On Tight Separation for Blum Measures Applied to Turing Machine Buffer Complexity

Jiří Šíma[a,*], Stanislav Žák[a]

[a]*Institute of Computer Science, Academy of Sciences of the Czech Republic, P.O. Box 5, 182 07 Prague 8, Czech Republic*

Abstract

We formulate a very general tight diagonalization method for the Blum complexity measures satisfying certain additional axioms. We apply this method to two new so-called distance and buffer complexity measures for Turing machine computations. These measures are sensitive to long-distance transfers of information on the worktape and they prove to be mutually related. In particular, the buffer complexity counts the number of necessary block uploads into a virtual double-block buffer of the worktape which is divided into blocks. Thus, they can be used for investigating the buffering aspects of Turing computations. We start this study by proving a tight separation which shows that a very small increase in the buffer (or distance) complexity bound (roughly from $f(n)$ to $f(n+1)$) brings provably more computational power to both deterministic and nondeterministic Turing machines even for unary languages. We also obtain hierarchies of the distance and buffer complexity classes.

Keywords: Turing machine, hierarchy, buffer complexity, diagonalization

1. Introduction

The theory of computational complexity is one of the major attempts to understand the phenomenon of computation. One of the key tasks of the theory is to find out how an increase or decrease of limits set on the computational resources can influence the computational power of different types of computational devices. In history, the efforts to answer questions of this type led to a long sequence of various separation and hierarchy results for particular computational devices and complexity measures, e.g. chronologically [1, 2, 3, 4, 5, 6, 7, 8].

The present paper follows this direction of research. We formulate a very general tight diagonalization method issuing from [5] which works for the Blum complexity measures [9] satisfying certain additional axioms. We apply this method to two new nontraditional measures, namely the so-called *distance* and

[*]Corresponding author
Email addresses: `sima@cs.cas.cz` (Jiří Šíma), `stan@cs.cas.cz` (Stanislav Žák)

buffer complexities, which are introduced for both deterministic and nondeterministic Turing machines (TM) with one worktape. These measures are sensitive to long transfers of information on the worktape of a Turing machine while the short transfers are not counted.

In particular, a given computation by TM is characterized, among others, by a sequence of worktape head positions $(h_t)_{t\geq 0}$ where h_t is the head position on the worktape just after t computational steps (i.e. at time instant t). The distance complexity d^δ is the minimum length of its so-called δ-distance subsequence $(h_{t_i})_{i\geq 0}$ which, starting with $t_0 = 0$, is defined inductively as follows. If one restricts only to the segment of computation since time t_i, then the next t_{i+1} is the first time instant at which the worktape head is exactly at distance $\delta(n)$ (measured in the number of tape cells) from some of its previous positions within the computational segment under consideration. Note that parameter $\delta(n)$ depends on the input length n.

The buffer complexity b^δ is defined similarly by using a so-called δ-buffer subsequence such that the distance in the definition of t_{i+1} is measured from the previous member h_{t_i} of this subsequence. The δ-buffer subsequence thus divides the worktape into disjoint blocks of size $\delta(n)$. The buffer complexity proves to be related to the distance measure by $d^{2\delta} \leq b^\delta \leq d^\delta$. Furthermore, consider a virtual buffer of the worktape whose size is two such blocks. Then the buffer complexity measures the number of necessary block uploads into this buffer. In this way, the buffer or distance measure can be used for investigating the buffering aspects of Turing computations.

We start our study by separation and hierarchy results for the distance and buffer complexity which are surprisingly very tight. This indicates that the new complexity measures are appropriate tools for classifying the computations. The tightness in the results requires that the worktape alphabet is fixed and the measure is not applied to TM computations directly but instead to their simulations on a fixed universal Turing machine. The results are of the form that a shift by one in the argument of the complexity bound (and of parameter δ plus an additive constant) leads to a strictly greater computational power. In the case of a linear complexity bound, the increase in the bound by an additive constant is sufficient to gain more power. For the hierarchies of complete languages the increase in the bound is slightly larger. The main tool of the proof is the general diagonalization method derived from [5] which is newly formulated for suitable Blum complexity measures. The results are valid even for unary languages, which strengthens a preliminary version of this paper [10] containing the proofs only for binary alphabet.

The paper is organized as follows. After a brief review of basic definitions regarding Turing machines and complexity measures in Section 2, we define a diagonalizer and prove a general diagonalization theorem for Blum complexity measures in Section 3. In Section 4, the distance and buffer complexity measures are introduced and related to each other. The separation for these measures is proven in Section 5 while the corresponding hierarchies are presented in Section 6. The results are summarized in Section 7 where possible further research directions are outlined.

