Separating Deterministic, Nondeterministic, and Co-Nondeterministic Time Complexity Classes for Single-Tape Computations

Wiedermann, Jiří
1995

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Separating Deterministic, Nondeterministic, and Co–Nondeterministic Time Complexity Classes for Single–Tape Computations

Jiří Wiedermann
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague
Czech Republic

Technical report No. 628

March 1995

Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague 8, Czech Republic
phone: (+422) 66414244 fax: (+422) 8585789
e-mail: wieder@uivt.cas.cz

INSTITUTE OF COMPUTER SCIENCE

ACADEMY OF SCIENCES OF THE CZECH REPUBLIC

Separating Deterministic, Nondeterministic, and Co–Nondeterministic Time Complexity Classes for Single–Tape Computations

Jiří Wiedermann[1]
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod vodárenskou věží 2, 182 07 Prague
Czech Republic

Technical report No. 628
March 1995

Abstract

It is shown that for any well behaved function $T(n)$, a single–tape nondeterministic Turing Machine of time complexity $T(n)$ can be simulated by a single–tape Σ_2–machine in time $T(n)/\log T(n)$. Consequently, a separation of all three related fundamental complexity classes — viz. single–tape deterministic, nondeterministic, and co–nondeterministic, respectively, time–bounded classes, is shown.

Keywords

Single tape computations, Turing machines, complexity classes

[1]This research was supported by GA ČR Grant No. 201/95/0976, and partly by Cooperative Action IC 1000 (Project "ALTEC") of the European Union

1 Introduction

Single–tape Turing machines (TMs) are generally considered as the simplest, albeit from computational point of view somewhat cumbersome, universal computing model. Therefore the related complexity classes differ in some details, especially in cases when polynomial factors cannot be neglected, from that of multitape TMs. However, it appears that it is just the relative simplicity of single–tape TMs that enables to obtain for them results that are so far, or due to the idiosyncracy of the machines involved, unattainable for multitape TMs. In both cases the respective results are of immense value for theoretical studies in complexity theory since they can serve at least as inspiration, or as a guide for our intuition, for similar studies in the field of more realistic models of computation.

In this context, among the central open problems that attract a lot of theoretical attention even in case of single–tape computations, are problems dealing with the "non–trivial" relationship among complexity classes related to fundamental computational resources — viz. determinism, nondeterminism, and co–nondeterminism, respectively. Unfortunately, so far the situation here has been much similar to that in the realm of multitape TMs: mostly only partial answers to only few of the related question has been known.

To illustrate the recent situation in the respective field, for single–tape TMs, especially the following results are to be reported.

In [3] or in [4] it is shown that any nondeterministic single–tape TM of time complexity $O(T(n))$ can be simulated in linear time by another nondeterministic single–tape machine in space $O(\sqrt{T(n)})$. In [4], moreover a tight time hierarchy for nondeterministic single tape TMs has been shown. For the deterministic case, even speed–ups have been be achieved: in [7], a single–tape deterministic machine has been speeded-up by an alternating machine (i.e., using an unbounded number of alternations) down to $O(\sqrt{T(n)})$ time. In [2] it is shown that the same machine, equipped, moreover, by a read–only input tape, can be simulated in time $O(T(n)^{2/3} \log T(n))$ by a multitape Σ_4–machine, i.e., by a bounded number of alternations. Along these lines, so–far the best result seems to be the result from [5], where the same result as before has been proved by using a substantially less powerful simulating machine than in [2] — namely a single–tape Σ_2-machine.

Unfortunately, even though the last mentioned result has lead to some separation results between *single–tape* deterministic time, and *multitape* nondeterministic time, the separation of determinism from nondeterminism, or from co–nondeterminism, for single–tape machines alone, and for arbitrary time–bounded complexity classes, has not followed.

It appears that sharper results are necessary if we want to achieve a separation. In this respect, the key observation seems to be the following one: the speed–ups achieved so far have been achieved for quite a high price — namely too many powerful computational resources have had to be added to achieve the speed–up effect. E.g., in the above mentioned latter case [5], for achieving a speed–up, both addition of nondeterminism,

and of one additional alternation, have been necessary.

