Electronic Colloquium on Computational Complexity, Report No. 36 (2015)

# Verifying whether One-Tape Turing Machines Run in Linear Time

David Gajser
IMFM, Jadranska 19, 1000 Ljubljana, Slovenija
david.gajser@fmf.uni-lj.si

February 17, 2015

**Abstract**

We discuss the following family of problems, parameterized by integers $C \geq 2$ and $D \geq 1$: Does a given one-tape $q$-state Turing machine make at most $Cn + D$ steps on all computations on all inputs of length $n$, for all $n$?

Assuming a fixed tape and input alphabet, we show that these problems are co-NP-complete and we provide good lower bounds. Specifically, these problems can not be solved in $\mathrm{o}(q^{(C-1)/4})$ non-deterministic time by multi-tape Turing machines. We also show that the complements of these problems can be solved in $\mathrm{O}(q^{C+2})$ non-deterministic time and not in $\mathrm{o}(q^{(C-1)/4})$ non-deterministic time by multi-tape Turing machines.

**Keywords.** one-tape Turing machine; crossing sequence; linear time; running time; lower bounds

## 1 Introduction

For a function $T : \mathbb{N} \to \mathbb{R}^+$, let us define the problem $\mathrm{HALT}_{T(n)}$ as

*Given a non-deterministic multi-tape Turing machine, does it run in time $T(n)$?*

In other words, $\mathrm{HALT}_{\mathrm{T(n)}}$ is the set of all Turing machines that make **at most** $T(n)$ steps on each computation on each input of length $n$, for all $n$. Note that there is no big O notation in the definition of the problem $\mathrm{HALT}_{T(n)}$, i.e. we are not asking whether a given Turing machine runs in time $\mathrm{O}(T(n))$. This is because it is undecidable even whether a given deterministic one-tape Turing machine runs in constant, i.e. $\mathrm{O}(1)$ time[1]. However, the problem $\mathrm{HALT}_C$ is decidable for any constant $C$ and one would hope that $\mathrm{HALT}_{T(n)}$ is decidable also for linear functions $T$. It was proven in [6] that this is not the case in general, but if we restrict the input to one-tape Turing machines, we get a decidable problem [3]. This motivates the definitions of the problems that we study in our paper.

For $C, D \in \mathbb{N}$, we consider the problem $\mathrm{HALT}_{(C,D)}$ defined as

*Given a non-deterministic one-tape Turing machine, does it run in time $Cn + D$?*

and the problem $\mathrm{DHALT}_{(C,D)}$ defined as

*Given a deterministic one-tape Turing machine, does it run in time $Cn + D$?*

Now that we restricted the input to one-tape Turing machines, can we also verify superlinear time bounds? It was shown in [3] that there is no algorithm that would verify a time bound $T(n) = \Omega(n \log n)$, $T(n) \geq n + 1$, for a given one-tape Turing machine. But if $T(n) = \mathrm{o}(n \log n)$ is tangible enough, then

[1]This is a folkloric result, see also [3]

ISSN 1433-8092

there is an algorithm that verifies whether a given one-tape Turing machine runs in time $T(n)$. It is also shown in [3] that a one-tape Turing machine that runs in time $\mathrm{o}(n \log n)$ must actually run in linear time[2], which implies that the most "natural" algorithmically verifiable time-bound for one-tape Turing machines is the linear one.

For the rest of the paper, we fix an input alphabet $\Sigma$, $|\Sigma| \geq 2$, and a tape alphabet $\Gamma \supseteq \Sigma$. It follows that the length of most standard encodings of $q$-state one-tape Turing machines is $\mathrm{O}(q^2)$. To make it simple, we assume that each code of a $q$-state one-tape Turing machines has length $\Theta(q^2)$ and when we will talk about the complexity of the problems $\mathrm{HALT}_{(C,D)}$, we will always use $q$ as a measure for the length of the input. We discuss the encoding in detail in Section 4.1.

The main result of this paper, proven in Section 4, is the following.

**Theorem 1.1.** *For integers $C \geq 2$ and $D \geq 1$, all of the following holds.*

*i) The problems* $\mathrm{HALT}_{(C,D)}$ *and* $\mathrm{DHALT}_{(C,D)}$ *are* co-NP-complete.

*ii) The problems* $\mathrm{HALT}_{(C,D)}$ *and* $\mathrm{DHALT}_{(C,D)}$ ***can not** be solved in time* $\mathrm{o}(q^{(C-1)/4})$ *by non-deterministic multi-tape Turing machines.*

*iii) The complements of the problems* $\mathrm{HALT}_{(C,D)}$ *and* $\mathrm{DHALT}_{(C,D)}$ ***can** be solved in time* $\mathrm{O}(q^{C+2})$ *by a non-deterministic multi-tape Turing machine.*

*iv) The complement of the problem* $\mathrm{HALT}_{(C,D)}$ ***can not** be solved in time* $\mathrm{o}(q^{(C-1)/2})$ *by a non-deterministic multi-tape Turing machine.*

*v) The complement of the problem* $\mathrm{DHALT}_{(C,D)}$ ***can not** be solved in time* $\mathrm{o}(q^{(C-1)/4})$ *by a non-deterministic multi-tape Turing machine.*

To put the theorem in short, for $\delta = 0.2$, the problems $\mathrm{HALT}_{(C,D)}$ and $\mathrm{DHALT}_{(C,D)}$ are co-NP-complete with a non-deterministic and a co-non-deterministic time complexity lower bound $\Omega(q^{\delta C})$ and a co-non-deterministic time complexity upper bound $\mathrm{O}(q^{C+2})$. This result can be compared to the one of Adachi, Iwata and Kasai [1] from 1984, where they proved good deterministic lower bounds for some problems that are complete in P.

To prove the lower bounds, we make reductions from hard problems for which hardness is proven by diagonalization. The diagonalization in Proposition 4.7 (non-deterministic lower bound) is straightforward and the diagonalization in Proposition 4.5 (co-non-deterministic lower bound) is implicit in the non-deterministic time hierarchy [9, 12]. The reductions are not so trivial and we describe the main idea in the following paragraph.

Suppose a one-tape non-deterministic Turing machine $M$ solves a computationally hard problem $L$. Then for any input $w$, we can decide whether $w \in L$ by **first** constructing a one-tape Turing machine $M_w$ that runs in time $Cn + D$ iff $M$ rejects $w$ **and then** solving $\mathrm{HALT}_{(C,D)}$ for $M_w$. The machine $M_w$ is supposed to simulate $M$ on $w$, but only for long enough inputs because we do not want to violate the running time $Cn+D$. Hence, $M_w$ will on the inputs of length $n$ first only measure the input length using at most $(C-1)n+1$ steps to assure that $n$ is large enough and then it will simulate $M$ on $w$ using at most $n$ steps. It turns out that the main challenge is to make $M_w$ effectively measure the length of the input with not too many steps and also not too many states. The latter is important because we do not want the input $M_w$ for $\mathrm{HALT}_{(C,D)}$ to be blown up too much so that we can prove better lower bounds. We leave the details for Section 4. Let us mention also Section 5.1, where we argue that our method of measuring the length of the input is optimal which implies that using our methods, we can not get much better lower bounds.

To prove the upper bounds in Theorem 1.1, we use Theorem 3.1, which we refer to as *the compactness theorem* and which is interesting in its own right. We use crossing sequences to state it in its full power, but a simple corollary of it is the following.

[2]In other words, no one-tape Turing machine can run in superlinear time and also in time $\mathrm{o}(n \log n)$.

**Corollary 1.2.** *For positive integers $C$ and $D$, a one-tape $q$-state Turing machine runs in time $Cn + D$ iff for each input of length $n \leq \mathrm{O}(q^{2C})$ it makes at most $Cn + D$ steps.*

To rephrase the corollary, we can solve $\mathrm{HALT}_{(C,D)}$ for an input Turing machine $M$ by just verifying the running time of $M$ on the inputs of length at most $\mathrm{O}(q^{2C})$. Behind the big O is hidden a polynomial in $C$ and $D$ (see Lemma 3.2). The result is interesting not only because it allows us to algorithmically solve the problem $\mathrm{HALT}_{(C,D)}$[3], but also because it gives a new insight into one-tape linear time computations. There was quite some work done in this area and a summary from 2010 can be found here [10].

A main tool in the analysis of one-tape linear time Turing machines are crossing sequences, which we define in Section 2.2. They were first studied in1960s by Hennie [5] and Trakhtenbrot [11] who proved one of the most well known properties of one-tape linear-time deterministic Turing machines: they recognize only regular languages. Hartmanis [4] extended this result to the one-tape deterministic Turing machines that run in time $\mathrm{o}(n \log n)$ and it was furthermore extended to the non-deterministic Turing machines by Kobayashi [7], Pighizzini [8] and Tadaki, Yamakami and Lin [10]. Hence, it is currently well known that one-tape linear time non-deterministic Turing machines accept only regular languages.

All of the above results were proven via crossing sequences, which are also the main tool in proving the compactness theorem. The methods we use to prove it are not new, rather they consist of standard cutting and pasting of the portions of the tape between cell boundaries where identical crossing sequences are generated. We show that a Turing machine that runs in time $Cn + D$ must produce some identical crossing sequences on each computation, if the input is long enough. Thus, when considering some fixed computation, we can partition the input on some parts where identical crossing sequences are generated, and analyze each part independently. We prove that it is enough to consider small parts of the input.

# 2 Preliminaries

Let $\mathbb{N}$ be the set of all non-negative integers. All logarithms with no base written have base 2. We use $\epsilon$ for the empty word and $|w|$ for the length of a word $w$. For words $w_1$ and $w_2$, let $w_1 w_2$ denote their concatenation.

We will use multi-tape Turing machines to solve decision problems. If not stated otherwise, lower and upper complexity bounds will be for this model of computation. We will not describe the model (any standard one will do). We will use the notation DTM and NTM for deterministic and non-deterministic Turing machines, respectively.

