Accelerating Nondeterministic Single-Tape Off-Line Computations by One Alternation

Wiedermann, Jiří

1997

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Accelerating Nondeterministic Single–Tape Off-Line Computations by One Alternation

Jiří Wiedermann

Technical report No. 725

October, 1997

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+4202) 66 05 35 20 fax: (+4202) 85 85 789
e-mail: wieder@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Accelerating Nondeterministic Single–Tape Off-Line Computations by One Alternation

Jiří Wiedermann[1]

Technical report No. 725
October, 1997

Abstract

It is shown that for a well–behaved function $T(n)$ any nonderministic single–tape off–line Turing machine of time complexity $T(n)$ can be speeded–up by one extra alternation by the factor $\log\log T(n)/\sqrt{\log T(n)}$. This leads to the separation $\mathsf{NTIME}_{1+\mathrm{I}}(T(n)) \subset \Sigma_2 - \mathsf{TIME}_{1+\mathrm{I}}(T(n))$ of the respective complexity classes. Analogous result holds also for the complementary classes $\mathsf{co}\text{-}\mathsf{NTIME}_{1+\mathrm{I}}(T(n))$ and $\Pi_2\text{-}\mathsf{TIME}_{1+\mathrm{I}}(T(n))$. This is the first occasion where such results have been proved for a restricted type of multitape nondeterministic machines. For the general case of multitape nondeterministic machines similar results are not known to hold.

Keywords
Turing Machines, Nondeterminism, Alternation, Complexity Classes

[1]This research was partially supported by GA ČR Grant No. 201/95/0976 "HYPERCOMPLEX" and partly by EU Grant No. INCO–COP 96–0195 "ALTEC-KIT" and by OK 304 Grant of Ministry of Education of the Czech Republic

1 Introduction

The separation of complexity classes within the Σ hierarchy
$\mathsf{DTIME}(T(n)) \subseteq \mathsf{NTIME}(T(n)) \subseteq \Sigma_2\text{-}\mathsf{TIME}(T(n)) \subseteq \ldots \mathsf{ATIME}(T(n)) \subseteq \mathsf{DSPACE}(T(n))$
is one of the central problems in complexity theory. The first proof that nondeterminism is stronger than determinism comes probably from Hennie [2] who in 1965 proved this result for single tape machines recognizing non–palindroms. For multitape machines the separation[2] $\mathsf{DTIME}(n) \subset \mathsf{NTIME}(n)$ has been proved in 1983 by Paul, Pippenger, Szemerédi, and Trotter [11]. In 1973 Paterson [9] proved that deterministic space is more powerful than deterministic time for single–tape machines. The analogous result for multitape machines $\mathsf{DTIME}(T(n)) \subset \mathsf{DSPACE}(T(n))$ followed in 1977 by Hopcroft, Paul and Valiant [5]. In 1980 Paul, Prauss, and Reischuk [10] proved that unbounded number of alternations can speedup single–tape computations. The separation for multitape machines ensued in 1983 by Dymond and Tompa [3]: $\mathsf{DTIME}(T(n)) \subset \mathsf{ATIME}(T(n))$. Finally, Kannan in 1983 and Maass and Schorr in 1987 proved increasingly better results that bounded number of alternations can speedup deterministic single tape computations (also with a separate input tape). For the case of multitape machines the respective result $\mathsf{DTIME}(T(n)) \subset \Sigma_2\text{-}\mathsf{TIME}(T(n))$ proved Gupta [4] in 1988. Thus, merely two alternation were enough to separate the latter complexity classes.

There are two lessons to be taken from the previous short historical excursion in the context of the present paper. The first lesson is that in all cases more general results for multitape machines always followed only after proving similar results for simpler types of machines. Thus, results for restricted machines served as inspiration for looking for similar results on more general models of computing. Second, as a rule, all the respective results separate *deterministic* time from some higher complexity class in the above mentioned hierarchy. This is because it seemed that there were principal reasons that prevented the application of analogous speed–up techniques also in the case of nondeterministic time. Roughly, in some cases it was the impossibility efficiently rerunning a piece of nondeterministic computation twice along the same computational path (cf. [5]), or "unavailability" of nondeterminism without adding a further alternation (cf. [8]).

The problem of efficient speedup of nondeterministic computations has been identified as the major roadblock that prevents any further progress in separating other complexity classes as before (cf. [4]).

Nevertheless, it appears that this problem has a solution, so far at least for restricted Turing machines. In order to see the key idea of such a solution we have to return to the idea of the best separation result that separates single–tape off–line deterministic time bounded computations from Σ_2 single tape off–line computations [8].

This proof, and also other simulation proofs achieving speedup by two alternations (cf. [4]) share roughly the same strategy. In the first, nondeterministic phase a space efficient description of size $o(T(n)$ of the original deterministic computation is guessed. The correctness of this guess is in turn verified in the second phase by invoking the

[2]The inclusion symbol '$\subset$' denotes the proper containment.

parallelism offered by co–nondeterminism.

