Complexity Theory
WS 2009/10

Prof. Dr. Erich Grädel

Mathematische Grundlagen der Informatik
RWTH Aachen

Contents

1 Deterministic Turing Machines and Complexity Classes

1.1 Turing machines

The simplest model of a Turing machine (TM) is the deterministic 1-tape Turing machine. Despite its simplicity, this model is sufficiently general to capture the notion of computability and allows us to define a very intuitive concept of computational complexity. During this course we will also use more general models of computation with the following facilities:

- a separate read-only input tape;
- a separate write-only output tape;
- more general types of memory, e.g., k linear tapes (for $k \geq 1$), higher-dimensional memory space, etc.

The corresponding definitions of configurations, computations, etc. need to be adjusted accordingly. We will do this for one specific model.

Definition 1.1. A *(deterministic) Turing machine with separate input and output tapes and k working tapes* is given by

$$M = (Q, \Gamma_{\text{in}}, \Gamma_{\text{out}}, \Sigma, q_0, F, \delta)$$

where

- Q is a finite set of states,
- Σ is the finite working alphabet, with a distinguished symbol $\square$ (blank),
- $\Gamma_{\text{in}}, \Gamma_{\text{out}}$ are the input and output alphabets (often $\Gamma_{\text{in}}, \Gamma_{\text{out}} = \Sigma$),
- $q_0 \subseteq Q$ is the initial state,
- $F \subseteq Q$ is the set of final states, and

- $\delta : (Q \setminus F) \times \Gamma_{\text{in}} \times \Sigma^k \to$
  $Q \times \{-1, 0, 1\} \times \Sigma^k \times \{-1, 0, 1\}^k \times (\Gamma_{\text{out}} \cup \{*\})$
  is the transition function.

A *configuration* is a complete description of all relevant data at a certain moment of the computation (state, memory contents, input, etc.). It is useful to distinguish between *partial* and *total* configurations.

Definition 1.2. Let M be a Turing machine. A *partial configuration* of M is a tuple $C = (q, w_1, \ldots, w_k, p_0, p_1, \ldots, p_k) \in Q \times (\Sigma^*)^k \times \mathbb{N}^{k+1}$, where

- q is the current state,
- $w_1, \ldots, w_k$ are the inscriptions on the working tapes,
- p_0 is the position on the input tape, and
- $p_1, \ldots, p_k$ are the positions on the working tapes.

The inscription of the ith working tape is given by a finite word $w_i = w_{i_0} \ldots w_{i_m} \in \Sigma^*$. There are only blanks on the fields $j > m$ of the infinite tape. When, in addition to a partial configuration, the inscriptions of the input and output tapes are given, one obtains a *total configuration* of M.

The *total initial configuration* $C_0(x)$ of M on $x \in \Gamma_{\text{in}}^*$ is given by

$$C_0(x) = (q_o, \varepsilon, \ldots, \varepsilon, 0, \ldots, 0, x, \varepsilon)$$

with

- the initial state q_0,
- empty working tapes, that is, $w_1 = w_2 = \cdots = w_k = \varepsilon$, (we denote the empty word by ε),
- position 0 on all tapes,
- the inscription x on the input tape, and
- the inscription ε on the output tape.

Remark 1.3. A *final configuration* is a configuration $C = (q, \overline{w}, \overline{p}, x, y)$ with $q \in F$. The word y (the inscription on the output tape) is the *output* of the final configuration C.

Successor configuration. Let $C = (q, w_1, \ldots, w_k, p_0, p_1, \ldots, p_k, x, y)$ be a (total) configuration of a Turing machine M. The transition

to the next configuration is determined by the value of the transition function δ on the current state q, and the values that have been read while in C, i.e., the symbols x_{p_0} read from the input tape and the symbols $w_{1\,p_1}, \ldots, w_{k\,p_k}$ read from the working tapes. Let $\delta(q, x_{p_0}, w_{1\,p_1}, \ldots, w_{k\,p_k}) = (q', m_0, a_1, \ldots, a_k, m_1, \ldots, m_k, b)$. Then $\Delta(C) := (q', \overline{w}', \overline{p}', x, y')$ is a successor configuration of C if

- w_i' results from w_i by replacing the symbol $w_{i\,p_i}$ with a_i,
- $p_i' = p_i + m_i$ (for $i = 0, \ldots, k$) and
- $y' = \begin{cases} y & \text{if } b = *, \\ yb & \text{if } b \in \Gamma_{\text{out}}. \end{cases}$

Notation. We write $C \vdash_M C'$, if $C' = \Delta(C)$.