## 2.1 Basic definitions

*A one-tape NTM* is an 8-tuple $M = (Q, \Sigma, \Gamma, \llcorner\!\lrcorner, \delta, q_0, q_{\text{ACC}}, q_{\text{REJ}})$, where $Q$ is a set of states, $\Sigma \supseteq \{0, 1\}$ an input alphabet, $\Gamma \supseteq \Sigma$ a tape alphabet, $\llcorner\!\lrcorner \in \Gamma \backslash \Sigma$ a blank symbol, $\delta : Q \backslash \{q_{\text{ACC}}, q_{\text{REJ}}\} \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{-1, 1\}) \backslash \{\emptyset\}$ a transition function and $q_0, q_{\text{ACC}}, q_{\text{REJ}} \in Q$ pairwise distinct starting, accepting and rejecting states. Here $\mathcal{P}$ denotes the power set.

As can be seen from the definition, the head of $M$ must move on each step and at the end of each finite computation the head of $M$ is in a halting state ($q_{\text{ACC}}$ or $q_{\text{REJ}}$). We can assume this without the loss of generality, but especially the property that the head of $M$ must move on each step will be convenient when discussing crossing sequences because it will hold that the sum of the lengths of all crossing sequences equals the number of steps. If we allowed the head to stay in place, we would have to change the definition of the length of a computation on a part (just before Theorem 3.1). However, what we really need is that $\Sigma$ contains at least two symbols and we denote them by 0 and 1, because a unary input alphabet would not suffice to prove our lower bounds.

[3] However, this is not enough to prove the upper bound in Theorem 1.1, for which we need the more powerful compactness theorem.

For one-tape NTMs $M_1$ and $M_2$, the *composition* of $M_1$ and $M_2$ is the NTM that starts computing as $M_1$, but has the starting state of $M_2$ instead of $M_1$'s accepting state. When the starting state of $M_2$ is reached, it computes as $M_2$. If $M_1$ rejects, it rejects.

*A one-tape DTM* is a one-tape NTM where each possible configuration has at most one successive configuration.

The number of steps that a Turing machine $M$ makes on some computation $\zeta$ will be called *the length of* $\zeta$ and denoted by $|\zeta|$. For a function $T : \mathbb{N} \to \mathbb{N}$, we say that a Turing machine $M$ *runs in time* $T(n)$ if $M$ makes **at most** $T(n)$ steps on all computations on all inputs of length $n$, for all $n \in \mathbb{N}$.

## 2.2 Crossing Sequences

For a one-tape Turing machine $M$, we can number the cells of its tape with integers so that the cell 0 is the one where $M$ starts its computation. Using this numbering we can number the boundaries between cells as shown in Figure 1. Whenever we say that an input is written on the tape, we mean that its $i$th symbol is in the cell $(i-1)$ and all other cells contain the blank symbol $\llcorner\!\lrcorner$.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CELLS: | $\cdots$ | $-3$ | $-2$ | $-1$ | $0$ | $1$ | $2$ | $3$ | $4$ | $5$ | $\cdots$ |
| BOUNDARIES: | $\cdots$ $-3$ | $-2$ | $-1$ | $0$ | $1$ | $2$ | $3$ | $4$ | $5$ | $\cdots$ | |

Figure 1: Numbering of tape cells and boundaries of a one-tape Turing machine. The shaded part is a potential input of length 4.

Intuitively, a *crossing sequence generated by a one-tape NTM $M$ after $t$ steps of a computation $\zeta$ on an input $w$ on a boundary $i$* is a sequence of states of $M$, in which $M$ crosses the $i$th boundary of its tape, when considering the first $t$ steps of the computation $\zeta$ on the input $w$. A more formal definition is given in the next paragraph.

Suppose that a one-tape NTM $M$ on the first $t \in \mathbb{N} \cup \{\infty\}$ steps of a computation $\zeta$ on an input $w$ crosses a boundary $i$ of its tape at steps $t_1, t_2 \dots$ (this sequence can be finite or infinite). If $M$ was in state $q_j$ after the step $t_j$ for all $j$, then we say that $M$ produces the *crossing sequence* $\mathcal{C}_i^t(M, \zeta, w) = q_1, q_2 \dots$ and we denote its length by $|\mathcal{C}_i^t(M, \zeta, w)| \in \mathbb{N} \cup \{\infty\}$. Note that this sequence contains all information that the machine carries across the $i$th boundary of the tape in the first $t$ steps of the computation $\zeta$. If we denote $\mathcal{C}_i(M, \zeta, w) = \mathcal{C}_i^{|\zeta|}(M, \zeta, w)$, the following trivial identity holds:

$$|\zeta| = \sum_{i=-\infty}^{\infty} |\mathcal{C}_i(M, \zeta, w)|.$$

# 3 The Compactness Theorem

In this section, we present the first result of this paper, the compactness theorem. Simply put, if we want to verify that an NTM $M$ runs in time $Cn + D$, we only need to verify the number of steps that $M$ makes on inputs of some bounded length. The same result can be found in [3], but the bound in the present paper is much better.

Before we formally state the theorem, let us introduce some notation. For a one-tape NTM $M$, define

$$\mathcal{S}_n(M) = \{\mathcal{C}_i^t(M, \zeta, w);\ |w| = n,\ 1 \le i \le n,\ \zeta \text{ computation on input } w,\ t \le |\zeta|\},$$

so $\mathcal{S}_n(M)$ is the set of all possible beginnings of the crossing sequences that $M$ produces on the inputs of length $n$ on the boundaries $1, 2 \dots n$.

The definition of $t_M(w, \mathcal{C})$ is more involved. Intuitively, $t_M(w, \mathcal{C})$ is the maximum number of steps that a one-tape NTM $M$ makes on a **part** $w$ of an imaginary input, if we only consider such computations on which $M$ produces the crossing sequence $\mathcal{C}$ on both left and right boundaries of $w$. To define it more formally, we will describe a valid *computation of $M$ on a part $w$ with ending crossing sequence $\mathcal{C} = (q_1, q_2 \dots q_l)$*[4]. We will use the term *standard case* to refer to the definition of a computation of an NTM on a given input (not on a part). Assume $|w| = n \geq 1$ and let $M = (Q, \Sigma, \Gamma, \llcorner\!\lrcorner, \delta, q_0, q_{\text{ACC}}, q_{\text{REJ}})$.

- A *valid configuration* is a 5-tuple $(\mathcal{C}_1, \tilde{w}, i, \tilde{q}, \mathcal{C}_2)$, where $\mathcal{C}_1$ is the *left crossing sequence*, $\tilde{w}$ is some word from $\Gamma^n$, $0 \leq i \leq n-1$ is the position of the head, $\tilde{q} \in Q$ is the current state of $M$ and $\mathcal{C}_2$ is the *right crossing sequence*. Intuitively, $\mathcal{C}_1$ and $\mathcal{C}_2$ are the endings of $\mathcal{C}$ that still need to be matched.

- The *starting configuration* is $((q_2, q_3 \dots q_l), w, 0, q_1, (q_1, q_2 \dots q_l))$. As in the standard case, we imagine the input being written on the tape of $M$ with the first bit in the cell 0 (where also the head of $M$ is). The head will never leave the portion of the tape where the input is written. Note that $q_1$ is missing in the left crossing sequence because we pretend that the head just moved from the cell -1 to the cell 0.

- Valid configurations $A = (\mathcal{C}_{1A}, w_A, i, q_A, \mathcal{C}_{2A})$ and $B = (\mathcal{C}_{1B}, w_B, j, q_B, \mathcal{C}_{2B})$ are *successive*, if one of the following holds:
  - the transition function of $M$ allows $(w_A, i, q_A)$ to change into $(w_B, j, q_B)$ as in the standard case, $\mathcal{C}_{1A} = \mathcal{C}_{1B}$ and $\mathcal{C}_{2A} = \mathcal{C}_{2B}$,
  - $i = j = 0$, $\mathcal{C}_{1A}$ is of the form $(\tilde{q}, q_B, \mathcal{C}_{1B})$, $w_A = a\tilde{w}$, $w_B = b\tilde{w}$, $(\tilde{q}, b, -1) \in \delta(q_A, a)$ and $\mathcal{C}_{2A} = \mathcal{C}_{2B}$,
  - $i = j = n-1$, $\mathcal{C}_{2A}$ is of the form $(\tilde{q}, q_B, \mathcal{C}_{2B})$, $w_A = \tilde{w}a$, $w_B = \tilde{w}b$ and $(\tilde{q}, b, 1) \in \delta(q_A, a)$ and $\mathcal{C}_{1A} = \mathcal{C}_{1B}$.

- There is a special *ending configuration* that can be reached from configurations of the form
  - $((q_l), a\tilde{w}, 0, \tilde{q}, ())$, if $(q_l, b, -1) \in \delta(\tilde{q}, a)$ for some $b \in \Gamma$ or
  - $((), \tilde{w}a, n-1, \tilde{q}, (q_l))$, if $(q_l, b, 1) \in \delta(\tilde{q}, a)$ for some $b \in \Gamma$.

- A *valid computation of $M$ on the part $w$ with ending crossing sequence $\mathcal{C}$* is any sequence of successive configurations that begins with the starting configuration and ends with an ending configuration.

Similar to the standard case, we can define $\mathcal{C}_i(M, \zeta, w, \mathcal{C})$ to be the crossing sequence generated by $M$ on the computation $\zeta$ on the part $w \in \Sigma^n$ with the ending crossing sequence $\mathcal{C}$ on the boundary $i$ $(1 \leq i \leq n-1)$ . We define

$$|\zeta| = |\mathcal{C}| + \sum_{i=1}^{n-1} |\mathcal{C}_i(M, \zeta, w, \mathcal{C})|$$

as the *length of the computation $\zeta$*. Figure 2 justifies this definition.

We define $t_M(w, \mathcal{C}) \in \mathbb{N} \bigcup \{-1\}$ as the length of the longest computation of $M$ on the part $w$ with the ending crossing sequence $\mathcal{C}$. If there is no valid computation of $M$ on the part $w$ with the ending crossing sequence $\mathcal{C}$ or $|\mathcal{C}| = \infty$, then we define $t_M(w, \mathcal{C}) = -1$.