Unfortunately, this natural idea of simulation cannot be straightforwardly transferred to the nondeterministic case. This is because in the above mentioned approach the verification phase requires replaying of some pieces of the original computation. Thus, when the original computation was a nondeterministic one it appeared that there was no way of its efficient verification during one subsequent alternation.

The first solution of the problem of speeding–up single–tape nondeterministic computations by one alternation has been devised in 1996 in [12].

The previously mentioned obstacle was roundabout by reorganizing the above mentioned two phase schema of similar proofs. Namely, the original nondeterministic computation was nonderministically split into very small pieces in order to achieve that among them many pieces were equal. The equal pieces of computations were eliminated and only the correctness of remaining different ones was checked, still in the first phase. In the second phase the correctness of split and that of the elimination was checked.

This has lead to a speedup by factor $\log T(n)$ by one extra alternation. A separation of the respective time complexity classes followed: $\mathsf{NTIME}_1(T(n)) \subset \Sigma_2\text{-}\mathsf{TIME}_1(T(n))$. In its machine category this has been a much stronger result when compared with all the previous results since it achieves a speed–up by the single alternation and separates two directly neighbouring complexity classes.

The present paper continues attacking the problem of speeding up nondeterministic computations by one alternation for the next more powerful type of restricted Turing machines — viz. Turing machines with one work tape and extra read–only two–way input tape (or shortly: single–tape off–line machines). Note that this type of machines is an intermediate type between single tape machines without an input tape, and two tape off–line machines. Due to the result of Book et al. [1] in nondeterministic case the latter machines are time equivalent to multitape off–line machines.

In Section 2 a speed–up theorem for single–tape off–line nondeterministic machines by one alternation is proved. As compared to previous cases a new simulation strategy has been developed here in order to reflect the qualitative change in machine architecture given by the addition of input tape as that of the second tape. This has called for two substantial changes. First, an idea of having a space efficient representation of a *log* of input head movements has been implemented. Second, verification of very short nondeterministic pieces of computations has been moved back to co–nonderministic phase where their correctness is checked by their *deterministic simulation* in exponential time w.r.t. their length. Efficient realization of both ideas leads to a speed–up of order $\log\log T(n)/\sqrt{\log T(n)}$.

Consequently, in Section 3 it is shown that for single–tape off–line machines nondeterministic time $T(n)$ is strictly contained in Σ_2–time $T(n)$.

The previous results hold also for the case of complementary machines. Thus, for deterministic single–tape off–line machines two alternations lead to more efficient computations than a single alternation.

The results presented in this paper have so far no counterpart in the case of nondeterministic multitape machines. Nevertheless, they open the door in the respective direction by pointing to a proof technique that is strong enough to capture the efficiency difference between single–tape and (restricted) two tape nondeterministic computations.

2 Speed-up

For the efficient speed–up simulation we are after it is essential that the nondeterministic machine to be simulated works in the small space. Fortunately, it is possible to make this assumption without any loss of generality due to the following theorem [7] which we shall give without a proof:

Theorem 2.1 *Let $\sqrt{T(n)\log n}$ be constructible in time $T(n)$, with $T(n) \geq n^2/\log n$. Than any $T(n)$–time bounded single–tape off–line nondeterministic Turing machine $\mathcal{M}$ can be simulated in linear time and in space $O(\sqrt{T(n)\log n})$ by a machine of the same type.*

Now we are in a position to formulate and prove our main result. The proof of the respective theorem has been partly inspired by the proof of a similar theorem that was proved in [12] for the case of single–tape nonderministic TM *without input tape*. Thus, the difference in current proof captures the presence of input tape; this, however, leads to non trivial changes both in simulation strategy, as well as in the respective data representation.

The idea is as follows. The computation of the machine to be simulated is represented with the help of so–called rectangular representation (this techniques goes back to Paterson [9]). The positions of input head at selected times are represented in a compressed way with the help of a so–called *log*, in which in chronological order the differences between any two subsequent positions of input head are recorded. The size of rectangles is selected so as to enable *deterministic* and w.r.t. $T(n)$ also a sublinear time verification of any *nondeterministic* piece of computation as described by the given rectangle, with the help of information from the above mentioned log. Thus, the simulation of the original machine $\mathcal{M}$ by the respective Σ_2–machine $\mathcal{S}$ consists of two phases: in the first, nondeterministic phase the respective rectangular representation and the log of input head movement is guessed and recorded and, in the second, co–nondeterministic (universal) phase the previous guesses are verified in parallel. Moreover, in order to make the verification process efficient enough, prior to the simulation the computation of the original machine is first transformed into an equivalent one, with the help of the previous theorem, and then "cut" into certain time segments that have the property that the contents of working tape is completely rewritten at the end of each time segment. As a result, the history of cell rewritings (that is needed to verify the correctness of rectangular representation) can be completely verified by performing the respective verification in parallel for each time segment.