Definition 1.4. A *computation* of M on x is a sequence $C_0, C_1, \ldots$ of (total) configurations of M with $C_0 = C_0(x)$ and $C_i \vdash_M C_{i+1}$ for all $i \geq 0$. The computation is *complete* if it is either infinite or it ends in a final configuration.

The *function computed by* M is a partial function $f_M : \Gamma_{\text{in}}^* \to \Gamma_{\text{out}}^*$. Thereby $f_M(x) = y$ iff the complete computation of M on x is finite and ends in a final configuration with output y.

Definition 1.5. A *k-tape acceptor* is a k-tape Turing machine M ($k \geq 1$), whose final states F are partioned into a set F^+ of *accepting* states and a set F^- of *rejecting* states. *M accepts x*, iff the computation of M on x halts in a state $q \in F^+$. *M rejects x*, iff the computation of M on x halts in a state $q \in F^-$.

Definition 1.6. Let $L \subseteq \Gamma_{\text{in}}^*$ be a language. *M decides L* if M accepts all $x \in L$ and rejects all $x \in \Gamma_{\text{in}}^* \setminus L$. L is *decidable* if there exists an acceptor M that decides L. We will write $L(M)$ to denote the set of inputs accepted by M.

In the following, we will often also use k-tape Turing machines without distinguished input and output tapes. In these cases the first working tape will also be the input tape while some other tape (or tapes) will overtake the role of the output tape.

Conventions. As long as not specified in a different way:

- a Turing machine (TM) shall be a k-tape Turing machine (for every $k \geq 1$), where k denotes the total number of tapes, possibly including separate input and output tapes;
- Γ shall stand for the input alphabet.

1.2 Time and space complexity classes

Definition 1.7. Let M be a Turing machine and x some input. Then $\text{time}_M(x)$ is the length of the complete computation of M on x and $\text{space}_M(x)$ is the total number of working tape cells used in the computation of M on x. Let $T, S : \mathbb{N} \to \mathbb{R}^{\geq 0}$ be monotonically increasing functions. A TM M is

- *T-time bounded* if $\text{time}_M(x) \leq T(|x|)$ for all inputs $x \in \Gamma^*$, and
- *S-space bounded* if $\text{space}_M(x) < S(|n|)$ for all inputs $x \in \Gamma^*$.

Definition 1.8.

(i) $\textsc{Dtime}_k(T)$ is the set of all languages L for which there exists a T-time bounded k-tape TM that decides L.
(ii) $\textsc{Dspace}_k(S)$ is the set of all languages L for which there exists a S-space bounded k-tape TM that decides L.
(iii) $\textsc{Dtime}(T) = \bigcup_{k \in \mathbb{N}} \textsc{Dtime}_k(T)$.
(iv) $\textsc{Dspace}(S) = \bigcup_{k \in \mathbb{N}} \textsc{Dspace}_k(S)$.
(v) $\textsc{Dtime-space}_k(T, S)$ is the set of all languages L for which there is a T-time bounded and S-space bounded k-tape TM that decides L.
(vi) $\textsc{Dtime-space}(T, S) = \bigcup_{k \in \mathbb{N}} \textsc{Dtime-space}_k(T, S)$.