2. Preliminaries

By a *language* L we mean any set of words over binary alphabet, that is, $L \subseteq \{0,1\}^*$. We will formulate our results for *unary languages* over one-symbol alphabet for which we assume $L \subseteq \{1\}^*$. In addition, we say that two languages L and L' are equivalent, that is, $L \sim L'$ iff they differ only in a finite number of words. For a class C of languages we define $\mathcal{E}(C) =_{\rm df} \{L' \,|\, (\exists L \in C)\, L \sim L'\}$. Clearly, $L \notin \mathcal{E}(C)$ implies that L differs from any language of $\mathcal{E}(C)$ on infinitely many words.

By a *Turing machine* we mean any finite-control machine with two-way read-only input tape and with one semi-infinite worktape (infinite to the right) with alphabet $\{0,1,b\}$ (b stands for blank) and endmarker # at the left end side (at the 0th cell of the worktape), allowing for both the deterministic or nondeterministic versions.

The *programs* are words from $\{0,1\}^+$. If p is a program, then M_p is a corresponding machine which implements p. For any machine M, we denote by $L(M)$ the language accepted by M, and define $L_p = L(M_p)$. By p_M we mean a program of M. For any input word u, a *universal machine* starts its computation with some program p stored at the left end side of the worktape and it simulates machine M_p on u.

By a *complexity measure* we mean any (partial) mapping $c : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbf{N} \cup \{\infty\}$ where $\mathbf{N}$ denotes the set of natural numbers (including 0). Informally, $c(p,u,v)$ is intended to measure the complexity of computation by machine M_p on input word u starting with v stored at the left end side of the worktape. In general, we assume that c satisfies *Blum axioms* [9], that is, $c(p,u,v) < \infty$ iff M_p with initial worktape content v halts on input u, and there is a Turing machine which, given $p,u,v \in \{0,1\}^*$ and $m \in \mathbf{N}$, decides whether $c(p,u,v) = m$. We also use notation $c(p,u) = c(p,u,\varepsilon)$ when ε is the empty word.

In Section 4, we will introduce the complexity measures whose definitions depend parametrically on the input length. This means that complexity measure c is in fact defined as a (uniform) sequence of complexity measures $(c^n)_{n=0}^{\infty}$ where c^n is employed for input words of length n, that is, $c(p,u) = c^{|u|}(p,u)$ where $|u|$ is the length of word u. For such c we denote by c' the complexity measure specified by a shifted sequence $(c^{n+1})_{n=0}^{\infty}$ so that $c'(p,u) = c^{|u|+1}(p,u)$ for any input word $u \in \{0,1\}^*$. For nonparametric measures, c' and c coincide.

Let $P \subseteq \{0,1\}^+$ be a recursively enumerable set of programs of the machines in question (e.g. nondeterministic Turing machines), that is, there exists a Turing machine which will enumerate all valid programs of P. For any $p \in P$, for any complexity measure c and for any complexity bound $f : \mathbf{N} \rightarrow \mathbf{N}$, we define language $L_p(c,f) =_{\rm df} \{u \in L_p \,|\, c(p,u) \leq f(|u|)\}$ to be a set of words accepted by machine M_p within the complexity bound f measured by c. The *complexity class* $c(f) = \{L_p(c,f) \,|\, p \in P\}$ contains all such languages for $p \in P$ while the respective class of *complete* languages is denoted by a corresponding capital letter and defined as $C(f) = \{L_p \,|\, L_p = L_p(c,f)\,,\, p \in P\}$. Clearly, $C(f) \subseteq c(f)$. For the purpose of a tight separation of complexity classes we will measure the complexity of a computation by machine M as the complexity of simulating p_M on a fixed universal machine U. In particular, for any complexity measure c we define its universal version as $c_U(p,u) =_{\mathrm{df}} c(p_U,u,p)$.

In the notation of complexity classes, $f(n+1)$ stands for complexity bound $f' : \mathbf{N} \to \mathbf{N}$ such that $f'(n) = f(n+1)$ for each $n \in \mathbf{N}$. Notice that inclusion $c(f) \subseteq c(f(n+1))$ need not, in general, be true even for nondecreasing f since some of the languages $L_p(c,f)$ cut out of L_p by complexity bound f cannot be delimited by $f(n+1)$. Recall that complexity bound f is recursive if there is a Turing machine that computes f.

