Nondeterministic Computations for Which Space is More Powerful than Time

Wiedermann, Jiří
2016

Institute of Computer Science
The Czech Academy of Sciences

Nondeterministic Computations for Which Space is More Powerful Than Time

Jiří Wiedermann

Technical report No. V-1231

June 2016

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: +420 266 053 520, fax: +420 286 585 789,
e-mail:wieder@cs.cas.cz

Institute of Computer Science
The Czech Academy of Sciences

Nondeterministic Computations for Which Space is More Powerful Than Time[1]

Jiří Wiedermann

Technical report No. V-1231

June 2016

Abstract:

In their seminal 1978 paper Hopcroft, Paul and Valiant have shown that for deterministic multitape Turing machines TIME(T(n)) $\subseteq$ SPACE(T(n)/ $\log$ T(n)), i.e., space is more powerful than time as a computational resource. For a large class of nondeterministic computations we show a similar relation: NTIME(T(n)) $\subseteq$ NSPACE(T(n)/ $\log$ T(n)). Moreover, the previous relation is inherent to any computation from that class meaning that no simulation is needed in order to establish the previous relation between the respective time and space complexity classes. Consequently, for any $\varepsilon > 0$ any nondeterministic Turing machine of time complexity $T(n)$ from the class under consideration can be simulated deterministically in time $2^{\varepsilon T(n)}$. The main analytical tool for proving these results are crossing sequences generalized for the case of multitape machines.

Keywords:
Nondeterministic computation, crossing sequences, complexity

[1]This research was partially supported by the institutional fund RVO 67985807.

1 Introduction

The contents of this paper is related to the seminal 1978 paper by Hopcroft, Paul and Valiant where the authors have shown that every deterministic multitape Turing machine of time complexity $T(n)$ can be simulated by a deterministic Turing machine of space complexity $T(n)/\log T(n)$ [5]. In this way they showed, for the first time, that for deterministic multitape Turing machines space is more powerful than time as a computational resource. In order to save space in a given deterministic computation its portions are repeatedly re-computed thus avoiding the necessity of storing intermediate results. Within this context the authors have noted: *"It is an interesting open problem whether* $\mathsf{NTIME}(t) \subseteq \mathsf{NSPACE}(t/\log t)$. *The difficulty here is that in going back and repeating a portion of a computation we cannot be sure that the same sequence of choices is made the second time."*

For a large class of nondeterministic computations this paper shows that there is an intrinsic relation between these two measures of a similar type that holds for the deterministic case: $\mathsf{NTIME}(\mathsf{T}(\mathsf{n})) \subseteq \mathsf{NSPACE}(\mathsf{T}(\mathsf{n})/\log \mathsf{T}(\mathsf{n}))$, for any $\varepsilon > 0$ and a sufficiently large n. This is achieved by exploiting a classical analytical tool from computational complexity theory — viz crossing sequences generalized for the case of multitape machines. Crossing sequences allow insight into the structure of the flow of data in multitape computations and enable derivation of lower and upper bounds on their time complexity.

Crossing sequences have been known since nineteen sixties when they have served as a tool for investigation of single-tape computations — cf. the works by Hennie [4] and Hartmanis [3]. Subsequent efforts for generalizing the definition of crossing sequences for the case of multitape computations were futile. For a more detailed overview of the related developments cf. [8]. The next progress occurred in 1979 when G. Wechsung [7] proposed a generalized definition of crossing sequences usable in the framework of multitape Turing machine computations and introduced the notion of crossing complexity measure for such machines. He showed that constant bounds on crossing complexity yield an infinite hierarchy of complexity classes and remarked briefly that the new measure allows to prove lower time bounds on multitape computations. More than thirty years later the latter idea has been elaborated in [8]. Here, the original Wechsung's definition of the crossing systems has been simplified and the results concerning the relation between time complexity and the number of crossing systems have been shown. Unfortunately, the results presented in that paper have not lead to a proof that nondeterministic space is more powerful than nondeterministic time.