In order to avoid worries concerning the time–constructibility of $T(n)$ we shall make use of the following definition of time complexity for the alternating machines (cf. [10]). We shall say that an alternating machine $\mathcal{M}$ is of *time complexity* $T(n)$ if for every accepted input of length n the respective computation tree of $\mathcal{M}$ stays accepting if it is pruned at depth $T(n)$.

Theorem 2.2 *Let $T(n) \geq n^2/\log n$. Than any $T(n)$–time bounded single–tape off–line nondeterministic Turing machine $\mathcal{M}$ can be simulated by a single–tape off–line Σ_2–machine $\mathcal{S}$ in time $O(T(n)\log\log T(n)/\sqrt{\log T(n)})$.*

Proof Outline. According to the statement of theorem 2.1, w.l.o.g. we can assume that $\mathcal{M}$ is of space complexity $O(\sqrt{T(n)\log n})$.

Split now the computation of $\mathcal{M}$ into $O(T^{1/3}(n))$ *time segments* of length $O(T^{2/3}(n))$ each and introduce a so–called *sweep* at the end of each time segment. A sweep consists of one complete traversal of $\mathcal{M}$'s working head over the entire rewritten part of $\mathcal{M}$'s working tape — i.e., the working head moves to the right end of the rewritten part of working tape, then to the left end and finally returns to its marked original position. Make each sweep a part of the respective time segment. Clearly, this transformation does not influence the time complexity of the resulting machine substantially — the resulting machine still works in time $O(T(n))$.

Now, consider the respective time–space computational diagram (i.e., the sequence of instantaneous descriptions of $\mathcal{M}$'s working tape, written one above the other, starting with the initial and ending with the final accepting instantaneous description), with the recorded trajectory of $\mathcal{M}$'s working head movement during the computation. Split this diagram by vertical lines into *slots* of equal size $b(n) = \lceil\sqrt{\log(T(n)/\log^3 T(n))}\rceil$ (the last slot can be shorter).[3] Consider the crossing sequence at the boundaries between individual slots (i.e., the sequence of points where the trajectory of input head crosses the above mentioned vertical lines). By shifting all slot boundaries simultaneously along the tape from its origin to the right, while keeping them equidistant, at some position j, with $0 < j < b(n)$, a situation must occur that the sum of lengths of crossing sequences at the current slot boundaries does not exceed $\ell(T(n) = T(n)/b(n)$. Namely, in the opposite case, if there was not such a position j, then the total sum of lengths of crossing sequences in between every tape cell would exceed $T(n)$, what is impossible.

Fix the first slot boundary at the position j. In the resulting diagram, draw horizontal lines to denote the boundaries between any two subsequent time segments. This will split the diagram into $T^{1/3}(n).\sqrt{T(n)\log n}/b(n) = O(T^{5/6}(n))$ of so–called *first order rectangles.* Further, split horizontally each first order rectangle into *second order rectangles* whose size is maximized, subject to the satisfaction of either of the following two conditions:

- none of the two respective vertical sides is crossed by $\mathcal{M}$'s working head more often than $b(n)$ times;
- the total time spent by $\mathcal{M}$'s working head in a given second order rectangle must not exceed $b^2(n)$.

Clearly, second order rectangles can be created in each first order rectangle, with the possible exception of "too short" slots, or in remainders of slots that are "artificially" cut by the line separating time segments. Call the respective second order rectangles, that could not be created in the "full size", as required by the previous two conditions, as *small rectangles.*

As a result we obtain at most $O(T(n)/b^2(n))$ second order full size rectangles (since the computation within each rectangle "consumes" either $b(n)$ crossing sequence elements, or time $b^2(n)$), plus at most $O(T^{5/6}(n)/b(n))$ small ones (since the number of small rectangles

[3]The particular choice of $b(n)$ will be justified at the end of the proof.

does not exceed that of first order ones). Thus the total number of second order rectangles is safely bounded be $O(T(n)/b^2(n))$.

Each second order rectangle will be completely represented by its two horizontal sides of length $b(n)$, giving the contents of the corresponding block on $\mathcal{M}$'s working tape at the respective time steps, and by the description of the history of crossing its two vertical sides by the working head of $\mathcal{M}$. For each side the history of crossing is described by the so–called *crossing sequence* of length ℓ_1 for the left side and ℓ_2 for the right side, respectively, with ℓ_1, $\ell_2 \le b(n)$. Any crossing sequence consists from so–called *crossing sequence elements* that are ordered chronologically in that order in which the head has crossed the respective rectangle side. Each element of a crossing sequence is represented by a pair $\{q, d\}$. Here q denotes the state of $\mathcal{M}$ when crossing the vertical side at hand and $d \in \{left, right\}$ records the direction of the crossing.

Hence, the size of each second order rectangle representation is $\Theta(b(n))$, what in a total gives $O(T(n)/b(n))$ for all rectangles.