In the paper at hand we shall show that a mere addition of an extra alternation is enough to speed–up a single–tape nondeterministic computation. More specifically, we show that under a certain reasonable technical conditions concerning the function $T(n)$, any $T(n)$–time bounded single–tape nondeterministic TM can be simulated by a single–tape Σ_2–machine in time $O(T(n)/\log T(n))$. The same speed–up can be achieved also for the complementary machines (i.e., a single–tape Π_1 (co–nondeterministic) machine can be speeded–up by a single–tape Π_2 machine). As a consequence the separation of the four related complexity classes — viz. single-tape deterministic, nondeterministic co–nondeterministic, and Σ_2, respectively, time–bounded classes, will be shown.

2 Speed–up of Nondeterministic Computations by Additional Alternation

The following theorem states that under certain technical assumptions, any computation of a nondeterministic, single–tape Turing machine can be speeded–up by a single–tape TM performing two alternations. The proof of this theorem is based on a space efficient, so–called *rectangular representation* of single–tape computations, that goes back to Paterson [6] (see also [1] or [3] for a similar approach). Our proof, however, differs substantially from proofs based on the similar representation, in at least two important aspects:

- first, the computation of the original machine, that is to be speeded–up, is first transformed into an equivalent computations that enables, later on, its efficient simulation;
- second, by making use of a different distribution of related computational tasks into nondeterministic, and co–nondeterministic, phase of the simulating machine, the speed–up of a *nondeterministic machine*, by only two alternations, is enabled.

Theorem 2.1 *Let $\sqrt{T(n)}$ be fully time and space constructible, with $T(n) \geq n^2$. Than any T(n)–time–bounded nondeterministic single–tape TM M can be simulated by a single-tape Σ_2–TM in time $O(T(n)/\log T(n))$.*

Proof Outline. Following [3], for $T(n)$ as above, w.l.o.g. we can assume that M is of space complexity $O(\sqrt{T(n)})$. Moreover, from [4] it follows that w.l.o.g. we can assume also that M is a *strongly accepting TM* — i.e. such that always halts after performing $T(n)$ moves, and either accepts, or rejects its input.

Next, for specific reasons that will become clear in a sequel we shall split the computation of M into $\Theta(T^{1/3}(n))$ *time segments*, each of length $\Theta(T^{2/3}(n))$, and, at the end of each time interval, we let the machine perform a complete sweep over its tape.

A sweep starts at the position of the machine head at the end of time interval at hand, proceeds by moving the head to the right end of the tape, then in the opposite direction to the left end, and finally returns to the original head position. During a sweep the contents of the tape is not altered. The execution of a sweep will be included into the respective time segment. It is clear that the time complexity of performing a single sweep is proportional to the tape length, and thus complexity of the resulting machine remains bounded by $O(T(n))$.

The reason for introducing sweeps over the entire tape at the end of each time segments lies in the fact that doing so it is possible to verify the "history" of cell rewritings during a computation by returning back in the time to the distance of only $\Theta(T^{2/3}(n))$. (This is the first idea where our proof deviates from the schema of similar proofs — say from [5].)

Now, for the machine M prepared as above we construct the rectangular representation of its computation.

To obtain this representation for any fixed computation, it is helpful to see the computation at hand as a sort of a diagram, that consists of individual instantaneous descriptions (ID's) of M, for subsequent time steps $t = 0, 1, \ldots$, written one above an other. Besides the current contents of M's tape in each ID also the current head position is recorded by writing down the current state of M's finite control over the position just scanned by M's head. In this way also the trajectory of M's head, on a given input, is recorded in our diagram.

Further, starting from some position j, with $0 \leq j < b(n)$, the tape of M can be partitioned into blocks of length $b(n)$ (the value of $b(n)$ to be determined later) in such a way that the sum of length of crossing sequences over the block boundaries will be at most $T(n)/b(n)$. Such a position j must exist, since in the opposite case, would the respective sum be greater than $T(n)/b(n)$ for each j, the total length of the crossing sequence, taken over all tape cells, would be greater than $T(n)$.

The block boundaries will split the computational diagram at hand into at most $\sqrt{T(n)}/b(n)$ vertical slots. These slots will be changed into so–called *first order rectangles* by drawing horizontal lines in between the ID's that separate the individual time segments.