The present paper reports a further progress in the question of nondeterministic time-space relation. Namely, we show that under a certain technical assumption concerning the computations of the nondeterministic Turing machine at hand and relation between its space complexity and the size of its crossing sequences there is no need at all to take some extra measures in order to achieve a computation in a smaller space. This is because there is an inherent relation between time and space of the respective computations: any such machine of time complexity $T(n)$ is "automatically" of space complexity $T(n)/\log T(n)$. This result holds for both deterministic and nondeterministic two tape Turing machines (and hence for all nondeterministic Turing machines thanks to the fact that two tape nondeterministic machines can simulate any multitape Turing machine in linear time and space [2]). The technical assumptions we have in mind are twofold. First, we only consider two-tape nondeterministic Turing machines, without an extra input tape, which at each step perform a move by both its heads. Second, we require that the space complexity $S(n)$ of the machine under consideration grows asymptotically faster than the product $nC(n)$ where $C(n)$ is a complexity measure (defined in the paper) related to the size of crossing sequences of the underlying computation: $\lim_{n\to\infty} nC(n)/S(n) = 0$. Thus, within the previous restrictions this result solves the open problem posed in [5].

As a consequence, under the same assumptions we get that for any $\varepsilon > 0$ a nondeterministic Turing machine of time complexity $T(n)$ can be simulated by a deterministic Turing machine in time $O(2^{\varepsilon T(n)})$. Within the given restrictions this again answers positively the question raised recently in [6].

The present paper can be seen as a follow-up of the previous paper [8] from which it differs in some important aspects since it at least partially resolves the general question of time versus space in nondeterministic multitape Turing machines computations. The current paper also brings a significant simplification of the basic definitions and arguments.

The structure of the paper is as follows. Section 2 describes the main framework for considering crossing sequences and a special case of Turing machines under consideration — the so-called diagonal Turing machine. Section 3 recalls the fundamental notion of crossing systems and crossing sequences and proves an important theorem stating that in any diagonal computation no crossing sequence can occur more than once. Section 4 presents the main result concerning the separation of time and space for a large class of nondeterministic computations. As a consequence a solution to the open problem from [6] — an improved upper bound on the time complexity of exhaustive search algorithms — is given. Section 5 contains the conclusions.

Every effort has been made in order to make the paper self-contained, independent from the previous papers [7] or [8]. It contains all the necessary definitions and facts needed for the exposition of the main results of the paper.

2 Preliminaries

In the sequel we consider a standard model of a k-tape nondeterministic Turing machine $\mathcal{M}$, $k \geq 1$, with set of states Q, alphabet Σ, transition relation δ and input w of length n initially written on one of the tapes. That is, the first n cells on this tape are used to store the input and this space is counted in the space complexity of the respective machine. However, in what follows we will only deal with 2-tape machines since it is known that the latter machines can simulate any k-tape machines in linear time and space (cf. [2]). The notions of the *head configuration*, of the *configuration*, of the *computation,* of the *acceptance*, and those of time and space complexity are defined in a usual way (cf. [1]).

Any computation of a two tape machine can be captured in a three dimensional *computational diagram.* In it, the x-axis (the y-axis) corresponds to the working head position on the first (second) tape, while the t-axis represents the flow of time (in discrete steps). A point $[x, y, t]$ in such a diagram corresponds to the situation when the first (second) head scans cell number x (y) at time t.

In such a computational diagram the movement of working heads in any computation can be captured by a *computational trajectory*, or *computational curve c.* It is an oriented curve starting in point $[1, 1, 1]$ connecting, in chronological order, the centers of those cells in the diagram (which are three-dimensional cubes) that correspond to the tape-cells visited during a computation. For definitiveness, in the three-dimensional space we will always imagine that the axes x, y, and t are oriented as usually in such cases; i.e., axis y goes left to right from the coordinate origin, axis x goes "towards us", and axis t goes up.

The smallest three-dimensional block (rectangular parallelepiped) enclosing the computational curve is called the *computational space.*

In the computational diagram we will consider the so-called *grid-planes.* Grid-planes are two-dimensional planes that are parallel with either the (x, t)-plane (such grid-planes are called *sagittal grid-planes*) or the (y, t)-plane (*frontal grid-planes*) and go through the centers of the grid cells they intersect. Obviously, sagittal planes are perpendicular to the frontal planes. For any tape, grid-plane number i goes through the centers of cells corresponding to tape position i on that tape. For definiteness, let tape one correspond to x-axis and tape two to y-axis. The cells on each tape are numbered from 1 onwards.