The rectangular representation of $\mathcal{M}$'s computation pertinent to the given input that is written on $\mathcal{S}$'s input tape will be represented on $\mathcal{S}$'s tape in the following order, from left to right: it is the sequence of individual second order rectangles that is generated for time segment by time segment, and within each time segment, first order rectangle by first order rectangle, and within each first order rectangle, second order rectangle by second order rectangle, in chronological order. Boundaries between individual (first and second order) rectangles, and time segments, respectively, are marked by special symbols on a special track.

To represent the computations of $\mathcal{M}$ in accordance with the idea mentioned before the statement of the theorem 2.2 we need moreover to record the position of $\mathcal{M}$'s input head that corresponds to each crossing sequence element in each second order rectangle. Consider the sequence of crossing sequence elements ordered chronologically — i.e., in that order in which the boundaries between slots are crossed during $\mathcal{M}$'s computation and consider also the respective associate sequence of corresponding input head positions. This associate sequence has as many elements as is the length $\ell(T(n))$ of our chronologically ordered crossing sequence and the respective elements are integers in the range $< 1..n >$. Now, instead of recording the "absolute" positions of $\mathcal{M}$'s input head on the input tape, record only the *differences* $d_i = p_i - p_{i-1}$ between any two absolute positions p_i and p_{i-1}, respectively, for $i = 0, 1, \ldots, \ell(T(n))$ and $p_0 = 1$. Thus, the differences are integers from the interval $< -n + 1..n - 1 >$. The size of representation of any $|d_i|$ is at most $\log n$. Superpose now the resulting sequence that, starting from p_0 enables to compute $\mathcal{M}$'s input head positions, with the above mentioned chronologically ordered crossing sequence and call the resulting merged sequence a *log* of $\mathcal{M}$'s head movement. Thus the log elements take the form $\{q, d, d_i\}$ of triples, where q denotes the state of $\mathcal{M}$, d the direction of $\mathcal{M}$'s working head movement, and d_i the difference between current and previous position of $\mathcal{M}$'s input head, with all values within the i–th triple pertinent to the moment when $\mathcal{M}$'s working head is crossing for the i–th time a boundary between any two slots. Ignoring items of constant size, for the log elements it holds that $\sum_{i=1}^{\ell(T(n))} |d_i| \le T(n)$. Therefore, the length of the corresponding representation can be bounded by

$$\sum_{i=1}^{\ell(T(n))} \lceil \log |d_i| \rceil = O(\log \prod_{i=1}^{\ell(T(n))} |d_i|) \le O\left(\ell(T(n)) \log \frac{\sum_{i=1}^{\ell(T(n))} |d_i|}{\ell(T(n))}\right) = O\left(T(n) \frac{\log\log T(n)}{\sqrt{\log T(n)}}\right)$$

The log representation will be also represented on $\mathcal{S}$'s working tape in a natural way from left to right, with special separators in between the time segments that split the log into segments that correspond to the computations of $\mathcal{M}$ of length $T^{2/3}(n)$.

We will insist that both rectangular representation and the log representation will be written one above the other on two parallel tracks on $\mathcal{S}$'s working tape in such a way that the respective time segment milestones both in the rectangular representation and in the log will find themselves at the same positions. This can be achieved by prolonging the length of rewritten part of $\mathcal{S}$'s tape at most four time. The reason is that the same crossing sequence elements that are stored in the triples of the log part corresponding to any time segment find themselves also among the crossing sequence elements represented within the corresponding part of the rectangular representation. Due to the chosen representation of second order rectangles each crossing sequence element from the log finds itself at most two times in the rectangular representation (namely, once in the right crossing sequence of some rectangle, and once in the left crossing sequence of the right adjacent rectangle). Further, the size of rectangles has been selected in such a way that to each crossing sequence element there are at most two elements (tape symbols) in the upper and lower horizontal sides of the respective rectangle. Thus, the length of the time segment in the rectangular representation is at most four times greater then is the number of crossing sequence elements in the corresponding time segment in the log. Since in the triples of the log also the differences between $\mathcal{M}$'s input head positions are stored we get the final estimate that the length of one time segment in the log is not greater than four times the length of the corresponding time segment in the rectangular representation plus the space needed to represent differences in input head positions stored in the triples.

Thus, data representing the corresponding time segments both in the rectangular representation and in the log can indeed be written on two parallel tracks in such a way that all time segments are separated by common markers that will be called *segment separators.*

The length of the above data, pertinent to one time segment of the rectangular representation, is at least $\Omega(\sqrt{T(n)\log n})$ (since there must be at least $\Omega(\sqrt{T(n)\log n}/b(n))$ second order rectangles, each of size $O(b(n))$), and at most $O(T^{2/3}(n))$ (since within the time segment of duration $T^{2/3}(n)$, at most $O(T^{2/3}(n)/b(n))$ different rectangles can be visited by $\mathcal{M}$'s head). Thus, the length of any time segment in the joint representation of both rectangles and log is at most $O(T^{2/3}(n)\log n)$. This estimate will be important for the complexity estimation of actions performed within one time segment. Nevertheless, it is obvious that the total length of the joint representation of the log and of all second order rectangles is bounded by the length of the log, i.e., is of order $O(T(n)\log\log T(n)/\sqrt{\log T(n)})$.

