Complexity and Cryptography

Thomas Zeugmann

Hokkaido University
Laboratory for Algorithmics

https://www-alg.ist.hokudai.ac.jp/~thomas/COCRB/

Lecture 6: Introducing Complexity Classes

Turing Machines I

A one-tape Turing machine consists of an infinite tape which is divided into cells. Each cell can contain exactly one of the tape-symbols. Initially, we assume that all cells of the tape contain the symbol $*$ except those in which the actual input has been written. Moreover, we enumerate the tape cells as shown in Figure 1.

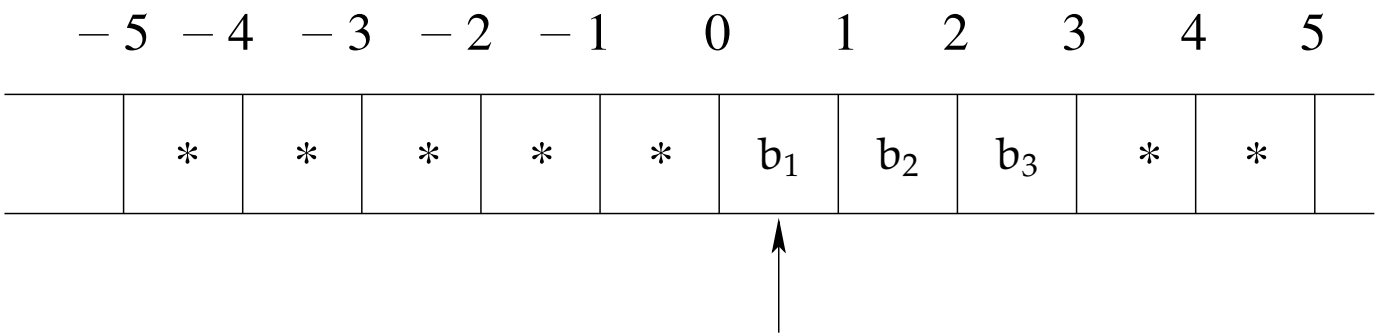


Figure 1: The tape of a Turing machine with input $b_1 b_2 b_3$.

Turing Machines II

Below we use TM as an abbreviation for Turing machine. The TM possesses a read-write head. This head can observe one cell at a time. Additionally, the TM has a finite number of states it can be in and a set of instructions it can execute. Initially, it is always in the start state z_s.

Turing Machines II

Below we use TM as an abbreviation for Turing machine. The TM possesses a read-write head. This head can observe one cell at a time. Additionally, the TM has a finite number of states it can be in and a set of instructions it can execute. Initially, it is always in the start state z_s.

Then, the TM works as follows: When in state z and reading the tape symbol b it writes the tape symbol b' into the observed cell, changes its state to z' and moves the head either one position to the left (denoted by L) or one position to the right (denoted by R) or does not move the head (denoted by N) provided (z, b, b', H, z') is in the instruction set of the TM, where $H \in \{L, N, R\}$. The execution of one instruction is called *step*.

When the TM reaches a distinguished state z_f (the final state), it stops.

Turing Machines III

Formally we define a TM as follows:

Definition 1

$M = [B, Z, A]$ is called a *deterministic one-tape TM* if B, Z, A are non-empty finite sets such that $B \cap Z = \emptyset$ and

(1) $|B| \geqslant 2$ $(\ B = \{*, |, \ldots\}\ )$ (tape-symbols);

(2) $|Z| \geqslant 2$ $(\ Z = \{z_s, z_f, \ldots\})$ (set of states);

(3) $A \subseteq Z \setminus \{z_f\} \times B \times B \times \{L, N, R\} \times Z$ (instruction set), where for every $z \in Z \setminus \{z_f\}$ and every $b \in B$ there is precisely one 5-tuple $(z, b, \cdot, \cdot, \cdot)$.

Often, we represent the instruction set A in a table.

Turing Machines IV

Let Σ denote any finite alphabet, or synonymously, a set of symbols. Then we use Σ^* to denote the free monoid over Σ. In the following we shall use λ to denote the *empty string*. We set $\Sigma^+ =_{\mathtt{df}} \Sigma^* \setminus \{\lambda\}$. Note that $\lambda \neq *$.

Any set $L \subseteq \Sigma^*$ is called a *language*.

Now, we define what does it mean that a TM accepts a language L.

Turing Machines V

Definition 2

A *TM M accepts a language* $L \subseteq \Sigma^*$ if for every string $w \in \Sigma^*$ the following conditions are satisfied:
If w is written on the empty tape of M (beginning in cell 0) and the TM M is started on the leftmost symbol of w in state z_s then M stops after having executed finitely many steps in state z_f.
Moreover,

(1) if $w \in L$ then the cell observed by M in state z_f contains a $|$.
In this case we also write $M(w) = |$.

(2) If $w \notin L$ then the cell observed by M in state z_f contains a $*$.
In this case we also write $M(w) = *$.

Turing Machines V

Definition 2

A *TM M accepts a language* $L \subseteq \Sigma^*$ if for every string $w \in \Sigma^*$ the following conditions are satisfied:
If w is written on the empty tape of M (beginning in cell 0) and the TM M is started on the leftmost symbol of w in state z_s then M stops after having executed finitely many steps in state z_f.
Moreover,

(1) if $w \in L$ then the cell observed by M in state z_f contains a $|$.
In this case we also write $M(w) = |$.

(2) If $w \notin L$ then the cell observed by M in state z_f contains a $*$.
In this case we also write $M(w) = *$.

Of course, in order to accept a language $L \subseteq \Sigma^*$ by a TM $M = [B, Z, A]$ we always have to assume that $\Sigma \subseteq B$.

Turing Machines VI

Next, we formally define a time complexity for TMs. In order to do so, we need the following notations: For any string w, we use $|w|$ to denote the length of w. Furthermore, for every alphabet Σ and $n \in \mathbb{N}$ we set

$$\Sigma^n =_{df} \{w \mid w \in \Sigma^* \ \wedge \ |w| = n\}.$$

Turing Machines VI

Next, we formally define a time complexity for TMs. In order to do so, we need the following notations: For any string w, we use $|w|$ to denote the length of w. Furthermore, for every alphabet Σ and $n \in \mathbb{N}$ we set

$$\Sigma^n =_{df} \{w \mid w \in \Sigma^* \ \wedge \ |w| = n\}.$$

Definition 3

Let $M = [B, Z, A]$ be a any TM, and let $w \in B^*$. We set

$$T_M(w) \quad =_{df} \quad \text{the number of steps performed by } M \text{ on input } w \text{ when started on the leftmost symbol of } w \text{ in state } z_s \text{ until reaching state } z_f .$$

Furthermore, we define $T_M(n) =_{df} \max\{T_M(w) \mid w \in \Sigma^n\}$.

