I F I G
RESEARCH
REPORT

INSTITUT FÜR INFORMATIK

DETERMINISTIC TURING MACHINES IN THE RANGE BETWEEN REAL-TIME AND LINEAR-TIME

Andreas Klein Martin Kutrib

IFIG RESEARCH REPORT 0002
FEBRUARY 2000

Institut für Informatik
JLU Gießen
Arndtstraße 2
D-35392 Giessen, Germany
Tel: +49-641-99-32141
Fax: +49-641-99-32149
mail@informatik.uni-giessen.de
www.informatik.uni-giessen.de

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

IFIG Research Report
IFIG Research Report 0002, February 2000

Deterministic Turing Machines in the Range between Real-Time and Linear-Time

Andreas Klein[1] and Martin Kutrib[2]

Institute of Informatics, University of Giessen
Arndtstr. 2, D-35392 Giessen, Germany

Abstract. Deterministic k-tape and multitape Turing machines with one-way, two-way and without a separated input tape are considered. We investigate the classes of languages acceptable by such devices with time bounds of the form $n + r(n)$ where $r \in o(n)$ is a sublinear function. It is shown that there exist infinite time hierarchies of separated complexity classes in that range. For these classes weak closure properties are proved. Finally, it is shown that similar results are valid for several types of acceptors with the same time bounds.

CR Subject Classification (1998): F.1.3, F.1.1, F.4.3

[1]E-mail: andreas.klein@math.uni-giessen.de
[2]E-mail: kutrib@informatik.uni-giessen.de

1 Introduction

When one is particularly interested in computations with small time bounds, lets say in the range between real-time and linear-time, most of the relevant Turing machine results have been published in the early times of computational complexity.

In the sequel we are concerned with time bounds of the form $id+r$ where id denotes the identity function on integers and $r \in o(id)$ a sublinear function. So nondeterministic Turing machines would not be fruitful devices for investigations. From [7] we know that the real-time and linear-time classes are identical for one-tape machines $\mathsf{NTIME}_1(id) = \mathsf{NTIME}_1(\mathrm{LIN})$. In [2] it has been shown that the complexity class Q which is defined by nondeterministic multitape real-time computations ($\mathsf{NTIME}(id)$) is equal to the corresponding linear-time languages ($\mathsf{NTIME}(\mathrm{LIN})$). Moreover, it has been shown that two working tapes and a one-way input tape $(2:1)$ are sufficient to accept the languages from Q in real-time. With other words, $\mathsf{NTIME}_{2:1}(id) = \mathsf{Q} = \mathsf{NTIME}(\mathrm{LIN})$. Thus, for almost all nondeterministic Turing machines there is no difference between real-time and linear-time.

The same does not hold true for deterministic machines. Though in [7] for one-tape the identity $\mathsf{DTIME}_1(id) = \mathsf{DTIME}_1(\mathrm{LIN})$ has been proved, for a total of at least two tapes the real-time languages are strictly included in the linear-time languages: In [11] a language belonging to $\mathsf{DTIME}_{1:1}(\mathrm{LIN})$ but not to $\mathsf{DTIME}(id)$ has been presented. Consequently, the investigations have to be in terms of deterministic Turing machines.

Another aspect that, at first glance, might attack the time range of interest is a possible speed-up. The well-known [6] linear speed-up from $t(n)$ to $id + \varepsilon \cdot t(n)$ for arbitrary $\varepsilon > 0$ yields complexity classes close to real-time (i.e. $\mathsf{DTIME}(\mathrm{LIN}) = \mathsf{DTIME}((1+\varepsilon)\cdot id))$ for k-tape and multitape machines but does not allow assertions on the range between real-time and linear-time. An application to the time bound $id + r$, $r \in o(id)$, would result in a slow-down to $id + \varepsilon \cdot (id + r) \geq id + \varepsilon \cdot id$.

Let us recall known time hierarchy results. For a number of $k \geq 2$ tapes in [4, 10] the hierarchy $\mathsf{DTIME}_k(t') \subset \mathsf{DTIME}_k(t)$, if $t' \in o(t)$ and t is time-constructible, has been shown. By the linear speed-up we obtain the necessity of the condition $t' \in o(t)$. The necessity of the constructibility property of t follows from the well-known gap theorem.

Since in case of multitape machines one needs to construct a Turing machine with a fixed number of tapes that simulates machines even with more tapes, the proof of a corresponding hierarchy involves a reduction of the number of tapes. This costs a factor log for the time complexity. The hierarchy $\mathsf{DTIME}(t') \subset \mathsf{DTIME}(t)$, if $t' \cdot \log(t') \in o(t)$ and t is time-constructible, has been proved in [6].

Due to the necessary condition $t' \in o(t)$ resp. $t' \cdot \log(t') \in o(t)$, again, the range between real-time and linear-time is not affected by the known time hierarchy results. On the other hand, the hierarchy $\mathsf{DTIME}_k(t') \subset \mathsf{DTIME}_k(t)$ is tight above linear-time what follows immediately from the condition $t' \in o(t)$ and the linear speed-up. For example, the trivial inclusions $\mathsf{DTIME}_k(3 \cdot id) \supseteq \mathsf{DTIME}_k(2 \cdot id + r) \supseteq \mathsf{DTIME}_k(2 \cdot id)$ become equalities for $\varepsilon = \frac{1}{3}$ by $\mathsf{DTIME}_k(3 \cdot id) = \mathsf{DTIME}_k(id + \varepsilon \cdot 3 \cdot id) = \mathsf{DTIME}_k(2 \cdot id)$. We conclude that there are no infinite hierarchies for time bounds of the form $t + r$, $r \in o(id)$, if $t \geq c \cdot id$, $c > 1$. In this sense the range between real-time and linear-time is a white area in the map. In the following we are going to color it.

The basic notions and preliminary results of a technical flavor are the objects of the next section. Section 3 is devoted to the hierarchies between real-time and linear-time. In particular, by generalizing a well-known equivalence relation to time complexities above real-time it is shown that specific languages which are constructed dependent on the given time complexity are not acceptable by multitape Turing machines obeying the smaller time bound. Conversely, it is proved by construction that these languages are acceptable by one-tape Turing machines with a two-way input tape whereby the larger time bound is obeyed. For the remaining case of one-tape machines with a one-way input tape a hierarchy is shown by easing the condition that relates each two time complexities. In Section 4 the question whether or not the hierarchies may be refined are discussed. By relating the hypothesis to a speed-up result it will turn out that some of the hierachies are optimal. The weak closure properties of the complexity classes in question are studied in Section 5. Since the proofs of our negative results depend on a equivalence relation we can show that similar results are valid for several types of acceptors as long as the number of distinguishable equivalence classes is bounded similarly.

2 Preliminaries

We denote the rational numbers by $\mathbb{Q}$, the integers by $\mathbb{Z}$, the positive integers $\{1, 2, ...\}$ by $\mathbb{N}$ and the set $\mathbb{N} \cup \{0\}$ by $\mathbb{N}_0$. The empty word is denoted by ε and the reversal of a word w by w^R. For the length of w we write $|w|$. We use $\subseteq$ for inclusions and $\subset$ if the inclusion is strict. For a function $f : \mathbb{N}_0 \to \mathbb{N}$ we denote its i-fold composition by $f^{[i]}$, $i \in \mathbb{N}$. If f is increasing then its inverse is defined according to $f^{-1}(n) = \min\{m \in \mathbb{N} \mid f(m) \geq n\}$. The identity function $n \mapsto n$ is denoted by id. As usual we define the set of functions that grow strictly less than f by $o(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0\}$. In terms of orders of magnitude f is an upper bound of the set $O(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid \exists\, n_0, c \in \mathbb{N} : \forall\, n \geq n_0 : g(n) \leq c \cdot f(n)\}$. Conversely, f is a lower bound of the set $\Omega(f) = \{g : \mathbb{N}_0 \to \mathbb{N} \mid f \in O(g)\}$.

A Turing machine with $k \in \mathbb{N}$ tapes consists of a finite-state control and k one-dimensional infinite two-way tapes. On each tape a read-write head is positioned. At the outset of a computation the Turing machine is in the designated initial state and the input is the inscription of one of the tapes, all the others are blank. The read-write head of the nonblank tape scans the leftmost symbol of the input whereas all the other heads are positioned on arbitrary tape cells. Dependent on the current state and the currently scanned symbols on the k tapes, the Turing machine changes its state, rewrites the symbols at the head positions and moves the heads independently one cell to the left, one cell to the right or not at all. With an eye towards language recognition the machines have no extra output tape but the states are partitioned in accepting and rejecting states. More formally:

Definition 1 *A* deterministic Turing machine *with $k \in \mathbb{N}$ tapes* (DTM$_k$) *is a system $\langle S, T, A, \delta, s_0, F \rangle$, where*

1. *S is the finite set of* internal states,
2. *T is the finite set of* tape symbols *containing the* blank *symbol* $\sqcup$,
3. *$A \subseteq T$ is the set of* input symbols,
4. *$s_0 \in S$ is the* initial state,
5. *$F \subseteq S$ is the set of* accepting states,
6. *$\delta : S \times T^k \to S \times T^k \times \{-1, 0, 1\}^k$ is the* partial transition function.

The set of rejecting states is implicitly given by the partitioning, i.e. $S \setminus F$. The numbers -1, 0 and 1 correspond to the left, no and right moves of the read-write heads.

If the set of tape symbols is a Cartesian product of some smaller sets $T = T_1 \times T_2 \times \cdots \times T_l$ we will use the notion *register* for the single parts of a symbol. The concatenation of a register of all tape cells of a tape forms a *track*.

Let $\mathcal{M}$ be a DTM$_k$. A *configuration* of $\mathcal{M}$ at some time $t \geq 0$ is a description of its global state which is a $(2k+1)$-tuple $(s, f_1, \ldots, f_k, p_1, \ldots, p_k)$ where $s \in S$ is the current state, $f_i : \mathbb{Z} \to T$ is a function that maps the tape cells of the ith tape to their current contents, and $p_i \in \mathbb{Z}$ is the current position of the head of the ith tape, $1 \leq i \leq k$.