Now, the idea of simulation is first to guess and record the above rectangular representation simultaneously with the log of $\mathcal{M}$'s computations and then to verify the correctness of the above guesses. The verification process consists of two main phases. First, we have to verify whether the guess of rectangular representation was correct — i.e., whether all rectangles 'fit' together and whether the size ant the format of rectangular representation and of the log have been guessed correctly (a so–called *global correctness*). Second, we have to attest whether each rectangle represents a valid piece of M's computation. Such a computation starts in "partial" configuration as described by the upper horizontal side of the rectangle at hand and ends in a configuration as described by the lower side of the rectangle. Moreover, in such a computation $\mathcal{M}$'s working head must leave and re–enter rectangles in accordance both with the respective two crossing sequence at both rectangle's vertical sides and with the symbol read by $\mathcal{M}$'s input head at that time (so–called *local correctness*).

This leads to the design of the simulation scheme in which $\mathcal{M}$ is simulated by a single–tape off–line Σ_2-machine $\mathcal{S}$ in two main phases: in the first, nondeterministic one, all guesses will be performed, whereas in the second, universal one, the verification of all previous guesses will be done.

Phase 1 - existential:

A. Simultaneous generation of the rectangular representation and of the log. On two special tracks, $\mathcal{S}$ guesses and writes down, in a single sweep over its working tape, the rectangular representation and the log of $\mathcal{M}$'s computation pertinent to the given input that is written on $\mathcal{S}$'s input tape. Both representations are split into time segments. The shorter of the two representations that corresponds to one time segment is filled by special symbols up to the length of the longer one, and then a joint *segment separator* is inserted separating two consecutive time segments.

In order to be able to verify later the correctness of format and size of the rectangular representation and of the log we shall further insert so–called milestones at the end of each time segment. These milestones will contain the guessed values of certain quantities.

The *rectangular representation milestones* contain the number of second order rectangles within the individual time segments. The *crossing sequence milestones* give the partial sum of the number of crossing sequence elements, starting with the first time segment and ending in the given time segment; The *input head milestones* give the absolute address of the first input head position within given log segments. The *working head milestones* give the corresponding number (index) of the slot in which the working head finds itself at times pertinent to the values of input head milestones. In the first segment the last two milestones are set to one.

As explained above, the length of the rewritten part of the tape will be at most equal to some constant multiple of $T(n)\log\log T(n)/\sqrt{\log T(n)}$.

Phase 2 — universal:

A. Checking the global correctness: This phase consists in fact of three independent verification processes that can be run also in parallel:

1. *verification of horizontal boundaries in rectangular representation:* First, it is verified whether the upper boundary of any rectangle from the "first row", so to speak, as seen in the time–space computational diagram of $\mathcal{M}$, is created by a sequence of blanks. Next, it is verified whether the respective pairs of rectangles fit at their common horizontal boundary. Due to the chosen representation of rectangles on $\mathcal{M}$'s tape (see subphase 1.1), within the same time interval, the lower and the upper horizontal side of two neighbouring rectangles within the same slot are at the distance of at most $O(b(n))$. Their equivalence can be easily verified in time $O(b^2(n))$ by invoking a special parallel process for each rectangle.

   When there are two neighbouring rectangles within the same slot, but in different time segments, then the distance of the corresponding horizontal boundaries, that have to be compared, is at most $O(T^{2/3}(n))$.

   The last verification is not completely trivial, since except the horizontal boundary of a rectangle at hand, also a counter of size $O(\log T(n)) = O(b^2(n))$ must be carried along each time segment, that "counts" the rectangles and enables thus to identify the corresponding rectangles that are the neighbours within the same slot in the rectangular representation.

   Thus, parallel time $O(T^{2/3}(n)\log T(n))$ is enough to perform all the necessary comparisons for all time segments.

2. *verification of vertical boundaries in rectangular representation:* For the kind of verification at hand it is important to realize that due to the sweeps involved at the end of each time segment, only crossing sequences between horizontally neighbouring rectangles *within the same time segment* must be compared. Thus, it is enough for each

crossing sequence element from the right side of some rectangle to find its "companion" in the left side of a horizontally neighbouring rectangle; this companion will be located at the distance of at most $O(T^{2/3}(n))$. This can be done in parallel, extra for each element and extra for each time segment: we only must keep track of element's relative position on the vertical boundary between the respective slots, within the given time segment. This amounts to shifting a counter of size $O(\log T(n))$ along the tape, to the distance of at most $O(T^{2/3}(n))$.