Turing Machines VII

Next, let $T\colon \mathbb{N} \to \mathbb{N}$ be any function. Then we define the *time complexity class* generated by T as follows:

$$\begin{aligned} \mathit{Time}(T(n)) \quad =_{\mathtt{df}} \quad \{L \mid L \subseteq \Sigma^* \text{ and there is a TM } M \text{ accepting } L \\ \text{such that } T_M(n) \leqslant T(n) \\ \text{for all but finitely many } n\}\,. \end{aligned}$$

Turing Machines VII

Next, let $T\colon \mathbb{N} \to \mathbb{N}$ be any function. Then we define the *time complexity class* generated by T as follows:

$$\begin{aligned} \mathit{Time}(T(n)) \quad =_{df} \quad \{L \mid L \subseteq \Sigma^* \text{ and there is a TM } M \text{ accepting } L \\ \text{such that } T_M(n) \leqslant T(n) \\ \text{for all but finitely many } n\}\,. \end{aligned}$$

This is a good place to recall our deterministic TM accepting the language of all palindromes over the two letter alphabet $\Sigma = \{a, b\}$ from our course *Theory of Computation*.

Palindromes I

Let $L_{pal} =_{df} \{w \mid w \in \Sigma^*,\ w = w^T\}$ and let $M = [B, Z, A]$, where $B = \{*, |, a, b\}$, $Z = \{z_s, z_1, z_2, z_3, z_4, z_5, z_6, z_f\}$ and A is given by the following table:

	a	b	$*$	$\|$
z_s	$*Rz_1$	$*Rz_2$	$\|Nz_f$	$\|Nz_f$
z_1	aRz_1	bRz_1	$*Lz_3$	$\|Nz_f$
z_2	aRz_2	bRz_2	$*Lz_4$	$\|Nz_f$
z_3	$*Lz_5$	$*Nz_f$	$\|Nz_f$	$\|Nz_f$
z_4	$*Nz_f$	$*Lz_5$	$\|Nz_f$	$\|Nz_f$
z_5	aLz_6	bLz_6	$\|Nz_f$	$\|Nz_f$
z_6	aLz_6	bLz_6	$*Rz_s$	$\|Nz_f$

Figure 2: Instruction set of a TM accepting L_{pal}.

Palindromes II

So, this TM remembers the actual leftmost and rightmost symbol, respectively. Then it is checking whether or not it is identical to the rightmost and leftmost symbol, respectively. Thus, the time complexity of this TM is $O(n^2)$.

Palindromes II

So, this TM remembers the actual leftmost and rightmost symbol, respectively. Then it is checking whether or not it is identical to the rightmost and leftmost symbol, respectively. Thus, the time complexity of this TM is $O(n^2)$.
Of course, we can design a TM that works *mutatis mutandis* as the TM above, but which memorizes two symbols each time, or more generally, k symbols. Nevertheless, the resulting time complexity is still $O(n^2)$. Thus, it is only natural to ask whether or not we can do any better. Is there a TM accepting L_{pal} having time complexity $o(n^2)$? The negative answer is provided by Theorem 1 below which was found by Jānis Bārzdiņš.
Before we can show it, we need a technique to prove lower bounds.

Traces of Turing Computations I

The key ingredient are *traces* of Turing computations (also called *crossing sequences* or *scheme*). Consider the computation process of a TM M on input the string $w = x_1 \cdots x_n$, where $x_i \in \Sigma$, and started in the initial configuration. Let us fix a border on the tape at position j (cf. Figure 3 for the case $j = 2$).

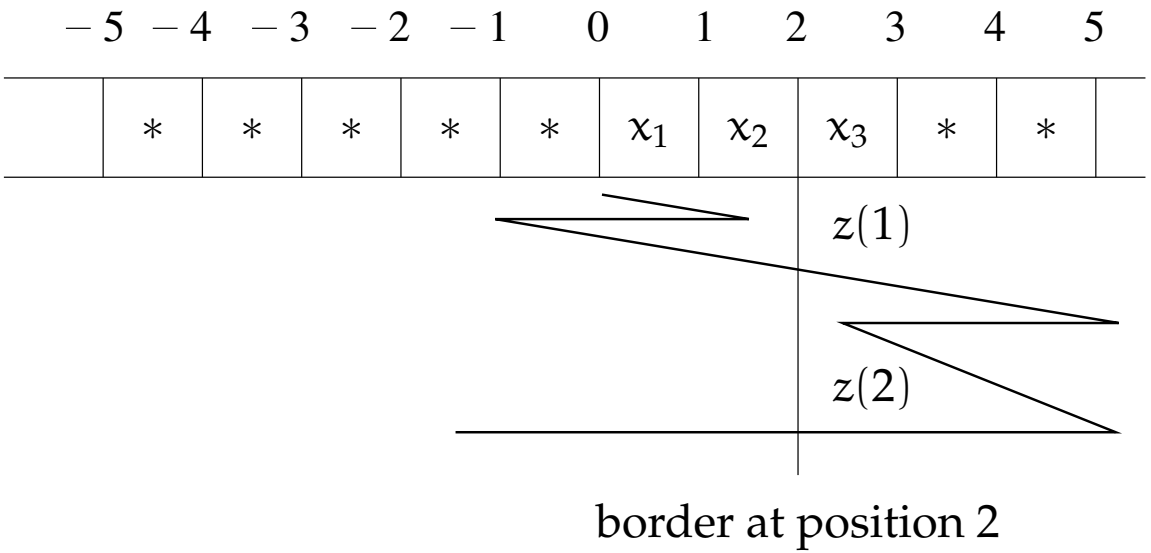


Figure 3: A fragment of the trace at border 2.

Traces of Turing Computations II

The trace of M on input w at position j is a string $\mathfrak{z} \in Z^*$ defined as follows:

(0) If the read-write head is never crossing the border j, then $\mathfrak{z} =_{df} \lambda$.

(1) Assume the read-write head crosses the border j exactly s times. The first time the border is crossed, M is in state $z(1)$, the second time in state $z(2)$, ..., and the sth time in state $z(s)$. Then, we set $\mathfrak{z} =_{df} z(1)z(2)\cdots z(s)$.

Traces of Turing Computations II

The trace of M on input w at position j is a string $\mathfrak{z} \in Z^*$ defined as follows:

(0) If the read-write head is never crossing the border j, then $\mathfrak{z} =_{df} \lambda$.

(1) Assume the read-write head crosses the border j exactly s times. The first time the border is crossed, M is in state $z(1)$, the second time in state $z(2)$, ..., and the sth time in state $z(s)$. Then, we set $\mathfrak{z} =_{df} z(1)z(2)\cdots z(s)$.

We denote the trace of M at position j on input w by $TR_M(w, j)$.
Of course, if $j > 0$, then the first time the head crosses the border at position j is a move from left to right, the second time from right to left, and so on.
If $j \leqslant 0$, then the first time the head crosses the border at position j is a move from right to left, the second time from left to right, and so on.