In the sequel for simplification of our considerations we will require that no part of length greater or equal 1 of the computational curve will be parallel with any axis. Such a curve can either diagonally cross a grid-plane or it can "touch" it in a single point (making a "v turn" at that point). Technically, this means that in each step each working head of the machine must perform a move. Such machines will be called *diagonal machines.* In [8] it has been shown that any deterministic machine can be simulated by a diagonal machine of the same type in linear time and space. Thus, the result also holds for nondeterministic machines. Computations of any diagonal Turing machine are called *diagonal computations.*

3 Basic Notions: Crossing Systems and Crossing Sequences

Now we proceed to the definition of crossing sequences. Our approach is inspired by that of G. Wechsung [7]. In order to understand the rest of the paper it is necessary to introduce all formal definitions related to the notion of crossing sequences. First we will define the so-called crossing system as a certain set of grid-planes crossed by the computational curve in a diagonal computation. The cut-points of the curve with these grid-planes correspond to those tape-cells whose contents are mutually dependent in the given computation (cf. Fig. 3.1 and 3.2, and Proposition 3.3). The consideration of diagonal computations has allowed a simplified definition of a crossing system than that in [7] because in diagonal computations the head moves within the grid-planes need not be considered. Moreover, our approach differs from that given in [7] and [8] since it explicitly considers the time dimension of computations.

In what follows we will say that a computational curve of a diagonal computation *cuts a grid-plane* (or that a *grid-plane cuts the curve*) if and only if the curve crosses the grid-plane in a diagonal manner at some point. That is, we do not say that a curve crosses a grid-plane when, making a v-turn, it only touches the grid-plane at some point.

Definition 3.1 (Crossing system) *Let c be a computational curve of a diagonal computation, let f be a grid-plane cutting c.*

A crossing system $\mathcal{S}$ *on c is a set of grid-planes that is constructed recursively as follows:*

- $f \in \mathcal{S}$;
- *if some g already in $\mathcal{S}$ crosses c in one or more points (so-called* cut-points*), then all frontal and sagittal grid-planes incidental with such cut-points are added to $\mathcal{S}$.*

The construction of $\mathcal{S}$ terminates when no further grid-planes can be added to it in accordance with the previous rule.

Note that there must be two perpendicular grid-planes incidental with each cut-point thanks to the fact that we consider diagonal computations. Also note that in the case when c only touches a grid-plane f there is no grid-plane perpendicular to f at that point (unless it is generated by some cut point from a different part of c).

For a given computation there can be many different crossing systems. For a given computational trajectory each crossing system is uniquely determined by any of its grid-planes. This means that no grid-plane can belong to two different crossing systems. This has an important consequence:

Proposition 3.2 *For a given diagonal computation all crossing systems are disjoint.*

Point $[i, j, t]$ on computational curve c is called the *crossing point* for grid-plane g if and only if c cuts g at time t. Note that there can be crossing points $[i, j, t_1]$ and $[i, j, t_2]$ on g with $t_1 \neq t_2$.

Obviously, for any diagonal computation c of a machine M a concrete crossing system $\mathcal{S}$ can also be defined with the help of its crossing points. If $C_1, C_2, \ldots, C_s$ are all crossing points of $\mathcal{S}$ listed in a chronological order w.r.t. their crossing times of grid-planes within $\mathcal{S}$, then the respective crossing system will be denoted as $\mathcal{S}_c[C_1, C_2, \ldots, C_s]$.

In fact, such a view of a crossing system gives a motivation for its definition capturing the inter-dependence of values stored in tape cells pertinent to that crossing system. This is illustrated by the following proposition.

Proposition 3.3 *Let $\mathcal{S}_c[C_1, C_2, \ldots, C_s]$ be a crossing system defined via its crossing points, let C_i, with $1 \leq i \leq s$ be any crossing point of $\mathcal{S}$ pertinent to time t and tape cells S_1 and S_2, respectively.*

Then in addition to the current state of the machine the value stored at S_1 at that time depends only on the value stored at that cell at time when this cell was visited by the respective head for the last time before time t (if ever), and similarly for the cell S_2. This information can be inferred from $\mathcal{S}_c[C_1, C_2, \ldots, C_s]$.

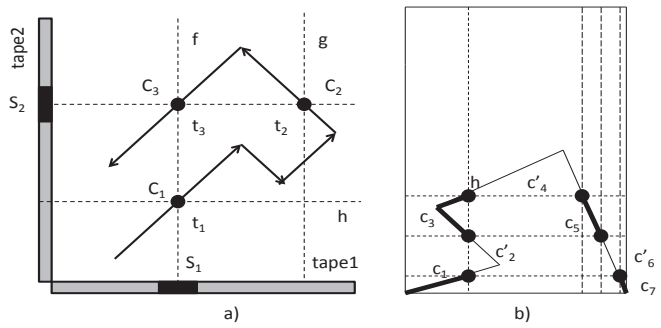

Figure 3.1: a) Interdependence of values pertinent to the crossing points b) Mixing the computations from Fig. 3.2.

