SIAM J. COMPUT.
Vol. 16, No. 1, February 1987

SPEED-UP OF TURING MACHINES WITH ONE WORK TAPE AND A TWO-WAY INPUT TAPE*

WOLFGANG MAASS† AND AMIR SCHORR‡

Abstract. In this paper we consider the next more powerful restricted type of Turing machine, which cannot be handled by the existing lower bound arguments: Turing machines with one work tape and a two-way input tape. We show that one can simulate a deterministic Turing machine of this type with time bound $O(n^3)$ by a Σ_2-Turing machine *of the same type* with time bound $O(n^2 \cdot \log^2 n)$. This implies the new separation result $\Sigma_2\,\mathrm{TIME}_1(n) \not\subseteq \mathrm{DTIME}_1(n^3 \cdot \log^6 n)$. Further, we improve Kannan's separation result $\mathrm{NTIME}(n) \not\subseteq \mathrm{DTIME}_1(n^{1.104})$ to $\mathrm{NTIME}(n) \not\subseteq \mathrm{DTIME}_1(n^{1.22})$. Finally we show that with Turing machines of the considered type that use $2k$ alternations, the achieved speed-up increases and for large k approximates $t^{1/2}(n)$. In view of the close relationship between spacebounded Turing machines and time-bounded Turing machines with unboundedly many alternations, this refinement provides a link between our speed-up result and the well-known space-compression result for offline 1-tape Turing machines.

Key words. Turing machines, complexity classes, lower bound arguments

AMS(MOS) subject classifications. 3D10, 3D15, 68C40

1. Introduction. A major goal of machine based complexity theory is the separation of complexity classes that arise from Turing machines with different control structures: deterministic, nondeterministic, and alternating. Since an alternating Turing machine may be viewed as a model for a highly parallel computer, such separation results are relevant for the comparison between sequential and parallel computational models (in addition to their interest for the P versus NP problem). With regard to such separation results one is currently in the following situation. On one hand one can separate deterministic linear time and nondeterministic linear time for the most powerful types of Turing machines: multi-tape Turing machines. But the demonstrated difference between deterministic and nondeterministic complexity classes is "infinitesimal": $\mathrm{NTIME}(n) \not\subseteq \mathrm{DTIME}(n \cdot (\log^* n)^{1/4})$ (Paul, Pippenger, Szemeredi and Trotter [11]). On the other hand more significant time differences between determinism and nondeterminism have been demonstrated for restricted types of Turing machines: a quadratic difference for 1-tape Turing machines without input tape by Hennie [3], and for 1-tape Turing machines with an additional 1-way input tape by Maass [7].

Since the more abstract methods that yield the separation result for multi-tape Turing machines can probably not be used to prove more significant time differences (see the discussion in [11]), it appears reasonable to look for more concrete arguments that yield larger time differences for increasingly more powerful *restricted* types of Turing machines (the goal is to approximate the power of 2-tape Turing machines from below). In this paper we attack the next more powerful restricted type of Turing machine that cannot be handled by previously developed concrete lower bound arguments: Turing machines with one work tape and a *two*-way input tape.

In § 2 we show that one can speed up deterministic Turing machines of this type by Turing machines of the same type (i.e. one work tape and a 2-way input tape) that use a bounded number of alternations. In particular we show that

$$\mathrm{DTIME}_1(t(n)) \subseteq \Sigma_2\,\mathrm{TIME}_1(t^{2.3}(n) \cdot \log^2 t(n)) \quad \text{for } t(n) \geqq n^3,$$

* Received by the editors October 16, 1985; accepted for publication (in revised form) April 28, 1986.

† Department of Mathematics, Statistics, and Computer Science, University of Illinois, Chicago, Illinois 60680. This author's work was supported in part by the National Science Foundation under grant DCR-8504247.

‡ Department of Computer Science, Tel-Aviv University, Ramat-Aviv, Tel-Aviv, Israel.

and more generally, that

$$\mathrm{DTIME}_1(t(n)) \subseteq \Sigma_{2k}\,\mathrm{TIME}_1(t^{(k+1)/(2k+1)}(n)\cdot \log^2 t(n))$$

$$\text{for } k \geqq 1 \text{ and } t(n) \geqq n^{2+1/k}$$

(see below for definitions).