Traces of Turing Computations III

If M's computation on input w does not stop, then the trace at a position j may become infinite. But we consider terminating computations only. So all traces have a finite length which is denoted by $|TR_M(w, j)|$.

Traces of Turing Computations III

If M's computation on input w does not stop, then the trace at a position j may become infinite. But we consider terminating computations only. So all traces have a finite length which is denoted by $|TR_M(w,j)|$.

The following observation explains the importance of traces:

Observation 1

$$T_M(w) \geqslant \sum_{j \in \mathbb{Z}} |TR_M(w,j)|.$$

Proof. The observation holds, since every time the head crosses the border, a step has to be performed. ∎

Traces of Turing Computations III

If M's computation on input w does not stop, then the trace at a position j may become infinite. But we consider terminating computations only. So all traces have a finite length which is denoted by $|TR_M(w,j)|$.

The following observation explains the importance of traces:

Observation 1

$$T_M(w) \geqslant \sum_{j \in \mathbb{Z}} |TR_M(w,j)|.$$

Proof. The observation holds, since every time the head crosses the border, a step has to be performed. ∎

So, if we prove that for every TM accepting a language L there must be infinitely many strings such that at sufficiently many positions sufficiently long traces occur, then we obtain a lower bound for the time needed to accept L. How can we prove this?

Traces of Turing Computations IV

Assume the trace $z(1) \cdots z(s)$ of a TM M on input w at position j to be known. For the sake of presentation, let $j > 0$. We cut the tape at position j and remove the right part of the tape cut. So we can still simulate the whole computation of M on the left part of the tape. As long as M does not cross the border j, the TM M works as before.

Traces of Turing Computations IV

Assume the trace $z(1) \cdots z(s)$ of a TM M on input w at position j to be known. For the sake of presentation, let $j > 0$. We cut the tape at position j and remove the right part of the tape cut. So we can still simulate the whole computation of M on the left part of the tape. As long as M does not cross the border j, the TM M works as before.
If M crosses the border for the first time (in state $z(1)$), we stop M, put the head back on the rightmost position of the left part of tape, and switch its state to $z(2)$. Now, M works again on the left part of the tape. Note that M behaves precisely the same way as if we have not cut the tape. This holds, since we restarted M in state $z(2)$ and the only possibility for M to memorize something it has done on the right part of the tape that may be needed while working on the left part of it, is its state. All what is left is to iterate this construction.

Traces of Turing Computations V

Let M be any TM, let $w = x_1 \cdots x_m$ and $\tilde{w} = \tilde{x}_1 \cdots \tilde{x}_n$ be any strings, and let $i, j \in \mathbb{N}$ with $0 < i < m$, $0 < j < n$ be such that $TR_M(w, i) = TR_M(\tilde{w}, j)$.
Furthermore, let $\tilde{w}_0^j = \tilde{x}_1 \cdots \tilde{x}_j$ and $\tilde{w}_j^n = \tilde{x}_{j+1} \cdots \tilde{x}_n$ as well as $w_0^i = x_1 \cdots x_i$ and $w_i^m = x_{i+1} \cdots x_m$. Then we have

$$TR_M(\tilde{w}, j) = TR_M(w, i) = TR_M(\tilde{w}_0^j w_i^m, j) = TR_M(w_0^i \tilde{w}_j^n, i) \ .$$

Traces of Turing Computations V

Let M be any TM, let $w = x_1 \cdots x_m$ and $\tilde{w} = \tilde{x}_1 \cdots \tilde{x}_n$ be any strings, and let $i, j \in \mathbb{N}$ with $0 < i < m$, $0 < j < n$ be such that $TR_M(w, i) = TR_M(\tilde{w}, j)$.
Furthermore, let $\tilde{w}_0^j = \tilde{x}_1 \cdots \tilde{x}_j$ and $\tilde{w}_j^n = \tilde{x}_{j+1} \cdots \tilde{x}_n$ as well as $w_0^i = x_1 \cdots x_i$ and $w_i^m = x_{i+1} \cdots x_m$. Then we have

$$TR_M(\tilde{w}, j) = TR_M(w, i) = TR_M(\tilde{w}_0^j w_i^m, j) = TR_M(w_0^i \tilde{w}_j^n, i) \ .$$

Consider M's behavior when started on input $\tilde{w}_0^j w_i^m$.

Case 1. $|TR_M(\tilde{w}, j)|$ is even.

Then the read-write head is left from position j when M terminates its computation. Moreover, the head observes in its final position a $|$ if $\tilde{w} \in L(M)$ and a $*$ provided $\tilde{w} \notin L(M)$.

Traces of Turing Computations VI

Case 2. $|TR_M(\tilde{w}, j)|$ is odd.

Then M stops its computation right from position j. Moreover, the head observes in its final position a $|$ if $w \in L(M)$ and a $*$ provided $w \notin L(M)$.

Traces of Turing Computations VI

Case 2. $|TR_M(\tilde{w}, j)|$ is odd.

Then M stops its computation right from position j. Moreover, the head observes in its final position a $|$ if $w \in L(M)$ and a $*$ provided $w \notin L(M)$.

Now, the scenario that $\tilde{w}$ and w are equivalent with respect to acceptance by M is of particular importance. By *equivalence with respect to acceptance by* M we mean that either both $\tilde{w}, w \in L(M)$ or both $\tilde{w}, w \notin L(M)$.
Taking the observation just made into account, we directly get the following *Replacement Lemma*, where we use the notations introduced above:

Traces of Turing Computations VII

Lemma 1 (Replacement lemma)

Let M be any TM, and assume $\tilde{w} = \tilde{w}_0^j \tilde{w}_j^n$ and $w = w_0^i w_i^m$ to be strings such that $\tilde{w}$ and w are equivalent with respect to acceptance by M and such that $TR_M(\tilde{w}, j) = TR_M(w, i)$. Then, the four strings $\tilde{w}$, w, $\tilde{w}_0^j w_i^m$ and $w_0^i \tilde{w}_j^n$ are pairwise equivalent with respect to acceptance by M.

Now, we are ready to prove the desired lower bound for L_{pal}.

Traces of Turing Computations VII

Lemma 1 (Replacement lemma)

Let M be any TM, and assume $\tilde{w} = \tilde{w}_0^j \tilde{w}_j^n$ and $w = w_0^i w_i^m$ to be strings such that $\tilde{w}$ and w are equivalent with respect to acceptance by M and such that $TR_M(\tilde{w}, j) = TR_M(w, i)$. Then, the four strings $\tilde{w}$, w, $\tilde{w}_0^j w_i^m$ and $w_0^i \tilde{w}_j^n$ are pairwise equivalent with respect to acceptance by M.

Now, we are ready to prove the desired lower bound for L_{pal}.

Theorem 1 (Bārzdiņš (1965), Hennie (1965))

For every deterministic one-tape Turing machine M accepting L_{pal} there is a constant $c_M > 0$ such that $T_M(n) \geqslant c_M \cdot n^2$ for all but finitely many $n \in \mathbb{N}$.