The initial configuration $(s_0, f_1, \ldots, f_k, 1, 0, \ldots, 0)$ at time 0 is defined by the input word $w = x_1 \cdots x_n \in A^*$, the initial state s_0 and blank working tapes:

$$f_1(m) = \begin{cases} x_m & \text{if } 1 \leq m \leq n \\ \sqcup & \text{otherwise} \end{cases}$$

$$f_i(m) = \sqcup \quad \text{if } 2 \leq i \leq k$$

Subsequent configurations are computed according to the *global transition function* Δ:

Let $(s, f_1, \ldots, f_k, p_1, \ldots, p_k)$ be a configuration and $\delta(s, f_1(p_1), \ldots, f_k(p_k))$ defined to be $(\tilde{s}, x_1, \ldots, x_k, d_1, \ldots, d_k)$. Then the successor configuration is as follows, $1 \leq j \leq k$:

$$(s', f_1', \ldots, f_k', p_1', \ldots, p_k') = \Delta\big((s, f_1, \ldots, f_k, p_1, \ldots, p_k)\big) \iff$$

$$s' = \tilde{s}$$

$$f_i'(m) = \begin{cases} f_i(m) & \text{if } m \neq p_i \\ x_i & \text{if } m = p_i \end{cases}$$

$$p_i' = p_i + d_i$$

Thus, the global transition function Δ is induced by δ.

Up to now it is supposed that the input is written on one of the k (working) tapes of a DTM$_k$. Often in the literature Turing machines with an additional write protected input tape are considered. Needless to say, if the input tape would not be write protected then we simply had $k + 1$ tapes.

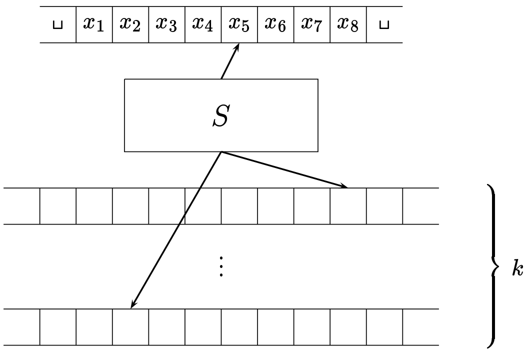

Figure 1: Turing machine with k working tapes and an input tape.

In the following we denote Turing machines with a write protected two-way input tape and $k \in \mathbb{N}$ working tapes by DTM$_{k:2}$. The write protection is realized by the definition of the transition function δ that now maps from $S \times (A \cup \{\sqcup\}) \times T^k$ to $S \times T^k \times \{-1, 0, 1\}^{k+1}$. Since the input tape cannot be rewritten we need no new symbol for its current tape cell. Due to the same fact, δ may only expect symbols from $A \cup \{\sqcup\}$ on the input tape.

A further restriction is a write protected one-way input tape (i.e. the input tape head is not allowed to move to the left). Such Turing machines with $k \in \mathbb{N}$ working tapes are denoted by DTM$_{k:1}$. Again the restriction is realized by the transition function that as in the previous case maps from $S \times (A \cup \{\sqcup\}) \times T^k$ now to $S \times T^k \times \{0, 1\} \times \{-1, 0, 1\}^k$.

The global transition functions for $\mathrm{DTM}_{k:2}$ and $\mathrm{DTM}_{k:1}$ are induced by the local ones in a straightforward manner. For consistency we often use the notation $\mathrm{DTM}_{k:0}$ instead of DTM_k.

The last class of Turing machines we are dealing with are the so-called multitape machines: $\mathrm{DTM} = \bigcup_{k\in\mathbb{N}} \mathrm{DTM}_k$

A Turing machine *halts* iff the transition function is undefined for the current configuration. An input word w is accepted by a Turing machine if the machine halts at some time in an accepting state, otherwise it is rejected.

Definition 2 *Let $\mathcal{M} = \langle S, T, A, \delta, s_0, F\rangle$ be a Turing machine.*

1. *A word $w \in A^*$ is* accepted *by $\mathcal{M}$ if $\mathcal{M}$ on input w halts at some time in an accepting state.*
2. *$L(\mathcal{M}) = \{w \in A^* \mid w$ is accepted by $\mathcal{M}\}$ is the* language accepted *by $\mathcal{M}$.*
3. *Let $t : \mathbb{N}_0 \to \mathbb{N}$, $t(n) \geq n+1$, be a function. A Turing machine is said to be* t-time-bounded *or of* time complexity t *iff it halts on every input of length n after at most $t(n)$ time steps.*

The family of all languages which can be accepted by $\mathrm{DTM}_{k:i}$ with time complexity t is denoted by $\mathsf{DTIME}_{k:i}(t)$. For multitape machines it holds $\mathsf{DTIME}(t) = \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}_k(t) = \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}_{k:2}(t) = \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}_{k:1}(t)$.

If t equals the function $id + 1$ acceptance is said to be in *real-time*. The *linear-time* languages are defined according to

$$\mathsf{DTIME}_{k:i}(\mathrm{LIN}) = \bigcup_{c\in\mathbb{Q}, c\geq 1} \mathsf{DTIME}_{k:i}(c \cdot id)$$

Since time complexities are mappings to positive integers and have to be greater than or equal to $id + 1$, actually, $c \cdot id$ means $\max\{\lceil c \cdot id\rceil, id + 1\}$. But for convenience we simplify the notation in the sequel.

In order to prove tight time hierarchies in almost all cases honest time bounding functions are required. Usually the notion "honest" is concretized in terms of computability or constructibility of the functions with respect to the device in question.

Definition 3 *Let $k \geq 1$. A function $f : \mathbb{N}_0 \to \mathbb{N}$ is said to be* DTM_k-time-constructible *iff there exists an $O(f)$-time-bounded DTM_k which for every $n \in \mathbb{N}_0$ on input 1^n writes the binary representation of $f(n)$ onto (one of) its working tape(s) and halts.*

Here a function f is called time-constructible if there exists a Turing machine that computes the binary representation of the value $f(n)$ from the unary representation of its argument n. Moreover, the machine has to be $O(f)$-time-bounded.

Another common definition of time-constructibility demands the existence of a Turing machine that halts after exactly $f(n)$ time steps when given the unary representation of the input n. Both definitions have been proven to be equivalent for multitape machines [8]. Since here we are also dealing with Turing machines with a fixed number of tapes and are naturally interested in rich families of constructible functions, we will use the next lemma for the proofs in the following sections.

Lemma 4 *Let $k \geq 1$ and $f : \mathbb{N}_0 \to \mathbb{N}$ be a DTM_k-time-constructible function. Then there exist a function $h : \mathbb{N}_0 \to \mathbb{N}$, $h \geq f$ and $h \in O(f)$ and a DTM_k which on input 1^n halts after exactly $h(n)$ time steps with its input head scanning the leftmost symbol of the input. The input is retained unchanged during the computation.*

Proof Let f be a DTM_k-time-constructible function and $\mathcal{M}$ be a witness for this fact. A DTM_k $\mathcal{M}'$ works as follows:

In a first phase $\mathcal{M}'$ simulates the constructor $\mathcal{M}$ whereby the input 1^n is conserved on an extra track. Subsequently, it moves the head of the tape that contains the binary representation of $f(n)$ to the tape cell containing the least significant bit of the representation. Up to this stage $\mathcal{M}'$ is $O(f)$-time-bounded since $\mathcal{M}$ is.

During a second phase $\mathcal{M}'$ generates successively the binary representations of $f(n)-1, f(n)-2, \ldots, 0$. Finally, it moves the head of the tape that contains the conserved input to the cell containing the leftmost symbol of the input and halts.

By calculating an upper bound for the number of moves it is easily verified that $\mathcal{M}'$ needs no more than $O(f(n))$ steps for successively decreasing the binary counter from $f(n)$ to 0. (Note that, for example, during every second decrementation only the least significant bit has to be changed. See e.g. [9] for further details.) It follows that $\mathcal{M}'$ obeys a time complexity of order $O(f)$. On the other hand, $\mathcal{M}'$ needs at least $f(n)$ time steps for decreasing the counter.

Now let for every $n \in \mathbb{N}_0$ the function $h(n)$ be defined as the running time of $\mathcal{M}'$ on input 1^n. Obviously, $h \geq f$ and $h \in O(f)$ what proves the lemma. $\square$

It is obvious that the lemma remains valid for all common definitions of time-constructibility and, therefore, our results are independent of a specific definition.

The following definition summarizes the properties of honest functions and names them.

Definition 5

1. *The set of all increasing, unbounded DTM_k-time-constructible functions f with the property $O(f(n)) \leq f(O(n))$ is denoted by $\mathscr{T}(\mathrm{DTM}_k)$.*
2. *The set of their inverses is $\mathscr{T}^{-1}(\mathrm{DTM}_k) = \{f^{-1} \mid f \in \mathscr{T}(\mathrm{DTM}_k)\}$.*

The properties increasing and unbounded are straightforward. At first glance the property $O(f(n)) \leq f(O(n))$ seems to be restrictive, but it is not. It is easily verified that almost all of the commonly considered time complexities have this property. As usual here we remark that at least for $k \geq 2$ the family $\mathscr{T}(\mathrm{DTM}_k)$ is very rich. More details can be found for example in [1, 12].

3 Hierarchies Between Real-Time and Linear-Time

In this section we will present our main results, time hierarchies between real-time and linear-time. Due to the small time bounds the devices under investigation are too weak for diagonalization. In order to separate complexity classes counting arguments are used. The following equivalence relation is well-known. At least implicitly it has been used several times in connection with real-time computations, e.g. in [6, 11] for Turing machines and in [3] for iterative arrays.

Definition 6 *Let $L \subseteq A^*$ be a language over an alphabet A and $l \in \mathbb{N}_0$ be a constant.*

1. *Two words w and w' are l-equivalent with respect to L if*

$$ww_l \in L \iff w'w_l \in L \text{ for all } w_l \in A^l$$

2. *$N(n, l, L)$ denotes the number of l-equivalence classes of words of length $n - l$ with respect to L (i.e. $|ww_l| = n$).*

The underlying idea is to bound the number of distinguishable equivalence classes. The following lemma gives a necessary condition for a language to be $(id + r)$-time acceptable by a DTM.

Lemma 7 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. If $L \in \mathsf{DTIME}(id + r)$ then there exists a constant $p \in \mathbb{N}$ such that*

$$N(n, l, L) \leq p^{l+r(n)}$$

Proof Let $\mathcal{M} = \langle S, T, A, \delta, s_0, F \rangle$ be a $(id + r)$-time DTM that accepts a language L.

In order to determine an upper bound to the number of l-equivalence classes we consider the possible situations of $\mathcal{M}$ after reading all but l input symbols. The remaining computation depends on the current internal state and the contents of the $2(l + r(n)) + 1$ reachable cells on each tape.