In § 3 we derive from this speed-up result two new separation results for the considered type of Turing machine:

$$\Sigma_{2k}\,\mathrm{TIME}_1(n) \not\subseteq \mathrm{DTIME}_1(n^{(2k+1)/(k+1)}/\log^6 n) \quad \text{for all } k \geqq 1,$$

and

$$\mathrm{NTIME}(n) \not\subseteq \mathrm{DTIME}_1(n^{1.22}).$$

The following definitions are used in this paper. A k-tape Turing machine (TM) is a TM with k work tapes (each with one head) and an additional 2-way input tape. This 2-way input tape is an extra read-only tape that is occupied by the input together with a left and a right endmarker. The associated (read-only) input head can move in both directions ("2-way").

Alternating TM's were introduced by Chandra, Kozen and Stockmeyer [2]. Each state of an alternating TM is either existential or universal. One calls a node v in the computation tree of an alternating TM M for some input w *accepting* if

—either M is at v in an existential state and there exists a son of v that is accepting,

—or M is at v in a universal state and all the sons of v are accepting,

—or M is at v in an accepting final state.

M accepts w if the root of this computation tree is accepting. M is called a Σ_m-TM if for all inputs w the computation tree of M starts with an existential state and alternates along each computation path at most $m-1$ times between existential and universal states. Thus Σ_1-TM's are just nondeterministic TM's. Deterministic TM's ($=\Sigma_0$-TM's) may be viewed as special cases of nondeterministic TM's, where the transition function is single-valued.

For any such TM, we say that M *is of time complexity* $t(n)$ if for every accepted input w of length n the computation tree for input w stays accepting if it is pruned at depth $t(n)$ (see Paul, Prauss and Reischuk [9]). This definition is convenient insofar as one does not have to worry too much about the time-constructibility of $t(n)$ (otherwise one might demand instead that the depth the depth of the computation tree of M on input w is bounded by $t(n)$).

We define for arbitrary functions $t(n) \geqq n$ $\Sigma_m\,\mathrm{TIME}_k(t(n)) = \{L \subseteq \{0,1\}^* \mid L$ is accepted by a k-tape Σ_m-TM of time complexity $O(t(n))\}$.

Among related results, we would like to mention (besides the already discussed paper [11]) those by Paul, Prauss and Reischuk [9], [10] and Kannan [5]. It is shown in [9] that every nondeterministic 1-tape TM *without* input tape, of time complexity $t(n)$, can be simulated by an alternating 1-tape TM of time complexity $O(n+t^{1/2}(n))$, which uses an *unbounded* number of alternations. Kannan [5] shows that any deterministic 1-tape TM with two-way input tape of time complexity n^{2l} ($l \geqq 1$ integer) can be simulated by a *multitape* Σ_4-TM of time complexity $O(n^{1+(4/3)\cdot l}\log n)$. This implies the separation result $\mathrm{NTIME}(n) \not\subseteq \mathrm{DTIME}_1(n^{1.104})$, which we improve in this paper to $\mathrm{NTIME} \not\subseteq \mathrm{DTIME}_1(n^{1.22})$.

Compared with the preceding results on the speed-up of deterministic TM's by alternating TM's, we use in this paper a somewhat different strategy in order to speed

up the considered restricted type of deterministic TM by a TM of the *same* restricted type, which uses only a *constant* number of alternations. This allows us to separate complexity classes that arise from restricted TM's, which differ *only* in their control structure.

Our speed-up from $t(n)$ to $t^{(k+1)/(2k+1)}(n) \cdot \log^2 t(n)$ with $2k-1$ alternations approximates $t^{1/2}(n)$ (for $(2k-1 \to \infty)$. This fits nicely together with the known time-space tradeoff for the considered restricted TM's. Space-bounded TM's are closely related to TM's which use an unbounded number of alternations (it follows from [2] and [9] that $\mathrm{ATIME}_1(t(n)) \subseteq \mathrm{DSPACE}_1(t(n)) \subseteq \mathrm{ATIME}_1(t^2(n))$). Thus the well-known result $\mathrm{DTIME}_1(t(n)) \subseteq \mathrm{DSPACE}_1(t^{1/2}(n))$ of Paterson [8] (see Ibarra and Moran [4] and Kurtz and Maass [6] for some recent refinements) as well as the mentioned result from [9] may be viewed as limit cases of the speed-up result in this paper.

