TECHNISCHE UNIVERSITÄT DRESDEN

# FOUNDATIONS OF COMPLEXITY THEORY

**Lecture 13: Space Hierarchy and Gaps**

**David Carral**
**Knowledge-Based Systems**

TU Dresden, December 14, 2020

# Review



## Review: Time Hierarchy Theorems

**Time Hierarchy Theorem 12.12** If $f, g : \mathbb{N} \to \mathbb{N}$ are such that $f$ is time-constructible, and $g \cdot \log g \in o(f)$, then

$$\mathsf{DTime}_*(g) \subsetneq \mathsf{DTime}_*(f)$$

**Nondeterministic Time Hierarchy Theorem 12.14** If $f, g : \mathbb{N} \to \mathbb{N}$ are such that $f$ is time-constructible, and $g(n+1) \in o(f(n))$, then

$$\mathsf{NTime}_*(g) \subsetneq \mathsf{NTime}_*(f)$$

In particular, we find that P ≠ ExpTime and NP ≠ NExpTime:

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSpace} \subseteq \mathsf{ExpTime} \subseteq \mathsf{NExpTime} \subseteq \mathsf{ExpSpace}$$

$\neq$

$\neq$



# A Hierarchy for Space

## Space Hierarchy

For space, we can always assume a single working tape:

- Tape reduction leads to a constant-factor increase in space
- Constant factors can be eliminated by space compression

Therefore, $\text{DSpace}_k(f) = \text{DSpace}_1(f)$.

Space turns out to be easier to separate – we get:

**Space Hierarchy Theorem 13.1:** If $f, g : \mathbb{N} \to \mathbb{N}$ are such that $f$ is space-constructible, and $g \in o(f)$, then

$$\text{DSpace}(g) \subsetneq \text{DSpace}(f)$$

**Challenge:** TMs can run forever even within bounded space.



## Proving the Space Hierarchy Theorem (1)

**Space Hierarchy Theorem 13.1:** If $f, g : \mathbb{N} \to \mathbb{N}$ are such that $f$ is space-constructible, and $g \in o(f)$, then

$$\text{DSpace}(g) \subsetneq \text{DSpace}(f)$$

**Proof:** Again, we construct a diagonalisation machine $\mathcal{D}$. We define a multi-tape TM $\mathcal{D}$ for inputs of the form $\langle \mathcal{M}, w\rangle$ (other cases do not matter), assuming that $|\langle \mathcal{M}, w\rangle| = n$

- Compute $f(n)$ in unary to mark the available space on the working tape
- Initialise a separate countdown tape with the largest binary number that can be written in $f(n)$ space
- Simulate $\mathcal{M}$ on $\langle \mathcal{M}, w\rangle$, making sure that only previously marked tape cells are used
- Time-bound the simulation using the content of the countdown tape by decrementing the counter in each simulated step
- If $\mathcal{M}$ rejects (in this space bound) or if the time bound is reached without $\mathcal{M}$ halting, then accept; otherwise, if $\mathcal{M}$ accepts or uses unmarked space, reject



## Proving the Space Hierarchy Theorem (1)

**Proof (continued):** It remains to show that $\mathcal{D}$ implements diagonalisation:

**L**$(\mathcal{D}) \in \text{DSpace}(f)$:

- $f$ is space-constructible, so both the marking of tape symbols and the initialisation of the counter are possible in $\text{DSpace}(f)$
- The simulation is performed so that the marked $O(f)$-space is not left

There is $w$ such that $\langle \mathcal{M}, w\rangle \in$ **L**$(\mathcal{D})$ iff $\langle \mathcal{M}, w\rangle \notin$ **L**$(\mathcal{M})$:

- As for time, we argue that some $w$ is long enough to ensure that $f$ is sufficiently larger than $g$, so $\mathcal{D}$'s simulation can finish.
- The countdown measures $2^{f(n)}$ steps. The number of possible distinct configurations of $\mathcal{M}$ on $w$ is $|Q| \cdot n \cdot g(n) \cdot |\Gamma|^{g(n)} \in 2^{O(g(n)+\log n)}$, and due to $f(n) \geq \log n$ and $g \in o(f)$, this number is smaller than $2^{f(n)}$ for large enough $n$.
- If $\mathcal{M}$ has $d$ tape symbols, then $\mathcal{D}$ can encode each in $\log d$ space, and due to $\mathcal{M}$'s space bound $\mathcal{D}$'s simulation needs at most $\log d \cdot g(n) \in o(f(n))$ cells.