3. The Diagonalization Theorem

Our diagonalization method issuing from [5] is based on a *diagonalizer* for Blum complexity measure c and its recursive bound $f : \mathbf{N} \to \mathbf{N}$, which is a Turing machine Δ working on unary inputs $u \in \{1\}^*$. We will below describe its program p_Δ by using a high-level pseudocode. The program is composed of two main parts: precomputation A and simulation S. The precomputation A is formally defined as an infinite process which is terminated when the worktape head of Δ leaves a delimited working space.

```
Diagonalizer $\Delta$

input $u$

Precomputation $A$

    check whether the input $u$ is of the form $1^n$ and construct a working space
    $W_n$ of length $|W_n| = \log n$ on the worktape (e.g. using a binary counter)

    for every $i = 1,2,3,\ldots$ until $W_n$ suffices do

      Phase $G$: on the untapped part of $W_n$ generate the next element $p_i$
         of a sequence $(p_i)_{i=1}^{\infty}$ which contains each program $p \in P$ infinitely
         many times

      Phase $T$: using only $W_n$, test whether $u_i \overset{?}{\in} L_{p_i}(c_U,f)$ where $u_i \in \{1\}^*$
         is the shortest input to $\Delta$ for which $A$ generates $p_i$ (within $W_{|u_i|}$)

    enddo

Simulation $S$ { last generated $p_i$ is on the worktape }

    if $A$ terminates during its phase $G$ without generating $p_{i+1}$ then
      accept $u$ iff $u_i \notin L_{p_i}(c_U,f)$

    else { $A$ terminates during its phase $T$ without deciding $u_i \overset{?}{\in} L_{p_i}(c_U,f)$ }
      simulate machine $M_{p_i}$ on input $u1$ as universal machine $U$ would do
```

Note that the leaving condition which breaks precomputation A when the logarithmic working space W_n on the worktape is consumed can, in fact, be replaced by any suitable condition which can be simply tested so that the longer the input word to Δ, the more computation of A is performed.

Furthermore, we define function $z : \mathbf{N} \rightarrow \mathbf{N}$ so that $z(i)$ is the minimum $j \in \mathbf{N}$ such that A finishes its phase T on input u_i1^j and decides whether $u_i \overset{?}{\in} L_{p_i}(c_U, f)$. For any $i \in \mathbf{N}$, denote $R_i = \{u_i1^j \,|\, 0 \leq j < z(i)\}$ whose union $R = \bigcup_{i \geq 1} R_i$ contains the inputs on which Δ performs the simulation in S. Now we are ready to introduce our diagonalization theorem which proves a tight separation for suitable Blum complexity measures:

Theorem 1. *Let Δ be a diagonalizer for Blum complexity measure c and its recursive bound $f : \mathbf{N} \rightarrow \mathbf{N}$ which satisfy the following two conditions:*

1. *for any $u \in R_i$, $c'(p_\Delta, u) = c_U(p_i, u1)$*
2. *for any $u \notin R$, $c'(p_\Delta, u) \leq f(|u| + 1)$.*

Then $L =_{df} L_{p_\Delta}(c', f(n+1)) \in c'(f(n+1)) \setminus \mathcal{E}(c_U(f))$.

Proof. On the contrary, suppose that $L \in \mathcal{E}(c_U(f))$. Hence, $L \sim L_p(c_U, f)$ for some $p \in P$. By the definition of diagonalizer Δ we know that p occurs in the sequence $(p_i)_{i=1}^{\infty}$ generated during phase G of precomputation A infinitely many times. Thus, there is $i \in \mathbf{N}$ such that $L_{p_i}(c_U, f)$ differs from L only on words shorter than u_i.

For any $u_i1^j \in R_i$, we know that $u_i1^j \in L$ iff $u_i1^j \in L_{p_\Delta}$ & $c'(p_\Delta, u_i1^j) \leq f(|u_i1^j| + 1)$ (by definition of L) iff $u_i1^j \in L_{p_\Delta}$ & $c_U(p_i, u_i1^{j+1}) \leq f(|u_i1^{j+1}|)$ (by condition 1) iff $u_i1^{j+1} \in L_{p_i}(c_U, f)$ (by definition of Δ) iff $u_i1^{j+1} \in L$ since $L_{p_i}(c_U, f)$ differs from L only on words shorter than u_i. Hence, $u_i \in L$ iff $u_i1^{z(i)} \in L$ iff $u_i1^{z(i)} \in L_{p_\Delta}$ (as for $u = u_i1^{z(i)} \notin R$ we know $c'(p_\Delta, u) \leq f(|u| + 1)$ by condition 2) iff $u_i \notin L_{p_i}(c_U, f)$ (by definition of Δ) iff $u_i \notin L$, which is a contradiction, and thus $L \notin \mathcal{E}(c_U(f))$. $\square$