Lower Bound for Palindromes I

Proof. To avoid unnecessary technical details, assume that $4|n$.

Claim 1. *Let* M *be any deterministic one-tape TM accepting* L_{pal}. *Then there is a constant* $D_M > 0$ *such that for all sufficiently large* n *there are strings* $w \in L_{pal}$ *satisfying* $|w| = n$ *and*
$|TR_M(w, j)| \geqslant D_M \cdot n \quad$ *for all* $j = \frac{n}{4} + 1,\ \frac{n}{4} + 2,\ \ldots,\ \frac{n}{2}$.

Lower Bound for Palindromes I

Proof. To avoid unnecessary technical details, assume that $4|n$.

Claim 1. *Let* M *be any deterministic one-tape TM accepting* L_{pal}. *Then there is a constant* $D_M > 0$ *such that for all sufficiently large* n *there are strings* $w \in L_{pal}$ *satisfying* $|w| = n$ *and*
$|TR_M(w,j)| \geqslant D_M \cdot n$ *for all* $j = \frac{n}{4}+1,\ \frac{n}{4}+2,\ \ldots,\ \frac{n}{2}$.

Before proving Claim 1, we show that it directly implies the assertion of Theorem 1. Using Observation 1, we get

$$T_M(w) \ \geqslant \sum_{j=n/4+1}^{n/2} |TR_M(w,j)| \ \geqslant \ \frac{n}{4} \cdot D_M \cdot n = \frac{D_M}{4} \cdot n^2 ,$$

and hence Theorem 1 follows by setting $c_M = D_M/4$.

Lower Bound for Palindromes II

To show Claim 1, let $M = [B, Z, A]$ be a TM with $L(M) = L_{pal}$. We prove that it suffices to set $D_M = 1/(8 \log_2 k)$, where $k = |Z|$. For $j = n/4 + 1,\ n/4 + 2,\ \ldots,\ n/2$, we let $N(j, n)$ be the number of palindromes w over $\{a, b\}$ of length n for which $|TR_M(w, j)| < n/(8 \log_2 k)$.

We claim that $N(j, n) \leqslant 2^{3n/8}$. Since there are $2^{n/2}$ many palindromes, this will prove Claim 1.

Lower Bound for Palindromes II

To show Claim 1, let $M = [B, Z, A]$ be a TM with $L(M) = L_{pal}$. We prove that it suffices to set $D_M = 1/(8 \log_2 k)$, where $k = |Z|$. For $j = n/4 + 1,\ n/4 + 2,\ \ldots,\ n/2$, we let $N(j, n)$ be the number of palindromes w over $\{a, b\}$ of length n for which $|TR_M(w, j)| < n/(8 \log_2 k)$.

We claim that $N(j, n) \leqslant 2^{3n/8}$. Since there are $2^{n/2}$ many palindromes, this will prove Claim 1.
Let $\sigma_1, \ldots, \sigma_p$ be all pairwise different traces of length less than $n/(8 \log_2 k)$. By $A(\alpha)$ we denote the number of all palindromes of length n having trace σ_α in position j. Then, we have

$$N(j, n) = A(1) + A(2) + \cdots + A(p)\ . \tag{1}$$

First, we estimate p.

Lower Bound for Palindromes III

Since $|Z| = k$ and traces are strings over Z, we obtain

$$\begin{aligned} p &\leqslant k + k^2 + \cdots + k^{n/(8\log_2 k)-1} \\ &= \frac{k^{n/(8\log_2 k)} - 1}{k-1} - 1 \leqslant k^{n/(8\log_2 k)} = 2^{n/8}. \end{aligned} \tag{2}$$

Lower Bound for Palindromes III

Since $|Z| = k$ and traces are strings over Z, we obtain

$$\begin{aligned} p &\leqslant k + k^2 + \cdots + k^{n/(8\log_2 k)-1} \\ &= \frac{k^{n/(8\log_2 k)} - 1}{k-1} - 1 \leqslant k^{n/(8\log_2 k)} = 2^{n/8}. \end{aligned} \quad (2)$$

Next, we estimate $A(\alpha)$. Let $\tilde{w} = \tilde{w}_0^j \tilde{w}_j^n$ and $w = w_0^j w_j^m$ be palindromes of length n and let $j \leqslant n/2$. If M generates in position j the same traces on input $\tilde{w}$ and on input w, then $\tilde{w}_0^j = w_0^j$ by Lemma 1.
So all strings contributing to $A(\alpha)$ must have the same initial part, say v. The number of palindromes with initial part v is $2^{(n/2)-j}$, i.e., $A(\alpha) \leqslant 2^{(n/2)-j}$. By using (1) and (2), we have

$$N(j,n) \leqslant p \cdot 2^{n/2-j} \leqslant 2^{n/8} \cdot 2^{n/2-n/4} = 2^{3n/8}.$$

Thus, Claim 1 is proved and the theorem follows. ∎

Regular Languages I

As we already know, every regular language can be accepted by a finite automaton. Thus, for every $L \in \mathcal{REG}$ there is also a TM M accepting it such that $T_M(n) = n$ for all $n \in \mathbb{N}$. Thus, Theorem 1 directly implies that $L_{pal} \notin \mathcal{REG}$.

Regular Languages I

As we already know, every regular language can be accepted by a finite automaton. Thus, for every $L \in \mathcal{REG}$ there is also a TM M accepting it such that $T_M(n) = n$ for all $n \in \mathbb{N}$. Thus, Theorem 1 directly implies that $L_{pal} \notin \mathcal{REG}$.

So, we already know that time $O(n^2)$ is enough to accept non-regular languages by using deterministic one-tape TMs. But what happens if we add less resources?

To answer this question, we introduce the following notations, where $M = [B, Z, A]$ stands for any deterministic one-tape TM, and $w \in \Sigma^*$ as well as $n \in \mathbb{N}$:

$$TR_M(w) =_{df} \max\{|TR_M(w, j)| \mid j \in \mathbb{Z}\} \quad (3)$$

$$TR_M(n) =_{df} \max\{TR_M(w) \mid |w| = n\}. \quad (4)$$

First, we prove the following lemma:

Regular Languages II

Lemma 2 (Hennie (1965))

Let M *be any deterministic one-tape TM. If there is a constant* $c > 0$ *such that* $TR_M(n) \leqslant c$ *for all* $n \in \mathbb{N}$ *then* $L(M) \in \mathcal{REG}$.

Regular Languages II

Lemma 2 (Hennie (1965))

Let M be any deterministic one-tape TM. If there is a constant $c > 0$ such that $TR_M(n) \leqslant c$ for all $n \in \mathbb{N}$ then $L(M) \in \mathcal{REG}$.