**Theorem 3.1** (The compactness theorem)**.** *Let $M$ be a one-tape NTM with $q$ states and let $C, D \in \mathbb{N}$. Denote $\ell = D + 8q^C$, $r = D + 12q^C$ and $\mathcal{S} = \bigcup_{n=1}^{\ell} \mathcal{S}_n(M)$. It holds:*

*$M$ runs in time $Cn + D$* ***if and only if***

*a) for each input $w$ of length at most $\ell$ and for each computation $\zeta$ of $M$ on $w$, it holds $|\zeta| \leq C|w| + D$* ***and***

[4]The author is not aware that this description would already be present in literature, although it has most likely been considered. A slightly less general description is given in [8].

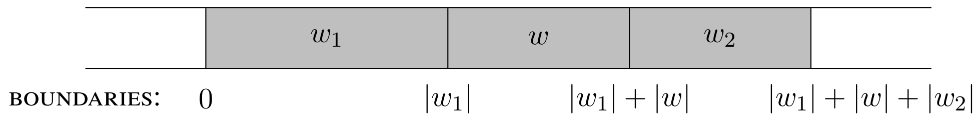


Figure 2: Suppose an input $w_1 w w_2$ is given to $M$, $|w_1|, |w| \geq 1$ and let a computation $\zeta$ produce the same crossing sequence $\mathcal{C}$ on boundaries $|w_1|$ and $|w_1| + |w|$. If $\zeta_1$ is the corresponding computation of $M$ on part $w$, then $M$ on the computation $\zeta$ spends exactly $|\zeta_1|$ steps on the part $w$. What is more, if the input $w_1 w_2$ is given to $M$ (we cut out $w$) and we look at the computation $\zeta_2$ which corresponds to $\zeta$ thus forming a crossing sequence $\mathcal{C}$ on the boundary $|w_1|$, then $|\zeta_2| = |\zeta| - |\zeta_1|$. Such considerations will be very useful in the proof of the compactness theorem.

b) *for each $\mathcal{C} \in \mathcal{S}$ and for each part $w$ of length at most $r$, for which $t_M(w, \mathcal{C}) \geq 0$, it holds $t_M(w, \mathcal{C}) \leq C|w|$.*

Before going to the proof, let us argue that the theorem is in fact intuitive. If a Turing machine $M$ runs in time $Cn + D$, then a) tells us that $M$ must run in that time for small inputs and b) tells us that on small parts $w$ that can be "inserted" into some input from a), $M$ must make at most $C|w|$ steps. For the opposite direction, one can think about constructing each input for $M$ from several parts from b) inserted into some input from a) on appropriate boundaries, which results in running time $Cn + D$.

The following lemma already proves one direction of the compactness theorem.

**Lemma 3.2.** *Let everything be as in Theorem 3.1. If b) does not hold, then there exists some input $z$ of length at most $\ell + (Cr + D)r$ such that $M$ makes more than $C|z| + D$ steps on $z$ on some computation.*

**Proof.** If b) does not hold, then there exists some finite crossing sequence $\mathcal{C} \in \mathcal{S}$, a part $w$ of length at most $r$ and a valid computation $\zeta_1$ of $M$ on the part $w$ with the ending crossing sequence $\mathcal{C}$, such that $|\zeta_1| \geq C|w| + 1$. From the definition of $\mathcal{S}$ we know that there exist words $w_1$ and $w_2$ such that $|w_1| \geq 1$ and $|w_1| + |w_2| \leq \ell$, $t \in \mathbb{N}$ and a computation $\zeta_2$, such that $\mathcal{C}$ is generated by $M$ on the input $w_1 w_2$ on the computation $\zeta_2$ on the boundary $|w_1|$ after $t$ steps. As in Figure 2, we can now insert $w$ between $w_1$ and $w_2$, in fact we can insert as many copies of $w$ between $w_1$ and $w_2$ as we want, because the crossing sequence $\mathcal{C}$ will always be formed between them.

Let us look at the input $z = w_1 w^{Cr+D} w_2$ for $M$. Let $\zeta$ be a computation of $M$ on $z$ that on the part $w_1$ (and left of it) and on the part $w_2$ (and right of it) acts like the first $t$ steps of $\zeta_2$ and on the parts $w$ it acts like $\zeta_1$. Note that after $\zeta$ spends $t$ steps on the parts $w_1$ and $w_2$, crossing sequence $\mathcal{C}$ is generated on the boundaries $|w_1|, (|w_1| + |w|) \ldots (|w_1| + (Cr + D)|w|)$ and by that time $M$ makes at least $t + (Cr + D)(C|w| + 1)$ steps. Using $t \geq 1$ and $r \geq \ell \geq |w_1| + |w_2|$, we see that $M$ makes at least

$$C(Cr + D)|w| + C(|w_1| + |w_2|) + D + 1 = C|z| + D + 1$$

steps on the computation $\zeta$ on the input $z$. Because $|w| \leq r$ and $|w_1| + |w_2| \leq \ell$, we have $|z| \leq \ell + (Cr + D)r$ and the lemma is proven. $\square$

Next, we prove the main lemma for the proof of the other direction of the compactness theorem.

**Lemma 3.3.** *Let $C, D$ be non-negative integers, $M$ a one-tape $q$-state NTM and $w$ an input for $M$ of length $n$. Assume that $M$ makes at least $t \leq Cn + D$ steps on the input $w$ on a computation $\zeta$ and suppose that each crossing sequence produced by $M$ on $\zeta$ after $t$ steps on the boundaries $1, 2 \ldots n$ appears at most $k$ times. Then $n \leq D + 4kq^C$.*

**Proof.** We know that $Cn + D \geq t \geq \sum_{i=1}^{n} |\mathcal{C}_i^t(M, \zeta, w)|$, thus

$$\begin{aligned} n &\leq D + (C+1)n - \sum_{i=1}^{n} |\mathcal{C}_i^t(M, \zeta, w)| \\ &= D + \sum_{i=1}^{n} (C + 1 - |\mathcal{C}_i^t(M, \zeta, w)|) \\ &\leq D + \sum_{j=0}^{C+1} \sum_{\substack{i=1 \\ |\mathcal{C}_i^t(M,\zeta,w)|=j}}^{n} (C + 1 - j) \\ &\leq D + \sum_{j=0}^{C+1} kq^j (C + 1 - j) \\ &\leq D + 4kq^C, \end{aligned}$$

where the last inequality follows by Lemma A.1 proven in the appendix. □

Before going into the proof of the compactness theorem, let us define $w(i, j)$ as the subword of a word $w$, containing characters from $i$th to $j$th, including $i$th and excluding $j$th (we start counting with 0). Alternatively, if $w$ is written on a tape of a Turing machine, $w(i, j)$ is the word between the $i$th and $j$th boundary.

**Proof of the compactness theorem (Theorem 3.1).** If $M$ runs in time $Cn+D$, then a) obviously holds and b) holds by Lemma 3.2. Now suppose that a) and b) hold. We will make a proof by contradiction, so suppose that $M$ does not run in time $Cn + D$. Let $w$ be a shortest input for $M$ such that there exists a computation of $M$ on $w$ of length more than $C|w| + D$. Denote this computation by $\zeta$ and let $n = |w|$, $t = Cn + D$.

Before we continue, let us give an outline of what follows in one paragraph. Our first goal is to find closest such boundaries $j_1$ and $j_2$ such that $M$ produces the same crossing sequence $\mathcal{C} = \mathcal{C}_{j_1}^{t+1}(M, \zeta, w) = \mathcal{C}_{j_2}^{t+1}(M, \zeta, w)$ on them after the $(t)$th and $(t+1)$st step of the computation $\zeta$ (see Figure 3). Then using the fact that $w$ is a shortest input for $M$ such that there exists a computation of $M$ on $w$ of length more than $C|w| + D$, we argue that $t_M(w(j_1, j_2), \mathcal{C}) > C|w(j_1, j_2)|$. Now the most important part of $w$ is between the boundaries $j_1$ and $j_2$, so we want to cut out the superfluous parts left of $j_1$ and right of $j_2$ (see Figure 4). After the cutting out we get an input $w_1 w(j_1, j_2) w_2$ on which $M$ on the computation corresponding to $\zeta$ on the time-step corresponding to $t$ generates the crossing sequence $\mathcal{C}$ on boundaries $|w_1|$ and $(|w_1| + j_2 - j_1)$ and all other crossing sequences are generated at most 3 times on boundaries $1, 2 \ldots (|w_1|+|w(j_1, j_2)|+|w_2|)$: once left from $w(j_1, j_2)$, once on the boundaries of $w(j_1, j_2)$ and once right from $w(j_1, j_2)$. Using Lemma 3.3 twice, we see that $|w_1 w_2| \leq \ell$ and $|w_1 w(j_1, j_2) w_2| \leq r$, which implies $\mathcal{C} \in \mathcal{S}$ and $|w(j_1, j_2)| \leq r$, which contradicts b) because $t_M(w(j_1, j_2), \mathcal{C}) > C|w(j_1, j_2)|$.

As we stated in the above outline, our first goal is to find boundaries $j_1$ and $j_2$. From a) it follows that $n > \ell = D + 4 \cdot 2q^C$, so by Lemma 3.3 there exist at least three identical crossing sequences produced by $M$ on the input $w$ on the computation $\zeta$ after $t$ steps on the boundaries $1, 2 \ldots n$. Let these crossing sequences be generated on boundaries $i_1 < i_2 < i_3$ (see Figure 3). Because $\mathcal{C}_{i_1}^t(M, \zeta, w)$ and $\mathcal{C}_{i_3}^t(M, \zeta, w)$ are of equal length, the head of $M$ is, before the $(t + 1)$st step of the computation $\zeta$, left of the boundary $i_1$ or right of the boundary $i_3$. Without the loss of generality we can assume that the head is right from $i_3$ (if not, we can rename $i_1 = i_2$ and $i_2 = i_3$ and continue with the proof). Thus, no crossing sequence on the boundaries $i_1, (i_1 + 1) \ldots i_2$ changes in the $(t + 1)$st step of the computation $\zeta$. Let $i_1 \leq j_1 < j_2 \leq i_2$ be closest boundaries such that $\mathcal{C}_{j_1}^{t+1}(M, \zeta, w) = \mathcal{C}_{j_2}^{t+1}(M, \zeta, w)$. Then the crossing sequences $\mathcal{C}_j^t(M, \zeta, w)$, for $j_1 \leq j < j_2$, are pairwise distinct and do not change in the $(t+1)$st step of the computation $\zeta$.