The proof follows from the definition of a crossing system and the respective crossing points.

For instance (cf. Fig. 3.1 a)), let C_3 be a crossing point in the intersection of two grid-planes f and g, respectively, pertinent to time t_3, C_2 at time t_2 be a crossing point on g, and finally C_1 at t_1 be a crossing point at f. Let there be no crossing points on g and f corresponding to times between t_2 and t_3, and t_1 and t_3, respectively, with $t_1 < t_2 < t_3$.

Then the value in cells S_1 and S_2, respectively, at time t_3 depends only on the values stored at S_2 at time t_2 and S_1 at time t_1, respectively (and, of course, on the state of the machine at time t_3).[2]

Similar claims can also be formulated for any other configurations of crossing points.

For our further purposes we will want to abstract from the concrete crossing systems given by the enumeration of their crossing points. We will be interested only in their general properties that can be defined without referring to the concrete position of grid-planes and concrete crossing times.

First we define the so-called *crossing pattern*. For a given crossing system it defines the chronological order in which trajectory c visits the respective crossing points.

Definition 3.4 (Crossing pattern) *Let $\mathcal{S} = \{h_1, \ldots, h_k, v_1, \ldots, v_\ell\}$ be a crossing system for a diagonal computation c, with the frontal grid-planes $h_1, \ldots, h_k$ and the sagittal grid-planes $v_1, \ldots, v_\ell$, for some $k \geq 1$ and $\ell \geq 1$. Then the sequence $\tau_{\mathcal{S}} = (p_1, \ldots, p_s)$ is called the* crossing pattern *of $\mathcal{S}$ if and only if*

- *c crosses $\mathcal{S}$ exactly s times, and*
- *for all $1 \leq k \leq s$ it holds: if the k-th crossing point of c with $\mathcal{S}$ lays on the grid-planes h_i and v_j then $p_k = (i, j)$ (the crossing points are ordered chronologically along the direction of the trajectory c).*

Note that in the notion of the crossing pattern there is neither information on the location of the grid-planes in the computational diagram nor on the crossing times when the trajectory passes through the grid-planes.

Our next notion will be that of the *crossing sequence*.

Definition 3.5 (Crossing sequence) *Let $\mathcal{S}$ be a crossing system and $\tau_{\mathcal{S}} = (p_1, \ldots, p_s)$ its crossing pattern, both pertinent to a diagonal computation c of machine M. Let $\delta_i \in \delta$ for $i = 1, 2, \ldots, s$ be the instructions from the instruction set of M that was used by M when leaving the i-th crossing point.*

Then $\Gamma_{\mathcal{S},c} = (p_1, \delta_1), (p_2, \delta_2), \ldots, (p_s, \delta_s)$ is called the crossing sequence *of c on $\mathcal{S}$. Items (p_i, δ_i) are called the* elements *of $\Gamma_{\mathcal{S},c}$. Number s is called the* length of the crossing sequence $\Gamma_{\mathcal{S},c}$.

Now we are ready to state and prove the following theorem dealing with the case when in a diagonal computation we have two crossing systems with the same crossing sequences.

[2] All figures in this paper depict a two-dimensional projection of a computational curve into plane (x, y).

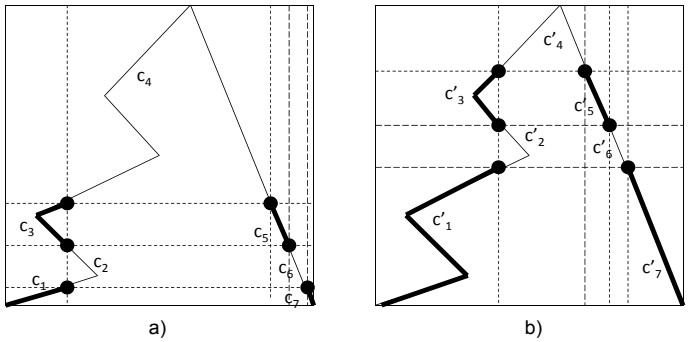

Figure 3.2: A computation with two identical crossing sequences. The computation without the redundant part is depicted in Fig. 3.1 b).