Clearly, in this way we obtain at most $O(T^{5/6}(n)/b(n))$ first order rectangles.

First order rectangles will be further split by drawing a horizontal line at suitable points, 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 the TM head more often than at $b(n)$ times;
- the total time spent by the TM 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 in each vertical slot, with the possible exception of "too short" slots, or in remainders of

first order rectangles 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. Thus the total number of second order rectangles is safely bounded by $O(T(n)/b^2(n))$.

Each second order rectangle will be thus represented by its two horizontal sides of length $b(n)$, giving the contents of the corresponding block at the respective time steps, and by the two vertical sides of length ℓ_1 and ℓ_2, respectively, with $\ell_1, \ell_2 \leq b(n)$, giving the respective crossing sequences in chronological order for each side separately.

Hence, the size of each second order rectangle representation is at most $4b(n)$, what in a total gives $O(T(n)/b(n))$ for all rectangles.

Now, the idea of simulation is first to guess the above rectangular representation, and then to verify whether the guess was correct — i.e., whether all rectangles 'fit' together (so–called *global correctness*) and, whether each rectangle represents a valid piece of M's computation — i.e., such that starts in "partial" configuration as described by the upper horizontal side of the rectangle at hand, ends in a configuration as described by the lower side of the rectangle, and where the M's head leaves and re–enters the rectangle in accordance with the crossing sequence that corresponds to the vertical rectangle boundaries (so–called *local correctness*).

This seems to lead straightforwardly to the design of the simulation scheme in which M is simulated by a single–tape Σ_2-machine in two main phases: in the the first, nondeterministic one, all guesses will be performed, whereas in the second, universal one, the verification of all previous guesses will be done.

Nevertheless the implementation of the above idea is complicated by the fact that verification of local correctness requires nondeterminism, and therefore must be performed during the first phase, i.e., sequentially. Doing this straightforwardly for each rectangle in turn will require a time of order $\Omega(T(n))$, what would prevent any speed–up. The way out of this problem is to define the size of each rectangle so small that there will be many equal rectangles in our rectangular representation; then it will be enough to identify, and verify only the different rectangles.

This is the second main point where our simulation departs significantly from the "standard" schema as used e.g. in [1], or in [5]. As we shall see later on, this departure will require also related, quite involved "preparation" actions in the first, and verification actions in the second, phase of simulation.

We have already noticed that the size of a rectangle representation is at most $4b(n)$. Thus, for a given TM M there are at most $R(n) = c^{4b(n)}$ different rectangles, for some constant $c > 0$ that depends on the size of M's alphabet and on the number of its stats. In order to have many equal rectangles in our rectangular representation of M's computations, $R(n)$ must be asymptotically less than their total number — i.e., less that $O(T(n)/b^2(n))$. Choose $b(n) = 1/24 \log_c T(n)$. Then $R(n) = T^{1/6}(n)$, what, as

we shall see later on, is quite appropriate for our purposes.

Let us describe now the simulation itself. Let S be the simulating single–tape Σ_2–TM.

The nondeterministic phase of its computation consists of the following four subphases.

Subphase 1.1 — generating a rectangular representation. For a given computation of M the rectangular representation, with $b(n)$ as above, is guessed and written down on the S's tape in the following order: time segment by time segment, and within each time segment, first order rectangle by first order rectangle, from left to right, 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.

The length of the above data, pertinent to one time segment, is at least $\Omega(\sqrt{T(n)})$ (since there must be at least $\Omega(\sqrt{T(n)}/b(n))$ 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 M's head).

However, it is clear that the entire rectangular representation can be generated in linear time w.r.t. its length — i.e., in time $O(T(n)/\log T(n))$.

Subphase 1.2 — preparing for duplicate rectangle identification. On a special track, guess and write down all the *different rectangles* from M's rectangular representation, in the sorted order. Call the resulting string a *set-of-rectangles*. Then "copy", still onto the same special track, nondeterministically, the previous string of length at most $4R(n)b(n)$, above the beginning of each time segment, $T^{1/3}(n)$ times in a row. Due to the nondeterminism involved, the entire "nondeterministic copy" operation, for all segments, can be performed in a single traversal over S's tape. Hence the time complexity of this operation is $O(T(n)/b(n))$.