Without loss of generality, we assume that the input w is read at least once. Consider the *echo mapping* of $M = [B, Z, A]$ on input w. Let $\zeta = z(2), \cdots, z(2n)$ be any fixed finite sequence from Z^*. The echo $z(1), \ldots, z(2n-1)$ of ζ of M on w is defined as follows:
We start M on w as usual. By assumption, M leaves the input sometimes for the first time on the right hand side. Let $z(1)$ be the state of M when crossing position $|w|$. We put M into state $z(2)$ and the head back to the cell where the last symbol of the input was or still is. After some time, M's head crosses again $|w|$. Let $z(3)$ be M's state when this event happens.

Regular Languages III

We put M into state $z(4)$ and the head again back to cell where the last symbol of the input was or still is, and so on. If M is not crossing often enough the position $|w|$ to the right, then the echo is *not* defined. For $\mathfrak{z} \in Z^*$ we use $\mathrm{echo}(w, \mathfrak{z})$ to denote the echo of $\mathfrak{z}$ on input w of M.

Regular Languages III

We put M into state $z(4)$ and the head again back to cell where the last symbol of the input was or still is, and so on. If M is not crossing often enough the position $|w|$ to the right, then the echo is *not* defined. For $\mathfrak{z} \in Z^*$ we use $\mathrm{echo}(w, \mathfrak{z})$ to denote the echo of $\mathfrak{z}$ on input w of M.

Next, we define a relation $\sim$ over Σ^* as follows: Let $u, v \in \Sigma^*$; we define

$$\begin{aligned} u \sim v \quad & \text{iff} \quad u \text{ and } v \text{ are equivalent w.r.t. acceptance by M ,} \\ & \text{and} \quad \forall \mathfrak{z}[\mathfrak{z} \in Z^* \ \wedge \ |\mathfrak{z}| \leqslant c \ \Rightarrow \ \mathrm{echo}(u, \mathfrak{z}) = \mathrm{echo}(v, \mathfrak{z})] \, . \end{aligned}$$

One easily verifies that $\sim$ is an equivalence relation. Thus, Σ^* is partitioned by $\sim$ into equivalence classes. Let $\Sigma^*_{/\sim}$ denote this set of equivalence classes.

Regular Languages IV

Claim 1. $\left|\Sigma^*_{/\sim}\right|$ *is finite.*

Since Z is finite, we know that $|\{\mathfrak{z} \mid \mathfrak{z} \in Z^* \ \wedge \ |\mathfrak{z}| \leqslant c\}| < \infty$. By the definition of the echo mapping, the length of $\text{echo}(w, \mathfrak{z})$ is, for all $\mathfrak{z} \in Z^*$ with $|\mathfrak{z}| \leqslant c$, also bounded by c. Hence, for every $w \in \Sigma^*$ there are only finitely many echo mappings with length less than or equal to c. But the set of all echo mappings on input w of M completely determines the equivalence class generated by w. Thus, $\left|\Sigma^*_{/\sim}\right|$ is finite and Claim 1 follows.

Regular Languages IV

Claim 1. $\left|\Sigma^*_{/\sim}\right|$ *is finite.*

Since Z is finite, we know that $|\{\mathfrak{z} \mid \mathfrak{z} \in Z^* \ \wedge \ |\mathfrak{z}| \leqslant c\}| < \infty$. By the definition of the echo mapping, the length of $\text{echo}(w, \mathfrak{z})$ is, for all $\mathfrak{z} \in Z^*$ with $|\mathfrak{z}| \leqslant c$, also bounded by c. Hence, for every $w \in \Sigma^*$ there are only finitely many echo mappings with length less than or equal to c. But the set of all echo mappings on input w of M completely determines the equivalence class generated by w. Thus, $\left|\Sigma^*_{/\sim}\right|$ is finite and Claim 1 follows.

By assumption, $TR_M(n) \leqslant c$ for all $n \in \mathbb{N}$. Hence, if $u \sim v$ then

$$TR_M(uw, |u|) = TR_M(vw, |v|) \quad \text{for all } w \in \Sigma^* . \tag{5}$$

Therefore, we may conclude that

$$uw \in L(M) \iff vw \in L(M) \text{ for all } w \in \Sigma^* . \tag{6}$$

But this means that L(M) satisfies the Nerode relation $\sim_N$.

Regular Languages V

Since $u \sim v$ implies $u \sim_N v$, we conclude that $\left|\Sigma^*_{/\sim_N}\right|$ is finite, too. By Nerode's theorem, it follows that $L(M) \in \mathcal{REG}$. ∎

Regular Languages V

Since $u \sim v$ implies $u \sim_N v$, we conclude that $\left|\Sigma^*/_{\sim_N}\right|$ is finite, too. By Nerode's theorem, it follows that $L(M) \in \mathcal{REG}$. ∎

Next, we sharpen Lemma 2 by proving a gap for $TR_M(n)$.

Lemma 3

Let M be any deterministic one-tape TM and assume $TR_M(n) = o(\log n)$. Then there is a constant $c > 0$ such that $TR_M(n) < c$ for all $n \in \mathbb{N}$.

Regular Languages V

Since $u \sim v$ implies $u \sim_N v$, we conclude that $\left|\Sigma^*_{/\sim_N}\right|$ is finite, too. By Nerode's theorem, it follows that $L(M) \in \mathcal{REG}$. ∎

Next, we sharpen Lemma 2 by proving a gap for $TR_M(n)$.

Lemma 3

Let M be any deterministic one-tape TM and assume $TR_M(n) = o(\log n)$. Then there is a constant $c > 0$ such that $TR_M(n) < c$ for all $n \in \mathbb{N}$.

Proof. The proof is indirect. Suppose that there is an infinite sequence of strings $(v_i)_{i\in\mathbb{N}}$ such that $TR_M(v_{i+1}) > TR_M(v_i)$ for all $i \in \mathbb{N}$. Moreover, without loss of generality, we can choose the strings v_i in a way such that for all $i \in \mathbb{N}$ we have

$$TR_M(u) < TR_M(v_i) \quad \text{for all strings } u \text{ with } |u| < |v_i| \,. \tag{7}$$

Regular Languages VI

Claim 1. *Then among all traces* $TR_M(v_i, j)$ *with* $0 \leqslant j \leqslant |v_i|$ *there are no three identical ones.*

Suppose the converse, i.e., there are j_1, j_2, j_3 with $0 \leqslant j_1 < j_2 < j_3 \leqslant |v_i|$ such that $TR_M(v_i, j_1) = TR_M(v_i, j_2) = TR_M(v_i, j_3)$. Let $j_* \in \mathbb{Z}$, $0 \leqslant j_* \leqslant |v_i|$, be such that $TR_M(v_i, j_*) \geqslant TR_M(v_i, j)$ for all $j \in \{0, \ldots, |v_i|\}$. If there is more than one j_*, we take the smallest one.