Let $\zeta_1$ be the computation on part $w(j_1, j_2)$ with ending crossing sequence $\mathcal{C}$ that corresponds to $\zeta$ and let $\zeta_2$ be a computation on input $w(0, j_1)w(j_2, n)$ that in first $(t + 1 - |\zeta_1|)$ steps corresponds to the

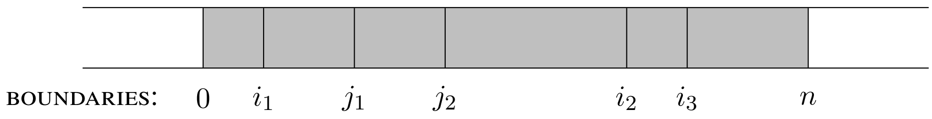


Figure 3: Finding boundaries $j_1$ and $j_2$. The shaded area represents the input $w$. First, we find boundaries $i_1$, $i_2$ and $i_3$ on which the same crossing sequence is generated after $t$ steps of the computation $\zeta$. Because the crossing sequences generated on the boundaries $i_1$, $i_2$ and $i_3$ are of the same length, after $t$ steps of the computation $\zeta$ the head of $M$ is on some cell left of the boundary $i_1$ or on some cell right of the boundary $i_3$, hence either the crossing sequences generated on the boundaries between (and including) $i_1$ and $i_2$ remain intact in the $(t+1)$st step of the computation $\zeta$, either the crossing sequences generated on the boundaries between (and including) $i_2$ and $i_3$ remain intact in the $(t+1)$st step of the computation $\zeta$. Without the loss of generality we may assume that the former holds. We choose $i_1 \leq j_1 < j_2 \leq i_2$ to be closest boundaries such that $\mathcal{C}_{j_1}^{t+1}(M, \zeta, w) = \mathcal{C}_{j_2}^{t+1}(M, \zeta, w)$.

first $(t+1)$ steps of $\zeta$. Because the input $w(0, j_1)w(j_2, n)$ is strictly shorter than $n$, $M$ makes at most $C(|w(0, j_1)| + |w(j_2, n)|) + D$ steps on any computation on this input, thus

$$\begin{aligned} t + 1 - |\zeta_1| &\leq |\zeta_2| \\ &\leq C(|w(0, j_1)| + |w(j_2, n)|) + D. \end{aligned}$$

From $t = Cn + D$ and $n = |w(0, j_1)| + |w(j_2, n)| + j_2 - j_1$ it follows that

$$\begin{aligned} |\zeta_1| &\geq t + 1 - C(|w(0, j_1)| + |w(j_2, n)|) - D \\ &= C(j_2 - j_1) + 1, \end{aligned}$$

thus $t_M(w(j_1, j_2), \mathcal{C}) > C|w(j_1, j_2)|$.

Next, we will cut out some pieces of $w$ to eliminate as many redundant parts as possible (if they exist), while leaving the part of $w$ between the boundaries $j_1$ and $j_2$ intact. Redundant parts are those where identical crossing sequences are generated on the computation $\zeta$ after $t$ steps. We will cut out parts recursively and the result will not necessarily be unique (see Figure 4).

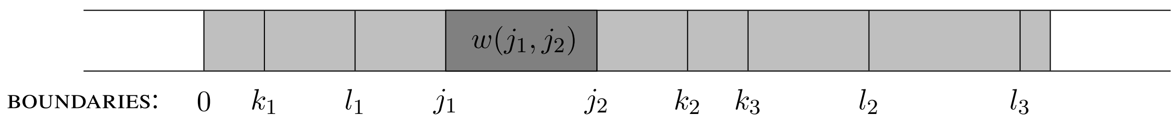


Figure 4: Cutting out parts of $w$ left and right of $w(j_1, j_2)$. If $M$ on the input $w$ (shaded) on the computation $\zeta$ after $t$ steps produces the same crossing sequence on boundaries $k_1$ and $l_1$, then we can cut out $w(k_1, l_1)$. The same holds also for pairs $(k_2, l_2)$ and $(k_3, l_3)$. What is more, we can cut out both $w(k_1, l_1)$ and $w(k_2, l_2)$ if $\mathcal{C}_{k_1}^t(M, \zeta, w) = \mathcal{C}_{l_1}^t(M, \zeta, w)$ and $\mathcal{C}_{k_2}^t(M, \zeta, w) = \mathcal{C}_{l_2}^t(M, \zeta, w)$. However, we can not cut out both $w(k_2, l_2)$ and $w(k_3, l_3)$ because they overlap, and we may get a different outcome if we cut out $w(k_2, l_2)$ or $w(k_3, l_3)$.

Suppose that $\mathcal{C}_k^t(M, \zeta, w) = \mathcal{C}_l^t(M, \zeta, w)$ for $1 \leq k < l \leq j_1$ or $j_2 \leq k < l \leq n$. Cut out the part of $w$ between the $k$th and $l$th boundary. Let $w'$ be the new input. Let the boundaries $j_1'$ and $j_2'$ for the input $w'$ correspond to the boundaries $j_1$ and $j_2$ for the input $w$. Let $\zeta'$ be a computation on $w'$ that corresponds to $\zeta$ (at least for the first $t$ steps of $\zeta$) and let $t'$ be the step in the computation $\zeta'$ that corresponds to the step $t$ of the computation $\zeta$. Now recursively find new $k$ and $l$. The recursion ends, when there are no $k, l$ to be found.

From the recursion it is clear that at the end we will get an input for $M$ of the form $w_0 = w_1 w(j_1, j_2) w_2$, where $|w_1| \geq 1$. Let $\zeta_0$ be a computation that corresponds to $\zeta$ after the cutting out (at least for the first $t$ steps of $\zeta$) and let $t_0$ be the step in $\zeta_0$ that corresponds to $t$. If we denote $n_0 = |w_0|$, then it holds $t_0 \leq Cn_0 + D$ because either there was nothing to remove and $w_0 = w$, $t_0 = t$ or $w_0$ is a shorter input than $w$ and $t_0 \leq Cn_0 + D$ must hold by the definition of $w$. From the construction it is clear that $M$ on input $w_0$ on computation $\zeta_0$ after $t_0$ steps generates the crossing sequence $\mathcal{C}$ on the boundaries $|w_1|$ and $(|w_1| + j_2 - j_1)$. What is more, the crossing sequences on the boundaries $1, 2 \dots |w_1|$ are pairwise distinct. The same is true for the crossing sequences on the boundaries $(|w_1| + 1), (|w_1| + 2) \dots (|w_1| + j_2 - j_1)$ and the crossing sequences on the boundaries $(|w_1| + j_2 - j_1), (|w_1| + j_2 - j_1 + 1) \dots n_0$. Because $t_0 \leq Cn_0 + D$, we get that $n_0 \leq D + 4 \cdot 3q^C = r$ by Lemma 3.3, hence $|w(j_1, j_2)| \leq r$.

Denote $\tilde{w} = w_1 w_2$ and $\tilde{n} = |w_1| + |w_2|$. Let the computation $\tilde{\zeta}$ on $\tilde{w}$ be a computation that corresponds to $\zeta_0$ (at least for the first $t_0$ steps of $\zeta_0$) and let $\tilde{t}$ be the time step of $\tilde{\zeta}$ that corresponds to the time step $t_0$ of $\zeta_0$. Because $\tilde{n} < n_0 \leq n$ and because $w$ is a shortest input for $M$ that violates the $Cn + D$ bound, $M$ makes at most $C\tilde{n} + D$ steps on any computation on the input $\tilde{w}$, thus also on the computation $\tilde{\zeta}$. Note that no three crossing sequences from $\{\mathcal{C}_i^{\tilde{t}}(M, \tilde{\zeta}, \tilde{w}); 1 \leq i \leq \tilde{n}\}$ are identical, thus by Lemma 3.3, $\tilde{n} \leq D + 4 \cdot 2q^C = \ell$. Because $\mathcal{C}_{|w_1|}^{\tilde{t}}(M, \tilde{\zeta}, \tilde{w}) = \mathcal{C}$, it follows that $\mathcal{C} \in \mathcal{S}$, which together with $|w(j_1, j_2)| \leq r$ and $t_M(w(j_1, j_2), \mathcal{C}) > C|w(j_1, j_2)|$ contradicts b). $\square$

If we combine the compactness theorem with Lemma 3.2, we get **Corollary 1.2**:

*For positive integers $C$ and $D$, a one-tape $q$-state Turing machine runs in time $Cn + D$ iff for each input of length $n \leq \mathrm{O}(q^{2C})$ it makes at most $Cn + D$ steps.*

Behind the big O is hidden a polynomial in $C$ and $D$ (see Lemma 3.2).

# 4 The Complexity of Solving $\mathrm{HALT}_{(C,D)}$ and $\mathrm{DHALT}_{(C,D)}$

In this section, we prove the main result of this paper, Theorem 1.1, which is about the complexity of the problems

$$\mathrm{HALT}_{(C,D)} = \{M | M \text{ is a one tape NTM that runs in time } Cn + D\} \quad \text{and}$$
$$\mathrm{DHALT}_{(C,D)} = \{M | M \text{ is a one tape DTM that runs in time } Cn + D\},$$

for $C, D \in \mathbb{N}$. We use an overline to refer to the complements of the problems, like

$$\overline{\mathrm{HALT}_{(C,D)}} = \{M | M \text{ is a one tape NTM that does not run in time } Cn + D\}.$$

## 4.1 The Encoding of One-Tape Turing Machines

As already stated in the introduction, we assume a fixed input alphabet $\Sigma \supseteq \{0, 1\}$ and a fixed tape alphabet $\Gamma$, hence we will actually be analyzing the problems $\mathrm{HALT}_{(C,D)}(\Sigma, \Gamma)$, which will enable us to have the codes of $q$-state one-tape Turing machines of length $\Theta(q^2)$. Because $q$ will describe the length of the code up to a constant factor, we will express the complexity of algorithms with a $q$-state one-tape NTM (or DTM) as input in terms of $q$ instead of $n = \Theta(q^2)$.

Let us state the properties that should be satisfied by the encoding of one-tape Turing machines.