The *set-of-rectangles* strings will be subsequently, in the universal phase of simulation (see subphase 2.3), be used for testing, whether all the rectangles from the original rectangular representation are duplicates of the rectangles that are included in the above set.

Subphase 1.3 — local verification. Verify, whether all the rectangles from *set-of-rectangles* present a valid "piece" of M's computation. This is done by performing a nondeterministic computation as dictated by rectangle sides, for each rectangle from the first occurrence of *set-of-rectangles*. Since there are $R(n)$ rectangles, and the verification of any of them is bounded by $b^2(n)$, the verification of all rectangles requires time $O(R(n)b^2(n))$.

Subphase 1.4 — acceptance verification. Check, whether among all rectangles in the rectangular representation (i.e., not in the *set-of-rectangles* (!)) there is a rectangle in which the acceptance of M occurs. For this purpose, by scanning the appropriate track of S's tape guess the right rectangle, and verify, by replaying corresponding nonde-

terministic computation of M, whether the accepting state of M is achieved. This takes time proportional to the length of S's tape - i.e., $O(T(n)/b(n))$.

Then the universal phase follows. It consists of three subphases that are run in parallel.

Subphase 2.1 — verifying the global correctness. This phase consists in fact of two independent verification processes that can be run also in parallel:

- *verification of horizontal boundaries in rectangular representation:* For the rectangles that are the first ones in each slot, we have to verify, that their upper horizontal boundaries correspond to the initial content of M's tape. For the remaining rectangles we have to verify, whether the lower horizontal boundary of any rectangle, that is not the last one in the given slot, is the same as the upper horizontal boundary of a rectangle that follows the previous rectangle in a given slot.

  The correctness of rectangles from the "first row", so to speak, is easily to be verified, by shifting the rectangles that carry parts of the input, towards the input, that is being kept on the special track at the beginning of the S's tape. The remaining rectangles, that do not carry the input, should contain blanks in the corresponding parts. All this can be done in time $O(T(n)^{2/3}b(n))$ for each rectangle in parallel.

  For the remaining rectangles, note that due to the chosen representation of rectangles on 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 $O(T^{2/3}(n))$. Thus, again, the necessary comparisons for all time segments can be done in parallel, in time $O(T^{2/3}(n)b(n))$.

  The last verification is not completely trivial, since except the horizontal boundary of a rectangle at hand, also a counter of size $O(\log T(n)) = O(b(n))$ 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.

- *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 $O(T^{2/3}(n))$.

Subphase 2.2 — verifying the correctness of "nondeterministic copy" operation from subphase 1.2: universally split into $O(T^{1/3}(n))$ copies of S, (exactly: as many copies as there are time segments, minus one) and for each pair of neighbouring time segments verify, whether the string *set-of-rectangles* is the same in both time segments. This can be done by moving the respective string along the tape from the beginning of the first time segment at hand above the beginning of the second time segment. Since the length of two time segments on S's tape is at most $O(T^{2/3}(n))$, moving the string *set-of-rectangles* of length $R(n)$ along this distance requires time $O(T^{2/3}(n).R(n).b(n)) = O(T^{5/6}(n)b(n))$.

Subphase 2.3. — verifying duplicates of rectangles: we have to verify, whether every rectangle in a given time segment finds itself also among the rectangles within *set-of-rectangles*, that is located at the beginning of each time segment on a special track (see subphase 1.2). This can be done easily in parallel, extra for each time segment, by subsequently moving along the tape each rectangle from the rectangular representation towards the beginning of the respective segment, and by verifying, whether the segment at hand finds itself also in *set-of-rectangles*. Similarly as before the cost of this operation is of order $O(T^{5/6}(n)b(n))$.

Note that there can be some "additional" rectangles in the *set-of-rectangles* that have passed the test in subphase 1.3, nevertheless they do not find themselves within the rectangular representation of M's computations, but, obviously, this is harmless.

This is the end of the simulation: if all the previous verifications in Phase 1 and 2, respectively, end successfully, S accepts its input.