Let $p_1 = \max\{|T|, |S|\}$.

For the $2(l+r(n))+1$ cells per tape there are at most $p_1^{2(l+r(n))+1}$ different inscriptions. For some $k \in \mathbb{N}$ tapes we obtain altogether at most $p_1^{k(2l+2r(n)+1)+1}$ different situations what bounds the number of l-equivalence classes:

$$N(n,l,L) \leq p_1^{k(2l+2r(n)+1)+1}$$

The lemma follows for $p = p_1^{3k+1}$. □

Since a $\mathrm{DTM}_{k:i}$ has at most $k+1$ tapes and the previous lemma holds for multitape machines and, thus, for arbitrary k, it follows immediately:

Corollary 8 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function, $k \geq 1$ and $i \in \{0,1,2\}$. If $L \in \mathsf{DTIME}_{k:i}(id + r)$ then there exists a constant $p \in \mathbb{N}$ such that*

$$N(n,l,L) \leq p^{l+r(n)}$$

From the next theorem the hierarchies for all but $\mathrm{DTM}_{1:1}$ are derived. Moreover, it says that the additional time needed in order to obtain a strict superclass cannot be compensated by any number of additional tracks. Any time-constructible function which is not constant would be greater than or equal to id, but since here we are interested in sublinear functions r, the inverses of the honest functions are used.

Theorem 9 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ and $r' : \mathbb{N}_0 \to \mathbb{N}$ be two functions and $k \geq 1$. If $r \in \mathscr{T}^{-1}(\mathrm{DTM}_k)$ and $r' \in o(r)$ then*

$$\mathsf{DTIME}_{k:2}(id + r) \setminus \mathsf{DTIME}(id + r') \neq \emptyset$$

Proof The first part of the proof is to define a witness language for the assertion.

Since $r \in \mathscr{T}^{-1}(\mathrm{DTM}_k)$ there exists a function $f_r \in \mathscr{T}(\mathrm{DTM}_k)$ such that $r = f_r^{-1}$. Due to Lemma 4 one can always find a function $h_r \in O(f_r)$, $h_r \geq f_r$, and a DTM_k $\mathcal{C}$ such that $\mathcal{C}$ when given the unary representation of $n \in \mathbb{N}$ runs for exactly $h_r(n)$ time steps and halts on the first symbol of its preserved input.

Now we are prepared to define the language dependent on h_r (and thus on r):

$$\begin{aligned} L_{h_r} = \Big\{ \mathsf{a}^{2m}\mathsf{b}^{h_r(2m)} w_1 \$ w_2 \$ \cdots \$ w_l ¢ y ¢ \mid\ & l, m \in \mathbb{N} \\ & \land\ y, w_i \in \{\mathtt{0},\mathtt{1}\}^+, 1 \leq i \leq l \\ & \land\ \exists\, j \in \{1,\ldots,l\} : y = w_j \\ & \land\ |w_1 \$ w_2 \$ \cdots \$ w_l ¢ y ¢| = m \Big\} \end{aligned}$$

In order to complete the proof $L_{h_r} \in \mathsf{DTIME}_{k:2}(id + r)$ and $L_{h_r} \notin \mathsf{DTIME}(id + r')$ has to be shown what will be done by Lemma 11 and Lemma 12. □

Before presenting the proofs a taste of the hierarchies is given that is based on natural functions:

Example 10 Since $\mathscr{T}(\mathrm{DTM}_2)$ is closed under composition and contains 2^{id} and id^c, $c \geq 1$, the functions $\log^{[i]}$, $i \geq 1$, and $\sqrt[c]{id}$ are belonging to $\mathscr{T}^{-1}(\mathrm{DTM}_2)$. (Actually, the inverses of 2^{id} and id^c are $\lceil \log \rceil$ and $\lceil id^{\frac{1}{c}} \rceil$ but as mentioned before we simplify the notation for convenience.) Therefore, an application to the hierarchy theorem yields

$$\begin{aligned}&\mathsf{DTIME}_{2:2}(id+1) \subset \cdots \\ &\qquad\cdots \subset \mathsf{DTIME}_{2:2}(id+\log^{[i+1]}) \subset \mathsf{DTIME}_{2:2}(id+\log^{[i]}) \subset \cdots \\ &\qquad\qquad \cdots \subset \mathsf{DTIME}_{2:2}(\mathrm{LIN})\end{aligned}$$

and

$$\begin{aligned}&\mathsf{DTIME}_{2:2}(id+1) \subset \cdots \\ &\qquad\cdots \subset \mathsf{DTIME}_{2:2}(id+id^{\frac{1}{i+1}}) \subset \mathsf{DTIME}_{2:2}(id+id^{\frac{1}{i}}) \subset \cdots \\ &\qquad\qquad \cdots \subset \mathsf{DTIME}_{2:2}(\mathrm{LIN})\end{aligned}$$

or in combinations e.g.,

$$\begin{aligned}&\mathsf{DTIME}_{2:2}(id+1) \subset \cdots \\ &\qquad\cdots \subset \mathsf{DTIME}_{2:2}(id+(\log^{[j+1]})^{\frac{1}{i+1}}) \subset \mathsf{DTIME}_{2:2}(id+(\log^{[j+1]})^{\frac{1}{i}}) \subset \cdots \\ &\qquad\cdots \subset \mathsf{DTIME}_{2:2}(id+(\log^{[j]})^{\frac{1}{i+1}}) \subset \mathsf{DTIME}_{2:2}(id+(\log^{[j]})^{\frac{1}{i}}) \subset \cdots \\ &\qquad\qquad \cdots \subset \mathsf{DTIME}_{2:2}(\mathrm{LIN})\end{aligned}$$

The next part of the proof of Theorem 9 shows by construction that the language L_{h_r} is acceptable by a $\mathrm{DTM}_{k:2}$.

Lemma 11 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function and $k \geq 1$ such that $r \in \mathscr{T}^{-1}(\mathrm{DTM}_k)$ then*

$$L_{h_r} \in \mathsf{DTIME}_{k:2}(id + r)$$

Proof In what follows a $\mathrm{DTM}_{k:2}$ $\mathcal{M}$ is constructed that accepts L_{h_r} with time complexity $id + r$. Since $\mathsf{DTIME}_{k:2}(id + r)$ is closed under intersection with regular sets we may restrict our considerations to inputs of the form

$$\mathtt{a}^+\mathtt{b}^+\big(\{\mathtt{0},\mathtt{1}\}^+\$\big)^*\{\mathtt{0},\mathtt{1}\}^+ ¢\{\mathtt{0},\mathtt{1}\}^+ ¢$$

Lets say $w = \mathtt{a}^p\mathtt{b}^q u ¢ y ¢$ where $u = w_1\$w_2\$\cdots\$w_l$ for $p, q, l \geq 1$ and $w_1, \cdots, w_l, y \in \{\mathtt{0},\mathtt{1}\}^+$.

$\mathcal{M}$ is designed to perform three tasks sequentially. The first one is to copy the $\mathtt{a}$'s onto a working tape and to check whether the number of $\mathtt{b}$'s is correct with respect to the number of $\mathtt{a}$'s. During the second task it is verified that $p = 2\cdot|u¢y¢|$. Finally, the third task is to ensure that $w_j = y$ for some $1 \leq j \leq l$. The input is accepted if and only if all tasks succeed.

In this case if we set $m = |u¢y¢|$ we have $p = 2\cdot|u¢y¢| = 2\cdot m$ and $q = h_r(p) = h_r(2m)$ and, hence, $w \in L_{h_r}$.

Task 1 $\mathcal{M}$ starts its computation with blank working tapes. During its first p time steps $\mathcal{M}$ copies the $\mathtt{a}$'s from its input tape onto the first working tape whereby each two $\mathtt{a}$'s are written in one tape cell. Subsequently it simulates the DTM_k $\mathcal{C}$ on its k working tapes. During the simulation one input symbol $\mathtt{b}$ is read at each time step.

The task succeeds if the simulation stops at that time step the input head has moved out of the $\mathtt{b}$'s, i.e. $q = h_r(p)$ has been verified. Due to Lemma 4 the head of $\mathcal{M}$'s first working tape is located again at the first symbol of its preserved input such that the $\mathtt{a}$'s are still available on the first tape.

Task 1 requires $p + q = p + h_r(p)$ time steps.

Task 2 This task starts with the input head located at the first symbol of the subword u. $\mathcal{M}$ copies the remaining input on a second track of its first working tape. Since by construction the $\mathtt{a}$'s are 2-fold compressed available on the first track it is easily verified that $2 \cdot |u¢y¢|$ equals p.

The time needed for task 2 is $|u¢y¢| = \frac{p}{2}$.

If Task 1 and Task 2 succeed it holds $|w| = p + h_r(p) + \frac{p}{2}$. For $m = \frac{p}{2}$ this is $|w| = 2m + h_r(2m) + m$.

Task 3 The last task starts with the heads of the first working tape and input tape located at the right hand end of the inscriptions $w_1\$ \cdots \$w_l¢y¢$, respectively. During the next $|y|+1$ time steps the working tape head moves back to the separating symbol $¢$ between w_l and y. Subsequently, y is compared with w_l symbolwise from right to left. Afterwards the head of the input tape is moved back again to the symbol $¢$ following y. The comparison process is now repeated for $w_{l-1}, \ldots, w_2$ and w_1.

The head of the input tape moves back and forth over the inscription y. It may move back when all the symbols of y are compared with symbols of w_i or vice versa. This needs $2 \cdot \min\{|w_i|+1, |y|+1\}$ time steps. Additionally $\max\{|w_i| - 2|y| - 1, 0\}$ time steps could be needed by the head of the first working tape for passing over remaining leading symbols of w_i. Altogether the comparison process needs

$$\sum_{i=l}^{1} 2 \cdot \min\{|w_i| + 1, |y| + 1\} + \max\{|w_i| - 2|y| - 1, 0\}$$

time steps. In order to resolve the sum for each $1 \leq i \leq l$ we are concerned with three cases:

1.	$2 \cdot (\|y\|+1) \leq \|w_i\|+1$:	$2 \cdot (\|y\|+1) + \|w_i\| - 2\|y\| - 1 = \|w_i\|+1$
2.	$\|y\|+1 \leq \|w_i\|+1 < 2 \cdot (\|y\|+1)$:	$2 \cdot (\|y\|+1) + 0 \leq 2 \cdot (\|w_i\|+1)$
3.	$\|w_i\|+1 < \|y\|+1$:	$2 \cdot (\|w_i\|+1) + 0 = 2 \cdot (\|w_i\|+1)$

Thus, the comparison process needs at most $2 \cdot (|w_i|+1)$ time steps for each w_i and at most $2 \cdot |w_1\$\cdots\$w_l¢|$ time steps in total.