Important complexity classes are:

- $\textsc{Logspace} := \bigcup_{d \in \mathbb{N}} \textsc{Dspace}(d \log n)$,
- $(\textsc{Ptime} =)\ \mathrm{P} := \bigcup_{d \in \mathbb{N}} \textsc{Dtime}(n^d)$,
- $\textsc{Pspace} := \bigcup_{d \in \mathbb{N}} \textsc{Dspace}(n^d)$,
- $\textsc{Exptime} := \bigcup_{d \in \mathbb{N}} \textsc{Dtime}(2^{n^d})$,
- $\textsc{Expspace} := \bigcup_{d \in \mathbb{N}} \textsc{Dspace}(2^{n^d})$.

Attention: Some authors may also define $\textsc{Exptime}$ as $\bigcup_{d \in \mathbb{N}} \textsc{Dtime}(2^{dn})$.

Elementary observations on the relationship between time and space complexity lead to the following statements.

Theorem 1.9.

(a) $\textsc{Dtime}(T) \subseteq \textsc{Dspace}(O(T))$ for all functions $T : \mathbb{N} \to \mathbb{N}$.
(b) $\textsc{Dspace}(S) \subseteq \textsc{Dtime}\big(2^{O(S)}\big)$ for all functions $S\colon \mathbb{N} \to \mathbb{N}$ with $S(n) \geq \log n$.

Proof. (a) A k-tape Turing machine can visit at most k fields in one step. (b) Because $L \in \textsc{Dspace}(S)$, we can assume that L is decided by a TM M with one input tape and k working tapes using space S.

For every input x (and $n = |x|$), any partial configuration is obtained at most once during the computation of M on x. Otherwise, M would run in an endless loop and could not decide L. The number of partial configurations with space $S(n)$ is bounded by

$$|Q| \cdot (n+1) \cdot S(n)^k \cdot |\Sigma|^{S(n)} = 2^{O(S(n))}, \text{ whenever } S(n) \geq \log n.$$

Here, $(n+1)$ is the number of possible positions of the input tape, $S(n)^k$ the number of positions of the working tapes and $|\Sigma|^{k \cdot S(n)}$ the number of possible memory contents. Thus, $\text{time}_M(x) \leq 2^{O(S(n))}$. Q.E.D.

Corollary 1.10. $\textsc{Logspace} \subseteq \mathrm{P} \subseteq \textsc{Pspace} \subseteq \textsc{Exptime}$.

Theorem 1.11 (Tape reduction)**.** Let $S(n) \geq n$. Then

$$\textsc{Dtime-space}(T, S) \subseteq \textsc{Dtime-space}_1(O(T \cdot S), S).$$

Proof. (Simulation of a k-tape TM using a 1-tape TM.) Let M be a T-time bounded and S-space bounded k-tape TM that decides L. The idea is to simulate the k tapes of M using $2k$ tracks on a single tape of a 1-tape TM M'. Track $2j-1$ of the tape of M' will contain the inscription on tape j of M and track $2j$ a mark $(*)$ at the current head position of tape j of M.

Before simulating a single step of M, the head of M' is at the first (leftmost) mark. The simulation is accomplished in three phases.

(i) M' moves to the right up to the last mark and saves (in the state set) the symbols at the current positions of M, that is, the information needed to determine the transition of M. Time needed: at most $S(n)$ steps.

(ii) M' determines the transition taken by M. This takes one step.

(iii) M' returns to the first mark performing on its way back all necessary changes on the tape. Time needed: at most $S(n)$ steps.

M' accepts (or rejects) iff M accepts (or rejects). At most $S(n)$ fields contain information. Therefore, the marks are at most $S(n)$ fields apart. The simulating 1-tape TM thus needs $O(S(n))$ steps and no additional memory to simulate a step of M. The claim follows. Q.E.D.

Where did we use that $S(n) \geq n$? Consider an S-space bounded 2-tape Turing machine M, where $S(n) < n$ and where the first tape is a separate input tape. As long as M is reading the whole input, the simulating 1-tape TM will have to go to the rightmost position on the first tape to set the marks. This way, the two marks can be more than $S(n)$ fields away from each other.

Corollary 1.12. $\textsc{Dtime}(T) \subseteq \textsc{Dtime}_1(O(T^2))$.