Theorem 3.6 *Let c be a diagonal computation of a two-tape deterministic or nondeterministic Turing machine $\mathcal{M}$ and let there exist two different crossing systems on c, $\mathcal{S}_c[C_1, C_2, \ldots, C_s]$ and $\mathcal{S}'_c[C'_1, C'_2, \ldots, C'_s]$, respectively, such that neither crossing system is incidental with the input and their crossing sequences are identical: $\Gamma_{\mathcal{S},c} = \Gamma_{\mathcal{S}',c}$.*

Then there is an equivalent diagonal computation c' of $\mathcal{M}$ with only one crossing system with the crossing sequence $\Gamma_{\mathcal{S},c}$.

Sketch of the proof: Let $s = c_1, c_2, \ldots$ be a segmentation of c induced by the crossing points $C_1, C_2, \ldots$ of the crossing system $\mathcal{S}$ for computation c (cf. Fig. 3.2 a)) and analogously, let $s' = c'_1, c'_2, \ldots$ be a segmentation of c induced by the crossing points $C'_1, C'_2, \ldots$ of the crossing system $\mathcal{S}'$ (Fig. 3.2 b)). Note that each segment finds itself in a "box" bounded by the corresponding grid-planes or by the boundary of the computational space.

The next observation is that in both crossing systems the odd or even indexed segments, respectively, can be colored by the same color (black or white, respectively) and that the equally colored segments lay in the same vertical or horizontal strips (cf. Fig. 3.2). This holds because a segment changes its color at each occasion when it diagonally passes a cut point proceeding from one box to the next one. (Note that a diagonal crossing is the only possibility for a curve to move between boxes.) At each such occasion, the curve changes both its vertical and its horizontal strip. It follows that in each strip the segments must be of the same color.

An other important observation is that the computational trajectory in the equally colored segments can be computed independently of the computations in segments colored by the other color. This is due to the fact that the computations in segments of one color depend only on the data in the corresponding parts of tapes (called *sectors* hereafter) that are disjoint with the data pertinent to computations of the other color and on the state of the machine when entering a segment.

That is, the computation in the chronologically first segment is performed over the empty tape sectors. Subsequently, when computing chronologically the next segment of the same color in the same strip, it is always computed over the tape contents left by the lastly performed computation over these sectors in the equally colored segments. Upon entering a new segment the initial head configuration is always given by the corresponding member of the crossing sequence $\Gamma_{\mathcal{S},c}$. The interface between the differently colored segments is also provided by the elements of the respective crossing sequence.

Finally, note that thanks to our assumption the interface between the black and white segments in both crossing systems $\mathcal{S}$ and $\mathcal{S}'$ is the same. This means that we can combine segments with the same interface from both crossing systems. W.l.o.g. we can assume that segment c_1 is the prefix of the segment c'_1. Then $c' = c_1, c'_2, c_3, c'_4, \ldots$ is a possible computational curve of $\mathcal{M}$. In this "combined" computation the ends of the respective segments are separated by a crossing system with the crossing sequence $\Gamma_{\mathcal{S},c}$. The resulting combined computation is depicted in Fig. 3.1 b).

Remark: if on c there exist more than two different crossing systems with the same crossing sequences, then the previous process of eliminating redundant parts in a computation can be repeated. □

The previous theorem reveals the main sense of the former tedious definitions since it offers a tool for eliminating redundant parts in nondeterministic computations. Note that in the case of deterministic computations an occurrence of two crossing systems with the same crossing sequence would mean that there is an infinite loop in such a computation.

Corollary 3.7 *To any diagonal accepting computation of a two-tape nondeterministic Turing machine $\mathcal{M}$ there exists an equivalent accepting computation in which each crossing sequence not incidental with the input occurs at most once.*

The reason why we have only considered the crossing sequences not incidental with the input is that the parts of computations incidental with the input cannot be eliminated since this would effectively mean that we have omitted some parts of the input.

4 Time Versus Space in Nondeterministic Computations

For diagonal computations we define a new complexity measure related to the notion of crossing sequences.

Definition 4.1 *Let $\mathcal{M}$ be a diagonal two-tape nondeterministic Turing machine. The* crossing complexity *$C(n)$ of $\mathcal{M}$ is the minimum of the maximal lengths of all crossing sequences taken over all possible inputs to $\mathcal{M}$ of length n and all possible accepting computations on these inputs.*

It is straightforward that for a diagonal deterministic or nondeterministic two-tape Turing machine of time complexity $T(n)$, space complexity $S(n)$ and crossing complexity $C(n)$ it holds $n \leq S(n) \leq T(n) \leq S(n)C(n)$ and $1 \leq C(n) \leq T(n)$.