The task succeeds if one of the w_i matches y. Altogether it requires at most

$$|y| + 1 + 2 \cdot |w_1\$\cdots\$w_l¢| \leq 2 \cdot |w_1\$\cdots\$w_l¢y¢| = 2 \cdot m$$

time steps.

At the end of task 2 $\mathcal{M}$ has read the whole input w exactly one symbol per time step. Due to the verifications $|w|$ is known to be $2m + h_r(2m) + m$ and thus for task 3 there is still $r\big(3m + h_r(2m)\big)$ time. Since f_r belongs to $\mathscr{T}(\mathrm{DTM}_k)$ it is increasing and therefore $r = f_r^{-1}$ is increasing, too. We obtain

$$r\big(3m + h_r(2m)\big) \geq r\big(h_r(2m)\big)$$

By Lemma 4 and construction it holds $h_r \geq f_r$. It follows

$$r\big(h_r(2m)\big) \geq r\big(f_r(2m)\big) = f_r^{-1}\big(f_r(2m)\big) = 2 \cdot m$$

We conclude that the time complexity $id + r$ is obeyed by $\mathcal{M}$ and thus $L_{h_r} \in \mathsf{DTIME}_{k:2}(id + r)$. □

Now the lemma that bounds the number of distinguishable equivalence classes is applied. By proving that L_{h_r} induces more equivalence classes than are distinguishable by any multitape Turing machine (with respect to the given time bound) the last part of the proof of Theorem 9 is shown.

Lemma 12 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ and $r' : \mathbb{N}_0 \to \mathbb{N}$ be two functions and $k \geq 1$. If $r \in \mathscr{T}^{-1}(\mathrm{DTM}_k)$ and $r' \in o(r)$ then*

$$L_{h_r} \notin \mathsf{DTIME}(id + r')$$

Proof Contrarily assume L_{h_r} is acceptable by some $\mathrm{DTM}_{k'}$ $\mathcal{M}$ with time complexity $id + r'$.

We consider words $\mathtt{a}^{2m}\mathtt{b}^{h_r(2m)}w_1\$w_2\$\cdots\$w_l¢y¢$ from L_{h_r} such that $|w_1| = \cdots = |w_l| = |y| = l$ and, therefore, we have $m = (l+1)^2$. The situation at time $n - (l+1)$ where n denotes the length of the input word is as follows. Two words

$$\mathtt{a}^{2m}\mathtt{b}^{h_r(2m)}w_1\$\cdots\$w_l¢ \quad \text{and} \quad \mathtt{a}^{2m}\mathtt{b}^{h_r(2m)}w_1'\$\cdots\$w_l'¢$$

are $(l+1)$-equivalent with respect to L_{h_r} if and only if the sets $\{w_1,\ldots,w_l\}$ and $\{w'_1,\ldots,w'_l\}$ are equal. There are exactly $\binom{2^l}{l}$ different subsets of $\{0,1\}^l$ with l elements. It follows:

$$\begin{aligned}
N(n,l+1,L_{h_r}) &= N\big(3m+h_r(2m),l+1,L_{h_r}\big)\\
&\geq \binom{2^l}{l} > \left(\frac{2^l-l}{l}\right)^l\\
&\geq \left(\frac{2^{\frac{l}{2}}}{l}\right)^l = \left(2^{\frac{l}{2}-\log(l)}\right)^l\\
&> 2^{\frac{l^2}{4}} = 2^{\Omega(l^2)}
\end{aligned}$$

for all sufficiently large l.

On the other hand, by Lemma 7 the number $N(n,l+1,L_{h_r})$ of equivalence classes distinguishable by $\mathcal{M}$ is bounded by $p^{l+1+r'(n)}$ for a constant $p\in\mathbb{N}$:

$$\begin{aligned}
N(n,l+1,L_{h_r}) &= N\big(3m+h_r(2m),l+1,L_{h_r}\big)\\
&= N\big((3(l+1)^2+h_r\big(2(l+1)^2\big),l+1,L_{h_r}\big)\\
&\leq p^{l+1+r'(3(l+1)^2+h_r(2(l+1)^2))}
\end{aligned}$$

Define $r''(n+1)=\max\{r'(n+1),r''(n)\}$. Obviously, $r'\leq r''$ and we obtain

$$\leq p^{l+1+r''(3(l+1)^2+h_r(2(l+1)^2))}$$

Since f_r belongs to $\mathscr{T}(\mathrm{DTM}_k)$ it is increasing and unbounded and it holds $f_r\geq id$. By construction we have $h_r\geq f_r$ and $h_r\in O(f_r)$ and, thus, $h_r\geq id$. By definition r'' is increasing. We conclude

$$\begin{aligned}
&\leq p^{l+1+r''(4\cdot h_r(2(l+1)^2))}\\
&\leq p^{l+1+r''(4\cdot c_1\cdot f_r(2(l+1)^2))}, \text{ for some } c_1\in\mathbb{N}\\
&= p^{l+1+r''(c_2\cdot f_r(2(l+1)^2))}, \text{ for some } c_2\in\mathbb{N}
\end{aligned}$$

From $r=f_r^{-1}$ it follows that r is increasing. By $r'\in o(r)$ and the construction of r'' we conclude $r''\in o(r)$. Furthermore, we know $O(f_r(n))\leq f_r(O(n))$. Thus

$$\begin{aligned}
&\leq p^{l+1+r''(f_r(c_3\cdot(l+1)^2))}, \text{ for some } c_3\in\mathbb{N}\\
&= p^{l+1+o(r(f_r(c_3\cdot(l+1)^2)))}\\
&= p^{l+1+o(c_3\cdot(l+1)^2)}
\end{aligned}$$

$$
\begin{aligned}
&= p^{l+1+o((l+1)^2)} \\
&= p^{o((l+1)^2)} \\
&= p^{o(l^2)} \\
&= 2^{o(l^2)}
\end{aligned}
$$

Now we have the contradiction that previously $N(n, l+1, L_{h_r})$ has been calculated to be at least $2^{\Omega(l^2)}$ what proves the lemma. $\square$

The inclusions

$$\mathsf{DTIME}_{k:i}(id + r') \subseteq \mathsf{DTIME}_{k:i}(id + r) \quad \text{and} \quad \mathsf{DTIME}(id + r') \subseteq \mathsf{DTIME}(id + r)$$

are trivial for $r' \leq r$. Applications of Theorem 9 yield the hierarchies:

Corollary 13 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ and $r' : \mathbb{N}_0 \to \mathbb{N}$ be two functions. If $r \in \mathscr{T}^{-1}(\mathrm{DTM}_k)$ and $r' \in o(r)$ then*

$$\mathsf{DTIME}_{k:2}(id + r') \subset \mathsf{DTIME}_{k:2}(id + r), \ k \geq 1$$

$$\mathsf{DTIME}_{k:1}(id + r') \subset \mathsf{DTIME}_{k:1}(id + r), \ k \geq 2$$

Proof The strictness of the first assertion has been shown by Lemma 11 and Lemma 12.

Observe that in the proof of Lemma 11 Task 2 and Task 3 do not use the working tapes $2, \ldots, k$. Since for the second assertion k has to be at least 2, Task 2 can be modified such that the subword $u¢y¢$ is additionally copied onto the second working tape. Subsequently the second working tape simulates the two-way input tape in a straightforward manner. $\square$

Corollary 14 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ and $r' : \mathbb{N}_0 \to \mathbb{N}$ be two functions and $k \geq 2$. If $r \in \mathscr{T}^{-1}(\mathrm{DTM}_{k-1})$ and $r' \in o(r)$ then*

$$\mathsf{DTIME}_k(id + r') \subset \mathsf{DTIME}_k(id + r)$$

Proof Here the working tape containing the input has to simulate the two-way input tape. Therefore, only the remaining $k - 1$ tapes are available for the simulation of the time-constructor. $\square$

Corollary 15 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ and $r' : \mathbb{N}_0 \to \mathbb{N}$ be two functions. If $r \in \mathscr{T}^{-1}(\mathrm{DTM})$ and $r' \in o(r)$ then*

$$\mathsf{DTIME}(id + r') \subset \mathsf{DTIME}(id + r)$$

There are two cases for which Theorem 9 does not yield a hierarchy: DTM_1 and $\mathrm{DTM}_{1:1}$. The first one is trivial. By the results in [7] $\mathsf{DTIME}_1(id) = \mathsf{DTIME}_1(\mathrm{LIN})$ is known and, thus, there is no hierarchy between real-time and linear-time.

The one-tape Turing machines with one-way input tape are too weak to accept the language L_{h_r} in $(id + r)$-time. So Lemma 11 does not hold for $\mathsf{DTIME}_{1:1}(id + r)$. But nevertheless a hierarchy can be proven if the time for the acceptance is slightly increased.

Theorem 16 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ and $r' : \mathbb{N}_0 \to \mathbb{N}$ be two functions. If $r \in \mathscr{T}^{-1}(\mathrm{DTM}_1)$ and $r' \in o(r)$ then*

$$\mathsf{DTIME}_{1:1}(id + r') \subset \mathsf{DTIME}(id + r^{\frac{3}{2}})$$

Proof The language L_{h_r} of Theorem 9 is slightly modified as follows:

$$L'_{h_r} = \Big\{ \mathtt{a}^{2m}\mathtt{b}^{h_r(2m)} w_1 \$ w_2 \$ \cdots \$ w_l ¢ y ¢ \mid l, m \in \mathbb{N} \wedge y, w_i \in \{\mathtt{0}, \mathtt{1}\}^l, 1 \le i \le l$$
$$\wedge\, \exists\, j \in \{1, \dots, l\} : y = w_j \wedge |w_1 \$ w_2 \$ \cdots \$ w_l ¢ y ¢| = m \Big\}$$

The difference between L_{h_r} and L'_{h_r} is that in the suffixes $w_1\$w_2\$\cdots\$w_l¢y¢$ of the words in L'_{h_r} are as many subwords w_i as there are symbols in the subword y, and that $w_1, \dots, w_l$ and y are of the same length. Thus, for $|y| = l$ it holds $|w_1\$w_2\$\cdots\$w_l¢y¢| = (l+1)^2$. Moreover, we have $L'_{h_r} \subset L_{h_r}$. Since the words in L'_{h_r} have been used to prove Lemma 12 it follows immediately $L'_{h_r} \notin \mathsf{DTIME}_{1:1}(id + r')$.