Regular Languages VI

Claim 1. *Then among all traces* $TR_M(v_i, j)$ *with* $0 \leqslant j \leqslant |v_i|$ *there are no three identical ones.*

Suppose the converse, i.e., there are j_1, j_2, j_3 with $0 \leqslant j_1 < j_2 < j_3 \leqslant |v_i|$ such that $TR_M(v_i, j_1) = TR_M(v_i, j_2) = TR_M(v_i, j_3)$. Let $j_* \in \mathbb{Z}$, $0 \leqslant j_* \leqslant |v_i|$, be such that $TR_M(v_i, j_*) \geqslant TR_M(v_i, j)$ for all $j \in \{0, \ldots, |v_i|\}$. If there is more than one j_*, we take the smallest one.

Case 1. $j_* = j_1$.

We can delete all symbols in v_i between j_2 and j_3 without changing $TR_M(v_i, j_*)$, thus obtaining a contradiction to (7).

Regular Languages VI

Claim 1. *Then among all traces* $TR_M(v_i, j)$ *with* $0 \leqslant j \leqslant |v_i|$ *there are no three identical ones.*

Suppose the converse, i.e., there are j_1, j_2, j_3 with $0 \leqslant j_1 < j_2 < j_3 \leqslant |v_i|$ such that $TR_M(v_i, j_1) = TR_M(v_i, j_2) = TR_M(v_i, j_3)$. Let $j_* \in \mathbb{Z}$, $0 \leqslant j_* \leqslant |v_i|$, be such that $TR_M(v_i, j_*) \geqslant TR_M(v_i, j)$ for all $j \in \{0, \ldots, |v_i|\}$. If there is more than one j_*, we take the smallest one.

Case 1. $j_* = j_1$.

We can delete all symbols in v_i between j_2 and j_3 without changing $TR_M(v_i, j_*)$, thus obtaining a contradiction to (7).

Case 2. $j_* \neq j_1$.

Then either $j_* \in (j_1, j_2)$, $j_* \in (j_2, j_3)$ or it does not fall in any of the two intervals (j_1, j_2) and (j_2, j_3). So there is one interval such that $j_* \notin (j_a, j_b)$, $a, b \in \{1, 2, 3\}$, $a \neq b$. We can delete all symbols between j_a and j_b without changing $TR_M(v_i, j_*)$, a contradiction to (7). Claim 1 follows.

Regular Languages VII

Hence, $|v_i|$ is bounded by twice the number of *different* traces of length at most $TR_M(v_i)$. Let $M = [B, Z, A]$, and recall that $|Z| \geqslant 2$. We obtain

$$|v_i| \quad \leqslant \quad 2 \cdot \sum_{j=0}^{TR_M(v_i)} |Z|^j = 2 \cdot \frac{|Z|^{TR_M(v_i)+1} - 1}{|Z| - 1} .$$

Regular Languages VII

Hence, $|v_i|$ is bounded by twice the number of *different* traces of length at most $TR_M(v_i)$. Let $M = [B, Z, A]$, and recall that $|Z| \geqslant 2$. We obtain

$$|v_i| \quad \leqslant \quad 2 \cdot \sum_{j=0}^{TR_M(v_i)} |Z|^j = 2 \cdot \frac{|Z|^{TR_M(v_i)+1} - 1}{|Z| - 1} .$$

Thus, taking logarithms to the base $|Z|$ directly yields

$$\log_{|Z|} |v_i| \quad \leqslant \quad \log_{|Z|} \left(\frac{2}{|Z| - 1} \right) + TR_M(v_i) + 1 .$$

So there is a constant $c > 0$ such that $\log_2 |v_i| \leqslant c \cdot TR_M(v_i)$ and we have $\log_2 |v_i| = O(TR_M(v_i))$. Thus, $TR_M(n)/\log n \geqslant TR_M(v_i)/\log |v_i| \geqslant 1/c$, a contradiction to $TR_M(n) = o(\log n)$. ∎

Regular Languages VIII

Now we are in a position to show the following gap-theorem:

Theorem 2 (Trakhtenbrot (1964), Hartmanis (1968))

Let M *be a deterministic one-tape TM and assume* $T_M(n) = o(n \log n)$. *Then* $L(M) \in \mathcal{REG}$.

Regular Languages VIII

Now we are in a position to show the following gap-theorem:

Theorem 2 (Trakhtenbrot (1964), Hartmanis (1968))

Let M be a deterministic one-tape TM and assume $T_M(n) = o(n \log n)$. Then $L(M) \in \mathcal{REG}$.

Proof. Let $M = [B, Z, A]$ and suppose, $L(M) \notin \mathcal{REG}$. By Lemmata 2 and 3, we conclude that $TR_M(n)$ is unbounded. As in the proof of Lemma 3 we choose an infinite sequence $(v_i)_{i \in \mathbb{N}}$ of strings such that $TR_M(v_{i+1}) > TR_M(v_i)$ for all $i \in \mathbb{N}$, and such that (7) is fulfilled. Also, there is a constant $c > 0$ such that $\log_2 |v_i| \leqslant c \cdot TR_M(v_i)$.

Regular Languages VIII

Now we are in a position to show the following gap-theorem:

Theorem 2 (Trakhtenbrot (1964), Hartmanis (1968))

Let M be a deterministic one-tape TM and assume $T_M(n) = o(n \log n)$. Then $L(M) \in \mathcal{REG}$.

Proof. Let $M = [B, Z, A]$ and suppose, $L(M) \notin \mathcal{REG}$. By Lemmata 2 and 3, we conclude that $TR_M(n)$ is unbounded. As in the proof of Lemma 3 we choose an infinite sequence $(v_i)_{i \in \mathbb{N}}$ of strings such that $TR_M(v_{i+1}) > TR_M(v_i)$ for all $i \in \mathbb{N}$, and such that (7) is fulfilled. Also, there is a constant $c > 0$ such that $\log_2 |v_i| \leqslant c \cdot TR_M(v_i)$.
By construction there are at least $(1/2)|v_i|$ many different traces in positions $j = 0, \ldots, |v_i|$. The number of traces of length k is $|Z|^k$. Assume that the traces of minimum possible length occur.

Regular Languages IX

We lower bound the minimum a such that $\sum_{k=0}^{a} |Z|^k \geqslant \frac{1}{2}|v_i|$.

This results in $(|Z|^{a+1} - 1)/(|Z| - 1) \geqslant (1/2)|v_i|$, and therefore, we have the condition $|Z|^{a+1} \geqslant (1/2)|v_i|(|Z| - 1)$. Taking logarithms to the base $|Z|$ yields
$a + 1 \geqslant \log_{|Z|}((1/2)|v_i|(|Z| - 1))$. Since $|Z| \geqslant 2$ and since $\log n$ is monotonically increasing, we obtain

$$a \geqslant \log_{|Z|}\left(\frac{1}{2}|v_i|\right) - 1 \,. \tag{8}$$

Regular Languages IX

We lower bound the minimum a such that $\sum_{k=0}^{a} |Z|^k \geqslant \frac{1}{2}|v_i|$.