In order to prove non-trivial relations among the crossing complexity and time and space complexity we will need the following lemma estimating the number of different crossing sequences of a given size.

Lemma 4.2 *There exists a constant $b > 0$ such that for any $s \geq 1$ the number $D(s)$ of different crossing sequences of size s is at most $D(s) \leq b^s$.*

Proof: We start with the upper bound on $D(S)$. In accordance with Definition 3.5 consider a computational curve c generating a crossing sequence $\Gamma_{\mathcal{S},c}$. Reaching a crossing point C_i on that curve which is not chronologically the last one on c, the curve proceeds towards the next crossing point. Doing so, the curve cannot cross any of the grid-planes in $\mathcal{S}$ in a point that is not the cut-point of the grid-planes in $\mathcal{S}$ since this would generate a new crossing point not included in $\Gamma_{\mathcal{S},c}$. Thus, the next crossing point C_{i+1} can find itself in one of the upper corners of the adjacent hyper-rectangles, or immediately above the C_i (seen in the 3D computational diagram). In general, there are 4 hyper-rectangles adjacent to C_i and therefore c can proceed to an upper corner of any of them, or "right up", toward a point lying above the current point. That is, in general c has 9 possible choices to proceed.

It follows that when starting from the initial (i.e., chronologically the first) crossing point, the curve has at each time at most 9 possible choices how to proceed, and this holds for any crossing point until the last, the s-th crossing point is reached. Therefore, each possible sequence of crossing points visited by a computational curve in crossing system $\mathcal{S}$ can be seen as a path of length s in an 9-ary tree having in its vertices the cut-points of c with the respective grid-planes from $\mathcal{S}$. Obviously, there are 9^s different paths of length s in such a tree and each of them corresponds to a possible crossing sequence of length s in the crossing system at hand.

Next, the above mentioned crossing sequences can differ by the assignment of instructions into the respective crossing points. An instruction in an element of a crossing sequence can be chosen in $d = 4r^2|\Sigma|^4$ different ways where 4 is the number of possible combinations of moves of both heads (stationary moves are excluded), factor r^2 corresponds to the number of different pairs of the current and the new states, and $|\Sigma|^4$ corresponds to the number of possible ways of rewriting the pair of the currently scanned symbols by the new symbols. Thus, there are at most d^s different sequences of instructions of length s each of which can be assigned to a possible arrangement of crossing points as estimated above.

Altogether, the final upper estimate of the number of different crossing sequences of length $s \geq 1$ is $D(s) \leq s9^s d^s \leq b^s$, for some constant $b > 0$.

□

Lemma 4.3 *There exists at least $d^s = (4r^2|\Sigma|^4)^s$ different crossing sequences of length s.*

Proof: In the previous lemma we have estimated that there are $d^s = (4r^2|\Sigma|^4)^s$ ways of assigning an instruction to the elements of a crossing sequence of length s. This number can be taken as a lower bound on the number of different crossing sequences of length s.

□

Theorem 4.4 *Let $\mathcal{M}$ be a diagonal deterministic or nondeterministic two-tape Turing machine of time complexity $T(n)$, space complexity $S(n)$ and crossing complexity $C(n)$ with input w, $|w| = n$.*

If $\lim_{n\to\infty} nC(n)/S(n) = 0$ then

(i) $C(n) = \Omega(\log S(n))$

(ii) $T(n) = \Omega(S(n) \log S(n))$

(iii) $S(n) = O(T(n)/\log T(n))$.

Proof: In order to prove i) we will have to get an upper bound on $S(n)$ in terms of $C(n)$. Clearly, $S(n)$ is not greater than the sum of the number of cells occupied by all crossing sequences of length at most $C(n)$ since each element of a crossing sequence occupies one cell at any tape and the crossing sequences pertinent to different crossing systems are disjoint (cf. Proposition 3.2).

However, there is a complication with the crossing systems incidental with that part of a tape where input w is written. Namely, we cannot claim that all corresponding crossing sequences must be different because in a computation we cannot skip a part of the input what may happen when applying Theorem 3.7. Therefore, we can only assume that solely sequences whose crossing points are not incidental with the input must be different. There are at most n crossing systems incidental with the input. The respective sequences thus occupy space of size at most $n(C(n) + 1)$. The sequences incidental with the remaining part of space must all be different. Thanks to our assumption on the asymptotic growth of $S(n)$ such sequences exist.