2. Speed-up by bounded alternation.

THEOREM 2.1. *For every function* $t(n) \geqq n$

$$\mathrm{DTIME}_1(t(n)) \subseteq \Sigma_2\,\mathrm{TIME}_1(t^{2/3}(n) \cdot \log^2 t(n) + t^{1/3}(n) \cdot n).$$

In particular for every function $t(n) \geqq n^3$

$$\mathrm{DTIME}_1(t(n)) \subseteq \Sigma_2\,\mathrm{TIME}_1(t^{2/3}(n) \cdot \log^2 t(n)).$$

Proof. Let M be a deterministic TM of time-complexity $t(n)$ with one work tape and a two-way input tape. We construct a Σ_2-TM $\tilde{M}$ with one work tape and a two-way input tape, that accepts the same inputs as M and is of the desired time complexity.

For any fixed input w of length n that is accepted by M, the simulating TM $\tilde{M}$ guesses a suitable separator for the computation graph of M. Subsequently $\tilde{M}$ verifies in parallel that for each component of the remaining graph the associated guesses are consistent.

We first describe the design of the separator. One partitions the work tape of M into blocks B of length $t^{1/3}(n)$ in such a way that the work head of M crosses at most $t^{2/3}(n)$ often a block boundary. Such partition exists, since there are $t^{1/3}(n)$ different partitions into blocks of length $t^{1/3}(n)$. Further, the *sum* of the number of crossings of block boundaries for all $t^{1/3}(n)$ partitions is bounded by $t(n)$ (since at every step the work head of M crosses for at most one partition a block boundary).

For each of those blocks B of the fixed partition of M's work tape such that M's head spends altogether at least $t^{2/3}(n)$ steps in B or leaves and re-enters B at least $t^{1/3}(n)$ often, one partitions the sequence of steps where M's work head is inside B into time blocks T_B. Each time block T_B is chosen as large as possible, subject to these two constraints:

1) T_B consists of at most $t^{2/3}(n)$ steps.

2) T_B contains at most $t^{1/3}(n)$ steps where the work head of M leaves B at the next step (note that in such a case the next element of T_B is the first subsequent step where the head re-enters B—unless we have reached the end of T_B).

For the previously described partition of space and time $\tilde{M}$ writes (during the $\exists$-space of the accepting subtree of its computation on input w) the following "guessed" data on its work tape. These data are arranged in a particular way, to support the subsequent "verification" during $\tilde{M}$'s $\forall$-phase. For the fixed $n = |w|$ we write from now on t instead of $t(n)$.

Starting with the leftmost tape block B that is visited by M's work head, $\tilde{M}$ writes from left to right for each block B the following guessed data associated with B on its work tape.

If those steps, where M's work head is inside B, have been partitioned into time blocks T_B, $\tilde{M}$ writes from left to right for each time block T_B the inscription of B at the beginning and at the end of T_B on two tracks of a tape segment S of length $t^{1/3}$. On two additional tracks $\tilde{M}$ marks on tape segment S the position of M's work head at the beginning and end of T_B together with the state of M and the position of M's input head (represented by a number from $\{1, \cdots, n\}$ in binary code) at these two steps. To the left and right of the considered segment S of $\tilde{M}$'s work tape $\tilde{M}$ writes those entries of the crossing sequence of M at the left and right ends of block B, that are associated with steps in time block T_B. By our choice of the length of T_B there are at most $t^{1/3}$ such entries.

Each entry in a crossing sequence for a boundary b on M's work tape consists in the present construction of the current state and input head position of M, together with the number of preceding crossings of this boundary b (represented by a number between 0 and $t-1$ in binary code). The latter number will be useful in the subsequent verification phase, where we have to verify in particular that the guessed crossing sequence for the right end of a tape block B agrees with the guessed crossing sequence for the left end of the next tape block B' to the right of B. The bit-length of a segment of a crossing sequence with up to $t^{1/3}$ entries is of the order $O(t^{1/3} \cdot \log t)$.