From the previous complexity estimates of individual parts of S's computation it is seen that the most time consuming computation has been performed in phase 1, and hence the time complexity of the whole simulation is $O(T(n)/b(n))$.

□

Note that the selection of a few "parameters" in the proof of the previous theorem (like the size of a rectangle, or the length of a time segment) was somewhat arbitrary: however, as seen from the proof and its analysis, although there are also other possibilities, within a certain range, none of them would lead to asymptotically better performance of the simulation. Namely, our proof "works" only in the case when there are less than $O(T(n)/\log^2 T(n))$ different rectangles in the rectangular representation at hand, what leads straightforwardly to the logarithmic size of block representation. Thus our simulation cannot be improved merely by tuning some of its parameters.

Therefore, the general question, whether there exist any asymptotically faster simulation of the above type, remains open.

We shall end this section by restating the statement of the previous theorem in terms of relation between the respective complexity classes. To this purpose, let $NTIME_1(T(n))$, and $\Sigma_2 - TIME(T(n))$, respectively, denote nondeterministic and Σ_2, respectively, $T(n)$–time bounded single–tape TM complexity classes. Then the following corollary of the theorem 2.1 holds true:

Corollary 2.1 *Let $\sqrt{T(n)}$ be fully time and space constructible, with $T(n) \geq n^2$. Then*

$$NTIME_1(T(n)) \subseteq \Sigma_2 - TIME_1(T(n)/\log T(n))$$

Proof Outline. The above inclusion follows directly from theorem 2.1; we only have to show that we can get rid of any constant hidden in the "big O" notation in the complexity estimate of simulating machine from the theorem 2.1. A short reflection reveals that this is indeed possible, since we can speed–up our simulating machine by an arbitrary constant factor, by using the standard technique of compressing groups of its tape symbols into symbols from a larger alphabet. There are slight problems with the compression of input - the first idea to compress it, prior to start of the simulation, in a sequential manner, would require time of order $\Omega(n^2)$, what is unacceptable (since some simulating machines can run in time $O(n^2/\log n)$, say). Therefore, a more complicated process must be chosen: first, nondeterministically guess, and write down, the compressed input, and later on, in the parallel computing phase, verify the correctness of the initial guess. Also note that in order to achieve the above constant speed–up the branching factor of nondeterministic or universal instructions of the original Σ_2 might be increased.

□

3 Speed–up of Co–Nondeterministic Computations by Additional Alternation

It appears that the previous results can be "reworked" to hold also for the case of co–nondeterministic computations — i.e., for computations performed by single–tape time bounded Π_1 machine. In this case, the simulating machine would be a single–tape Π_2 machine (a machine that starts its computation in a co-nondeterministic mode, and during its computation is allowed to perform one more alternation):

Theorem 3.1 *Let $\sqrt{T(n)}$ be fully time and space constructible, with $T(n) \geq n^2$. Than any T(n)–time–bounded co–nondeterministic single–tape TM can be simulated by a single-tape Π_2–TM in time $O(T(n)/\log T(n))$.*

We shall not present a proof here, since it is similar to the proof of theorem 2.1; in fact, the simulating machine is obtained from that from the proof of theorem 2.1 by exchanging its existential states with universal ones, and by mutually exchanging

its accepting and rejecting states (recall that in the proof of theorem 2.1 the simulated machine has been w.l.o.g. considered as a strongly $T(n)$–time bounded machine — i.e., such that ends in an accepting or rejecting state on any computational path, after performing at most $T(n)$ steps).

Further, when introducing the notion of two new complexity classes, namely that of $co-NTIME_1(T(n))$, and $\Pi_2-TIME_1(T(n))$, respectively, with the obvious meaning, than similarly as before, one can prove the following corollary:

Corollary 3.1 *Let $\sqrt{T(n)}$ be fully time and space constructible, with $T(n) \geq n^2$. Then*

$$co - NTIME_1(T(n)) \subseteq \Pi_2 - TIME_1(T(n)/\log T(n))$$

4 Separation Results

In the sequel we shall prove that the amount of speed–up achieved by the previous simulation is enough to prove interesting separation results concerning the basic complexity classes that are related to the computations at hand. To prove these results, we shall need two hierarchy theorems. The first one of them, that we shall reproduce here without a proof, is the theorem by Lorys and Liskiewicz [4], proving a tight hierarchy for single–tape nondeterministic time complexity classes:

Theorem 4.1 *Let $T_2(n)$ be fully time constructible, with $n^2 \in o(T_2(n))$, and let $T_1(n+1) \in o(T_2(n))$. Then*[2]

$$NTIME_1(T_1(n)) \subset NTIME_1(T_2(n))$$

The next theorem proves a similar hierarchy for single–tape co–nondeterministic time:

Theorem 4.2 *Let $T_2(n)$ be fully time constructible, with $n^2 \in o(T_2(n))$, and let $T_1(n+1) \in o(T_2(n))$. Then*

$$co - NTIME_1(T_1(n)) \subset co - NTIME_1(T_2(n))$$

Proof Outline: Consider any language $L \in NTIME_1(T_2(n)) - NTIME_1(T_1(n))$. According to the previous theorem, such a language must exist, and for its complement holds $co-L \in co-NTIME_1(T_2(n))$, and $co-L \notin co-NTIME_1(T_1(n))$, what proves the theorem.

□

Now we can return to our separation results. First we show that co–nondeterministic single–tape time is not included in nondeterministic single–tape time:

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

Theorem 4.3 *Let $T(n)$ be such that $\sqrt{T(n)}$ is a fully time and space constructible function, with $n^2 \in o(T(n))$, and $T(n+1)/\log T(n+1) \in o(T(n))$. Then*

$$co - NTIME_1(T(n)) \not\subseteq NTIME_1(T(n))$$

Proof Outline. Assume the opposite — i.e., $co - NTIME_1(T(n)) \subseteq NTIME_1(T(n))$, and consider a computation of any $T(n)$ time–bounded single–tape nondeterministic TM M, for $T(n)$ as above. Then, according to theorem 2.1, M can be simulated by a single–tape Σ_2-machine S in time $O(T(n)/\log T(n))$. In this machine, replace the co–nondeterministic part of computation by a nondeterministic one, what, according to our assumption, must be possible, without asymptotically deteriorating the computational time complexity of the resulting *nondeterministic machine* M'. Clearly, M' recognizes the same language as M does, and runs in time $O(T(n)/\log T(n))$.

Consider now any language $L \in NTIME_1(T(n)) - NTIME_1(T(n)/\log T(n))$. According to theorem 4.1 and our assumptions on $T(n)$, such a non–empty language L must exist. However, M' recognizes L in time $O(T(n)/\log T(n))$, what is a contradiction with the choice of L. Therefore, $co - NTIME_1(T(n)) \not\subseteq NTIME_1(T(n))$.

□

Obviously, from the previous theorem it already follows that nondeterminism is not closed under complement. Nevertheless, for establishing a clear picture as far as the exact relationship between nondeterminism, and co–nondeterminism is concerned, we shall show that similar theorem as the latter one holds also for nondeterministic time, that is not included in co–nondeterministic time:

Theorem 4.4 *Let $T(n)$ be such that $\sqrt{T(n)}$ is a fully time and space constructible function, with $n^2 \in o(T(n))$, and $T(n+1)/\log T(n+1) \in o(T(n))$. Then*

$$NTIME_1(T(n)) \not\subseteq co - NTIME_1(T(n))$$

The respective proof mirrors that of the previous theorem, by making use of theorem 3.1., instead of 2.1., and of theorem 4.2 instead of 4.1.

Next we show that determinism does not equal nondeterminism for single–tape time–bounded TM computations; for this purpose we introduce also the complexity class $DTIME_1(T(n))$, with obvious meaning.

Theorem 4.5 *Let $T(n)$ be such that $\sqrt{T(n)}$ is a fully time and space constructible function, with $n^2 \in o(T(n))$, and $T(n+1)/\log T(n+1) \in o(T(n))$. Then*

$$DTIME_1(T(n)) \subset NTIME_1(T(n))$$

Proof Outline: It is clear that $DTIME_1(T(n)) \subseteq NTIME_1(T(n))$; suppose that $DTIME_1(T(n)) = NTIME_1(T(n))$. Then $NTIME_1(T(n))$ must be closed under complement, since $DTIME_1(T(n))$ obviously is. But the previous theorem shows that $NTIME_1(T(n))$ is not closed under complement — a contradiction.