This follows from Theorem 1.11 using the fact that $\text{space}_M(x) \leq O(\text{time}_M(x))$ for all M and all x. We also obtain:

Corollary 1.13. $\textsc{Dspace}(S) \subseteq \textsc{Dspace}_1(S)$ for $S(n) \geq n$.

To simulate a k-tape TM using a 2-tape TM, the time complexity increases by a logarithmic factor only.

Theorem 1.14. $\textsc{Dtime}(T) \subseteq \textsc{Dtime}_2(O(T \cdot \log T))$ for $T(n) \geq n$.

Proof (Outline). A k-tape TM M is simulated using a 2-tape TM M':

- 2 tracks on the first tape of M' are created for every tape of M.
- The second tape of M' is only used as intermediate memory for copy operations.

The first tape of M' is divided into blocks $\ldots, B_{-i}, B_{-i+1}, \ldots, B_{-1}, B_0, B_1, \ldots, B_i$, where $|B_0| = 1, |B_j| = 2^{|j|-1}$ for $j \neq 0$. All characters currently read by M can be found in block B_0. If the head on one track of M moves to the left, M' moves the entire inscription on the corresponding tapes to the right. This way, the current character will again be at position B_0. A clever implementation of this idea leads to a simulation with at most logarithmic time loss: if M is T-time bounded, then M' is $O(T \cdot \log T)$-time bounded. Q.E.D.

The complete proof can be found, e.g., in J. Hopcraft, J. Ullmann: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley 1979, pp. 292–295.

1.3 Speed-up and space compression

Definition 1.15. For functions $f, g : \mathbb{N} \to \mathbb{R}$, we write $f = o(g)$ to denote $\lim_{n\to\infty} f(n)/g(n) = 0$.

Theorem 1.16 (Speed-up theorem)**.**

$$\textsc{Dtime}_k(T) \subseteq \textsc{Dtime}_k(\max(n, \varepsilon \cdot T(n)))$$

for all $k > 1$, $\varepsilon > 0$, and $T\colon \mathbb{N} \to \mathbb{R}^{\geq 0}$ with $n = o(T(n))$.

Proof. Let M be a k-tape TM that decides L in time $T(n)$. Choose m in such way that $\varepsilon \cdot m \geq 16$. Let Σ be the working alphabet of M. We will construct a k-tape TM M' that uses the working alphabet $\Sigma \cup \Sigma^m$ so that it can encode m symbols of M by a single symbol. This way the computation can be speeded up.

(1) M' copies the input to a different tape compressing m symbols into one. Then, M' treats this working tape as the input tape. Time needed: n steps for copying and $\lceil \frac{n}{m} \rceil$ steps to return the head to the first symbol of the compressed input.

(2) M' simulates m steps of M taking 8 steps at a time. The following operations are executed on the working tapes:

(a) M' saves the contents of both neighboring fields of the current field "in the state set". This needs 4 steps: one to the left, two to the right, and one to the left again.

(b) M' determines the result of the next m steps of M. This is hard-coded in the transition function of M'. In m steps, M can only use or change fields that are at most m steps away from each other. In other words, it can only visit the current field of M' and both neighboring fields. Hence, M' needs 4 steps to implement this change.

(c) M' accepts or rejects iff M accepts or rejects, respectively.

Let x be an input of length n. Then

$$\text{time}_{M'}(x) \leq n + \lceil n/m \rceil + 8\lceil T(n)/m \rceil \leq n + n/m + 8T(n)/m + 2.$$

Since $n = o(T(n))$, for every $d > 0$, there is an n_d so that $T(n)/n \geq d$ for all $n \geq n_d$. Therefore, $n \leq T(n)/d$ for $n \geq n_d$. For $n \geq max(2, n_d)$, we obtain $2n \geq n + 2$. Thus, M' needs at most

$$2n + \frac{n}{m} + 8\frac{T(n)}{m} \leq T(n)\left(\frac{2}{d} + \frac{1}{md} + \frac{8}{m}\right) = T(n)\left(\frac{2m + 8d + 1}{md}\right)$$

steps. Set $d = \frac{2m+1}{8}$. Then the number of steps of M' is bounded by

$$T(n)\left(\frac{8(2m + 1 + 2m + 1)}{m(2m + 1)}\right) = \frac{16}{m}T(n) \leq \varepsilon T(n)$$

for all $n \geq max(2, n_d)$. The finite number of inputs of length $< n_d$ can be accepted in n_d time. Q.E.D.