After $\tilde{M}$ has written on its work tape the described guesses associated with time block T_B, it writes to the right of it the analogous guesses for the next time block T'_B for the same tape block B. After all time blocks for tape block B are handled, $\tilde{M}$ writes to the right of these guesses those guesses that are associated with the next block B' to the right of B.

In the case that the steps where M's work head is inside B have *not* been partitioned into time blocks T_B, $\tilde{M}$ guesses, instead of the preceding, just the crossing sequences for the left and right ends of B. $\tilde{M}$ also places markers in distances of $t^{1/3}$ along the guessed string (to provide a tape segment that has exactly the same length as B for the subsequent simulation of the computation inside B). If the work head of M happens to be inside B during the first or last step of M's computation, $\tilde{M}$ guesses in addition the state and head positions of M for these two steps. Note that for the blocks B considered here, $\tilde{M}$ does not guess the initial or final tape content of B (in order to keep the total length of its guesses sufficiently short).

For the chosen partition of M's work tape there are at most $t^{2/3}$ crossings of block boundaries. Thus all guesses of segments of crossing sequences together require $O(t^{2/3} \cdot \log t)$ bits on the work tape of $\tilde{M}$.

It is also easy to see that there are altogether at most $O(t^{1/3})$ many time blocks T_B, since every time block either consists of $t^{2/3}$ steps, or involves $t^{1/3}$ crossings of a boundary of B, or is the last one of at least two time blocks for the same tape block B. To verify this estimate, we observe that there are altogether at most $t^{1/3}$ time blocks T_B that consist of $t^{2/3}$ steps (because M uses only t steps), and altogether at most $t^{1/3}$ time blocks T_B that contain $t^{1/3}$ steps where M's head leaves B at the next step (because at most $t^{2/3}$ crossings of block boundaries occur).

The preceding implies that the guessed initial and final inscriptions of blocks B for time blocks T_B occupy together at most $O(t^{2/3}(n))$ cells on the work tape of $\tilde{M}$. Thus the total guessing phase of $\tilde{M}$ requires no more than $O(t^{2/3} \cdot \log t)$ bits.

We would like to point out that M's work head may visit during its computation up to $t^{2/3}(n)+2$ many different tape blocks B. Therefore $\tilde{M}$ could not afford to guess

initial and final tape inscriptions for all these blocks B. This explains why blocks B with and without time blocks T_B were handled differently in the preceding.

In the following $\forall$-phase the Σ_2-TM $\tilde{M}$ verifies in parallel the following facts about the guessed data:

FACT 1. *For each time block T_B the following guessed data are consistent: the initial and final inscriptions of block B, the initial and final state and head positions of M, and the associated segments of the crossing sequences for the left and right boundary of B.*

To verify this consistency, $\tilde{M}$ simulates the corresponding computation steps of M, sweep by sweep. Each time when $\tilde{M}$ simulates another sweep over B, its work head moves first to the next entry of the corresponding crossing sequence. The length of this move can be estimated by the bit length of the segments of the two crossing sequences associated with T_B: $O(t^{1/3} \cdot \log t)$. Once $\tilde{M}$'s work head has arrived at the next entry of the crossing sequence, $\tilde{M}$ brings its input head into the position that is prescribed by the corresponding binary number between 1 and n in this entry of the crossing sequence.

This procedure takes $O(n)$ steps. We assume that the subsequent simulation of the next sweep of M's work head over B is simulated on the same tape segment of $\tilde{M}$'s work tape, where it has recorded its guess of the initial and final block content for T_B (we may use an extra track for this simulation). After the simulation of each sweep over B, $\tilde{M}$ moves its work head to the corresponding next entry of the crossing sequence, in order to compare its current state and input head position with the corresponding guessed data.

Of course $\tilde{M}$ starts the simulation of the first sweep over B that belongs to T_B with the guessed initial content of B and with the guessed state and head position. Analogously, after $\tilde{M}$ has simulated the last sweep of T_B over B, it compares the current content of the track that simulates B and the current state and head positions with the guessed data for T_B.