Note that condition 1 in Theorem 1 is naturally satisfied for Blum measures that are related to the space complexity. Clearly, the logarithmic space consumed within precomputation A is hidden in the space complexity of simulation S, which implies condition 1. For example, this is not true for the time complexity of Δ which is a sum of the times needed for performing A and S, respectively. In this case, the diagonalizer itself must control the time of simulation S [5]. Thus, for Blum complexity measures that do not meet condition 1 one can possibly modify Δ in part S so that Δ simulates M_{p_i} by U on input $u1$ and accepts iff U accepts within complexity bound $f(|u1|)$. In addition, the witness language L is defined as complete language L_{p_Δ}. Hence, for any $u_i1^j \in R_i$, we obtain that $u_i1^j \in L =_{\text{df}} L_{p_\Delta}$ iff $u_i1^{j+1} \in L_{p_i}$ & $c_U(p_i, u_i1^{j+1}) \leq f(|u_i1^{j+1}|)$ by this modified definitions of Δ and L which compensate for condition 1 in the proof of Theorem 1 accordingly.

The diagonalization Theorem 1 provides the witness language L which is unary and is thus outside the respective complexity class of languages over any multi-symbol alphabet. This strengthens our previous simpler diagonalizer working on binary inputs which was presented in a preliminary version of this paper [10].

4. The Distance and Buffer Measures

We introduce two new complexity measures which are sensitive to long-distance transfers of information on the worktape. For any computation by Turing machine M, denote by $h_t \in \mathbf{N}$ the head position on the worktape just after t computational steps by M (i.e. at time instant t) which equals the distance (in the number of tape cells) from the current worktape head position to the leftmost worktape cell with endmarker # whose position is thus zero (e.g. $h_0 = 1$). This defines a sequence $h_0, h_1, h_2, \ldots$ of the worktape head positions which is *finite* for halting computations and satisfies $|h_{t+1} - h_t| \leq 1$ for any $t \geq 0$. For any positive recursive function $\delta : \mathbf{N} \to \mathbf{N}$, we inductively define its so-called *δ-distance subsequence* $h_{t_0}, h_{t_1}, h_{t_2}, \ldots$ for a computation on input word u as follows:

1. $t_0 = 0$
2. $t_{i+1} = \min\{t \,|\, (\exists\, t')\ \ t_i \leq t' < t \ \ \&\ \ |h_t - h_{t'}| = \delta(|u|)\}$.

In other words, if we take into account only the worktape positions $h_{t_i}, h_{t_i+1}, h_{t_i+2}, \ldots$ visited by the head after t_i computational steps, then t_{i+1} is the first time instant at which the worktape head is exactly at distance $\delta(|u|)$ from some of these previous positions. Similarly, a so-called *δ-buffer subsequence* is defined when condition 2 above is replaced by

2'. $t_{i+1} = \min\{t \,|\, t > t_i \ \ \&\ \ |h_t - h_{t_i}| = \delta(|u|)\}$

in which t' is restricted to t_i, and hence $\delta(|u|)$ divides $h_{t_i} - 1$ for any $i \in \mathbf{N}$. This means that the δ-buffer subsequence divides the worktape into blocks of length $\delta(|u|)$.

For any program $p \in P$ and for any input word u, we define the *distance complexity* $d^\delta(p, u)$ to be the minimum length of δ-distance subsequence over all halting computations of M_p on u where δ is a paramater of the distance measure depending on the input length. In addition, we define formally $d^\delta(p, u) = \infty$ if M_p does not halt on u. Similarly, the *buffer complexity* measure b^δ is defined by using δ-buffer subsequences. Obviously, the distance and buffer complexity measures satisfy the Blum axioms. Moreover, they can mutually be related as follows.

Lemma 1. *Let $\delta : \mathbf{N} \to \mathbf{N}$ be a positive recursive function. For any program $p \in P$ and any input word u, inequality $d^{2\delta}(p, u) \leq b^\delta(p, u) \leq d^\delta(p, u)$ holds.*