Corollary 1.17.

$$\textsc{Dtime}(T(n)) = \textsc{Dtime}(max(n, \varepsilon \cdot T(n))$$

for all $T : \mathbb{N} \to \mathbb{R}$ with $n = o(T(n))$ and all $\varepsilon > 0$.

A similar but easier proof shows the following.

Theorem 1.18 (Space compression)**.**

$$\textsc{Dspace}(S) \subseteq \textsc{Dspace}(max(1, \varepsilon \cdot S(n))$$

for all functions $S : \mathbb{N} \to \mathbb{R}^{\geq 0}$ and all $\varepsilon > 0$.

1.4 The Gap Theorem

In this and the following section, we address the question whether one is able to solve more problems when more ressources are provided. If S_2 increases faster than S_1, does this mean that $\textsc{Dspace}(S_2) \supsetneq \textsc{Dspace}(S_1)$ (and analogously for time)? We will show that this does not hold in the general case. Towards this, we will first prove the following lemma.

Lemma 1.19. Let M be a k-tape acceptor with $\max_{|x|=n} \text{space}_M(x) \leq S(n)$ for almost all n (that is, all but finitely many) and let $L(M)$ be the set of all inputs accepted by M. Then, $L(M) \in \textsc{Dspace}(S)$.

Proof. We build a k-tape acceptor M' such that $L(M') = L(M)$ and $space_{M'}(x) \leq S(|x|)$ for all x. The set $X = \{x \ : \ \text{space}_M(x) > S(|x|)\}$ is finite by definition. Hence, for inputs $x \in X$, we can hard-code the answer to $x \in L(M)$ in the transition function M' without using additional space. Q.E.D.

Theorem 1.20 (Gap Theorem)**.** For any computable total function $g : \mathbb{N} \to \mathbb{N}$ with $g(n) \geq n$, there exists a computable function $S : \mathbb{N} \to \mathbb{N}$ such that $\textsc{Dspace}(S) = \textsc{Dspace}(g \circ S)$.

Proof. Let $M_0, M_1, \ldots$ be a recursive enumeration of all Turing machines. Consider the function $S_i(n) := \max_{|x|=n} \text{space}_{M_i}(x) \cup \{\infty\}$ which returns the space required by Turing machine M_i on words of length n.

Lemma 1.21. The set $R := \{(i, n, m) \ : \ S_i(n) = m\}$ is decidable.

Proof. For every triple (i, n, m), there is a time bound $t \in \mathbb{N}$ on computations of M_i which, on inputs of length at most n, use at most m tape

Algorithm 1.1. $S(n)$

Input: n
$y := 1$
while *there is an* $i \leq n$ *with* $(i, n, y) \in P$ **do**
  choose the smallest such i
  $y := S_i(n)$
endwhile
$S(n) := y$

cells while no configuration occurs more than once. This bound t is computable from (i, n, m). By simulating t steps of M_i on the (finitely many) different inputs of length n, one can decide whether $S_i(n) = m$. Q.E.D.