□

Similarly, we show that determinism does not equal co–nondeterminism either:

Theorem 4.6 *Let $T(n)$ be such that $\sqrt{T(n)}$ is a fully time and space constructible function, with $n^2 \in o(T(n))$, and $T(n+1)/\log T(n+1) \in o(T(n))$. Then*

$$DTIME_1(T(n)) \subset co - NTIME_1(T(n))$$

Proof Outline: It is clear that $DTIME1(T(n)) \subseteq co - NTIME_1(T(n))$. If $DTIME_1(T(n)) = co - NTIME_1(T(n))$, then $co - NTIME_1(T(n))$ would be closed under complement — a contradiction with theorem 4.3.

□

Now consider the relationship of the previous classes to the single tape Σ_2 time. It appears that this class can be also strictly separated from the classes considered above:

Theorem 4.7 *Let $T(n)$ be such that $\sqrt{T(n)}$ is a fully time and space constructible function, with $n^2 \in o(T(n))$, and $T(n+1)/\log T(n+1) \in o(T(n))$. Then*

$$NTIME_1(T(n)) \subset \Sigma_2 - TIME_1(T(n))$$

Proof Outline. If $NTIME_1(T(n)) = \Sigma_2 - TIME_1(T(n))$, then from corollary 2.1 we get a contradiction in form $NTIME_1(T(n)) \subseteq NTIME_1(T(n)/\log T(n)))$ with the hierarchy from theorem 4.1.

□

Theorem 4.8 *Let $T(n)$ be such that $\sqrt{T(n)}$ is a fully time and space constructible function, with $n^2 \in o(T(n))$, and $T(n+1)/\log T(n+1) \in o(T(n))$. Then*

$$co - NTIME_1(T(n)) \subset \Sigma_2 - TIME(T(n))$$

Proof Outline. If $co - NTIME_1(T(n)) = \Sigma_2 - TIME_1(T(n))$, then from theorem 4.7 we get $NTIME_1(T(n)) \subset co - NTIME_1(T(n))$, what is in a contradiction with theorem 4.4.

□

Diagrammatically, the relation among all above considered complexity classes is depicted in Fig.1. In this picture, unlike in the most of other similar pictures, the position of an oval inside an other oval, really means a strict containment.

5 Conclusions

Separation results for the fundamental time bounded complexity classes, related to single–tape TM computations, for a large spectrum of complexity bounds, have been shown. This seems to be the first occasion where similar results for a universal model of computation have been achieved.

A lot of work remains to be done. Among the first questions that could be amenable for the further research is the extension of the previous results to single–tape

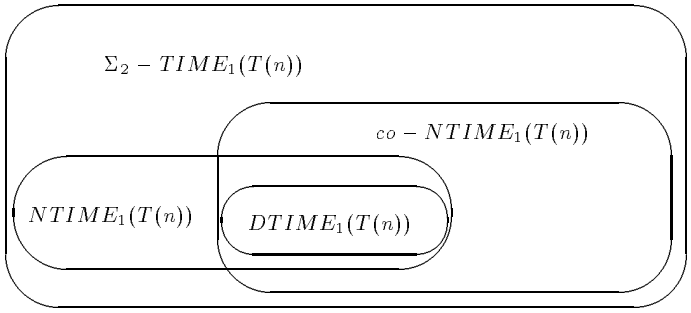

Figure 4.1: Relations among the basic single–tape complexity classes

off-line TMs. This may bring us some steps further on our way towards the solution of similar problems for multitape TMs.

Acknowledgement. The author thanks to his colleague Stanislav Žák for his helpful comments on the first version of the present paper.

Bibliography

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

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

[3] Li, M. — Neuféglise, H. — Torenvliet, L. — van Emde Boas, P.: On Space Efficient Simulations. *ITLI Prepublication Series*, CS–89–03, University of Amsterdam, 1989

[4] Lorys, K. — Liskiewicz, 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

[5] 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

[6] Paterson, M.: Tape Bounds for Tape–Bounded Turing Machines, *JCSS*, Vol. 6, 1972,pp. 115–124

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