It is obvious that the described verification of the consistency of the guessed data for a time block T_B requires at most

$$O(t^{2/3} \cdot \log t + t^{1/3} \cdot n)$$

steps.

FACT 2. *For each block B without time blocks the guessed data are consistent.*

This is verified by simulating all sweeps of M's work head over B, in the same way as above (initially B is blank). The fact that B has no time blocks implies that the same time bound as in Fact 1 holds.

FACT 3. *For any two consecutive time blocks T_B and T'_B for the same tape block B the guessed final block inscription, state and head positions of T_B agree with the initial block inscription, state and head positions of T'_B.*

Here $\tilde{M}$ has to compare two strings of length $O(t^{1/3})$, that are written at most $O(t^{1/3} \cdot \log t)$ cells apart on $\tilde{M}$'s work tape (only the associated segments of crossing sequences are written in between). This procedure takes $O(t^{2/3} \cdot \log t)$ steps.

FACT 4. *For any two consecutive tape blocks B and B', each entry in the crossing sequence for the right end of B agrees with the corresponding entry in the crossing sequence for the left end of B'.*

More precisely $\tilde{M}$ checks here the following. If $\tilde{M}$ places markers above any entry e_1 in the crossing sequence at the right end of B and any entry e_2 in the crossing sequence at the left end of B', and if $\tilde{M}$ verifies that both entries have assigned the same running number (in their respective crossing sequence), then e_1 and e_2 agree.

The considered entries e_1, e_2 are located on $\tilde{M}$'s work tape at most $O(t^{2/3} \cdot \log t)$ cells apart. The work head of $\tilde{M}$ has to cover the distance in between at most $O(\log t)$ often in the described procedure. Therefore this procedure takes $O(t^{2/3} \cdot \log^2 t)$ steps.

FACT 5. *Consecutive entries in the same crossing sequence have been assigned consecutive running numbers.*

Two consecutive entries may be assigned to two different time blocks. However they are at most $O(t^{1/3} \cdot \log t)$ cells apart. Thus for any two consecutive entries $\tilde{M}$ can verify this in $O(t^{1/3} \cdot \log^2 t)$ steps.

FACT 6. *For exactly one block B the computation starts in B (with correct initial state and initial head positions) and for exactly one block B' the computation ends in B' in an accepting final state.*

$\tilde{M}$ can verify this with $O(1)$ sweeps over the guessed data.

The description of $\tilde{M}$'s $\forall$-phase is now complete.

The preceding arguments imply that if input w of length n is accepted by M with $t(n)$ steps, $\tilde{M}$ accepts w via a subtree of the computation tree that has depth $O(t^{2/3}(n) \cdot \log^2 t(n) + t^{1/3}(n) \cdot n)$. On the other hand one constructs $\tilde{M}$ so that whenever $\tilde{M}$ accepts some input w (with a subtree of arbitrary depth), then M also accepts w. Therefore $\tilde{M}$ accepts the same language as M, and $\tilde{M}$ is of time complexity $O(t^{2/3}(n) \cdot \log^2 t(n) + t^{1/3}(n) \cdot n)$.

THEOREM 2.2. *For all natural numbers $k \geqq 1$ and every function $t(n) \geqq n$*

$$\mathrm{DTIME}_1(t(n)) \subseteq \Sigma_{2k}\,\mathrm{TIME}_1(t^{(k+1)/(2k+1)}(n) \cdot \log^2 t(n) + t^{1/(2k+1)}(n) \cdot n).$$

In particular for $t(n) \geqq n^{2+1/k}$

$$\mathrm{DTIME}_1(t(n)) \subseteq \Sigma_{2k}\,\mathrm{TIME}_1(t^{(k+1)/(2k+1)}(n) \cdot \log^2 t(n)).$$