We will use this result to construct a function $S : \mathbb{N} \to \mathbb{N}$ such that, for every $i \in \mathbb{N}$, either $S_i(n) \leq S(n)$ for almost all n, or $S_i(n) \geq g(S(n))$ for infinitely many n. Towards this, consider the set $P := \{(i, n, y) \in \mathbb{N}^3 \ : \ y < S_i(n) \leq g(y)\}$. By Lemma 1.21 and since g is computable, we obtain that P is decidable. Let $S : \mathbb{N} \to \mathbb{N}$ be the function defined by Algorithm 1.1. As P is decidable, S is a computable total function. It remains to show that

$$\textsc{Dspace}(g \circ S) \setminus \textsc{Dspace}(S) = \varnothing.$$

For any $L \in \textsc{Dspace}(g \circ S)$ we have $L = L(M_i)$ for some $i \in \mathbb{N}$. As $L \in \textsc{Dspace}(g \circ S)$, by definition $S_i(n) \leq g(S(n))$ holds for all $n \in \mathbb{N}$. The way S was constructed, we have $S_i(n) \leq S(n)$ for all $n \geq i$. Otherwise $S(n) < S_i(n) \leq g(S(n))$ would hold for some $i \leq n$, which is excluded by the algorithm. Hence, $S_i(n) \leq S(n)$ for almost all n and, according to Lemma 1.19, we can conclude that $L = L(M_i) \in \textsc{Dspace}(S)$. Q.E.D.

Application. Consider $g(n) = 2^n$. There exists a function S such that $\textsc{Dspace}(2^S) = \textsc{Dspace}(S)$. That is, using more space does not necessarily allow to solve more problems.

Analogously, one can show the follwing theorem on time complexity.

Theorem 1.22 (Gap Theorem for time complexity)**.** For every computable function g, there exists a computable function T with $\textsc{Dtime}(T) = \textsc{Dtime}(g \circ T)$.

Hence, there are computable functions f, g, h so that,

- $\textsc{Dtime}(f) = \textsc{Dtime}(2^f)$.
- $\textsc{Dtime}(g) = \textsc{Dtime}(2^{2^g})$.
- $\textsc{Dtime}(h) = \textsc{Dtime}\left(2^{\overset{\left.\right\}h(n) \text{ times}}{2^{\cdot^{\cdot^{\cdot^{2}}}}}}\right)$.

1.5 The Hierarchy Theorems

In the previous section, we have shown that increasing complexity bounds does not always allow us to solve more problems. We will now investigate under which conditions a complexity class is fully contained in another one. As in the proof of the undecidability of the Halting Problem for Turing machines, we will use a diagonalization argument. The proof will be kept very general with a view to complexity measures beyond time and space.

Let $\mathcal{M}$ be a class of abstract machines (e.g., 2-tape Turing machines) and R a ressource defined for machines in $\mathcal{M}$ (e.g., time or space) such that, for every machine $M \in \mathcal{M}$ and every input x, $R_M(x) \in \mathbb{N} \cup \{\infty\}$ is defined. For a function $T\colon \mathbb{N} \to \mathbb{N}$, $R(T)$ denotes the complexity class of all problems that machines in $\mathcal{M}$ with T-bounded ressource R can decide:

$$R(T) = \{L : \text{there is an } M \in \mathcal{M} \text{ deciding } L \text{ with } R_M(x) \leq T(|x|) \text{ for all } x\}.$$

Furthermore, we assume that there is an function ρ encoding machines in $\mathcal{M}$ over the alphabet $\{0,1\}$ in such way that the structure and computational behavior of M can be extracted effectively from $\rho(M)$.

Let $T, t\colon \mathbb{N} \to \mathbb{N}$ be functions, $\mathcal{M}_1$ and $\mathcal{M}_2$ classes of acceptors,

and R, r ressources defined for $\mathcal{M}_1$ and $\mathcal{M}_2$. We thus obtain the complexity classes $R(T)$ and $r(t)$.

Definition 1.23. $R(T)$ allows *diagonalization* over $r(t)$ if there exists a machine $D \in \mathcal{M}_1$ such that:

- D is T-bounded in ressource R and stops on every input. In other words, $L(D) \in R(T)$.
- For every machine $M \in \mathcal{M}_2$ that is t-bounded in ressource r,

  $$\rho(M)\#x \in L(D) \Leftrightarrow \rho(M)\#x \notin L(M).$$

  holds for almost all $x \in \{0,1\}^*$.