Proof. Let $(h_{t_i^1})_{i\geq 0}$, $(h_{t_i^2})_{i\geq 0}$, and $(h_{t'_i})_{i\geq 0}$ be the δ-distance, 2δ-distance, and δ-buffer subsequences, respectively, for a computation of p on input word u. It suffices to prove $t_i^1 \leq t'_i \leq t_i^2$ for any meaningful i (note that the subsequences are typically of different length). By definition we know $t_0^1 = t'_0 = t_0^2 = 0$. On the contrary, let $j \geq 1$ be the minimum index such that $t_j^1 \leq t'_j < t'_{j+1} < t_{j+1}^1$ or $t'_j \leq t_j^2 < t_{j+1}^2 < t'_{j+1}$. Suppose first that $t_j^1 \leq t'_j < t'_{j+1} < t_{j+1}^1$. According to the definition of δ-distance subsequence, t_{j+1}^1 is the first time instant such that there is t' satisfying $t_j^1 \leq t' < t_{j+1}^1$ and $|h_{t_{j+1}^1} - h_{t'}| = \delta(|u|)$, but for $t' = t'_j$ we know $t_j^1 \leq t'_j < t_{j+1}^1$ and $|h_{t'_{j+1}} - h_{t'_j}| = \delta(|u|)$, which contradicts $t'_{j+1} < t_{j+1}^1$. Thus, assume $t'_j \leq t_j^2 < t_{j+1}^2 < t'_{j+1}$. According to the definition of δ-buffer subsequence, t'_{j+1} is the first time instant such that $t'_{j+1} > t'_j$ and $|h_{t'_{j+1}} - h_{t'_j}| = \delta(|u|)$, which implies $|h_{t_{j+1}^2} - h_{t'_j}| < \delta(|u|)$ and $|h_{t'_j} - h_{t_j^2}| < \delta(|u|)$. Hence, $2\delta(|u|) > |h_{t_{j+1}^2} - h_{t'_j}| + |h_{t'_j} - h_{t_j^2}| \geq |h_{t_{j+1}^2} - h_{t_j^2}| = 2\delta(|u|)$, which is a contradiction completing the argument. $\square$

The buffer complexity can be used for investigating the buffering aspects of Turing computations. In particular, imagine the control unit has a virtual buffer memory for the worktape whose capacity is two blocks of length $\delta(|u|)$ so that the worktape head position has to be within this buffer. This means that the buffer uploads the block from the worktape next to the right when the head leaves the buffer to the right, and the buffer contents are shifted to the left so that the head finds itself in the midpoint of the buffer while the block on the left is stored to the worktape. This is reminiscent of a super-head reading the whole block as a super-cell of length $\delta(|u|)$ and moving to the right. Similarly, the buffer moves to the left when the head leaves the buffer to the left. Thus, the buffer complexity measures the number of necessary buffer uploads.

5. The Separation Result

We first show a lemma concerning the distance complexity of simulating a machine on the universal Turing machine.

Lemma 2. *Let U be a fixed universal machine, p_M be a program of machine M, and $\delta : \mathbf{N} \to \mathbf{N}$ be a positive recursive function. Then $d^\delta(p_M, u) = d_U^{\delta+|p_M|}(p_M, u)$ for any input word u.*

Proof. (Sketch) Machine M is simulated by universal Turing machine U so that its program p_M is being shifted along the worktape following the shifts of the head of M. The distance δ on the worktape of M is thus transformed to the distance $\delta + |p_M|$ on the worktape of U. $\square$

A tight separation for the distance complexity measure is formulated in the following theorem. We point out that this is a quite strong result since a very small increase in the distance complexity bound (roughly from f to $f(n+1)$ plus a constant) brings provably more computational power to both deterministic and nondeterministic Turing machines working on unary inputs.

Theorem 2. *Let U be a fixed universal machine. Assume $\delta : \mathbf{N} \to \mathbf{N}$ and $f : \mathbf{N} \to \mathbf{N}$ are positive recursive functions, and $\delta(n+1) \geq \log_2$. Then $L = L_{p_\Delta}(d^{\delta(n+1)}, f(n+1)) \in d_U^{\delta(n+1)+|p_\Delta|}(f(n+1)) \setminus \mathcal{E}(d_U^\delta(f))$ where Δ is a diagonalizer for complexity measure d^δ and its bound f.*

Proof. We will employ Theorem 1 for the distance complexity d^{δ} and its bound f for which the two conditions have to be verified. For any input word $u = 1^n$ to Δ, the distance complexity $d^{\delta(n+1)}(p_A, u)$ of precomputation A equals 1 by the assumption of $\delta(n+1) \geq \log_2$ since the capacity of working space W_n is $\log n$. Thus, for input word $u \in R_i$, the distance complexity $d^{\delta(n+1)}(p_\Delta, u) = d_U^{\delta}(p_i, u1)$ is dominated by the complexity of simulation S, which secures condition 1 of Theorem 1. Condition 2 which has now the form $1 = d^{\delta(n+1)}(p_\Delta, u) \leq f(|u|+1)$ for any $u \notin R$ follows from the assumption of $f > 0$. By applying Theorem 1, we obtain language $L = L_{p_\Delta}(d^{\delta(n+1)}, f(n+1)) \in d^{\delta(n+1)}(f(n+1))$ which satisfies $L \notin \mathcal{E}(d_U^{\delta}(f))$. Moreover, $L \in d_U^{\delta(n+1)+|p_\Delta|}(f(n+1))$ by Lemma 2, which completes the proof. □

It is obvious that Lemma 2 and Theorem 1 are also valid for the buffer complexity b^{δ}, which can be verified simply by replacing d with b in their statements and proofs.