Thus, $S(n)$ equals at most the sum of space occupied by n crossing sequences (which need not all be different) incidental with the input elements plus the sum of all different crossing sequences of length $1, 2, \ldots, C(n)$.

According to Lemma 4.2 for each s the number of the latter crossing sequences of length s is upper-bounded by b^s, for some constant $b > 0$.

In the worst case a crossing sequence of length s can occupy up to $s + 1$ cells on two tapes — namely in the case when the crossing system consists of a single grid-plane cut by s perpendicular grid-planes.

Using this upper-bound we eventually reach our final estimate of $S(n)$: $S(n) \leq nC(n) + \sum_{s=1}^{C(n)}(s+1)b^s < nC(n) + C(n)(C(n)+1)b^{C(n)}$ Since for a sufficiently large n, $nC(n) \leq S(n)/2$ we get

$$S(n)/2 \leq b^{C(n)+\log_b 2C^2(n)} < q^{C(n)} \qquad (*)$$

for a suitable constant $q > 0$.

Taking base q logarithm of both sides we finally obtain $C(n) = \Omega(\log S(n))$.

Remark: A computation corresponding to space utilization estimated above need not exist at all. This, however, does not matter since we were merely interested in an upper bound on the space complexity.

As far as (ii) is concerned, consider an infinite sequence σ_1 in shortlex order of all crossing sequences of length s, for $s = 1, 2, \ldots$. Let σ_2 be the *finite* sequence of different crossing sequences used in computation c of $\mathcal{M}$ which are not incidental with the input and let this sequnce be in shortlex order. Now substitute the i-th sequence in σ_2 by the i-th member of σ_1, for $i = 1, 2, \ldots$. By this we get a finite sequence σ_3 which is a prefix of σ_1. Note that σ_2 and σ_3 have the same number of crossing

sequences and in σ_3 no crossing sequence is longer than the corresponding crossing sequence in σ_2. Hence the sum of the lengths of the crossing sequences of σ_3 is not greater than the sum of the lengths of the crossing sequences of σ_2.

Now, for each n the computational time $T(n)$ is not less than the sum of the lengths of the crossing sequences in σ_3. Let r be the maximal number such that σ_3 contains all crossing sequences of length $s = 1, 2, \ldots, r$. Then, using the lower bound from Lemma 4.4 the number r in computation c of length $T(n)$ must satisfy the inequality $T(n) \geq \sum_{s=0}^{r} sd^s \geq rd^r$ and, at the same time, the inequality $(*)$ (with $r = C(n)$). From the latter inequality we have $r \geq \log_q(S(n)/2)$ and therefore, using (i), we get our final estimate $T(n) = \Omega(rS(n)) = \Omega(S(n) \log S(n))$.

Finally, in order to prove (iii) we show that the asymptotic growth of $S(n) = O(T(n)/\log T(n))$ is a maximal one such that $T(n) = \Omega(S(n) \log S(n))$ still holds.

If $S(n) = O(T(n)/\log T(n))$ then $T(n) = \Omega(S(n) \log S(n))$ since $T(n) \geq S(n)$. This means that iii) holds.

Now suppose that $S(n)$ grows asymptotically faster than $T(n)/\log T(n)$. We show that then iii) cannot hold.

To that end assume $\lim_{n\to\infty} T(n)/(S(n) \log T(n)) = 0$. Then for any $\varepsilon > 0$ there exists an n_0 such that for all $n > n_0$, $T(n)/(S(n) \log T(n)) < \varepsilon$. Therefore

$$T(n) = \Omega(S(n) \log S(n)) = \Omega\left(\frac{T(n)}{\varepsilon \log T(n)} \log \frac{T(n)}{\varepsilon \log T(n)}\right) =$$

$$= \Omega\left(\frac{T(n)}{\varepsilon}\left[1 - \frac{\log(\varepsilon \log T(n))}{\log T(n)}\right]\right)$$

For a fixed ε and growing n the second expression in the square brackets tends to 0. Therefore, choosing a sufficiently small ε and considering a sufficiently large n the last Ω-expression cannot be made smaller than $T(n)$ and therefore the last relation cannot hold. Thus, $S(n)$ cannot grow asymptotically faster than $T(n)/\log T(n)$.