Theorem 1.24 (General Hierarchy Theorem)**.** If $R(T)$ allows diagonalization over $r(t)$, then $R(T) \setminus r(t) \neq \varnothing$.

Proof. Let D be the diagonalization machine from Definition 1.23. We will show that $L(D) \notin r(t)$. Otherwise, there would be a machine M that is t-bounded in ressource r with $L(D) = L(M)$. This, however, is impossible since for almost all x:

$$\rho(M)\#x \in L(D) \Leftrightarrow \rho(M)\#x \notin L(M)$$

holds. Therefore, $L(M) \neq L(D)$. Q.E.D.

Definition 1.25. A function $T\colon \mathbb{N} \to \mathbb{N}$ is called fully *time constructible* if there exists a Turing machine M such that $\text{time}_M(x) = T(|x|)$ for all x. Similarly, $S : \mathbb{N} \to \mathbb{N}$ is fully *space constructible* if $\text{space}_M(x) = S(|x|)$ holds for some Turing machine M and all x.

Time and space constructible functions are "proper" functions whose complexity is not much larger than their values. Most of the functions we usually consider are fully time and space constructible. Specifically, this is true for n^k, 2^n and $n!$. If two functions f and g have this property, the functions $f + g$, $f \cdot g$, 2^f and f^g do as well.

Theorem 1.26. Let $T, t : \mathbb{N} \to \mathbb{R}^{\geq 0}$ such that $T(n) \geq n$, with T time constructible and $t = o(T)$. Then $\textsc{Dtime}_k(t) \subsetneq \textsc{Dtime}(T)$ for all $k \in \mathbb{N}$.

Proof. We will show that $\textsc{Dtime}(T)$ allows diagonalization over $\textsc{Dtime}_k(t)$. Towards this, let D be a TM with the following properties:

(a) If M is a k-tape TM and $x \in \{0,1\}^*$, then, on input $\rho(M)\#x$, the machine D simulates the computation of M on $\rho(M)\#\rho(x)$.

(b) For each M, there is a constant c_M such that D needs at most c_M steps for the simulation of each step of M.

(c) At the same time, D simulates another TM N on separate tapes which executes precisely $T(n)$ steps on inputs of length n. By time constructability of T, such a machine exists.

(d) After $T(n)$ ($n = |\rho(M)\#x|$) steps, D stops and accepts $\rho(M)\#x$ iff the simulated computation of M on $\rho(M)\#x$ has rejected. Otherwise, if M has already accepted or the computation has yet not been completed, D rejects.

Let $L(D) = \{\rho(M)\#x \ : \ D \text{ accepts } \rho(M)\#x\}$. We have:

- $L(D) \in \textsc{Dtime}(T)$.
- For all M: $T(n) \geq c_M \cdot t(n)$ for almost all n (since $t = o(T)$). Therefore, D can simulate the computation of M on $\rho(M)\#x$ for almost all inputs $\rho(M)\#x$ in $T(n)$ steps.

Thus, $\rho(M)\#x \in L(D) \iff \rho(M)\#x \notin L(M)$. The claim follows from the General Hierarchy Theorem. Q.E.D.

Corollary 1.27 (Time Hierarchy Theorem)**.** Let $T(n) \geq n$, T be time-constructible and $t \cdot \log t = o(T)$. Then $\textsc{Dtime}(t) \subsetneq \textsc{Dtime}(T)$.

Proof. By Theorem 1.14, there is a constant c such that $\textsc{Dtime}(t) \subseteq \textsc{Dtime}_2(c \cdot t \cdot \log t)$. If $t \cdot \log t = o(T)$, then also $c \cdot t \cdot \log t = o(T)$ holds. Thus, by Theorem 1.26, there is a language

$$L \in \textsc{Dtime}(T) \setminus \textsc{Dtime}_2(c \cdot t \cdot \log t) \subseteq \textsc{Dtime}(T) \setminus \textsc{Dtime}(t). \text{ Q.E.D.}$$