3. *Checking the milestones:* by invoking parallel process for each pair of neighbouring time segments we check by shifting appropriate counters along the $\mathcal{S}$'s working tape:

    - whether the value of the next input head milestone corresponds to the sum of input head differences in between this, and the previous milestone;
    - whether the value of the next working head milestone corresponds to the working head movement within the given time segment. This is done as follows. We first set a special counter to the value of the corresponding working head milestone in the first time segment of the both in the pair at hand. Then scanning the log in this segment we keep track on the direction of moves of $\mathcal{M}$'s working head that are recorded in chronological ordering in crossing sequence elements of the log. From these information we can infer whether the head traversed across the respective rectangle from left to right (then we increase the respective counter by one), or from right to left (we decrease the counter by one), or whether the head only "visited" the rectangle without traversing across it (we do not change the value of the counter). When reaching the next working head milestone the value of counter must coincide with the value of that milestone;
    - whether the difference between two consecutive crossing sequence milestones is equal to the number of crossing sequence elements really present in the log and also in the rectangular representation and whether the total number of crossing sequence elements does not exceed the prescribed limit;
    - whether the number of rectangles in the second time segment of the both is equal to the difference between the respective rectangular representation milestones. In the case of the last time segment we check whether the total number of rectangles is within the prescribed limits. At this occasion we also check the format of rectangles and of log elements.

B. Checking the local correctness. In order to be able to perform the local verification of each second order rectangle, we first have to assign to each crossing sequence element from the rectangle's boundary the respective input head position of $\mathcal{M}$ at that moment to which the crossing sequence element is pertinent.

For such a purpose we will need a unique identification of each rectangle. Within a given time segment each second order rectangle is uniquely given by its position within the rectangular representation corresponding to the given time segment. This position is represented by two coordinates. The first coordinate is the number of slot in which the rectangle find itself. The second coordinate is the number (index) of the first crossing sequence element on the left side of the rectangle within the crossing sequence pertinent to the left side of the respective first order rectangle (i.e., we are indexing the crossing sequence elements along the respective slot boundary, in each time segment separately, starting from one). From effectiveness reasons we shall make use also of a third coordinate that is defined in analogous way as the second one, but instead of the left crossing sequence, it concerns the right one in the rectangle at hand.

For each second order rectangle, the respective three values will be stored as one triple in the variable called *rectangle_address*. The size of *rectangle_address* is $O(\log T(n)) = O(b^2(n))$.

The local verification consists thus from three main steps that are described in the sequel. For each rectangle these three steps have to be performed in a sequential manner, but in parallel for all second order rectangles. This is achieved by scanning the rectangular representation as a whole from left to right and making a universal split after passing each second order rectangle representation.

1. *Computing the address of a second order rectangle.* The first coordinate of the rectangle is computed by shifting a counter along the rectangular representation from the beginning of the respective time segment and counting the crossings of horizontal boundaries of the *first* order rectangles encountered.

   The second coordinate is computed in a similar manner by shifting an appropriate counter along the respective time segment in the rectangular representation, this time keeping track of the number of crossing sequence elements at the slot boundary we are interested in. Thus, the second coordinate (index of the first crossing sequence element within the rectangle) within *rectangle_address* must be equal to the sum of lengths of all left crossing sequences of all rectangles that were visited earlier within the same slot in the given time segment. The third coordinate is computed in a similar way.

2. *Assigning input head positions to each crossing sequence element in each rectangle.* We shall describe the respective procedure for the crossing sequence elements from the left side of a rectangle. For the right side the procedure will be performed in a similar way.

   First, similarly as in the previous step we will have to determine the indexes of all the crossing sequence elements in the left crossing sequence of the rectangle at hand. This is easy, since we already have the index of the first crossing sequence element stored in the second coordinate of the respective *rectangle_address*. The indices of the remaining elements are obtained by subsequently adding one to this initial value.

   Now follow the log from the respective milestone (i.e., in the part pertinent to the given time segment) and by shifting appropriate counters along it, compute the *absolute position* of input head of $\mathcal{M}$ on its input tape at the respective times when $\mathcal{M}$'s working head crosses the boundary between slots. Keep also track which slot boundary is working head crossing at that time. When the working head is crossing the slot boundary at (the left side of) which our rectangle finds itself, count also the number of crossings of this boundary. This can be done as follows.

   Absolute positions of the input head are determined with the help of differences stored in the log, by adding them to the respective input head milestone as they are encountered when traversing the log.

   Which slot boundary is working head crossing can be inferred in a similar way from the information about the directions of working head movements when crossing the respective slot boundaries in much the same way as the correctness of working head milestones has been verified in Phase 2.A.3.

   In this way we move along the log until we reach positions when the working head hits, one after the other, the crossing sequence elements from our rectangle's left boundary.

In such a case the number of current slot that is being crossed by the working head must match the (first coordinate of the) address of our rectangle and the number of crossings this boundary must equal the index of crossing sequence element from the rectangle's left boundary. At this moment the position of the input head can be assigned to this element.

For all elements from both sides of the rectangle all this can be accomplished by moving appropriate counters of size at most $\log T(n)$ along the respective time segment $b(n)$ times, in a total time $O(T^{2/3}(n)b(n)\log T(n)\log n)$.