- Given a code of a $q$-state one-tape NTM $M$, a multi-tape NTM can simulate each step of $M$ in $\mathrm{O}(q^2)$ time. Similarly, given a code of a $q$-state one-tape DTM $M$, a multi-tape DTM can simulate each step of $M$ in $\mathrm{O}(q^2)$ time.
- A code of a composition of two one-tape Turing machines can be computed in linear time by a multi-tape DTM.
- The code of a $q$-state one-tape Turing machine has to be of length $\Theta(q^2)$. This is a technical requirement that makes arguments easier and it gives a concrete relation between the number of states of a one-tape Turing machine and the length of its code. We can achieve this because we assumed a fixed input and tape alphabet.

Now we describe an example of such an encoding. It is clear that we can easily convert any standard code of a one-tape Turing machine to ours and vice versa.

A *code* of a $q$-state Turing machine $M$ is a code of a $q \times q$ matrix $A$, where $A[i,j]$ is a (possibly empty) list of all the triples $(a,b,d) \in \Gamma \times \Gamma \times \{-1,1\}$ such that $M$ can come in one step from the state $q_i$ to the state $q_j$ replacing the symbol $a$ below the head by the symbol $b$ and moving in the direction $d$. In other words, $A[i,j]$ is a list of all the triples $(a,b,d) \in \Gamma \times \Gamma \times \{-1,1\}$ such that $(q_j,b,d) \in \delta(q_i,a)$. We assume that the indices $i$ and $j$ range from 0 to $(q-1)$ and that the index 0 corresponds to the starting state, the index $(q-2)$ corresponds to the accepting state and the index $(q-1)$ corresponds to the rejecting state.

Because the tape alphabet $\Gamma$ is fixed, a code of a $q$-state one-tape Turing machine is of length $\Theta(q^2)$ and it is clear that we can simulate each step of a given one-tape $q$-state Turing machine in $\mathrm{O}(q^2)$ time by a multi-tape Turing machine. Furthermore, the composition of two one-tape Turing machines can be computed in linear time by a multi-tape DTM as can be seen in Figure 5.

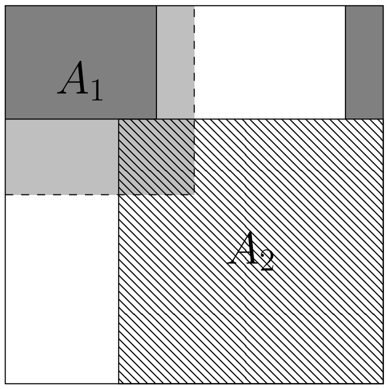


Figure 5: The code of a composition of two Turing machines. Suppose that we want to compute the code of a composition of Turing machines $M_1$ and $M_2$. Let $A_1$ and $A_2$ be the corresponding matrices that were used to encode $M_1$ and $M_2$. Then we can erase the last two lines of the matrix $A_1$ (they correspond to the halting states of $M_1$) and adjust $A_2$ "below" $A_1$ as shown on the figure. Note that the column of the accepting state of $M_1$ coincides with the column of the starting state of $M_2$. The last column of $A_1$ that corresponds to the rejecting state of $M_1$ is flushed to the right. To compute the code of the composition of two Turing machines, we have to compute the code of this matrix, which can be done in linear time given the codes of $A_1$ and $A_2$.

## 4.2 The Upper Bound

Let us define the problem

$$\mathrm{HALT}_{(\cdot,\cdot)} = \{(C,D,M) | C,D \in \mathbb{N} \text{ and } M \text{ is a one tape NTM that runs in time } Cn+D\}.$$

Hence, the problem $\mathrm{HALT}_{(\cdot,\cdot)}$ is the same as the problem $\mathrm{HALT}_{(C,D)}$, only that $C$ and $D$ are parts of the input.

**Proposition 4.1.** *There exists a multi-tape NTM that solves* $\overline{\mathrm{HALT}_{(\cdot,\cdot)}}$ *in time* $\mathrm{O}(p(C,D)q^{C+2})$ *for some quadratic polynomial* $p$.

**Proof.** Let us describe a multi-tape NTM $M_{\text{mult}}$ that solves $\overline{\mathrm{HALT}_{(\cdot,\cdot)}}$.

- On the input $(C,D,M)$, where $M$ is a $q$-state one-tape NTM, compute $\ell = D + 8q^C$ and $r = D + 12q^C$.

- Non-deterministically choose an input of length $n \leq \ell$ and simulate a non-deterministically chosen computation of $M$ on it. If $M$ makes more than $Cn + D$ steps, accept.
- Non-deterministically choose an input $w_0 = w_1 w_2 w_3$ such that $|w_1| \geq 1$, $1 \leq |w_2| \leq r$ and $|w_1| + |w_3| \leq \ell$. Initialize $\mathcal{C}_1$ and $\mathcal{C}_2$ to empty crossing sequences and counters $t_0 = C|w_0| + D$, $t_2 = C|w_2|$.
- Simulate a non-deterministically chosen computation $\zeta$ of $M$ on the input $w_0$. After each simulated step $t$ of $M$, do:
  - decrease $t_0$ by one,
  - if the head of $M$ is on some cell $|w_1| \leq i < |w_1| + |w_2|$, decrease $t_2$ by one,
  - update the crossing sequences $\mathcal{C}_1 = \mathcal{C}^t_{|w_1|}(M, \zeta, w_0)$ and $\mathcal{C}_2 = \mathcal{C}^t_{|w_1|+|w_2|}(M, \zeta, w_0)$.
  - If $t_0 < 0$, accept.
  - Non-deterministically decide whether to do the following:
    * If $\mathcal{C}_1 = \mathcal{C}_2$ and $t_2 < 0$, accept. Else, reject.
  - If $M$ halts, reject.

  Note that the counter $t_0$ counts the number of simulated steps, while the counter $t_2$ counts the number of steps done on the part $w_2$.

It is clear that $M_{\text{mult}}$ accepts if either a) or b) from the compactness theorem are violated and it rejects if $M$ runs in time $Cn + D$ and b) from the compactness theorem is not violated. Hence $M_{\text{mult}}$ correctly solves the problem $\overline{\text{HALT}_{(\cdot,\cdot)}}$.

Because the condition $\mathcal{C}_1 = \mathcal{C}_2$ is verified at most once during the algorithm and

$$\begin{aligned} |\mathcal{C}_1|, |\mathcal{C}_2| &\leq C|w_0| + D \\ &\leq C(\ell + r) + D \\ &= \mathrm{O}\left((CD + C + D + 1)q^C\right), \end{aligned}$$

testing whether $\mathcal{C}_1 = \mathcal{C}_2$ contributes $\mathrm{O}((CD + C + D + 1)q^{C+1})$ time to the overall running time. Because $M_{\text{mult}}$ needs $\mathrm{O}(q^2)$ steps to simulate one step of $M$'s computation and it has to simulate at most $C(2\ell + r) + D$ steps, $M_{\text{mult}}$ runs in time $\mathrm{O}((CD + C + D + 1)q^{C+2})$. $\square$

**Corollary 4.2.** *The problems* $\text{HALT}_{(C,D)}$ *and* $\text{DHALT}_{(C,D)}$ *are in* co-NP *and their complements can be solved in time* $\mathrm{O}(q^{C+2})$ *by a non-deterministic multi-tape Turing machine.*

### 4.3 The Lower Bounds

Let us again state the idea that we use to prove the lower bounds in Theorem 1.1. Suppose a one-tape non-deterministic Turing machine $M$ solves a problem $L$. Then for any input $w$, we can decide whether $w \in L$ by first constructing a one-tape Turing machine $M_w$ that runs in time $Cn + D$ iff $M$ rejects $w$ and then solving $\text{HALT}_{(C,D)}$ for $M_w$. If we choose $L$ to be a hard language, then we can argue that we can not solve $\text{HALT}_{(C,D)}$ fast. The next lemma gives a way to construct $M_w$.

**Lemma 4.3.** *Let $C \geq 2$ and $D \geq 1$ be integers, let $T(n) = Kn^k + 1$ for some integers $K, k \geq 1$ and let $M$ be a one-tape $q$-state NTM that runs in time $T(n)$. Then there exists an*

$$\mathrm{O}\left(T(n)^{2/(C-1)} + n^2\right)\text{-}\textit{time}$$

*multi-tape DTM that given an input $w$ for $M$, constructs a one-tape NTM $M_w$ such that*

*$M_w$ runs in time $Cn + D$ **iff** $M$ rejects $w$.*

**Proof.** Let us first describe the NTM $M_w$. The computation of $M_w$ on an input $\tilde{w}$ will consist of two phases. In the first phase, $M_w$ will use at most $(C-1)$ deterministic passes through the input to assure that $\tilde{w}$ is long enough. We will describe this phase in detail later.

In the second phase, $M_w$ will write $w$ on its tape and simulate $M$ on $w$. Hence $\mathrm{O}(|w|)$ states and $\mathrm{O}(T(|w|))$ time are needed for this phase (note that $q$ is a constant). If $M$ accepts $w$, $M_w$ starts an infinite loop, else it halts. Let $c$ be a constant such that $M_w$ makes at most $cT(|w|)$ steps in the second phase before starting the infinite loop.

To describe the first phase, define

$$\begin{aligned} m &= \left\lceil \left(cT(|w|)(C-2)!\right)^{1/(C-1)} \right\rceil \\ &= \mathrm{O}\left(T(|w|)^{1/(C-1)}\right), \end{aligned}$$

where $\lceil x \rceil$ denotes the smallest integer that is greater than or equal to $x$. In the first phase, the machine $M_w$ simply passes through the input $(C-1)$ times, each time verifying that $|\tilde{w}|$ is divisible by one of the numbers $(m+i)$, for $i = 0, 1 \ldots (C-2)$. If this is not the case, $M_w$ rejects. Else, the second phase is to be executed. It suffices to have $(m+i)$ states to verify in one pass if the length of the input is divisible by $(m+i)$, so we can make $M_w$ have

$$\begin{aligned} \mathrm{O}\left(\sum_{i=0}^{C-2}(m+i)\right) &= \mathrm{O}\left((C-1)m\right) \\ &= \mathrm{O}(m) \end{aligned}$$

states for the first phase such that it makes at most $(C-1)|\tilde{w}|+1$ steps before entering the second phase[5]. We assume that $M_w$ erases all symbols from the tape in the last pass of the first phase so that the second phase can begin with a blank tape.