6. The Hierarchies

In order to achieve a fine-grained hierarchy for the distance (or buffer) complexity we will employ the classes of complete languages $D_U^{\delta}(f)$. Unlike $d_U^{\delta}(f)$, these classes $D_U^{\delta}(f)$ prove to be linearly ordered with respect to inclusion for hierarchies of functions δ and f.

Lemma 3. *Let U be a fixed universal machine. Assume $\delta_1 : \mathbf{N} \to \mathbf{N}$, $\delta_2 : \mathbf{N} \to \mathbf{N}$ are positive recursive functions, and $f_1 : \mathbf{N} \to \mathbf{N}$, $f_2 : \mathbf{N} \to \mathbf{N}$ are recursive complexity bounds. If $\delta_1 \leq \delta_2$ and $f_1 \leq f_2$, then $D_U^{\delta_1}(f_1) \subseteq D_U^{\delta_2}(f_2)$.*

Proof. Let $L_p \in D_U^{\delta_1}(f_1)$, that is, $L_p = L(M_p) = L_p(d_U^{\delta_1}, f_1)$ for some $p \in P$. We will prove that $L_p(d_U^{\delta_1}, f_1) \subseteq L_p(d_U^{\delta_2}, f_2)$ which implies $L_p = L_p(d_U^{\delta_2}, f_2)$, and consequently $L_p \in D_U^{\delta_2}(f_2)$.

Suppose that $u \in L_p(d_U^{\delta_1}, f_1)$. For $k \in \{1, 2\}$, let $h_{t_0^k}, h_{t_1^k}, h_{t_2^k}, \ldots$ be the δ_k-distance subsequence for a computation of U on the input word u starting with p on the worktape. We will prove that $t_i^1 \leq t_i^2$ for any meaningful i. We know $t_0^1 = t_0^2 = 0$. On the contrary, let $j \geq 1$ be the minimum index such that $t_j^1 \leq t_j^2 < t_{j+1}^2 < t_{j+1}^1$. This means that there is t' such that $t_j^2 \leq t' < t_{j+1}^2$ and $|h_{t_{j+1}^2} - h_{t'}| = \delta_2(|u|) \geq \delta_1(|u|)$ by the definition of δ_2-distance subsequence, which contradicts the definition of δ_1-distance subsequence since $t_{j+1}^2 < t_{j+1}^1$ was assumed. Thus, $t_i^1 \leq t_i^2$, which implies $d_U^{\delta_2}(p, u) \leq d_U^{\delta_1}(p, u) \leq f_1(|u|) \leq f_2(|u|)$. Hence, $u \in L_p(d_U^{\delta_2}, f_2)$, which completes the argument for $D_U^{\delta_1}(f_1) \subseteq D_U^{\delta_2}(f_2)$. □

We will show that any language from complexity class $d^{\delta}(f)$ is complete for slightly larger distance parameter δ' and complexity bound f'.

Lemma 4. *Let $\delta : \mathbf{N} \to \mathbf{N}$ be a positive recursive function and $f : \mathbf{N} \to \mathbf{N}$ be a recursive complexity bound such that for any input word u, the binary representation of function values $\delta(|u|)$ and $f(|u|)$ can be computed by a Turing* *machine Γ so that $d^{\delta'}(p_\Gamma, u) \leq g_f^\delta(|u|)$ where $\delta' = \delta + 8\log\delta + 2\log f$, for some complexity bound $g_f^\delta : \mathbf{N} \to \mathbf{N}$. Then $L_p(d^\delta, f) \in D^{\delta'}(f + g_f^\delta)$ for any $p \in P$.*

Proof. Denote $L = L_p(d^\delta, f)$ and $M = M_p$. We will define machine M' that simulates M on any input word u and halts immediately before $d^\delta(p_M, u) > f(|u|)$, which implies $L(M') = L$. In addition, we will ensure that $L(M') \in D^{\delta'}(f + g_f^\delta)$, which gives desired $L \in D^{\delta'}(f + g_f^\delta)$. The main ideas of how M' computes follow. At the beginning, M' constructs on its worktape two segments of length $8\log\delta(|u|)$ and $2\log f(|u|)$, respectively, by using Γ. Then M' simulates M so that M' shifts the two segments along the worktape following the worktape head of M.