3. *Verifying individual rectangles.* For the second order rectangle at hand, that already has the input head position assigned to each of its vertical boundary crossing sequences by the previous process, replay the corresponding piece of the respective *nondeterministic* computation of $\mathcal{M}$. This means that we have to check that starting from the "partial" instantaneous description, as described by the upper horizontal side of a rectangle, there is a nondeterministic computation of $\mathcal{M}$ that after at most $b^2(n)$ steps reaches the partial instantaneous description as described by the lower horizontal side of our rectangle. During this computation, the input head must be at most $b(n)$ times repositioned at the input tape at each occasion when the working head reaches either side of the rectangle at hand. Thus, the time complexity of such a nondeterministic computation is $O(b^2(n) + nb(n))$ (the term $nb(n)$ reflects the complexity of head positioning). Note that only $b^2(n)$ nondeterministic moves are performed within the latter computation.

   However, this computation cannot be done straightforwardly, since there is no longer any nondeterminism at our disposal. Therefore, we have to *deterministically* simulate the respective piece of computation with the help of a suitable modification of the standard backtrack procedure. This procedure has to try all possible $c^{b^2(n)}$ paths in the nondeterministic computation at hand. Nevertheless, the total price of input head repositioning during the verification of all computational paths would be still asymptotically to high. Therefore, we will implement an other strategy. Namely, we will save the costs of the repeated input head relocation to the same places.

   Note that due to the choice of $b^2(n)$ as that of the maximal nondeterministic time that $\mathcal{M}$ can spent in each rectangle, the input head, once positioned to a corresponding place on the input tape, can move to the distance of at most $b^2(n)$ in either direction during the respective computation. Thus, at the beginning of the respective deterministic simulation we will position the input head once for each crossing sequence element in the respective rectangle and will copy the corresponding parts of the input tape that can be reached during the simulation to a special track in the vicinity of working head. This will give rise to the string *input* where the respective parts of the input of length $b^2(n)$ are stored one after the other, separated by suitable separators. Since there are at most $b(n)$ repositionings needed, there will be at most $b(n)$ of such parts. Therefore, the length of *input* will be at most $b^3(n)$.

   Now the simulation proceeds deterministically, but instead of reading the necessary inputs from the input tape $\mathcal{S}$ looks for them in the string *input*. The current position of input head in the respective part of input within the string *input* is marked by an appropriate mark. Reading of any input symbol requires thus $O(b^3(n))$ moves.

   This new simulation schema of one path of the nondeterministic computation leads to the time complexity of order $O(b^5(n))$ since $b^2(n)$ moves of $\mathcal{M}$ are to be simulated. To verify all paths we need $O(b^5(n)c^{b^2(n)}) = O(T(n)/\sqrt{\log T(n)})$ time. Note that in fact it was the last equality that has lead to fixing the block size $b(n)$ at the maximal value that would still enable to perform the deterministic simulation of nondeterministic computations within rectangles in the time that would not asymptotically affect the complexity of simulation process as a whole.

The simulation ends successfully when all the verifications performed in the second simulation phase terminate successfully and in the universal subphase 2(b) an accepting rectangle has been discovered.

The time complexity of simulation is dominated by the time needed to generate the log in the existential phase. Therefore its length $O(T(n)\log\log T(n)/\sqrt{\log T(n)})$ is at the same time the bound on the time complexity of the entire simulation.

$\square$

As seen from the above proof, it has been quite complicated (e.g., also in comparison with the similar proof for the case of simulation of deterministic TM [8]) due to the quite peculiar choice of log, in which the input head positions in chronological order have been encoded. However, the "natural" alternative choice — viz. is the storing of the input head positions along with each corresponding crossing sequence element, as exploited also in [8] — would lead to the simulation of time complexity $(T(n)\log n/\log T(n))$, as the author persuaded himself. The fact that within polynomial complexity classes the latter alternative presents no speed–up at all, has lead to the development of the above mentioned idea of a log.

3 Separation Result

In terms of complexity classes the previous theorem says that for a suitable $T(n)$ the class of nondeterministic $T(n)$ time bounded single–tape off–line computations is a subset of the $T(n)\log\log T(n)/\sqrt{\log T(n)}$-time bounded computations on a Σ_2 machine of the same type:

$$\mathsf{NTIME}_{1+\mathrm{I}}(T(n)) \subseteq \Sigma_2\text{-}\mathsf{TIME}_{1+\mathrm{I}}(T(n)\log\log T(n)/\sqrt{\log T(n)})$$

To prove a separation result related to the above mentioned complexity classes that are bounded by the same function we shall need the following hierarchy theorem for nondeterministic single–tape of–line machines by Loryś and Liśkiewicz [7]:

Theorem 3.1 *Let $T_2(n)$ be a fully time constructible function, with $n\log n \in o(T_2(n))$ and such that there exists a deterministic off–line Turing machine which for each input of length n writes on the work tape the binary representation of $T_2(n)$ in time $T_2(n)$. Let $T_1(n+1) \in o(T_2(n))$. Then*

$$\mathsf{NTIME}_{1+\mathrm{I}}(T_1(n)) \subset \mathsf{NTIME}_{1+\mathrm{I}}(T_2(n))$$