Therefore, there is $w$ for which $\mathcal{D}$ simulates $\mathcal{M}$ long enough to obtain (and flip) its output, or to detect that it is not terminating (and to accept, flipping again). □



## Space Hierarchies

Like for time, we get some useful corollaries:

**Corollary 13.2:** PSpace $\subsetneq$ ExpSpace

**Proof:** As for time, but easier. □

**Corollary 13.3:** NL $\subsetneq$ PSpace

**Proof:** Savitch tells us that NL $\subseteq \text{DSpace}(\log^2 n)$. We can apply the Space Hierarchy Theorem since $\log^2 n \in o(n)$. □

**Corollary 13.4:** For all real numbers $0 < a < b$, we have $\text{DSpace}(n^a) \subsetneq \text{DSpace}(n^b)$.

In other words: The hierarchy of distinct space classes is very fine-grained.

# The Gap Theorem



## Why Constructibility?

The hierarchy theorems require that resource limits are given by constructible functions

Do we really need this?

Yes. The following theorem shows why (for time):

**Special Gap Theorem 13.5:** There is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{DTime}(f(n)) = \mathrm{DTime}(2^{f(n)})$.

This has been shown independently by Boris Trakhtenbrot (1964) and Allan Borodin (1972).

**Reminder:** For this we continue to use the strict definition of $\mathrm{DTime}(f)$ where no constant factors are included (no hidden $O(f)$). This simplifes proofs; the factors are easy to add back.



## Proving the Gap Theorem

**Special Gap Theorem 13.5:** There is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{DTime}(f(n)) = \mathrm{DTime}(2^{f(n)})$.

**Proof idea:** We divide time into exponentially long intervals of the form:

$$[0, n], \quad [n+1, 2^n], \quad [2^n+1, 2^{2^n}], \quad [2^{2^n}+1, 2^{2^{2^n}}], \quad \cdots$$

(for some appropriate starting value $n$)

We are looking for gaps of time where no TM halts, since:

- for every finite set of TMs,
- and every finite set of inputs to these TMs,
- there is some interval of the above form $[m+1, 2^m]$

such none of the TMs halts in between $m+1$ and $2^m$ steps on any of the inputs.

The task of $f$ is to find the start $m$ of such a gap for a suitable set of TMs and words



## Gaps in Time

We consider an (effectively computable) enumeration of all Turing machines:

$$\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \ldots$$

**Definition 13.6:** For arbitrary numbers $i, a, b \in \mathbb{N}$ with $a \leq b$, we say that $\mathrm{Gap}_i(a, b)$ is true if:

- Given any TM $\mathcal{M}_j$ with $0 \leq j \leq i$,
- and any input string $w$ for $\mathcal{M}_j$ of length $|w| = i$,

$\mathcal{M}_j$ on input $w$ will halt in less than $a$ steps, in more than $b$ steps, or not at all.

**Lemma 13.7:** Given $i, a, b \geq 0$ with $a \leq b$, it is decidable if $\mathrm{Gap}_i(a, b)$ holds.

**Proof:** We just need to ensure that none of the finitely many TMs $\mathcal{M}_0, \ldots, \mathcal{M}_i$ will halt after $a$ to $b$ steps on any of the finitely many inputs of length $i$. This can be checked by simulating TM runs for at most $b$ steps. $\square$

## Find the Gap

We can now define the value $f(n)$ of $f$ for some $n \geq 0$:

Let $\mathsf{in}(n)$ denote the number of runs of TMs $\mathcal{M}_0, \ldots, \mathcal{M}_n$ on words of length $n$, i.e.,

$$\mathsf{in}(n) = |\Sigma_0|^n + \cdots + |\Sigma_n|^n \quad \text{where } \Sigma_i \text{ is the input alphabet of } \mathcal{M}_i$$

We recursively define a series of numbers $k_0, k_1, k_2, \ldots$ by setting $k_0 = 2n$ and $k_{i+1} = 2^{k_i}$ for $i \geq 0$, and we consider the following list of intervals:

$$\begin{array}{cccc} [k_0+1, k_1], & [k_1+1, k_2], & \cdots, & [k_{\mathsf{in}(n)}+1, k_{\mathsf{in}(n)+1}] \\ \parallel & \parallel & & \parallel \\ [2n+1, 2^{2n}], & [2^{2n}+1, 2^{2^{2n}}], & \cdots, & [2^{\cdot^{\cdot^{2n}}}+1, 2^{2^{\cdot^{\cdot^{2n}}}}] \end{array}$$

Let $f(n)$ be the least number $k_i$ with $0 \leq i \leq \mathsf{in}(n)$ such that $\mathsf{Gap}_n(k_i+1, k_{i+1})$ is true.