If the second phase begins, we know that

$$\begin{aligned} |\tilde{w}| &\geq \mathrm{lcm}\{m, (m+1)\ldots(m+C-2)\} \\ &\geq \frac{m^{C-1}}{(C-2)!} \\ &\geq cT(|w|), \end{aligned}$$

where we used the inequality

$$\mathrm{lcm}\{m, (m+1)\ldots(m+C-2)\} \geq m\binom{m+C-2}{C-2}$$

proven in [2]. Hence, $M_w$ makes at most $|\tilde{w}|$ steps in the second phase **iff** it does not go into an infinite loop. So we have proven that

$$M_w \text{ runs in time } Cn+1 \textbf{ iff } M_w \text{ runs in time } Cn+D \textbf{ iff } M \text{ rejects } w.$$

To construct $M_w$, we first compute $m$ which takes $\mathrm{O}(|w|)$ time and then in $\mathrm{O}(m^2)$ time we compute a Turing machine $M_1$ that does the first phase (the construction is straightforward). Finally we compose $M_1$ with the Turing machine $M$, only that $M$ first writes $w$ on the tape and $M$, instead of going to the accept state, starts moving to the right forever. Because $M$ is not a part of the input and because we can compute compositions of Turing machines in linear time, the description of $M_w$ can be obtained in $\mathrm{O}(m^2 + |w|^2)$ time, which is $\mathrm{O}\left(T(n)^{2/(C-1)} + n^2\right)$. □

We now combine Corollary 4.2 and Lemma 4.3 to show that $\mathrm{HALT}_{(C,D)}$ is co-NP-complete.

[5] The "plus one" in $(C-1)|\tilde{w}|+1$ is needed because each Turing machine makes at least one step on the empty input. This is also the reason for why we need $D \geq 1$ in the statement of the lemma.

**Proposition 4.4.** *The problems* $\text{HALT}_{(C,D)}$ *are* co-NP-complete *for all* $C \geq 2$ *and* $D \geq 1$.

**Proof.** Corollary 4.2 proves that these problems are in co-NP and Lemma 4.3 gives a Karp reduction of an arbitrary problem in co-NP to the above ones. □

The first lower bound for the problems $\text{HALT}_{(C,D)}$ follows. To prove it, we will use Lemma 4.3 to translate a hard problem to $\text{HALT}_{(C,D)}$.

**Proposition 4.5.** *For positive integers* $C$ *and* $D$*, the problem* $\overline{\text{HALT}_{(C,D)}}$ *can not be solved by a multi-tape NTM in time* $\mathrm{o}(q^{(C-1)/2})$.

**Proof.** For $C \leq 5$, the proposition holds (the length of the input is $\Theta(q^2)$), so suppose $C \geq 6$. By the non-deterministic time hierarchy theorem [9, 12] there exists a language $L$ and a multi-tape NTM $M$ that decides $L$ and runs in time $\mathrm{O}(n^{C-1})$, while no multi-tape NTM can decide $L$ in time $\mathrm{o}(n^{C-1})$. We can reduce the number of tapes of $M$ to get a one-tape NTM $M'$ that runs in time $\mathrm{O}(n^{2(C-1)})$ and decides $L$. By Lemma 4.3 there exists a multi-tape DTM $M_{\text{mult}}$ that runs in time $\mathrm{O}(n^4)$ and given an input $w$ for $M'$, constructs a one-tape $q_w$-state NTM $M_w$ such that

$$M_w \text{ runs in time } Cn + D \textbf{ iff } M' \text{ rejects } w.$$

Because the description of $M_w$ has length $\mathrm{O}(|w|^4)$, it follows that $q_w = \mathrm{O}(|w|^2)$.

If there was some multi-tape NTM that could solve $\overline{\text{HALT}_{(C,D)}}$ in time $\mathrm{o}\left(q^{(C-1)/2}\right)$, then for all $w$, we could decide whether $w \in L$ in $\mathrm{o}(n^{C-1})$ non-deterministic time: first run $M_{\text{mult}}$ to get $M_w$ and then solve $\overline{\text{HALT}_{(C,D)}}$ for $M_w$. By the definition of $L$ this is impossible, hence the problem $\overline{\text{HALT}_{(C,D)}}$ can not be solved by a multi-tape NTM in time $\mathrm{o}\left(q^{(C-1)/2}\right)$. □

For all of the rest lower bounds, we need to reformulate Lemma 4.3 a bit. We say that an NTM $M$ is a *two-choice* NTM if in each step it has at most two possible non-deterministic choices.

**Lemma 4.6.** *Let* $C \geq 2$ *and* $D \geq 1$ *be integers and let* $T(n) = Kn^k + 1$ *for some integers* $K, k \geq 1$. *Then there exists a multi-tape DTM* $M_{\text{mult}}$*, which given an input* $(M, w)$*, where* $w$ *is an input for a one-tape two-choice q-state NTM* $M$*, constructs a one-tape DTM* $M_w$ *such that*

$$M_w \text{ runs in time } Cn + D \textbf{ iff } M \text{ makes at most } T(|w|) \text{ steps on the input } w.$$

*We can make* $M_{\text{mult}}$ *run in time*

$$\mathrm{O}\left(T(|w|)^{4/(C-1)} + q^{\kappa} + |w|^2\right)$$

*for some integer* $\kappa \geq 1$*, independent of* $C$, $D$, $K$ *and* $k$.

**Proof.** The proof is based on the same idea as the proof of Lemma 4.3. The main difference is that this time we will have to count steps while we will simulate $M$ and we will have to use the symbols of an input of the DTM $M_w$ to simulate non-deterministic choices of $M$.

Again, we begin with the description of $M_w$. The computation of $M_w$ on an input $\tilde{w}$ will consist of two phases. In the first phase, $M_w$ will use at most $(C-1)$ deterministic passes through the input to assure that $\tilde{w}$ is long enough. This phase will be the same as in Lemma 4.3, only that we will need more states to measure longer inputs because the second phase will be more time consuming.

This time we define

$$\begin{aligned} m &= \left\lceil \left( \left( cT(|w|) \right)^2 (C-2)! \right)^{1/(C-1)} \right\rceil \\ &= \mathrm{O}\left(T(|w|)^{2/(C-1)}\right) \end{aligned}$$

for some constant $c$ defined later. In the first phase, the machine $M_w$ simply passes through the input $(C-1)$ times, each time verifying that $|\tilde{w}|$ is divisible by one of the numbers $(m+i)$, for $i = 0, 1 \ldots (C-2)$. If this is not the case, $M_w$ rejects. Else, the second phase is to be executed. It suffices to have $(m+i)$ states to verify in one pass if the length of the input is divisible by $(m+i)$, so we can make $M_w$ have

$$\mathrm{O}\left(\sum_{i=0}^{C-2}(m+i)\right) = \mathrm{O}(m)$$

states for the first phase such that it makes at most $(C-1)|\tilde{w}|+1$ steps before entering the second phase. We also need that while the Turing machine $M_w$ passes through the input in the first phase, it does not change it, except that it erases the first symbol of $\tilde{w}$ (if not, $M_w$ would need $(n+1)$ steps for one pass through the input). Additionally, if the input $\tilde{w}$ contains some symbol that is not 0 or 1, $M_w$ rejects[6].

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\tilde{w}$ | ␣ | ␣ | 0 | $w_1$ | 1 | $w_2$ | 1 | $\cdots$ | 1 | $w_n$ | ␣ | $1^{2T(\|w\|)}$ | |

Figure 6: The preparation for the simulation in the phase two. The input $\tilde{w}$ (without the first symbol) is much longer compared to what is on the right side of it (phase one takes care for that). After the phase one, the head of $M_w$ could be on the left side of the input $\tilde{w}$ or on the right side of it, depending on the parity of $C$. Let us assume that $C$ is even and hence the head of $M_w$ is on the right side of $\tilde{w}$ after the phase one. Before the simulation begins, $M_w$ writes the following on the right side of $\tilde{w}$: ␣␣0 followed by $w$ with the symbol 1 inserted between each two of its symbols. Then on the right of $w$ it computes $2T(|w|)$ in unary so that we get the situation as shown on the figure. If $C$ is odd, we can look on the tape of $M_w$ from behind and do everything the same as in the case of $C$ being even.

In the second phase, $M_w$ will compute $T(|w|)$ and simulate $M$ on $w$ for at most $T(|w|)$ steps, using the non-deterministic choices determined by $\tilde{w}$. If $M$ will not halt, $M_w$ will start an infinite loop, else it will halt. In Figure 6 we see how $M_w$ makes the preparation for the second phase. Let us call the part of the tape with the symbols of $\tilde{w}$ written on it the *non-deterministic part*, the part of the tape from 0 to $w_n$ the *simulating part* and the part of the tape with $1^{2T(|w|)}$ written on it the *counting part*. During the simulation, the following will hold.

- In the simulating part, it will always be the case that the symbols from the tape of $M$ will be in every second cell and between each two of them will always be the symbol 1, except on the left of the cell with the head of $M$ on it where it will be 0.
- There will always be at least two blank symbols left of the simulating part and there will always be at least one blank symbol right of the simulating part. This will be possible because before each simulated step of $M$, as explained below, the number of blank symbols left and right of the simulating part will be increased by two for each side, hence when simulating a step of $M$, the simulating part can be increased as necessary.
- Before each simulated step of $M$, $M_w$ will use the rightmost symbol of the non-deterministic part of the tape to determine a non-deterministic choice for $M$ and it will overwrite the two rightmost symbols of the non-deterministic part of the tape with two blank symbols.
- Before each simulated step of $M$, $M_w$ will overwrite the two leftmost symbols of the counting part of the tape with two blank symbols.
- If $M$ halts before the counting part of the tape vanishes, $M_w$ halts. Else, $M_w$ starts an infinite loop.

[6] Recall that in the definition of a one-tape Turing machine in Section 2 it was required that $0, 1 \in \Sigma$.