It remains to show $L'_{h_r} \in \mathsf{DTIME}_{1:1}(id + r^{\frac{3}{2}})$. In order to construct an appropriate $\mathrm{DTM}_{1:1}$ $\mathcal{M}$ we only have to modify Task 3 of Lemma 11 as follows.

Task 3' The task starts with the heads of the working tape and the input tape located at the right hand end of the inscriptions $w_1\$\cdots\$w_l¢y¢$ as Task 3 does. Since the input tape is one-way the remaining computations are on the working tape.

The head of the working tape sweeps back and forth over its inscription $w_1\$\cdots\$w_l¢y¢$ whereby the subword y is symbolwise copied to the subwords w_i and the number of subwords is checked as follows.

During a right to left sweep $\mathcal{M}$ marks the rightmost non-marked symbol of y and copies this (on an additional track) onto the rightmost empty register of each subword w_i. Additionally during the sweep, the rightmost non-marked separating symbol ($\$$ or the $¢$ between w_l and y) is marked.

During a left to right sweep the tape content is not rewritten.

Suppose now the input belongs to L'_{h_r}. During its last right to left sweep ($\mathcal{M}$ can detect when it has marked the leftmost symbol of y by its position next to the separating symbol ¢) $\mathcal{M}$ can check whether all of the separating symbols are marked, whether the lengths of all of the subwords w_i are equal to the length of y, and whether one of the subwords w_i matches its copy of y. Thus, whether $|w_1| = \cdots = |w_l| = |y| = l$ and $w_i = y$ for some $1 \leq i \leq l$.

Let $|w_1\$\cdots\$w_l¢y¢| = m$ then Task 3' needs less than $2 \cdot m \cdot l$ time steps. By $m = (l+1)^2 \geq l^2$ we conclude less than $2 \cdot m \cdot \sqrt{m} \leq (2m)^{\frac{3}{2}}$ time steps.

Suppose now the input does not belong to L'_{h_r}. If the lengths of the subwords are correct but none of the w_i matches y the time for Task 3' is again less than $(2m)^{\frac{3}{2}}$.

If the lengths are not correct, i.e. if $\mathcal{M}$ cannot find an unmarked separating symbol, or if $\mathcal{M}$ cannot find an empty register for at least one of the subwords w_i, or if on its last right to left sweep there remain empty registers or unmarked separating symbols, then $\mathcal{M}$ rejects the input immediately.

In these cases let $\mathcal{M}$ have performed $j-1$ sweeps successfully, $j \geq 1$. Then Task 3' needs less than $j \cdot 2 \cdot m$ time steps. Due to the successful sweeps there have to exist at least $j-1$ subwords w_i each of which have to consist of at least $j-1$ symbols. Together with the subword y and the separating symbols it follows $m \geq j^2$. Thus, Task 3' needs less than $2 \cdot m \cdot \sqrt{m} \leq (2m)^{\frac{3}{2}}$ time steps even if the input is rejected.

Recalling the final arguments of the proof of Lemma 11 for Task 3' there is $\big(r(3m + h_r(2m))\big)^{\frac{3}{2}}$ time. Since $r(3m + h_r(2m))$ has shown to be greater than $2m$ there is at least $(2m)^{\frac{3}{2}}$ time. We conclude that the time complexity $id + r^{\frac{3}{2}}$ is obeyed by $\mathcal{M}$ and hence $L'_{h_r} \in \mathsf{DTIME}_{1:1}(id + r^{\frac{3}{2}})$. □

4 Quality of the Hierarchies and Speed-up

This section is devoted to the question whether or not the presented hierarchies might be more refined. A refinement would necessarily require a weaker hypothesis.

Due to the well-known gap theorem we cannot relax the constructibility of the function r^{-1}. On the other hand, since the proof of the hierarchies uses actually Lemma 4 that in turn is provable with several different notions of time-constructibility, we have a very weak constructibility condition.

Now we take a closer look at the second hypothesis $r' \in o(r)$. Is it necessary that r' grows strictly less than r? Or is it possible to separate the complexity classes even under the condition $r' \leq \varepsilon \cdot r$ for some $0 < \varepsilon < 1$? In order to disprove the latter condition we are going to show a speed-up result that allows to speed-up the time beyond id linearly. Note that the widely known theorem which allows to speed-up from t to $id+\varepsilon\cdot t$, $\varepsilon>0$, does not help for time bounds of the form $id+r$, $r\in o(id)$. In such cases an application would yield a slow-down to linear-time.

In the following we consider Turing machines with one-way input tape. A speed-up from $id+r$ to $id+\varepsilon\cdot r$, $\varepsilon>0$, $r\in o(id)$, has to cope with the situation that only time steps at which no input symbol is read can be sped-up and, moreover, that these time steps might alternate with steps at which an input symbol is consumed. Therefore, a fast machine has to simulate two steps of a slow machine within *exactly one* step.

Theorem 17 *Let $r:\mathbb{N}_0\to\mathbb{N}$ be a function, $k\geq 1$ and $\varepsilon>0$. Then*

$$\mathsf{DTIME}_{k:1}(id+r)=\mathsf{DTIME}_{k:1}(id+\varepsilon\cdot r)$$

Proof Let $\mathcal{M}=\langle S,T,A,\delta,s_0,F\rangle$ be a $\mathrm{DTM}_{k:1}$ with time complexity $id+r$. We are going to construct a $\mathrm{DTM}_{k:1}$ $\mathcal{M}'$ that accepts $L(\mathcal{M})$ with time complexity $id+\varepsilon\cdot r$. The construction is shown for $\varepsilon'=\frac{1}{2}$ but can be iterated i times until $\frac{1}{2^i}\leq\varepsilon$. Since all k working tapes are handled identically it suffices w.l.o.g. to prove the theorem for $k=1$.

Basically, $\mathcal{M}'$ simulates the working tape of $\mathcal{M}$ 2-fold compressed (i.e. $\mathcal{M}'$ stores each two tape symbols of $\mathcal{M}$ into one tape cell). Since initially the working tape is blank $\mathcal{M}'$ needs no extra time to compress any tape inscription. Let us call the two tape symbols stored in one tape cell a block.

At every time step $\mathcal{M}'$ is designed to store one of the blocks internally as part of its state. The internal block does not appear on the working tape but the head of $\mathcal{M}'$ scans a cell containing one of the two possible neighboring blocks.

Another part of the internal state of $\mathcal{M}'$ remembers the currently scanned tape cell of $\mathcal{M}$. The crucial point is to construct $\mathcal{M}'$ such that this cell is always one of the two possible block components that are next to the border of the blocks. So we have to deal with four different situations as depicted in Figure 2.

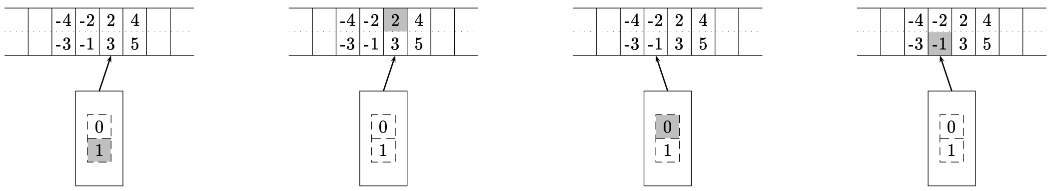

Figure 2: Four possible situations during a speed-up simulation. The gray shaded component indicates the currently scanned tape cell of the simulated machine.

Formally, $\mathcal{M}'=\langle S',T',A,\delta',s_0',F'\rangle$ is defined as follows: $S'=S\times T^2\times\{r,l\}\times\{i,e\}$ where $s\in S$ tracks the current state of $\mathcal{M}$, T^2 is for the internal block, r resp. l

indicates that the currently scanned block is the right resp. left neighboring block (of the internal block) and i resp. e indicates whether the internal resp. external border component is marked to be the currently scanned cell of $\mathcal{M}$. $T' = T^2$, $s_0' = \big(s_0, (\sqcup, \sqcup), r, i\big)$ and $F' = F \times T^2 \times \{r, l\} \times \{i, e\}$.

Due to the mechanism of tracking the current cell of $\mathcal{M}$, during two time steps of $\mathcal{M}$ only the contents of the internal and the currently scanned block of $\mathcal{M}'$ have to be rewritten. Obviously, this can be done by $\mathcal{M}'$ in one step.

During two steps $\mathcal{M}$ can move its head two cells to the right or left, one cell to the right or left or not at all. Correspondingly, we have to define δ' for these five possibilities with respect to the four situations of Figure 2. It remains to show that in any case the successor situation is again one of the situations of Figure 2. The formal definitions are tedious and hard to read. Exemplarily, we present the five successor situations of the leftmost situation in Figure 3.

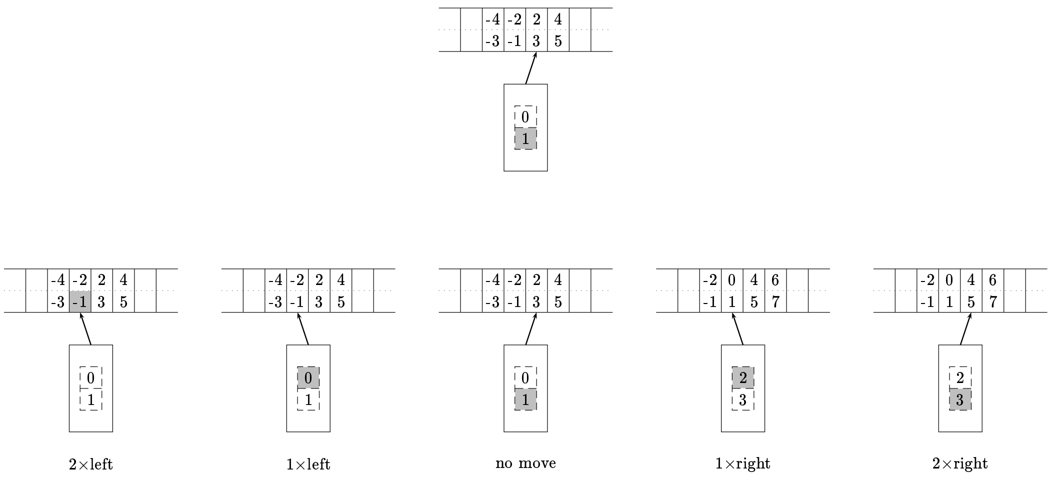

Figure 3: Five possible successor situations of the situation at the top.