Proof. The proof is analogous to that of Theorem 2.1. We use the additional alternations of the Σ_{2k}-TM $\tilde{M}$ to refine the partition of the given computation of the deterministic TM M. For a given input w of length n we cut the work tape of M in such a way into blocks B of length $t^{k/(2k+1)}$, that M's work head crosses in the computation on input w at most $t^{(k+1)/(2k+1)}$ often a block boundary (again we write t instead of $t(n)$). Further for those blocks B in which M spends $t^{2k/(2k+1)}$ steps, or whose boundary it crosses $t^{k/(2k+1)}$ often, one cuts the sequence of steps where M's work head is inside B into time blocks T_B with $\leqq t^{2k/(2k+1)}$ steps and $\leqq t^{k/(2k+1)}$ crossings. This can be done in such a way that only $O(t^{1/(2k+1)})$ time blocks arise. Thus $\tilde{M}$ can afford to guess for all time blocks T_B the intermediate contents of block B. In the subsequent $\forall$-phase $\tilde{M}$ starts to check in parallel that for each time block T_B the guessed data are consistent. Since a direct simulation of T_B would take too long, $\tilde{M}$ uses the second $\exists$-phase to cut T_B into smaller time blocks $T_B^{(2)}$ that consist of $\leqq t^{(2k-1)/(2k+1)}$ steps and have $\leqq t^{(k-1)/(2k+1)}$ crossings. Again this can be done so that each time block T_B consists of at most $O(t^{1/(2k+1)})$ smaller time blocks $T_B^{(2)}$. If necessary, one has to cut also for those blocks B that have no time blocks the steps that M spends inside B into smaller time blocks $T_B^{(2)}$ (of the same maximal size as above).

By continuing this process of guessing in parallel, one finally arrives in the kth $\exists$-phase at very small time blocks $T_B^{(k)}$ with $\leqq t^{(k+1)/(2k+1)}$ steps and $\leqq t^{1/(2k+1)}$ crossings. The consistency of the guessed data for each $T_B^{(k)}$ can be checked in parallel by a simulation of the corresponding segments of M's computation.

Similarly as in the proof of Theorem 2.1 several additional facts about the guessed data have to be verified to make sure that whenever $\tilde{M}$ accepts w (with an arbitrary number of steps) then M also accepts w. In particular one has to check that for any two subsequent smaller time blocks $T_B^{(i+1)}$ the guessed content of B at the end of the

first one agrees with the guessed content of B at the beginning of the second one. A straightforward verification of this fact would take too long. Therefore $\tilde{M}$ guesses in addition a partition of B into mini-blocks of length $t^{1/(2k+1)}$. Then $\tilde{M}$ can verify the desired agreement of guessed inscriptions of B in parallel, separately for each mini-block of B.

3. Application to separation problems.

THEOREM 3.1. *For every natural number* $k \geqq 1$

$$\Sigma_{2k}\,\mathrm{TIME}_1(n) \not\subseteq \mathrm{DTIME}_1\left(\frac{n^{(2k+1)/(k+1)}}{\log^6 n}\right).$$

Remark 3.2. Note that the TM's that are considered in Theorem 3.1 differ *only* in their control structure, not in their memory or input device.

Proof. This separation result follows from the preceding speed-up result (Theorem 2.2) in combination with an adaption of standard diagonalization and padding techniques to the considered type of TM.

Via diagonalization, one shows that

$$\mathrm{DTIME}_1(n^{2+1/k}) \not\subseteq \mathrm{DTIME}_1\left(\frac{n^{2+1/k}}{\log n \cdot \log^* n}\right)$$

(besides the program of the simulated machine the simulating machine also moves a step counter along with work head; this causes the factor $1/\log n$ on the right-hand side). Together with Theorem 2.2 this implies that

$$\Sigma_{2k}\,\mathrm{TIME}_1(n^{(k+1)/k} \cdot \log^2 n) \not\subseteq \mathrm{DTIME}_1\left(\frac{n^{2+1/k}}{\log n \cdot \log^* n}\right).$$

Via padding, one can derive from this

$$\Sigma_{2k}\,\mathrm{TIME}_1(n) \not\subseteq \mathrm{DTIME}_1\left(\frac{n^{(2k+1)/(k+1)}}{\log^{(6k+4)/(k+1)} n \cdot \log^* n}\right)$$

(this is straightforward but not completely trivial: in order to simulate a TM $\tilde{M}$ that processes the padded input by a TM M that processes the unpadded input, M has to record the current position of $\tilde{M}$'s input head via a binary counter). This implies the claim of the theorem.