We see that $M_w$, if it does not go into an infinite loop, finishes the second phase in time $\mathrm{O}(T(|w|)^2)$ using $\mathrm{O}(|w|+q)$ states. Note that to achieve that, the counting part of the tape really has to be computed and not encoded in the states, which takes $\mathrm{O}(T(|w|)^2)$ steps. A possible implementation of this would be to first write $|w|$ in binary ($|w|$ can be encoded in the states), then compute $T(|w|)$ in binary and extend it to unary.

To define the integer $c$ that is used in the first phase, suppose that $M_w$ makes at most $(cT(|w|))^2$ steps in the second phase before starting the infinite loop. Note that $c$ is independent of $M$ and $w$. If the second phase begins, we know that

$$\begin{aligned}|\tilde{w}| &\geq \mathrm{lcm}\{m,(m+1)\dots(m+C-2)\}\\ &\geq \frac{m^{C-1}}{(C-2)!}\\ &\geq (cT(|w|))^2\,,\end{aligned}$$

as in the proof of Lemma 4.3, thus $M_w$ makes at most $|\tilde{w}|$ steps in the second phase **iff** it does not go into an infinite loop. This inequality also implies that the non-deterministic part of the tape in the phase two is long enough so that it does not vanish during the simulation of $M$.

Now if $M$ makes at most $T(|w|)$ steps on all computations on the input $w$, then $M_w$ runs in time $Cn+1$. But if there exists a computation $\zeta$ on input $w$ such that $M$ makes more than $T(|w|)$ steps on it, then because $M$ is a two-choice machine, there exists a binary input $\tilde{w}$ for $M_w$ such that the non-deterministic part of the tape in the phase two corresponds to the non-deterministic choices of $\zeta$, hence $M_w$ on the input $\tilde{w}$ simulates more than $T(|w|)$ steps of $M$ which means that the counting part of the tape vanishes and thus $M_w$ does not halt on the input $\tilde{w}$. So we have proven that

$M_w$ runs in time $Cn+1$ **iff** $M_w$ runs in time $Cn+D$ **iff** $M$ makes at most $T(|w|)$ steps on the input $w$.

Now let us describe a multi-tape DTM $M_{\text{mult}}$ that constructs $M_w$ from $(M,w)$. First we prove that, for some integer $\kappa$ independent of $C,\ D,\ K$ and $k$, the DTM $M_{\text{mult}}$ can construct, in time $\mathrm{O}(q^\kappa+|w|^2)$, a one-tape DTM $M_2$ that does the second phase. To see this, let $M_T$ be a one-tape DTM that given a number $x$ in binary, its head never crosses the boundary $-1$ and it computes $T(x)$ in unary in time $\mathrm{O}(T(x)^2)$. Note that $M_T$ does not depend on the input $(M,w)$ for $M_{\text{mult}}$ and thus it can be computed in constant time. Now $M_2$ can be viewed as a composition of three deterministic Turing machines:

- The first DTM writes down the simulating part of the tape, followed by $|w|$ written in binary. $M_{\text{mult}}$ needs $\mathrm{O}(|w|^2)$ time to construct this DTM.

- The second DTM is $M_T$ and $M_{\text{mult}}$ needs $\mathrm{O}(1)$ time to construct it.

- The third DTM performs the simulation of $M$ on $w$ and $M_{\text{mult}}$ needs $\mathrm{O}(q^\kappa)$ time to construct it, where $\kappa$ is independent of $C,\ D,\ K$ and $k$.

Because the composition of Turing machines can be computed in linear time, we can construct $M_2$ in time $\mathrm{O}(q^\kappa+|w|^2)$.

Because the first phase does not depend on $M$ and we need $\mathrm{O}(m)$ states to do it, $M_{\text{mult}}$ can compute the DTM $M_1$ that does the first phase in time

$$\mathrm{O}(m^2)=\mathrm{O}\left(T(|w|)^{4/(C-1)}\right),$$

as in the proof of Lemma 4.3. Since $M_w$ is the composition of $M_1$ and $M_2$, $M_{\text{mult}}$ can construct $M_w$ in time

$$\mathrm{O}\left(T(|w|)^{4/(C-1)}+q^\kappa+|w|^2\right). \qquad \square$$

**Proposition 4.7.** *For positive integers C and D, the problem* $\mathrm{DHALT}_{(C,D)}$ *can not be solved by a multi-tape NTM in time* $\mathrm{o}(q^{(C-1)/4})$.

**Proof.** For $C \leq 9$, the proposition holds (the length of the input is $\Theta(q^2)$), so suppose $C \geq 10$. Let $\kappa$ be as in Lemma 4.6. Define a *padded code* of a one-tape NTM $M$ as a code of $M$, padded in front by any number of 0s followed by a redundant 1. Thus the padded code of a one-tape NTM can be arbitrarily long.

Let $M$ be the following one-tape NTM:

- On an input $w$, which is a padded code of a one-tape two-choice NTM $M'$, construct a one-tape $q_w$-state DTM $M_w$ such that

  $$M_w \text{ runs in time } Cn + D \textbf{ iff } M' \text{ makes at most } |w|^{\kappa(C-1)} \text{ steps on the input } w.$$

  The machine $M_w$ can be constructed by a multi-tape DTM in time $\mathrm{O}(|w|^{4\kappa})$ by Lemma 4.6, hence it can be constructed in time $\mathrm{O}(|w|^{8\kappa})$ by a one-tape DTM. It also follows that $q_w = \mathrm{O}(|w|^{2\kappa})$.

- Verify whether $M_w$ runs in time $Cn + D$. If so, start an infinite loop, else halt.

Now we make a standard diagonalization argument to prove the proposition. Suppose that $\mathrm{DHALT}_{(C,D)}$ can be solved by a multi-tape NTM in time $\mathrm{o}(q^{(C-1)/4})$. Then it can be solved in time $\mathrm{o}(q^{(C-1)/2})$ by a one-tape NTM. Using more states and for a constant factor more time, $\mathrm{DHALT}_{(C,D)}$ can be solved in time $\mathrm{o}(q^{(C-1)/2})$ by a one-tape two-choice NTM. If $M$ uses this machine to verify whether $M_w$ runs in time $Cn + D$, then considering $q_w = \mathrm{O}(|w|^{2\kappa})$ and $C \geq 10$, $M$ is a one-tape two-choice NTM that makes

$$\mathrm{O}\left(|w|^{8\kappa}\right) + \mathrm{o}\left(|w|^{\kappa(C-1)}\right) = \mathrm{o}\left(|w|^{\kappa(C-1)}\right)$$

steps on any computation on the input $w$, if it does not enter the infinite loop.

Let $w$ be a padded code of $M$. If $M$ makes at most $|w|^{\kappa(C-1)}$ steps on the input $w$, the Turing machine $M_w$ will run in time $Cn + D$ which implies that $M$ will start an infinite loop on some computation on the input $w$, which is a contradiction. Hence, $M$ must make more steps on the input $w$ than $|w|^{\kappa(C-1)}$ which implies that $M_w$ does not run in time $Cn + D$ and hence $M$ does not start the infinite loop, thus it makes $\mathrm{o}\left(|w|^{\kappa(C-1)}\right)$ steps. It follows that

$$\mathrm{o}\left(|w|^{\kappa(C-1)}\right) > |w|^{\kappa(C-1)}$$

which is impossible since the padding can be arbitrarily long. □

**Corollary 4.8.** *For positive integers $C$ and $D$, the problem $\mathrm{HALT}_{(C,D)}$ can not be solved by a multi-tape NTM in time $\mathrm{o}(q^{(C-1)/4})$.*

**Proof.** The result follows by Proposition 4.7 because a Turing machine that solves $\mathrm{HALT}_{(C,D)}$ also solves $\mathrm{DHALT}_{(C,D)}$. □

The following lemma is a "deterministic" analog of Lemma 4.3.

**Lemma 4.9.** *Let $C \geq 2$ and $D \geq 1$ be integers, let $T(n) = Kn^k + 1$ for some integers $K, k \geq 1$ and let $M$ be a one-tape two-choice $q$-state NTM that runs in time $T(n)$. Then there exists an*

$$\mathrm{O}\left(T(n)^{4/(C-1)} + n^2\right)\text{-time}$$

*multi-tape DTM that given an input $w$ for $M$, constructs a one-tape DTM $M_w$ such that*

$$M_w \text{ runs in time } Cn + D \textbf{\textit{ iff }} M \text{ rejects } w.$$

**Proof.** Let $M'$ be a one-tape two-choice $(q+1)$-state NTM that computes just like $M$ only that it starts an infinite loop whenever $M$ would go to the accepting state. It follows that

$$M' \text{ makes at most } T(|w|) \text{ steps on the input } w \textbf{ iff } M \text{ rejects } w.$$

Now we can use Lemma 4.6 to construct a DTM $M_w$ such that

$$M_w \text{ runs in time } Cn + D \textbf{ iff } M' \text{ makes at most } T(|w|) \text{ steps on the input } w \textbf{ iff } M \text{ rejects } w.$$

Because $M$ and $M'$ are only parameters in the lemma we are proving, we can construct $M_w$ in time

$$\mathrm{O}\left(T(|w|)^{4/(C-1)} + |w|^2\right)$$

by Lemma 4.6. □

We now combine Corollary 4.2 and Lemma 4.9 to show that DHALT$_{(C,D)}$ is co-NP-complete.

**Proposition 4.10.** *The problems* DHALT$_{(C,D)}$ *are* co-NP-complete *for all* $C \geq 2$ *and* $D \geq 1$.

**Proof.** Corollary 4.2 proves that these problems are in co-NP and Lemma 4.9 gives a Karp reduction of an arbitrary problem in co-NP to the above ones. □

To prove the last lower bound, we use Lemma 4.9 to translate a hard problem to DHALT$_{(C,D)}$, the same way as in Proposition 4.5.

**Proposition 4.11.** *For positive integers $C$ and $D$, the problem* $\overline{\text{DHALT}_{(C,D)}}$ *can not be solved by a multi-tape NTM in time* $\mathrm{o}(q^{(C-1)/4})$.