By construction $\mathcal{M}'$ is able to simulate two steps of $\mathcal{M}$ in exactly one time step if $\mathcal{M}$ does not consume an input symbol during the first of the steps. Otherwise $\mathcal{M}'$ simulates only one time step of $\mathcal{M}$.

Since the input tape of $\mathcal{M}$ is one-way the number of time steps at which no speed-up is possible is bounded by id. $\square$

It is evident that the previous proof does not hold for Turing machines without a separate or with a two-way input tape. In these cases the head of the tape containing the input may move at every time step and therefore at no time step at all a speed-up would be possible. But nevertheless, we can cope with this problem by adding an extra tape.

Corollary 18 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function, $k \geq 1$ and $\varepsilon > 0$. Then*

$$\mathsf{DTIME}_{k:2}(id + r) \subseteq \mathsf{DTIME}_{k+1:2}(id + \varepsilon \cdot r)$$

and

$$\mathsf{DTIME}_{k}(id + r) \subseteq \mathsf{DTIME}_{k+1}(id + \varepsilon \cdot r)$$

Proof In order to prove the corollary we need to show the inclusion $\mathsf{DTIME}_k(t) \subseteq \mathsf{DTIME}_{k:1}(t)$ for $k \in \mathbb{N}$ and arbitrary functions $t : \mathbb{N}_0 \to \mathbb{N}$. What makes the inclusion less obvious is the fact that a $\mathrm{DTM}_{k:1}$ fetches its input from a restricted tape whereas a DTM_k is allowed to operate unrestricted on all its tapes.

The inclusion becomes obvious by the following construction. A $\mathrm{DTM}_{k:1}$ $\mathcal{M}'$ that simulates a given DTM_k $\mathcal{M}$ uses its tapes $2, \ldots, k$ exactly as $\mathcal{M}$ does. In order to be able to operate on the input like $\mathcal{M}$, $\mathcal{M}'$ copies the input to its (initially blank) first tape. Since $\mathcal{M}'$ may not waste time for the copying process it copies the symbols on demand:

Whenever the head on the first tape scans the first blank tape cell at the right of the nonblank inscription an input symbol is read. The subsequent write operation is onto the first tape. When the head of the first tape scans a nonblank cell then the cell's content is used and rewritten without reading a symbol from the input tape.

Now the corollary follows from Theorem 17 by some trivial inclusions:

$$\begin{aligned}
\mathsf{DTIME}_{k:2}(id + r) &\subseteq \mathsf{DTIME}_{k+1}(id + r) \\
&\subseteq \mathsf{DTIME}_{k+1:1}(id + r) = \mathsf{DTIME}_{k+1:1}(id + \varepsilon \cdot r) \\
&\subseteq \mathsf{DTIME}_{k+1:2}(id + \varepsilon \cdot r)
\end{aligned}$$

and

$$\begin{aligned}
\mathsf{DTIME}_{k}(id + r) &\subseteq \mathsf{DTIME}_{k:1}(id + r) = \mathsf{DTIME}_{k:1}(id + \varepsilon \cdot r) \\
&\subseteq \mathsf{DTIME}_{k:2}(id + \varepsilon \cdot r) \\
&\subseteq \mathsf{DTIME}_{k+1}(id + \varepsilon \cdot r)
\end{aligned}$$

$\square$

Essentially, from the proof of the corollary we obtain a stronger result. A speed-up is possible if we add a one-way input tape to a DTM_k or if we add an extra working tape to a $\mathrm{DTM}_{k:2}$ though the two-way input tape can be replaced by a one-way one. For multitape Turing machines it follows immediately:

Corollary 19 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function and $\varepsilon > 0$. Then*

$$\mathsf{DTIME}(id + r) = \mathsf{DTIME}(id + \varepsilon \cdot r)$$

Back to the question at the beginning of the section the speed-up results have shown that the hypothesis $r' \leq \varepsilon \cdot r$ for some $\varepsilon > 0$ is not strong enough to obtain hierarchies of separated complexity classes. We conclude that those of the presented hierarchies where a speed-up is possible are in some sense optimal.

5 Closure Properties

Besides the fact that closure properties can shed some light on the structure of a complexity class they may be used as powerful reduction tool in order to simplify proofs or constructions. It will turn out that the complexity classes under investigation have weak closure properties.

Lemma 20 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function, $k \geq 1$ and $i \in \{0, 1, 2\}$. Then*

$$\mathsf{DTIME}_{k:i}(id + r) \text{ is closed under complement.}$$

Proof Since a $\mathrm{DTM}_{k:i}$ $\mathcal{M}$ works deterministically it suffices to define F' to be $S \setminus F$ in order to construct a $\mathrm{DTM}_{k:i}$ that accepts the complement of $L(\mathcal{M})$. □

The closure under complement is the only known closure of $\mathsf{DTIME}_{k:i}(id + r)$ under Boolean operations. It is an open problem whether or not these classes are closed under union or intersection, but they are closed under union and intersection with regular sets.

Lemma 21 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function, $k \geq 1$ and $i \in \{0, 1, 2\}$. Then*

$$\mathsf{DTIME}_{k:i}(id + r) \text{ is closed under union and intersection with regular sets.}$$

Proof The principle is not surprising. A $\mathrm{DTM}_{k:i}$ $\mathcal{M}'$ simulates the given $\mathrm{DTM}_{k:i}$ $\mathcal{M}$ and a finite automaton in parallel and decides dependent on the results of both simulations.

If $\mathcal{M}$ has a one-way input tape the simulation of the finite automaton is rather simple but for two-way input tapes we have to take account of left moves of the input tape head. Since the tape is write protected (for $i = 2$) it is not possible to mark the corresponding position on the tape in order to continue the simulation when the head reaches the mark again.

Instead, in some sense, a reversible finite automaton has to be simulated. Let $\mathcal{F} = \langle S, A, \delta, s_0, F \rangle$ be a deterministic finite automaton with internal states S, input symbols A, initial state s_0, accepting states F, and transition function $\delta : S \times A \to S$.

For the reversible automaton $\mathcal{F}' = \langle S', A, \delta_,', \delta_r', s_0', F' \rangle$ we provide two transition functions. $\delta' : S' \times A \to S'$ is applied if the input tape head moves to the right and $\delta_r' : S' \times A \to S'$ if it moves to the left. If no move occurs or if the head scans blank cells at the left of the input area then no transition is simulated. The simulation is stopped when the head moves for the first time to a cell at the right of the input area.

Define $S' = 2^S \times \{l, r\}$ where 2^S denotes the powerset of S and r resp. l indicates that the last move was a right resp. left move, $s_0' = \big(\{s_0\}, r\big)$, $F' = \big\{(\{s\}, r) \mid s \in F\big\}$, and δ' and δ_r' as follows. For all $Q \in 2^S$ and $a \in A$:

$$\begin{aligned}
\delta'\big((Q,r),a\big) &= \big(\{s \in S \mid \exists\, s' \in Q : \delta(s',a) = s\}, r\big) \\
\delta'\big((Q,l),a\big) &= (Q,r) \\
\delta_r'\big((Q,l),a\big) &= \big(\{s' \in S \mid \delta(s',a) \in Q\}, l\big) \\
\delta_r'\big((Q,r),a\big) &= (Q,l)
\end{aligned}$$

If $\mathcal{F}'$ would be simulated only with right moves then by construction it would behave precisely as $\mathcal{F}$ does.

Suppose $\mathcal{F}'$ is in a state $(Q,r) \in S'$ with its input head located at some tape cell i. It is easily proved by induction that after a sequence of moves to cells located at the left hand side of i, $\mathcal{F}'$ is again in the state (Q,r) when its input head again is located at the tape cell i.

Combining these two observations it follows that $\mathcal{F}'$ accepts precisely the language $L(\mathcal{F})$. □

The closure under union and intersection is settled for multitape Turing machines:

Lemma 22 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. Then*

$\mathsf{DTIME}(id + r)$ *is closed under complement, union and intersection.*

Proof The closure under complement is proved analogously to lemma 20. Now let $\mathcal{M}_1$ be a DTM_{k_1} and $\mathcal{M}_2$ be a DTM_{k_2} which are both of time complexity $id + r$. Due to the proof of Corollary 18 one can always find a $\mathrm{DTM}_{k_1:1}$ $\mathcal{M}_1'$ resp. a $\mathrm{DTM}_{k_2:1}$ $\mathcal{M}_2'$ that accepts the language $L(\mathcal{M}_1)$ resp. $L(\mathcal{M}_2)$ with the same time complexity.

A Turing machine $\mathcal{M}'$ for the union or the intersection of $L(\mathcal{M}_1)$ and $L(\mathcal{M}_2)$ simulates the machines $\mathcal{M}_1'$ and $\mathcal{M}_2'$ nearly parallel and decides at the end of both simulations dependent on both results whether the input belongs to $L(\mathcal{M}_1) \cap L(\mathcal{M}_2)$ or to $L(\mathcal{M}_1) \cup L(\mathcal{M}_2)$.

$\mathcal{M}'$ has a one-way input tape and $k_1 + k_2$ working tapes. On tapes $1, \dots, k_1$ the working tapes of $\mathcal{M}'_1$ and on tapes $k_1 + 1, \dots, k_1 + k_2$ the working tapes of $\mathcal{M}'_2$ are simulated directly.

During the computation of $\mathcal{M}'$ there may occur three different demands on the input tape head.

1. $\mathcal{M}'_1$ and $\mathcal{M}'_2$ are both requesting an input symbol. In this case $\mathcal{M}'$ simulates a step of $\mathcal{M}'_1$ as well as a step of $\mathcal{M}'_2$ whereby an input symbol is read.
2. If neither $\mathcal{M}'_1$ nor $\mathcal{M}'_2$ are requesting an input symbol, then $\mathcal{M}'$ simulates a step of $\mathcal{M}'_1$ and one of $\mathcal{M}'_2$ without reading an input symbol.
3. If $\mathcal{M}'_1$ requests an input symbol but $\mathcal{M}'_2$ does not (or vice versa) then $\mathcal{M}'$ simulates one step of $\mathcal{M}'_2$ (or $\mathcal{M}'_1$) only without reading an input symbol.