## Properties of $f$

We first establish some basic properties of our definition of $f$:

**Claim:** The function $f$ is well-defined.

**Proof:** For finding $f(n)$, we consider $\mathsf{in}(n)+1$ intervals. Since there are only $\mathsf{in}(n)$ runs of TMs $\mathcal{M}_0, \ldots \mathcal{M}_n$, at least one interval remains a "gap" where no TM run halts. □

**Claim:** The function $f$ is computable.

**Proof:** We can compute $\mathsf{in}(n)$ and $k_i$ for any $i$, and we can decide $\mathsf{Gap}_n(k_i+1, k_{i+1})$. □

Papadimitriou: "notice the fantastically fast growth, as well as the decidedly unnatural definition of this function."



## Finishing the Proof

We can now complete the proof of the theorem:

**Claim:** $\mathsf{DTime}(f(n)) = \mathsf{DTime}(2^{f(n)})$.

Consider any **L** $\in \mathsf{DTime}(2^{f(n)})$.
Then there is an $2^{f(n)}$-time bounded TM $\mathcal{M}_j$ with **L** = **L**$(\mathcal{M}_j)$.

For any input $w$ with $|w| \geq j$:

- The definition of $f(|w|)$ took the run of $\mathcal{M}_j$ on $w$ into account
- $\mathcal{M}_j$ on $w$ halts after less than $f(|w|)$ steps, or not until after $2^{f(|w|)}$ steps (maybe never)
- Since $\mathcal{M}_j$ runs in time $\mathsf{DTime}(2^{f(n)})$, it must halt in $\mathsf{DTime}(f(n))$ on $w$

For the finitely many inputs $w$ with $|w| < j$:

- We can augment the state space of $\mathcal{M}_j$ to run a finite automaton to decide these cases
- This will work in $\mathsf{DTime}(f(n))$

Therefore we have **L** $\in \mathsf{DTime}(f(n))$. □



## Discussion: The case $|w| < j$

**Borodin says:** It is meaningful to state complexity results if they hold for "almost every" input (i.e., for all but a finite number)

**Papadimitriou says:** These words can be handled since we can check the length and then recognise the word in less than $2j$ steps

Really?

- If we do these $< 2j$ steps before running $\mathcal{M}_j$, the modified TM runs in $\mathsf{DTime}(f(n) + 2j)$
- This does not show **L** $\in \mathsf{DTime}(f(n))$

A more detailed argument:

- Make the intervals larger: $[k_i + 1, 2^{k_i+2n} + 2n]$, that is $k_{i+1} = 2^{k_i+2n} + 2n$.
- Select $f(n)$ to be $k_i + 2n + 1$ if the least gap starts at $k_i + 1$.

The same pigeon hole argument as before ensures that an empty interval is found.

But now the $f(n)$ time bounded machine $\mathcal{M}_j$ from the proof will be sure to stop after $f(n) - 2n - 1$ steps, so a shift of $2j \leq 2n$ to account for the finitely many cases will not make it use more than $f(n)$ steps either

# Discussion: Generalising the Gap Theorem

- Our proof uses the function $n \mapsto 2^n$ to define intervals
- Any other computable function could be used without affecting the argument

This leads to a generalised Gap Theorem:

**Gap Theorem 13.8:** For every computable function $g : \mathbb{N} \to \mathbb{N}$ with $g(n) \geq n$, there is a computable function $f : \mathbb{N} \to \mathbb{N}$ such that DTime($f(n)$) = DTime($g(f(n))$).

**Example 13.9:** There is a function $f$ such that

$$\text{DTime}(f(n)) = \text{DTime}\left(\underbrace{2^{2^{\cdot^{\cdot^{2}}}}}_{f(n)\ \text{times}}\right)$$

Moreover, the Gap Theorem can also be shown for space (and for other resources) in a similar fashion (space is a bit easier since the case of short words $|w| < j$ is easy to handle in very little space)



# Discussion: Significance of the Gap Theorem

**What have we learned?**

- More time (or space) does not always increase computational power
- However, this only works for extremely fast-growing, very unnatural functions

"Fortunately, the gap phenomenon cannot happen for time bounds $t$ that anyone would ever be interested in"[1]

Main insight: better stick to constructible functions

[1] Allender, Loui, Reagan: Complexity Theory. In Computing Handbook, 3rd ed., CRC Press, 2014



# Summary and Outlook

Hierarchy theorems tell us that more time/space leads to more power:

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime} \subseteq \text{ExpSpace}$$

However, they don't help us in comparing different resources and machine types (P vs. NP, or PSpace vs. ExpTime)

With non-constructible functions as time/space bounds, arbitrary (constructible or not) boosts in resources do not lead to more power

**What's next?**

- The inner structure of NP revisited
- Computing with oracles (reprise)
- The limits of diagonalisation, proved by diagonalisation