The first segment of length $8\log\delta(|u|)$ serves for identifying the time instant t_{i+1} corresponding to the next worktape head position $h_{t_{i+1}}$ from the δ-distance subsequence $h_{t_0}, h_{t_1}, h_{t_2}, \ldots$ for a computation of M on u. Recall that t_{i+1} is the first time instant t since t_i for which there is t' such that $t_i \leq t' < t$ and $|h_t - h_{t'}| = \delta(|u|)$. For this purpose, it suffices to keep and update the current head position h_τ, the current minimum and maximum head positions $h_{\min} = \min\{h_t \,|\, t_i \leq t \leq \tau\}$ and $h_{\max} = \max\{h_t \,|\, t_i \leq t \leq \tau\}$ since time instant t_i as differences $h_\tau - h_{t_i}$, $|h_{\min} - h_{t_i}|$, and $|h_{\max} - h_{t_i}|$, respectively, which consumes $3\log\delta(|u|)$ worktape cells of the segment. Moreover, a test whether the current maximum distance $|h_{\min} - h_{t_i}| + |h_{\max} - h_{t_i}|$ equals $\delta(|u|)$ requires the value of $\delta(|u|)$ occupying additional $\log\delta(|u|)$ cells to be precomputed and shifted with the first segment. Similarly, the second segment of length $2\log f(|u|)$ serves for halting the computation after $f(|u|)$ members of the δ-distance subsequence. In particular, the value of $f(|u|)$ occupying $\log f(|u|)$ cells is computed at the beginning and decremented after each head position of the δ-distance subsequence is reached.

In fact, the implementation of the ideas above requires the full double length of the two segments since the worktape alphabet of M' is restricted to $\{0,1\}$ according to our definition of Turing machine. In particular, it suffices to replace each bit by a pair of bits. The first bit of this pair indicates "marked/non-marked", which is used e.g. for comparing two parts of segments, and the second one represents the value of the original bit. Hence, the length of the two segments follows, which guarantees $L(M') \in D^{\delta'}(f + g_f^\delta)$. □

Now we are ready to prove the hierarchy theorem for the distance complexity classes of complete languages.

Theorem 3. *Let U be a fixed universal machine. Assume $\delta : \mathbf{N} \to \mathbf{N}$ and $f : \mathbf{N} \to \mathbf{N}$ are positive nondecreasing recursive functions, and $\delta(n+1) \geq \log_2$. Define recursive functions $\delta' : \mathbf{N} \to \mathbf{N}$ and $f' : \mathbf{N} \to \mathbf{N}$ as*

$$\delta' = \delta(n+1) + 8\log(\delta(n+1)) + 2\log(f(n+1)) \qquad (1)$$
$$f' = f(n+1) + g_{f(n+1)}^{\delta(n+1)} \qquad (2)$$

where $g_{f(n+1)}^{\delta(n+1)} : \mathbf{N} \to \mathbf{N}$ is a nondecreasing recursive complexity bound such that for any input word u, the binary representation of function values of $\delta(|u|+1)$

and $f(|u|+1)$ can be computed by a Turing machine Γ so that $d^{\delta'}(p_\Gamma, u) \leq g_{f(n+1)}^{\delta(n+1)}(|u|)$. Then $D_U^\delta(f) \subsetneqq D_U^{\delta'+|p_\Delta|}(f')$ where Δ is a diagonalizer for distance complexity measure d^δ and its bound f.

Proof. Since δ, f, and $g_{f(n+1)}^{\delta(n+1)}$ are nondecreasing functions we know that $\delta \leq \delta' + |p_\Delta|$ and $f \leq f'$ according to (1) and (2), respectively. Hence, $D_U^\delta(f) \subseteq D_U^{\delta'+|p_\Delta|}(f')$ follows from Lemma 3. Define $L = L_{p_\Delta}(d^{\delta(n+1)}, f(n+1))$. According to Theorem 2, we know $L \notin \mathcal{E}(d_U^\delta(f)) \supseteq \mathcal{E}(D_U^\delta(f))$ which gives $L \notin D_U^\delta(f)$. On the other hand, $L \in D^{\delta'}(f')$ by Lemma 4, which implies $L \in D_U^{\delta'+|p_\Delta|}(f')$ according to Lemma 2. This completes the proof of the theorem. □

The argument is similar for the hierarchy of buffer complexity classes $B_U^\delta(f)$ of complete languages, which is formulated in the following theorem.