Since $\mathcal{M}'_1$ and $\mathcal{M}'_2$ are both equipped with a one-way input tape the number of time steps at which no input symbol is read is bounded by r. Therefore, the simulation delay of $\mathcal{M}'_1$ caused by $\mathcal{M}'_2$ is at most r. The same holds for the delay of $\mathcal{M}'_2$ caused by $\mathcal{M}'_1$. Thus $\mathcal{M}'$ obeys the time complexity $id + 2 \cdot r$. By Corollary 19 $\mathcal{M}'$ can be sped-up to $id + r$. □

Now we are exploring some closure properties concerning concatenations. It turns out that all the classes in question are neither closed under iteration nor under concatenation.

Theorem 23 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. If $r \in o(id)$ then*

$$\mathsf{DTIME}_{k:1}(id + r),\ \mathsf{DTIME}_{k:2}(id + r),\ k \geq 1,\ \textit{and}$$
$$\mathsf{DTIME}_{k}(id + r),\ k \geq 2,\ \textit{and}$$
$$\mathsf{DTIME}(id + r)$$

are not closed under left concatenation with regular sets.

Proof The language $L = \{y\$w¢y^R¢ \mid y \in \{0,1\}^+, w \in \{0,1,\$\}^+\}$ is a deterministic context-free language that is acceptable by a deterministic pushdown automaton without ε-transitions. Thus, it is a real-time $\mathrm{DTM}_{1:1}$ language and belongs to all the classes of the assertion.

Let $R = \{0,1,\$\}^*$ be a regular set. The concatenation RL contains all words of the form $w_1\$w_2\$\cdots\$w_l¢y¢$ where y^R matches one of the w_i, $1 \leq i \leq l$, and $|w_1| = \cdots = |w_l| = l$. Two such words are $(l+1)$-equivalent iff the sets of the subwords w_i are equal. There are $\binom{2^l}{l}$ different subsets of $\{0,1\}^l$ with l elements. As calculated in the proof of Lemma 12 the number $N\big((l+1)^2, l+1, RL\big)$ of equivalence classes is at least of order $2^{\Omega(l^2)}$.

But Lemma 7 and Corollary 8 say that the number of equivalence classes distinguishable by a Turing machine is bounded by $p^{l+1+r((l+1)^2)}$ for a constant $p \in \mathbb{N}_0$:

$$N\big((l+1)^2, l+1, RL\big) \le p^{l+1+r((l+1)^2)}$$

Since $r \in o(id)$ we obtain

$$= p^{l+1+o((l+1)^2)} = 2^{o(l^2)}$$

From the contradiction the theorem follows. □

Corollary 24 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. If $r \in o(id)$ then*

$$\begin{aligned}&\mathsf{DTIME}_{k:1}(id+r),\ \mathsf{DTIME}_{k:2}(id+r),\ k \ge 1,\ \text{and}\\ &\mathsf{DTIME}_k(id+r),\ k \ge 2,\ \text{and}\\ &\mathsf{DTIME}(id+r)\end{aligned}$$

are not closed under concatenation.

The technical reason why the proof does not work for the only excluded class $\mathsf{DTIME}_1(id+r)$ is simple: A $(id+r)$-time-bounded DTM_1 cannot accept the language L. In [5] it has been shown that the classes $\mathsf{DTIME}_1(id \cdot o(\log))$ are precisely the regular languages. The closure of $\mathsf{DTIME}_1(id+r)$ under concatenation and iteration follows immediately.

The proof of Theorem 23 yields a linear-time lower bound for the language RL even for multitape Turing machines.

In general, the non-closure under iteration is not an immediate corollary of the non-closure under concatenation.

Theorem 25 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. If $r \in o(id)$ then*

$$\begin{aligned}&\mathsf{DTIME}_{k:1}(id+r),\ \mathsf{DTIME}_{k:2}(id+r),\ k \ge 1,\ \text{and}\\ &\mathsf{DTIME}_k(id+r),\ k \ge 2,\ \text{and}\\ &\mathsf{DTIME}(id+r)\end{aligned}$$

are not closed under iteration.

Proof Let $L = \big\{y\$w¢y^R¢ \mid y \in \{\mathtt{0},\mathtt{1}\}^+, w \in \{\mathtt{0},\mathtt{1},\$\}^+\big\}$ and $R = \{\mathtt{0},\mathtt{1},\$\}^*$ be the same languages as in the proof of Theorem 23 and define $L' = L \cup R$. Since all the classes in question contain L and are closed under union with regular sets, L' belongs to all the classes either. Suppose contrarily that the classes are closed under iteration and therefore contain $(L')^*$ each. Since R is regular $R' = R¢R¢$ is regular, too. All the classes in question are closed under intersection with regular sets and, thus, containing $(L')^* \cap R'$. But this is a contradiction since $(L')^* \cap R'$ is precisely the language RL shown not to belong to all the classes in Theorem 23. $\square$

The negative closures of $\mathsf{DTIME}(id)$ under left concatenation with regular sets and iteration have been shown in [11]. Our results become interesting with respect to the corresponding open properties of the linear-time languages.

By the last results the question whether or not the classes are closed under a weaker kind of concatenation, i.e. marked concatenation, arises immediately. Obviously, all classes are closed under marked concatenation with regular sets. But for machines for which a speed-up is possible, i.e. $\mathrm{DTM}_{k:1}$ and multitape Turing machines, we can prove a stronger result:

Lemma 26 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be an increasing function and $k \geq 1$. Then*

$$\mathsf{DTIME}_{k:1}(id + r) \text{ and } \mathsf{DTIME}(id + r)$$

are closed under marked concatenation.

Proof Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two $(id + r)$-time-bounded $\mathrm{DTM}_{k:1}$. A $\mathrm{DTM}_{k:1}$ $\mathcal{M}$ for the marked concatenation of $L(\mathcal{M}_1)$ and $L(\mathcal{M}_2)$ works as follows.

In a first phase it simulates $\mathcal{M}_1$ on the left part of the input. If $\mathcal{M}_1$ would halt $\mathcal{M}$ moves the input tape head to the marking symbol and simulates $\mathcal{M}_2$ on the right part of the input.

Let $w = w_1 \bullet w_2$ be the input. Then $\mathcal{M}$ needs $|w_1| + r(|w_1|)$ time steps for the simulation of $\mathcal{M}_1$, at most $r(|w_1|)$ time steps in order to move the input tape head to the marking symbol, and additional $|w_2| + r(|w_2|)$ time steps for the simulation of $\mathcal{M}_2$.

Altogether $\mathcal{M}$ obeys the time complexity $|w_1| + |w_2| + 2r(|w_1|) + r(|w_2|)$. Since r is increasing this is at most $|w| + 3r(|w|) = id + 3r$. By Theorem 17 $\mathcal{M}$ can be sped-up to $id + r$.

For $\mathsf{DTIME}(id + r)$ the lemma is shown analogously. $\square$

Now we turn to the operation reversal. The linear-time languages $\mathsf{DTIME}(\mathrm{LIN})$ are trivially closed under reversal whereas the real-time languages are closed under right concatenation with regular sets but not under left concatenation, and therefore are not closed under reversal. Unfortunately, it is an open problem whether or not the classes between real-time and linear-time are closed under right concatenation with regular sets. But, fortunately, the non-closure under reversal can be shown by a certain witness language.

Theorem 27 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. If $r \in o(id)$ then*

$$\begin{aligned}&\mathsf{DTIME}_{k:1}(id + r),\ \mathsf{DTIME}_{k:2}(id + r),\ k \geq 1,\ \textit{and}\\ &\mathsf{DTIME}_k(id + r),\ k \geq 2,\ \textit{and}\\ &\mathsf{DTIME}(id + r)\end{aligned}$$

are not closed under reversal.

Proof The language

$$\begin{aligned}L = \big\{&w_l\$w_{l-1}\$\cdots\$w_1¢y¢ \mid l \in \mathbb{N}, w_i, y \in \{0,1\}^+, 1 \leq i \leq l, \text{ and}\\ &\exists\, 1 \leq i \leq l : (i \text{ odd } \wedge w_i^R = y) \vee (i \text{ even } \wedge w_i = y)\big\}\end{aligned}$$

does not belong to $\mathsf{DTIME}(id + r)$ for any $r \in o(id)$.

Two words $w_l\$\cdots\$w_1¢$ and $w'_l\$\cdots\$w'_1¢$, $|w_i| = |w'_i| = l$, $1 \leq i \leq l$, are $(l+1)$-equivalent iff the sets $\{w_1, \ldots, w_l\}$ and $\{w'_1, \ldots, w'_l\}$ are equal.

As shown in the proof of Lemma 12 the number $N\big((l+1)^2, l+1, L\big)$ of equivalence classes is at least of order $2^{\Omega(l^2)}$ whereas a $(id{+}r)$-time-bounded DTM can distinguish at most $2^{o(l^2)}$ classes.

Conversely, the reversal of L is real-time acceptable by a $\mathrm{DTM}_{1:1}$ as follows: The subword y is copied from the input tape to the working tape. At the end of this process the head of the working tape is located at the right hand side of y. Subsequently, it moves back and forth over the inscription y whereby the reversal of y is compared to w_1, y is compared to w_2, the reversal of y is compared to w_3 and so on. □

Since arbitrary erasing homomorphisms are a very powerful operation one expects that the classes are not closed under this kind of homomorphism. But they are not closed under weaker ε-free homomorphisms neither.

Theorem 28 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. If $r \in o(id)$ then*

$$\begin{aligned}&\mathsf{DTIME}_{k:1}(id + r),\ \mathsf{DTIME}_{k:2}(id + r),\ k \geq 1,\ \textit{and}\\ &\mathsf{DTIME}_k(id + r),\ k \geq 2,\ \textit{and}\\ &\mathsf{DTIME}(id + r)\end{aligned}$$

are not closed under ε-free homomorphisms.

Proof Let $L = \big\{y\$w¢y^R¢ \mid y \in \{0,1\}^+, w \in \{0,1,\$\}^+\big\}$ and $R = \{0,1,\$\}^*$ be defined as in the proof of Theorem 23 where it was shown that RL does not belong to $\mathsf{DTIME}(id + r)$ for any $r \in o(id)$.