This results in $(|Z|^{a+1} - 1)/(|Z| - 1) \geqslant (1/2)|v_i|$, and therefore, we have the condition $|Z|^{a+1} \geqslant (1/2)|v_i|(|Z| - 1)$. Taking logarithms to the base $|Z|$ yields
$a + 1 \geqslant \log_{|Z|}((1/2)|v_i|(|Z| - 1))$. Since $|Z| \geqslant 2$ and since $\log n$ is monotonically increasing, we obtain

$$a \geqslant \log_{|Z|}\left(\frac{1}{2}|v_i|\right) - 1\,. \tag{8}$$

Thus, by lower bounding $T_M(|v_i|)$ by the sum of the lengths of all traces of minimum length that must occur, by (8), we obtain the following:

Regular Languages X

$$
\begin{aligned}
T_M(|v_i|) &\geqslant 2 \cdot \sum_{k=0}^{a} k \cdot |Z|^k \geqslant 2a \cdot |Z|^a \\
&\geqslant 2 \cdot \left(\log_{|Z|} \left((1/2)|v_i| \right) - 1 \right) \left(|Z|^{\log_{|Z|}((1/2)|v_i|)-1} \right) \\
&= 2 \cdot \left(\log_{|Z|} \left(\frac{1}{2}|v_i| \right) - 1 \right) \left(\frac{|Z|^{\log_{|Z|}((1/2)|v_i|)}}{|Z|} \right) \\
&= \frac{1}{|Z|} \left(|v_i| \log_{|Z|} \left(\frac{1}{2}|v_i| \right) - |v_i| \right) \\
&= \frac{1}{|Z|} \left(|v_i| \log_{|Z|} |v_i| - |v_i|(1 + \log_{|Z|} 2) \right) .
\end{aligned}
$$

Regular Languages X

$$
\begin{aligned}
T_M(|v_i|) &\geqslant 2 \cdot \sum_{k=0}^{a} k \cdot |Z|^k \geqslant 2a \cdot |Z|^a \\
&\geqslant 2 \cdot \left(\log_{|Z|} \left((1/2)|v_i| \right) - 1 \right) \left(|Z|^{\log_{|Z|}((1/2)|v_i|)-1} \right) \\
&= 2 \cdot \left(\log_{|Z|} \left(\frac{1}{2}|v_i| \right) - 1 \right) \left(\frac{|Z|^{\log_{|Z|}((1/2)|v_i|)}}{|Z|} \right) \\
&= \frac{1}{|Z|} \left(|v_i| \log_{|Z|} \left(\frac{1}{2}|v_i| \right) - |v_i| \right) \\
&= \frac{1}{|Z|} \left(|v_i| \log_{|Z|} |v_i| - |v_i|(1 + \log_{|Z|} 2) \right) .
\end{aligned}
$$

We conclude $T_M(n) \neq o(n \log n)$, a contradiction. Hence, we have shown that $L(M) \in \mathcal{REG}$. ∎

Regular Languages XI

Note that the bound proved in Theorem 2 cannot be improved. For seeing this, we recommend to solve the following exercise:

Exercise 1. *Consider the language* $L = \{0^n 1^n \mid n \in \mathbb{N}\} \notin \mathcal{REG}$. *Prove that there is a deterministic one-tape Turing machine* M *accepting* L *in time* $n \log n$.

Remarks

On the one hand, the Theorems 1 and 2 are very strong. On the other hand, they also show that just having one tape is causing a lot of work which could be avoided if the TM would have more than one tape. The need for more than one tape is also evident if we wish to study space complexity.

Remarks

On the one hand, the Theorems 1 and 2 are very strong. On the other hand, they also show that just having one tape is causing a lot of work which could be avoided if the TM would have more than one tape. The need for more than one tape is also evident if we wish to study space complexity.

If we would consider again deterministic one-tape Turing machines, then the problem is ill posed, since we need already n cells to write the input on the tape and thus could not study sublinear space complexity classes. The idea is to consider Turing machines having at least an input-tape with a head that is only allowed to read, and a work-tape with read-write head. But our goal is a bit more far reaching, thus we are going to consider the more general case of having k tapes, where $k \geqslant 2$.

k-Tape Turing Machines I

Definition 4

A TM $M = [B, Z, A]$ is called *deterministic k-tape TM*, $k \geqslant 2$, provided M has an input-tape with read-only head and $k - 1$ many work-tapes each of which possesses exactly one read-write head.
M works as the previously defined one-taped TM except that now in every step k heads are moved and k cells (one on each tape) are observed. Thus formally, now we have

$$A \subseteq Z \setminus \{z_f\} \times B^k \times (B \times \{L, N, R\})^k \times Z\,,$$

with the restriction that M is not allowed to write on its input-tape.

k-Tape Turing Machines II

Initially, M is in state z_s (the start state) and all heads are observing the first cell located right to position 0. Thus, we also have to modify the definition of accepting a language. This is done as follows:

k-Tape Turing Machines III

Definition 5

A language $L \subseteq \Sigma^*$ is *accepted* by a deterministic k-tape TM M if for every string $w \in \Sigma^*$ the following conditions are satisfied:
If w is written on the empty input-tape of M beginning in cell 0 and the TM M is started such that the read-only head on the input-tape is put on the leftmost symbol of w and all other heads are put on the first cell located right to position 0 in state z_s then M stops after having executed finitely many steps in state z_f. Moreover,

(1) if $w \in L$, the cell observed by M on its first work-tape in state z_f contains a $|$.
In this case we also write $M(w) = |$.

(2) If $w \notin L$, the cell observed by M on its first work-tape in state z_f contains a $*$.
In this case we also write $M(w) = *$.

Space Complexity I

Again, we use $L(M)$ to denote the language accepted by TM M. Next, we define what is meant by space complexity.

Definition 6

Let M be a deterministic k-tape TM, $k \geqslant 2$, let $w \in \Sigma^*$ be M's input. Then we define the *space complexity* of M on input w to be the number of cells visited by M on its work-tapes when started in its initial configuration as described above until it stops.

Space Complexity I

Again, we use $L(M)$ to denote the language accepted by TM M. Next, we define what is meant by space complexity.

Definition 6

Let M be a deterministic k-tape TM, $k \geqslant 2$, let $w \in \Sigma^*$ be M's input. Then we define the *space complexity* of M on input w to be the number of cells visited by M on its work-tapes when started in its initial configuration as described above until it stops.