Theorem 4. *Let U be a fixed universal machine. Assume $\delta : \mathbf{N} \to \mathbf{N}$ and $f : \mathbf{N} \to \mathbf{N}$ are positive nondecreasing recursive functions, and $\delta(n+1) \geq \log_2$. Define recursive functions $\delta' : \mathbf{N} \to \mathbf{N}$ and $f' : \mathbf{N} \to \mathbf{N}$ as*

$$\delta' = \delta(n+1) + 4\log(\delta(n+1)) + 2\log(f(n+1)) \quad (3)$$

$$f' = f(n+1) + g_{f(n+1)}^{\delta(n+1)} \quad (4)$$

where $g_{f(n+1)}^{\delta(n+1)} : \mathbf{N} \to \mathbf{N}$ is a nondecreasing recursive complexity bound such that for any input word u, the binary representation of function values of $\delta(|u|+1)$ and $f(|u|+1)$ can be computed by a Turing machine Γ so that $b^{\delta'}(p_\Gamma, u) \leq g_{f(n+1)}^{\delta(n+1)}(|u|)$. Then $B_U^\delta(f) \subsetneqq B_U^{\delta'+|p_\Delta|}(f')$ where Δ is a diagonalizer for buffer complexity measure b^δ and its bound f.

Proof. The proof proceeds the same way as in the case of the distance complexity. In particular, the proof of Theorem 3 is based on Theorem 2 and Lemmas 2–4. We already know that the statements of Theorem 2 and Lemma 2, in which d is replaced with b, are valid for the buffer complexity. The same applies to Lemma 3 in whose proof the time instant t' coincides with t_j^2. The only slight change appears in the proof of the buffer complexity version of Lemma 4. In the definition of machine M', the first segment serves for identifying the time instant t_{i+1} corresponding to the next worktape head position $h_{t_{i+1}}$ from the δ-buffer subsequence $h_{t_0}, h_{t_1}, h_{t_2}, \ldots$, which is the first time instant t such that $t > t_i$ and $|h_t - h_{t'}| = \delta(|u|)$. Thus, it suffices to keep and update only the current head position h_τ since time instant t_i as a difference $h_\tau - h_{t_i}$ which consumes $\log \delta(|u|)$ worktape cells of the segment. Hence, the first segment is of half length $4 \log \delta(|u|)$ as compared to the distance complexity version, which appears in formula (3). □

7. Conclusions

In this paper we have introduced the new distance and buffer complexity measures for computations on Turing machines with one worktape. These measures can be used for investigating the buffering aspects of Turing computations. As a first step along this direction, we have proven quite strong separation and hierarchy results which are valid even for unary languages. Many questions concerning e.g. the comparison to other complexity measures, reductions, completeness and complexity classes remain open for further research.

We have also formulated our diagonalization method for the general Blum complexity measures satisfying additional axioms, which is interesting on its own. Analogous theorems can possibly be proven for other types of machines such as those with auxiliary pushdown or counter, or with oracle etc.

Acknowledgments

J.Š.'s research was partially supported by project GA ČR P202/12/G061 and RVO: 67985807. S.Ž.'s research was partially supported by project GA ČR P202/10/1333 and RVO: 67985807.

References

[1] S. A. Cook, A hierarchy for nondeterministic time complexity, Journal of Computer and System Sciences 7 (4) (1973) 343–353.

[2] J. I. Seiferas, Relating refined space complexity classes, Journal of Computer and System Sciences 14 (1) (1977) 100–129.

[3] J. I. Seiferas, M. J. Fischer, A. R. Meyer, Separating nondeterministic time complexity classes, Journal of the ACM 25 (1) (1978) 146–167.

[4] I. H. Sudborough, Separating tape bounded auxiliary pushdown automata classes, in: Proceedings of the STOC'77 Ninth Annual ACM Symposium on Theory of Computing, 1977, pp. 208–217.

[5] S. Žák, A Turing machine time hierarchy, Theoretical Computer Science 26 (1983) 327–333.

[6] E. Allender, R. Beigel, U. Hertrampf, S. Homer, Almost-everywhere complexity hierarchies for nondeterministic time, Theoretical Computer Science 115 (2) (1993) 225–241.

[7] V. Geffert, Space hierarchy theorem revised, in: Proceedings of the MFCS 2001 Twenty-Sixth Symposium on Mathematical Foundations of Computer Science, Vol. 2136 of LNCS, 2001, pp. 387–397.

[8] J. Kinne, D. van Melkebeek, Space hierarchy results for randomized models, in: Proceedings of the STACS 2008 Twenty-Fifth Annual Symposium on Theoretical Aspects of Computer Science, 2008, pp. 433–444.

[9] M. Blum, A machine-independent theory of the complexity of recursive functions, Journal of the ACM 14 (2) (1967) 322–336.

[10] S. Žák, J. Šíma, A Turing machine distance hierarchy, in: Proceedings of the LATA 2013 Seventh International Conference on Language and Automata Theory and Applications, Vol. 7810 of LNCS, 2013, pp. 570–578.