Define $R' = \{\mathtt{0}', \mathtt{1}', \mathtt{\$}'\}^*$. Since L is a real-time $\text{DTM}_{1:1}$ language, $R'L$ belongs to all classes of the assertion.

The ε-free homomorphism $h(\mathtt{0}') = h(\mathtt{0}) = \mathtt{0}$, $h(\mathtt{1}') = h(\mathtt{1}) = \mathtt{1}$, $h(\mathtt{\$}') = h(\mathtt{\$}) = \mathtt{\$}$ and $h(\mathtt{¢}) = \mathtt{¢}$ maps $R'L$ to RL what proves the lemma. □

The closure properties concerning homomorphisms are in some sense asymmetric. It is not known whether the classes are closed under inverse homomorphisms if r grows strictly faster than log. Moreover we need to express the condition $r \in O(\log)$ by the functional equation $\log(m \cdot n) = \log(m) + \log(n)$ in order to exclude some exotic functions with a strange behavior.

Theorem 29 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. If $r(m \cdot n) \leq r(m) + r(n)$ then*

$$\mathsf{DTIME}_{k:1}(id + r),\ \mathsf{DTIME}_{k:2}(id + r),\ k \geq 1,\ \textit{and}$$
$$\mathsf{DTIME}_{k}(id + r),\ k \geq 2,\ \textit{and}$$
$$\mathsf{DTIME}(id + r)$$

are closed under inverse homomorphisms.

Proof Let $\mathcal{M}$ be a $(id + r)$-time-bounded Turing machine and A its set of input symbols. A Turing machine $\mathcal{M}'$ of the same type as $\mathcal{M}$ that for a given homomorphism $h : B^* \to A^*$ accepts the language $h^{-1}\big(L(\mathcal{M})\big) = \big\{w \in B^* \mid h(w) \in L(\mathcal{M})\big\}$ with time complexity $id + r$ works as follows.

Internally $\mathcal{M}'$ maps the currently scanned input symbol according to the homomorphism h and simulates $\mathcal{M}$ on this image. Obviously, if the simulation accepts then $h(w) \in L(\mathcal{M})$ and, thus, $\mathcal{M}'$ accepts since $w \in h^{-1}\big(L(\mathcal{M})\big)$.

Define $c = \max\{|h(b)| \mid b \in B\}$ then for all $w \in B^*$ the image $h(w)$ is not longer than $c \cdot |w|$. Therefore $\mathcal{M}'$ is $(c \cdot id + r(c \cdot id))$-time-bounded.

A speed-up can be achieved by the methods shown in the proof of Theorem 17. Since here $\mathcal{M}'$ maps the current input symbol b to $|h(b)|$ input symbols of $\mathcal{M}$, the next $|h(b)|$ steps can be sped-up as far as a possibly two-way input head of $\mathcal{M}$ does not move out of the $|h(b)|$ symbols to the left. Due to its time bound $\mathcal{M}$ performs at most r left moves and, therefore, $\mathcal{M}'$ can be sped-up to $id + r(c \cdot id)$. From the condition on r it follows $r(c \cdot id) \leq r(c) + r(id) \leq c' + r(id)$ for some constant $c' \in \mathbb{N}$.

Since a Turing machine can always be sped-up by an additive constant as long as the time complexity does not fall below real-time $\mathcal{M}'$ obeys the time-bound $id + r$. □

6 Generalization to other Types of Acceptors

The equivalence relation of Definition 6 and the upper bound of distinguishable equivalence classes play an important role in the proofs of our results. Next we are going to show that similar results are valid for several types of acceptors as long as the number of distinguishable equivalence classes is bounded similarly.

In general, we consider acceptors that consist of a finite-state control and an arbitrary number of memory cells. In each of the cells a symbol from a finite set may be stored. After a certain number of time steps the result of the computation is indicated by the state of the finite-state control. Moreover, the input is fed serially to the machine. For example, models with parallel input mode as are cellular automata are not considered.

Definition 30

1. *A language acceptor has* polynomially limited memory access of degree d *if at any time step $i \in \mathbb{N}$ the number of memory cells that are potentially accessible during the next $j \in \mathbb{N}$ time steps is bounded by $c \cdot j^d$ for some constant $c \in \mathbb{N}$.*
2. *A class $\mathscr{M}$ of language acceptors has polynomially limited memory access of degree d if each $\mathcal{M} \in \mathscr{M}$ has this property.*

Example 31 The following classes of language acceptors have polynomially limited memory access.

1. $\mathrm{DTM}_{k:i}$, $k \geq 1$ and $i \in \{0, 1, 2\}$, degree 1
2. multitape Turing machines, degree 1
3. d-dimensional multihead multitape Turing machines, degree d
4. d-dimensional iterative arrays, degree d

Now Lemma 7 that bounds the number of distinguishable equivalence classes is generalized.

Lemma 32 *Let $r : \mathbb{N}_0 \to \mathbb{N}$ be a function. If a language L is $(id{+}r)$-time acceptable by a machine with polynomially limited memory access of degree d, then there exists a constant $p \in \mathbb{N}$ such that*

$$N(n, l, L) \leq p^{(l+r(n))^d}$$

The proof is a straightforward adaption of the proof of Lemma 7.

As a consequence we obtain:

Lemma 33 *Let $\mathcal{M}$ be an acceptor with polynomially limited memory access of degree d. Then the language*

$$L = \{w_1\$\cdots\$w_l ¢ y ¢ \mid y, w_i \in \{0,1\}^+, 1 \leq i \leq l \wedge \exists\, j \in \{1,\cdots,l\} : y = w_i\}$$

cannot be accepted by $\mathcal{M}$ in $(id + r)$-time if $r \in o(id^{\frac{1}{d}})$.

Proof L contains all words of the form $w_1\$\cdots\$w_{l^d} ¢ y ¢$ where y matches one of the w_i and $|w_1| = \cdots = |w_{l^d}| = l$. Two such words are $(l+1)$-equivalent if the sets of the subwords w_i are equal.

There are $\binom{2^l}{l^d}$ different of theses subsets. Generalizing the calculation in the proof of Lemma 12 the number $N\big((l^d+1)\cdot(l+1), l+1, L\big)$ of equivalence classes is at least of order $2^{\Omega(l^{d+1})}$.

But by Lemma 32 the number of distinguishable equivalence classes is bounded as follows.

$$N\big((l^d+1)\cdot(l+1), l+1, L\big) \leq p^{(l+1+r((l^d+1)\cdot(p+1)))^d}$$

Since $r \in o(id^{\frac{1}{d}})$ we obtain

$$\begin{aligned} &= p^{(l+1+o((l^{d+1}+l^d+l+1)^{\frac{1}{d}}))^d} \\ &= p^{(l+1+o(l^{\frac{d+1}{d}}))^d} \\ &= p^{l^d+o(l^{d+1})} = 2^{o(l^{d+1})} \end{aligned}$$

From the contradiction the lemma follows. □

For one-dimensional Turing machines the lemma yields a linear-time lower bound for the language L. Furthermore, the lemma implies that dependent on a honest function r (with respect to the device in question) a language L_r can be defined similarly to L_{h_r} that is not acceptable in $(id + r')$-time if $r' \in o(r)$. On the other hand, if L is $(id + id^{\frac{1}{d}})$-time acceptable then L_r is acceptable in $(id + r)$-time (cf. proof of Lemma 11). These observations lead to the generalization of the hierarchy theorem.

Theorem 34 *Let $\mathscr{M}$ be a class of acceptors with polynomially limited memory access of degree d and L be the language of Lemma 33. If L is acceptable by some $\mathcal{M} \in \mathscr{M}$ in $(id + id^{\frac{1}{d}})$-time, then there exists an infinite time hierarchy between real-time and $(id + id^{\frac{1}{d}})$-time.*

Regarding closure properties the negative results can be generalized.

Theorem 35 *Let $\mathscr{M}$ be a class of acceptors with polynomially limited memory access of degree d and $r : \mathbb{N}_0 \to \mathbb{N}$ be a function with $r \in o(id^{\frac{1}{d}})$.*

1. *If $L(\mathscr{M})$ contains the deterministic context-free language $\{y\$w¢y^R¢ \mid y \in \{\mathtt{0},\mathtt{1}\}^+, w \in \{\mathtt{0},\mathtt{1},\$\}^+\}$ and the regular language $\{\mathtt{0},\mathtt{1},\$\}^*$ then $L(\mathscr{M})$ is not closed under left concatenation with regular sets, nor under concatenation.*
2. *If $L(\mathscr{M})$ is additionally closed under union and intersection with regular sets then $L(\mathscr{M})$ is not closed under iteration.*
3. *If $L(\mathscr{M})$ contains the real-time $\mathrm{DTM}_{1:1}$ languages then it is not closed under reversal.*
4. *If $L(\mathscr{M})$ in addition to 1. is closed under marked left concatenation with regular sets then it is not closed under ε-free homomorphisms.*

References

[1] Balcázar, J. L., Díaz, J., and Gabarró, J. *Structural Complexity I*. Springer, Berlin, 1988.

[2] Book, R. V. and Greibach, S. A. *Quasi-realtime languages*. Math. Systems Theory 4 (1970), 97–111.

[3] Cole, S. N. *Real-time computation by n-dimensional iterative arrays of finite-state machines*. IEEE Trans. Comput. C-18 (1969), 349–365.

[4] Fürer, M. *The tight deterministic time hierarchy*. Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing (STOC '82), 1982, pp. 8–16.

[5] Hartmanis, J. *Computational complexity of one-tape turing machine computations*. J. Assoc. Comput. Mach. 15 (1968), 325–339.

[6] Hartmanis, J. and Stearns, R. E. *On the computational complexity of algorithms*. Trans. Amer. Math. Soc. 117 (1965), 285–306.

[7] Hennie, F. C. *One-tape, off-line turing machine computations*. Inform. Control 8 (1965), 553–578.

[8] Kobayashi, K. *On proving time constructibility of functions*. Theoret. Comput. Sci. 35 (1985), 215–225.

[9] Paul, W. J. *Komplexitätstheorie*. Teubner, Stuttgart, 1978.

[10] Paul, W. J. *On time hierarchies*. J. Comput. System Sci. 19 (1979), 197–202.

[11] Rosenberg, A. L. *Real-time definable languages*. J. Assoc. Comput. Mach. 14 (1967), 645–662.

[12] Wagner, K. and Wechsung, G. *Computational Complexity*. Reidel, Dordrecht, 1986.