**Proof.** The proof is the same as the proof of Proposition 4.5, only that we use Lemma 4.9 instead of Lemma 4.3. □

To sum up this section, we have proven **Theorem 1.1** which states
*For integers $C \geq 2$ and $D \geq 1$, all of the following holds.*

i) *The problems* HALT$_{(C,D)}$ *and* DHALT$_{(C,D)}$ *are* co-NP-complete.
Proposition 4.4 and Proposition 4.10 prove this.

ii) The problems HALT$_{(C,D)}$ and DHALT$_{(C,D)}$ **can not** be solved in non-deterministic time $\mathrm{o}(q^{(C-1)/4})$.
Proposition 4.7 and Corollary 4.8 prove this.

iii) The problems $\overline{\text{HALT}_{(C,D)}}$ and $\overline{\text{DHALT}_{(C,D)}}$ **can** be solved in non-deterministic time $\mathrm{O}(q^{C+2})$.
Corollary 4.2 proves this.

iv) The problem $\overline{\text{HALT}_{(C,D)}}$ **can not** be solved in non-deterministic time $\mathrm{o}(q^{(C-1)/2})$.
Proposition 4.5 proves this.

v) The problem $\overline{\text{DHALT}_{(C,D)}}$ **can not** be solved in non-deterministic time $\mathrm{o}(q^{(C-1)/4})$.
Proposition 4.11 proves this. □

# 5 Final Words

In this section we first show that our methods can not give essentially better lower bounds as in Theorem 1.1. Then we observe that, for Theorem 1.1, we needed $C \geq 2$ and $D \geq 1$ and we discuss what happens if $C = 0$, $C = 1$ or $D = 0$.

## 5.1 The Optimality of Our Measuring of the Length of an Input

Let us again have a look at how we proved the lower bound in Proposition 4.5. A very similar idea was also used to prove all the other lower bounds in Theorem 1.1, so what follows can be applied to any of them.

Let a one-tape non-deterministic Turing machine $M$ solve a problem $L$ in time $T(n)$. Then for any input $w$, we can decide whether $w \in L$ by first constructing a one-tape Turing machine $M_w$ that runs in time $Cn + D$ iff $M$ rejects $w$ and then solving $\mathrm{HALT}_{(C,D)}$ for $M_w$. On the inputs of length $n$, the Turing machine $M_w$ computed in two phases (see Lemma 4.3): in the first phase $M_w$ only measured the input length using at most $(C-1)n+1$ steps to assure that $n$ was large enough, specifically $n = \Omega(T(|w|))$, and then in the second phase it simulated $M$ on $w$ using at most $n$ steps. In our implementation $M_w$ used $\mathrm{O}(T(|w|)^{1/(C-1)})$ states for the first phase and *we claim that this is optimal.*

If we want $M_w$ to measure the length $T(|w|)$ in the first phase using at most $(C-1)n+1$ steps, then for each computation on inputs of length $T(|w|)$, it can not produce the same crossing sequence on two boundaries. By Lemma 3.3, $M_w$ has $\Omega(T(|w|)^{1/(C-1)})$ states which implies that our measuring of the length of the input was optimal. What is more, Lemma 3.3 is tight and our method for proving lower bounds can not give much better bounds.

## 5.2 An Open Problem

*For $D \in \mathbb{N}$, how hard are the problems* $\mathrm{HALT}_{(1,D)}$ *and* $\mathrm{DHALT}_{(1,D)}$*?*

It is clear that we can solve the problems $\mathrm{HALT}_{(C,0)}$ and $\mathrm{DHALT}_{(C,0)}$, for $C \in \mathbb{N}$, in constant time. The answer is always NO, since any Turing machine makes at least one step on the empty input.

It is also easy to see that we can solve the problems $\mathrm{HALT}_{(0,D)}$ and $\mathrm{DHALT}_{(0,D)}$, for $D \in \mathbb{N}$, in polynomial time. The algorithm would be to simulate a given one-tape Turing machine $M$ on all the inputs up to the length $D$ and accept iff the time bound was not violated. Now if the algorithm rejects, $M$ clearly does not run in time $D$ and if it accepts, then $M$ never reads the $(D+1)$st symbol of an input (see Figure 7) and hence it was enough to verify the running time on inputs up to the length $D$.

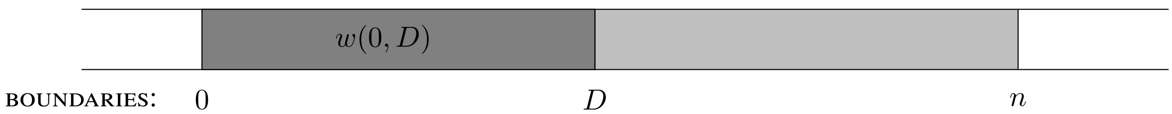


Figure 7: Suppose a Turing machine reads the $(D+1)$st symbol of an input $w$ of length $n$ on some computation. Then this Turing machine must cross the $D$th boundary on the input $w(0,D)$ and read the blank symbol behind. Hence it makes at least $(D+1)$ steps on this input of length $D$.

For $C \geq 2$ and $D \geq 1$, good complexity bounds for $\mathrm{HALT}_{(C,D)}$ and $\mathrm{DHALT}_{(C,D)}$ are given in Theorem 1.1. Hence only the bounds for $C = 1$ are missing[7]. For this case, we can prove the following proposition.

**Proposition 5.1.** *The problems* $\mathrm{HALT}_{(1,1)}$ *and* $\mathrm{DHALT}_{(1,1)}$ *are solvable in deterministic polynomial time.*

**Proof.** The main observation is that a one-tape NTM which runs in time $(n+1)$ never moves its head to the left, except possibly in the last two steps of a computation. To prove this, we suppose the opposite. Let $M$ be a one-tape NTM that runs in time $(n+1)$ and let $w$ be an input for $M$ such that on some computation on $w$, $M$ moves its head to the left for the first time on some time step $t < n = |w|$ and it makes at least two more steps afterwards. As can be seen in Figure 8, $M$ makes more than $(t+1)$ steps on some computation on the input $w(0,t)$ of length $t$, which is a contradiction.

[7] This paper also does not cover more general cases where $C, D \in \mathbb{Q}$ or even $C, D \in \mathbb{R}$. We did not think much about such generalizations.

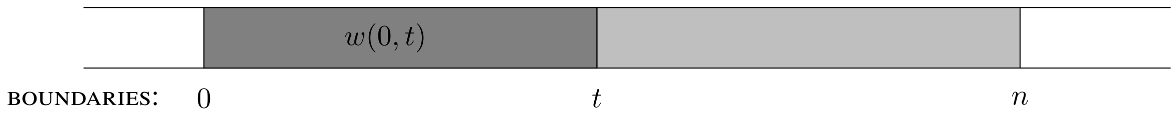


Figure 8: Suppose that a Turing machine $M$ on input $w$ of length $n$ moves its head to the left for the first time on a time step $t$ (the head turns left just before crossing the boundary $t$) and let $M$ make at least two more steps after this step (we assume some fixed computation). Then $M$ on the input $w(0,t)$ makes at least $(t+2)$ steps.

Hence, to solve $\text{HALT}_{(1,1)}$ and $\text{DHALT}_{(1,1)}$ for a given one-tape Turing machine $M$, we have to verify that the head of $M$ never moves to the left, except possibly in the last two steps of a computation, which can be verified in polynomial time. □

Does a similar proof go through for all problems $\text{HALT}_{(1,D)}$?

**Acknowledgements**

The author wishes to thank his research advisor Sergio Cabello for valuable comments and Valentine Kabanets for the help with the presentation of results. This work is partially funded by the Slovenian Research Agency.

## References

[1] A. Adachi, S. Iwata, and T. Kasai. Some combinatorial game problems require $\Omega(n^k)$ time. *J. ACM*, 31(2):361–376, 1984.

[2] B. Farhi. Nontrivial lower bounds for the least common multiple of some finite sequences of integers. *J. Number Theory*, 125(2):393 – 411, 2007.

[3] D. Gajser. Verifying time complexity of deterministic Turing machines. *CoRR*, abs/1307.3648, 2013.

[4] J. Hartmanis. Computational complexity of one-tape Turing machine computations. *J. ACM*, 15(2):325–339, 1968.

[5] F. C. Hennie. One-tape, off-line Turing machine computations. *Information and Control*, 8(6):553–578, 1965.

[6] P. Hájek. Arithmetical hierarchy and complexity of computation. *Theor. Comput. Sci.*, 8(2):227 – 237, 1979.

[7] K. Kobayashi. On the structure of one-tape nondeterministic Turing machine time hierarchy. *Theor. Comput. Sci.*, 40(2-3):175–193, 1985.

[8] G. Pighizzini. Nondeterministic one-tape off-line Turing machines and their time complexity. *J. Autom. Lang. Comb.*, 14(1):107–124, 2009.

[9] J. I. Seiferas, M. J. Fischer, and A. R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, 1978.

[10] K. Tadaki, T. Yamakami, and J. C. H. Lin. Theory of one-tape linear-time Turing machines. *Theor. Comput. Sci.*, 411(1):22–43, 2010.

[11] B. A. Trakhtenbrot. Turing computations with logarithmic delay. *Algebra i Logica 3*, pages 33–48, 1964. In Russian.

[12] S. Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327 – 333, 1983.

# A Appendix

We prove here the following technical lemma.

**Lemma A.1.** *For every $q \geq 2$ and $C \in \mathbb{N}$, it holds*

$$\sum_{j=0}^{C} q^j (C-j) = \frac{q^{C+1} - (C+1)q + C}{(q-1)^2} \leq 4q^{C-1}.$$

**Proof.**

$$\begin{aligned}
\sum_{j=0}^{C} q^j (C-j) &= C \sum_{j=0}^{C} q^j - q \frac{d}{dq} \left( \sum_{j=0}^{C} q^j \right) \\
&= C \frac{q^{C+1} - 1}{q-1} - q \frac{d}{dq} \left( \frac{q^{C+1} - 1}{q-1} \right) \\
&= \frac{q^{C+1} - (C+1)q + C}{(q-1)^2}.
\end{aligned}$$

It is easy to see that, for $q \geq 2$, it follows $\frac{q^{C+1}-(C+1)q+C}{(q-1)^2} \leq \frac{q^{C+1}}{(q-1)^2} \leq 4q^{C-1}$. □



ECCC ISSN 1433-8092

http://eccc.hpi-web.de