Now we are in a position to prove the separation result we are after:

Theorem 3.2 *Let $T_1(n) = T(n)$ and $T_2(n) = T(n)\log\log n$ fulfill the assumptions of Theorem 3.1 and let $T_2(n)$ fulfills the assumptions of Theorem 2.2. Then*

$$\mathsf{NTIME}_{1+\mathrm{I}}(T(n)) \subset \Sigma_2\text{-}\mathsf{TIME}_{1+\mathrm{I}}(T(n))$$

Proof Outline: From Theorem 3.1 we know the proper inclusion

$$\mathsf{NTIME}_{1+\mathrm{I}}(T(n)) \subset \mathsf{NTIME}_{1+\mathrm{I}}(T(n)\log\log T(n))$$

According to the Theorem 2.2 the latter class is contained in

$$\Sigma_2\text{-}\mathsf{TIME}_{1+\mathrm{I}}\left(T(n)\log\log T(n)\frac{\log\log(T(n)\log\log T(n))}{\sqrt{\log(T(n)\log\log T(n))}}\right) \subseteq \Sigma_2\text{-}\mathsf{TIME}_{1+\mathrm{I}}(T(n))$$

□

Thus, for nondeterministic single–tape off–line time bounded Turing machines one more alternations leads to provably more powerful machines.

It appears that all the previous theorems can be reworked to hold also for complementary machines, i.e., for classes $\mathsf{co}\text{-}\mathsf{NTIME}_{1+\mathrm{I}}(T(n))$ and $\Pi_2\text{-}\mathsf{TIME}_{1+\mathrm{I}}(T(n))$. However, from space reasons we will abstain from the formulation of the respective proofs (cf. [12] for a similar procedure for case of single tape machines without an input tape).

4 Conclusions

It has been shown that for a large class of identically time bounded computations, adding of one more alternation to a nondeterministic or co-nondeterministic Turing machine with one work tape and extra input tape leads to strictly larger complexity classes. Thus, for such machines the first and the second level of the respective alternating hierarchy of complexity classes do not collapse. This seems to be the first occasion where such a general result has been achieved for a restricted type of multitape nondeterministic Turing machines. We hope that our results will at least serve as an inspiration for proving similar results also for the general case of nondeterministic multitape machines. Note that for such a purpose it is enough to consider only two tape nondeterministic off–line machines, due to the result by Book et al. [1].

Bibliography

[1] Book, R.V. — Greibach, S.A. — Wegbreit, B.: Time– and Tape–Bounded Turing Acceptors and AFLs. *JCSS 4*, Vol. 6, Dec. 1970, pp. 602–621

[2] Hennie, F.C.: One–Tape, Off–line Turing Machine Computations. *Information and Control*, Vol. 8, 1965, pp. 553–578

[3] Dymond, P.W. — Tompa, M.: Speedups of Deterministic Machines by Synchronous Parallel Machines. In *Proc. 24th Annual IEEE Symposium on Foundations of Computer Science*, pp. 336–364, 1983

[4] Gupta, S.: Alternating Time Versus Deterministic Time: A Separation. *Proc. of Structure in Complexity*, San Diego, 1993

[5] Hopcroft, J. — Paul, W. — Valiant L.: On time versus space and related problems. *Proc. IEEE FOCS* **16**, 1975, pp. 57–64

[6] Kannan, R.: Alternation and the power of nondeterminism (Extended abstract). *Proc. 15–th STOC*, 1983, pp. 344–346

[7] Loryś, K. — Liśkiewicz, M.: Two Applications of Führers Counter to One–Tape Nondeterministic TMs. *Proceedings of the MFCS'88*, LNCS Vol. 324, Springer Verlag, 1988, pp. 445–453

[8] Maass, W. — Schorr, A.: Speed-up of Turing Machines with One Work Tape and Two–way Input Tape. *SIAM J. Comput.*, Vol. **16**, no. 1, 1987, pp. 195–202

[9] Paterson, M.: Tape Bounds for Time–Bounded Turing Machines, *JCSS*, Vol. 6, 1972, pp. 116–124

[10] Paul, W. — Prauss, E. J. — Reischuk, R.: On Alternation. *Acta Informatica*, **14**, 1980, pp. 243–255

[11] Paul, W.J. — Pippenger, N. — Szemerédi, E. — Trotter, W.T.: On determinism versus nondeterminism and related problems. In *Proc. 24th Annual IEEE Symposium on Foundations of Computer Science*, pp. 429–438, 1983

[12] Wiedermann, J.: Speeding–up Single– Tape Nondeterministic Computations by Single Alternation, with Separation Results. *Proceedings of the 23–rd International Colloquium on Automata, Languages, and Programming*, ICALP'96, LNCS Vol. 1099, Springer Verlag, Berlin, 1996