The following theorem improves a separation result by Kannan [4], who had shown that $\mathrm{NTIME}(n) \not\subseteq \mathrm{DTIME}_1(n^{1.104})$. It is well known that $\mathrm{NTIME}(n) = \mathrm{NTIME}_2(n)$ [1].

THEOREM 3.3. $\mathrm{NTIME}(n) \not\subseteq \mathrm{DTIME}_1(n^{1.22})$.

Proof. We use a method of Paul and Reischuk [10] to convert a computation with alternations into a deterministic computation. Assume for a contradiction that $\mathrm{NTIME}(n) \subseteq \mathrm{DTIME}_1(n^{1.22})$. With the help of padding one can show that this implies $\mathrm{NTIME}(n^{1.22}) \subseteq \mathrm{DTIME}_1(n^{1.22})^2 \cdot \log n)$. We apply both inclusions to a linear time Σ_2-TM M with one work tape and a two-way input tape such that

$$L(M) \in \Sigma_2\,\mathrm{TIME}_1(n) - \mathrm{DTIME}_1\left(\frac{n^{3/2}}{\log^6 n}\right)$$

(such TM M exists by Theorem 3.1). For an arbitrary fixed input w of length n one replaces in the computation tree of M on input w each maximal $\forall$-subtree by a deterministic computation of length $O(n^{1.22})$ (we use here the first inclusion). The resulting computation tree can be construed as a nondeterministic computation of

length $O(n^{1.22})$ on a two-tape TM $\tilde{M}$ (it would be nice if we could view $\tilde{M}$ as a TM with *one* work tape and a two-way input tape, because then we could replace $\mathrm{NTIME}(n)$ by $\mathrm{NTIME}_1(n)$ in the claim of the theorem; however the implanted deterministic TM of time complexity $O(n^{1.22})$ uses both the content of M's input tape and the content of M's work tape at the beginning of the $\forall$-phase as its input). Thus we can apply the second inclusion. In this way one derives

$$L(M) \in \mathrm{DTIME}_1(n^{(1.22)^2} \cdot \log n) \subseteq \mathrm{DTIME}_1(n^{3/2}/\log^6 n),$$

a contradiction to the choice of M.

Note that the constant 1.22 that arises in this separation result can be replaced by any number less than $\sqrt{3/2}$.

Acknowledgment. We would like to thank the referees for their helpful comments.

REFERENCES

[1] R. V. Book, S. A. Greibach and B. Wegbreit, *Time and tape bounded Turing acceptors and* AFL's, J. Comput. System Sci., 4 (1970), pp. 606–621.

[2] A. K. Chandra, D. Kozen and L. J. Stockmeyer, *Alternation*, J. Assoc. Comput. Mach., 28 (1981), pp. 114–133.

[3] F. C. Hennie, *One-tape, off-line Turing machine computations*, Inform. and Control, 8 (1965), pp. 553–578.

[4] O. H. Ibarra and S. Moran, *Some time-space trade-off results concerning single tape and offline Turing machines*, this Journal, 12 (1983), pp. 388–394.

[5] R. Kannan, *Alternation and the power of nondeterminism*, Proc. of the 15th ACM Symposium on the Theory of Computing, 1983, pp. 344–346.

[6] S. A. Kurtz and W. Maass, *Some time, space and reversal tradeoffs*, Inform. and Control, to appear.

[7] W. Maass, *Combinatorial lower bound arguments for deterministic and nondeterministic Turing machines*, Trans. Amer. Math. Soc., 292 (1985), pp. 675–693.

[8] M. Paterson, *Tape bounds for time-bounded Turing machines*, J. Comput. System Sci., 6 (1972), pp. 116–124.

[9] W. J. Paul, E. J. Prauss and E. J. Reischuk, *On alternation*, Acta Informatica, 14 (1980), pp. 243–255.

[10] W. J. Paul and E. J. Reischuk, *On alternation* II, Acta Inform., 14 (1980), pp. 391–403.

[11] W. J. Paul, N. Pippenger, E. Szemeredi and W. Trotter, *On determinism versus nondeterminism and related problems*, Proc. 24th IEEE Symposium on the Foundation of Computer Science, 1983, pp. 429–438.