 2.1 might be replaced by a larger factor. We shall show that it can be increased at most to "log N".

Theorem 5.1: For every N, there is a graph G in $H_3(N)$ such that, for every set J of vertices of G, either $|J|=\Omega(N/\log N)$ or some vertex in G-J has $\Omega(N/\log N)$ predecessors in G-J.

This theorem is proved by constructing a variant of the "Fast Fourier Transform" graph in $H_3(N)$. For some $M=\Omega(N/\log N)$, G has M inputs, M outputs and M^2 paths joining inputs to outputs. Furthermore, each vertex lies on O(M) of these paths. Thus, unless $|J|=\Omega(M)$, G-J contains an output with $\Omega(M)$ inputs as predecessors.

A more illuminating proof of Theorem 5.1 can be obtained from the following theorem.

Theorem 5.2: For every N, there is a graph G in $H_3(N)$ such that, for some $M=\Omega(N/\log N)$, G contains a homeomorphic image of every graph in $H^2(N)$.

This theorem is proved by constructing a universal graph (in the sense of Valiant [21]) in $H_3(N)$. Applying the theorem to grates yields Theorem 5.1.

The results of Sections 3 and 4 were restricted to multi-tape Turing machines because the Theorem 2.1 was restricted to graphs in $H_r(N)$. In the case of time versus space, the result of Hopcroft, Paul and Valiant [7] for multi-tape Turing machines have been extended to multi-dimensional and tree-structured tapes (see Paul and Reischuk [13] and Pippenger [16]). This extension was facilitated by the fact that general N-vertex graphs can be one-person pebbled in space $S=O(N/\log N)$. It seems natural, therefore, to inquire about the time-rounds trade-off for the two-person pebble game on general graphs. We have three results bearing on this matter.

Proposition 5.3: For any graph in $H^r(N)$ and any $2\le R\le N$, the Pebbler can win in R rounds and time $T=O(rN/\log R)$.

This proposition is proved by modifying the strategy of Dymond and Tompa [3] to take account of the allowed number of rounds.

Proposition 5.4: For every N and every $1\le R\le N$, there is a graph in $H^2(N)$ for which the Pebbler can win in R rounds only in time $T=\Omega(N/R)$.

This proposition is proved by a construction involving grates [18,19] and expanding graphs [5].

We conjecture that the upper bound of Proposition 5.3 is tight and that the lower bound of Proposition 5.4 can be sharpened to meet it. To this end we offer the following result.

Proposition 5.5: For every N and every $1 \leq R \leq N$, there is a graph in $H^2(N)$ for which the Pebbler can win in R rounds only in time $T=\Omega(N/\log (R \log N))$.

This result is prove by means of a simulation of the two-person pebble game by the one-person pebble game, together with the time-space trade-off for the one-person pebble game due to Lengauer and Tarjan [10]. When $\log R = \Omega(\log \log N)$, this lower bound matches the upper bound of Proposition 5.3. Thus only the case of few rounds remains.

It may seem at first glance that these results rule out any possibility of extending the results of Sections 2-4 to more general machine models. Careful consideration reveals, however, that such a pessimistic conclusion is unwarranted.

Firstly, even Turing machines with multi-dimensional and tree-structured tapes impose constraints on the graphs underlying their computations. We have shown that computation graphs of Turing machines with one-dimensional tapes cannot be linear-sized grates, but the case of two-dimensions is open. Expoliting these weaker constraints may be difficult, however: it is known that the computation graphs of Turing machines with two-dimensional tapes can be linear-sized superconcentrators (see Gabber and Galil [5]), though the case of one dimensional tapes is open.

Secondly, the order of the quantifiers in Proposotions 5.4 and 5.5 (... for every R, there exists a graph ...) is not as strong as it might be (... there exists a graph such that, for every R ...). Unless results with this stronger quantification hold, the possibility of results such as those in Section 4 cannot be excluded. This raises the question of whether there are "uniformly hard" graphs for the two-person pebble game, or whether there are only graphs that are hard for a particular number of rounds. Even for the one-person pebble game, the analogous question (whether there are only graphs that are hard for a particular amount of space) remains open (see Lengauer and Tarjan [10]).

6. References

[1] R. V. Book, S. A. Greibach and B. Wegbreit, "Time and Tape Bounded Turing Acceptors and AFL's", J. Comp. and Sys. Sci., 4 (1970) 606-621.

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

[3] P. Dymond and M. Tompa, "Speedups of Deterministic Machines by Synchronous Parallel Machines", STOC, 15 (1983).

[4] P. Erdős, R. L. Graham and E. Szemerédi, "On Sparse Graphs with Dense Long Paths", Comp. and Math. with Appl., 1 (1975) 365-369.

[5] O. Gabber and Z. Galil, "Explicit Constructions of Linear-Sized Superconcentrators", J. Comp. and Sys. Sci., 22 (1981)407-420.

[6] F. C. Hennie, "One-Tape, Off-Line Turing Machine Computations", Info. and Contr., 8 (1965) 553-578.

[7] J. E. Hopcroft, W. J. Paul and L. G. Valiant, "On Time versus Space", J. ACM, 24 (1977) 332-337.

[8] R. Kannan, "Towards Separarting Non-Deterministic Time from Deterministic Time", FOCS, 22 (1981) 335-343.

[9] R. Kannan, "Alternation and the Power of Non-Determinism", STOC, 15 (1983) 344-346.

[10] T. Lengauer and R. E. Tarjan, "Asymptotically Tight Bounds on Time-Space Trade-Offs in a Pebble Game", J. ACM, 29 (1982) 1087-1130.

[11] P. M. Lewis, R. E. Stearns and J. Hartmanis, "Memory Bounds for the Recognition of Context-Free and Context-Sensitive Languages", FOCS, 6 (1965) 191-202.

[12] R. J. Lipton and R. E. Tarjan, "A Separator

Theorem for Planar Graphs", SIAM J. Appl. Math., 36 (1979) 177-189.

[13] W. Paul and R. Reischuk, "On Time versus Space, II", J. Comp. and Sys. Sci., 22 (1981) 312-327.

[14] W. J. Paul, E. J. Prauss and R. Reischuk, "On Alternation", Acta Inf., 14 (1980) 243-255.

[15] W. J. Paul and R. Reischuk, "On Alternation, II", Acta Inform., 14 (1980) 391-403.

[16] N. Pippenger, "Probabilistic Simulations", STOC 14 (1982) 17-26.

[17] N. Pippenger, "Advances in Pebbling", ICALP 9 (1982) 407-417.

[18] G. Schnitger, "A Family of Graphs with Expensive Depth-Reduction", Theor. Comp. Sci., 18 (1982) 89-93.

[19] G. Schnitger, "On Depth-Reduction and Grates", Theor. Comp. Sci., to appear.

[20] M. Tompa, "A Pebble Game That Models Alternation", in preparation.

[21] L. G. Valiant, "Universal Circuits", STOC, 8 (1976) 196-203.

[22] L. G. Valiant, "Graph-Theoretic Arguments in Low-Level Complexity", MFCS 6 (1977).