Applications. As $T(n) = n^{d+1}$ is time-constructible for each $d \in \mathbb{N}$ and $t(n) = n^d \log n^d = O(n^d \log n) = o(n^{d+1}) = o(T(n))$, the following

holds:

$$\textsc{Dtime}(n^d) \subsetneq \textsc{Dtime}(n^{d+1}).$$

Corollary 1.28. For any time constructible and increasing function f with $\lim_{n\to\infty} f(n) = \infty$, the class P of all problems decidable in polynomial time is strictly included in $\textsc{Dtime}(n^{f(n)})$. In particular, $\mathrm{P} \subsetneq \textsc{Exptime}$.

Theorem 1.29 (Space Hierarchy Theorem)**.** Let $S, s : \mathbb{N} \to \mathbb{N}$ be two functions where S is space constructible and $S(n) \geq \log n$ and $s = o(S)$. Then, $\textsc{Dspace}(S) \setminus \textsc{Dspace}(s) \neq \varnothing$.

Proof. As we can reduce, by Theorem 1.11, the number of working tapes to one without increasing the space complexity, it is sufficient to consider a TM with one input and one working tape. Since M is s-space bounded, there are at most $|Q| \cdot (n+1) \cdot |\Sigma|^{s(n)} \cdot s(n) = (n+1)2^{O(s(n))}$ different partial configurations of M. The machine M therefore stops either after $\leq t(n) = 2^{c_M s(n) + \log(n+1)}$ steps or it never halts. Here, c_M denotes a constant which depends on M but not on n. Since S is space-constructible, there is a TM N with $\mathrm{space}_N(x) = S(|x|)$ for all x. It remains to show that $\textsc{Dspace}(S)$ allows diagonalization over $\textsc{Dspace}(s)$. Consider the machine D that operates on input $\rho(M)\#x$ as follows:

(a) At first, mark a range of $S(n)$ fields, by simulation of N. All subsequent operations will take place within this range. If other fields are accessed during the execution, D immediately stops and rejects the input.
(b) D initializes a counter to $t(n)$ and stores it on an extra tape.
(c) D simulates the computation of M on $\rho(M)\#x$ and decrements the counter at every simulated step.
(d) If the simulation accesses a non-marked field or M does not stop within $t(n)$ steps, then D rejects the input $\rho(M)\#x$. D also rejects if M accepts the input $\rho(M)\#x$. If D completes the simulation of a rejecting computation of M on $\rho(M)\#x$, then D accepts $\rho(M)\#x$.

We obtain:

- $L(D) \in \text{DSPACE}(S)$.
- It remains to show: if M is s-space bounded, then D can simulate the computation of M on $\rho(M)\#x$ for almost all x completely (or $t(n)$ steps of it).
  - Because $t(n) = 2^{O(s(n)+\log n)}$, $s = o(S)$ and $S(n) \geq \log n$, the counter $t(n)$ can be encoded by a word of length $S(n)$ for all n that are large enough.
  - Assuming M has an alphabet with d different symbols. Then D needs $\leq \log d$ fields to encode a symbol of M (note that this factor only depends on M but not on the input length).
  - For the simulation of M, the machine D needs at most space $\log d \cdot \text{space}_M(\rho(M)\#x) \leq \log d \cdot s(n) \leq S(n)$ for almost all n.

  For all sufficiently large x, the following holds: $\rho(M)\#x \in L(D) \iff \rho(M)\#x \notin L(M)$. Therefore, $\text{DSPACE}(S)$ allows diagonalization over $\text{DSPACE}(s)$. The claim follows with the General Hierarchy Theorem. Q.E.D.

Remark 1.30. As an immediate consequence we obtain $\text{LOGSPACE} \subsetneq \text{PSPACE}$. Thus, at least one of the inclusions $\text{LOGSPACE} \subseteq \text{P} \subseteq \text{PSPACE}$ must be strict. However, at the present time, we do not know whether $\text{LOGSPACE} \subsetneq \text{P}$ or $\text{P} \subsetneq \text{PSPACE}$.