We denote the space complexity of M on input w by $S_M(w)$. Furthermore, for every $n \in \mathbb{N}$ we set

$$S_M(n) \quad =_{df} \quad \max\{S_M(w) \mid w \in \Sigma^n\}.$$

Complexity Classes

The time complexity is defined as before. Finally, we define the following complexity classes. Let $f\colon \mathbb{N} \to \mathbb{N}$ be a function. We shall refer to f as a *bounding function*. Furthermore, we set

$$
\begin{aligned}
\mathit{Time}_k(f(n)) &=_{df} \{L(M) \mid M \text{ is a deterministic } k\text{-tape TM} \\
&\qquad \text{and } T_M(n) \leqslant f(n) \text{ for all } n \in \mathbb{N}\}\,. \\
\mathit{Space}_k(f(n)) &=_{df} \{L(M) \mid M \text{ is a deterministic } k\text{-tape TM} \\
&\qquad \text{and } S_M(n) \leqslant f(n) \text{ for all } n \in \mathbb{N}\}\,. \\
\mathit{TIME}(f(n)) &=_{df} \{L(M) \mid M \text{ is a deterministic TM} \\
&\qquad \text{and } T_M(n) \leqslant f(n) \text{ for all } n \in \mathbb{N}\}\,. \\
\mathit{SPACE}(f(n)) &=_{df} \{L(M) \mid M \text{ is a deterministic TM} \\
&\qquad \text{and } S_M(n) \leqslant f(n) \text{ for all } n \in \mathbb{N}\}\,.
\end{aligned}
$$

Constant Factor Speed-up

Next, we prove a constant factor speed-up theorem.

Theorem 3

For every constant $c > 0$, $c \in \mathbb{N}$, *and every function* $f\colon \mathbb{N} \to \mathbb{N}$, *such that* $c \cdot f(n) \geqslant n$ *for all* n *we have* $TIME(f(n)) = TIME(c \cdot f(n))$.

Constant Factor Speed-up

Next, we prove a constant factor speed-up theorem.

Theorem 3

For every constant $c > 0$, $c \in \mathbb{N}$, *and every function* $f\colon \mathbb{N} \to \mathbb{N}$, *such that* $c \cdot f(n) \geqslant n$ *for all* n *we have* $TIME(f(n)) = TIME(c \cdot f(n))$.

Proof. We only sketch the proof here. The remaining details are left as an exercise. Given a Turing machine $M = [B, Z, A]$, one can construct a Turing machine M' working c-times faster than M. Turing machine M' simulates in each step c steps of M by using the tape alphabet B^c and the appropriately defined state set. ∎

Constructibility

Thus, in the following it is always sufficient to show that $T_M(n) = O(f(n))$ instead of proving $T_M(n) \leqslant f(n)$.

Next, we define the important notions of (weak) space (abbr. S) and time (abbr. T) constructibility for functions.

Constructibility I

Definition 7

Let a function $f\colon \mathbb{N} \to \mathbb{N}$ be given.

(1) The function f is said to be *weakly S-constructible* if there is a deterministic Turing machine M such that $S_M(n) \leqslant f(n)$ and for every $n \in \mathbb{N}$ there exists a string $w \in \Sigma^*$ with $|w| = n$ and $S_M(w) = f(n)$.

(2) The function f is said to be *S-constructible* if there is a deterministic Turing machine M such that for all $w \in \Sigma^*$ the equality $S_M(w) = f(|w|)$ is fulfilled. M is then called a *marker* for space $f(n)$.

Constructibility II

Definition 8

Let a function $f\colon \mathbb{N} \to \mathbb{N}$ be given.

(1) The function f is said to be *weakly T-constructible* if there is a deterministic Turing machine M such that $T_M(n) \leqslant f(n)$ and for every $n \in \mathbb{N}$ there exists a string $w \in \Sigma^*$ with $|w| = n$ and $T_M(w) = f(n)$.

(2) The function f is said to be *T-constructible* if there is a deterministic Turing machine M such that for all $w \in \Sigma^*$ the equality $T_M(w) = f(|w|)$ is fulfilled. M is then called a *clock* for time $f(n)$.

Examples

We continue with some examples. The proof of the assertions made is left as exercise.

Examples

We continue with some examples. The proof of the assertions made is left as exercise.

The functions $f(n) = n$ and $f(n) = 2^n$ are both S-constructible and T-constructible.

Examples

We continue with some examples. The proof of the assertions made is left as exercise.

The functions $f(n) = n$ and $f(n) = 2^n$ are both S-constructible and T-constructible.

The sum and product of S-constructible and T-constructible functions are again S-constructible and T-constructible, respectively.
Consequently, all polynomials are both S-constructible and T-constructible.

Examples

We continue with some examples. The proof of the assertions made is left as exercise.

The functions $f(n) = n$ and $f(n) = 2^n$ are both S-constructible and T-constructible.

The sum and product of S-constructible and T-constructible functions are again S-constructible and T-constructible, respectively.
Consequently, all polynomials are both S-constructible and T-constructible.

However, the function $f(n) = n \log \log n$ is *not* T-constructible.

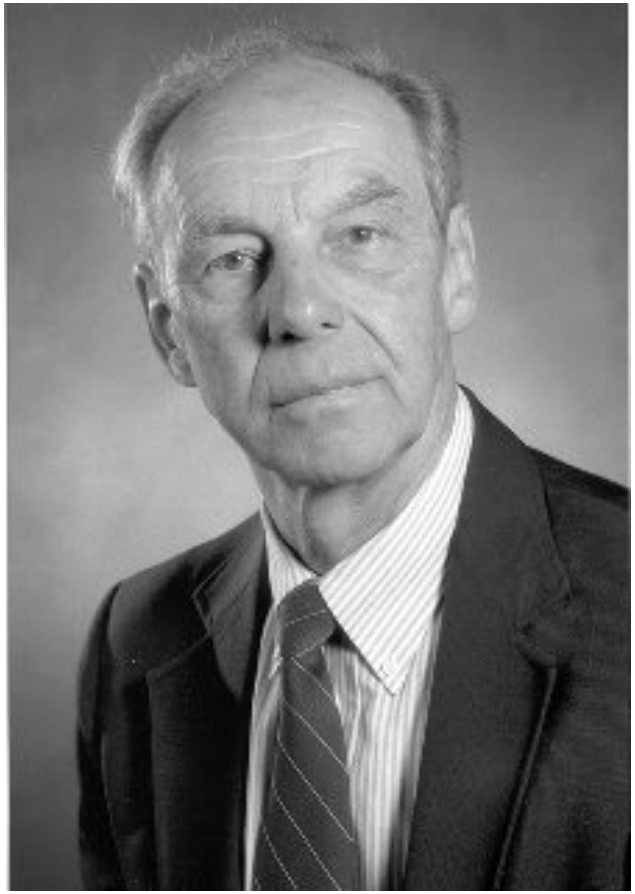
Remark

A technical remark is mandatory here. We defined bounding functions as functions that map $\mathbb{N}$ to $\mathbb{N}$. Therefore, when looking at functions such as $n \log n$, then we implicitly assume the logarithmic function $\log_2 n$ to the base 2 to be replaced by its integer valued counterpart, i.e., $\log n =_{df} 1$ if $n \leqslant 1$ and $\log n =_{df} \lceil \log_2 n \rceil$ if $n > 1$.

Thank you!

Jānis Bārzdiņš

Boris A. Trakhtenbrot



Juris Hartmanis