The above mentioned results hold for both deterministic and nondeterministic two-tape machines since in their derivation no specific properties of either type of machines have been used. □

A remark concerning the assumption on the asymptotic growth of $S(n)$ in the previous theorem is in order. The assumption that $\lim_{n\to\infty} nC(n)/S(n) = 0$ was chosen in order to ensure that in computations at hand there exist crossing systems not incidental with the input. This assumption mirrors the one needed for proving a similar theorem in the case of single tape computation [3]. For single tape computations the respective assumption is of form $\lim_{n\to\infty} n/S(n) = 0$. Note that this is a special form of our assumption since in the single-tape case a crossing system always consumes one unit of space whereas in the two-tape case it can consume up to $C(n)$ units of space.

Theorem 4.4 gives a partial answer to an open problem concerning efficient deterministic simulation of nondeterministic computations. Namely, in [6] the authors have asked whether for any $\varepsilon > 0$, $\mathsf{NTIME}(T(n)) \subseteq \mathsf{DTIME}(2^{\varepsilon T(n)})$. It appears that if the assumptions of Theorem 4.4 are satisfied then the above mentioned relation holds:

Theorem 4.5 *Let $\mathcal{M}$ be a diagonal nondeterministic two-tape Turing machine of time complexity $T(n)$, space complexity $S(n)$ and crossing complexity $C(n)$.*

If $\lim_{n\to\infty} nC(n)/S(n) = 0$ then for any $\varepsilon > 0$ $\mathcal{M}$ can be simulated by a deterministic Turing machine $\mathcal{M}'$ in time $O(2^{\varepsilon T(n)})$.

Proof: Under the assumption on the growth of $S(n)$ from Theorem 4.4 it follows that for machine $\mathcal{M}$ it holds $S(n) = O(T(n)/\log T(n)))$. This machine can be simulated by machine $\mathcal{M}'$ using the standard breadth-first simulation technique in time $2^{O(S(n))}$ (cf. [1]). Now, for any $\varepsilon > 0$ and a sufficiently large n we have $cT(n)/\log T(n) \leq \varepsilon T(n)$ for any constant $c > 0$. The claim of the theorem follows.

□

5 Conclusions

In this paper we have systematically applied the concept of crossing sequences to computations of nondeterministic multitape Turing machines. With the help of these novel techniques we were able to give positive answers to two open problems, albeit not in their full generality. The first open problem, from nineteen seventies [5], asked whether space is more powerful than time for nondeterministic computations. The second one recently asked whether one can improve, at least slightly, upper bounds on the standard exhaustive search [6]. For answering these problems we needed additional assumptions concerning the simultaneous movements of Turing machine heads and the relation between space and crossing complexity of the underlaying machines. Whether these problems can be answered positively under milder assumptions remains to be seen.

Both results illustrate the power and usefulness of crossing sequences for obtaining non-trivial results concerning the complexity of multitape Turing machine computations. For instance, in [8] we showed that nondeterministic multitape computations with crossing complexity $C(n)$ can be nondeterministically simulated in space $O(C(n) \log T(n))$.

By focusing to the information transfer among the tapes within multitape computations the concept of crossing sequences brings new insights into the nature of general Turing machine computations. It opens new paths towards obtaining lower and upper bounds on the respective computations. Our approach might present a first step for further and more significant results.

Bibliography

[1] Aho, A.V., Hopcroft, J.E., Ullman, J.D.: *The Design and Analysis of Computer Algorithms*, Reading, MA: Addison-Wesley; 1974.

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

[3] Hartmanis, J.: Computational Complexity of One-Tape Turing Machine Computations. JACM, Vol. 15, No. 2, April 1968, pp. 325–339.

[4] Hennie, F.C.: One-tape, off-line Turing machine computations. Information and Control 8, 1965, pp. 553–578.

[5] Hopcroft, J.E., Paul, W.J., Valiant, L.G.: On Time Versus Space. J. ACM, vol. 24, no. 2, pp. 332–337, 1977.

[6] Kalyanasundaram, S., Lipton, R.J., Regan, K.W., Shokrieh, F.: Improved simulation of nondeterministic Turing machines. Theor. Comput. Sci. 417: 66–73 (2012).

[7] Wechsung, G.: A Crossing Measure for 2-Tape Turing Machines. In: Proc. Mathematical Foundations of Computer Science MFCS 1979, LNCS vol. 74, Springer, 1979, pp. 508–516.

[8] Wiedermann, J.: Complexity of nondeterministic multitape computations based on crossing sequences. In: Proc. Descriptional Complexity of Formal Systems DCFS 2011, LNCS Vol. 6808, Springer Berlin Heidelberg, 2011, pp. 314–327.