# Computational complexity

lecture 3

# Announcement

Mid-term exam:

11.12.2018, during the lecture (Tuesday, 12:15)

# Universal machines

The definition of complexity was:

A language $L \subseteq \Sigma^*$ is decidable in time $T(n)$ / space $S(n)$ if
<u>there exists a Turing machine</u> that recognizes this language
and works in time $T(n)$ / space $S(n)$

But in real life we do not build a new computer if we want to solve a new problem. We rather use always the same computer, and we only write a new program.

# Universal machines

The definition of complexity was:

A language $L \subseteq \Sigma^*$ is decidable in time $T(n)$ / space $S(n)$ if
<u>there exists a Turing machine</u> that recognizes this language
and works in time $T(n)$ / space $S(n)$

But in real life we do not build a new computer if we want to solve a new problem. We rather use always the same computer, and we only write a new program.

- A Turing machine can be represented as a string (this is a simple observation, but has far reaching consequences)

# Universal machines

Some notation:

- $\langle M \rangle$ – a word encoding a machine $M$
- ➔ $M(w)$ – the “effect” of running machine $M$ on input $w$:
  - ➔ ”$M$ rejects”
  - ➔ “$M$ loops”
  - ➔ “$M$ accepts and outputs word $v$”
- $M(u,w)$ – the “effect” of running machine $M$ on the pair $(u,w)$
  (we fix some encoding of pairs of words in words)

# Universal machines

Theorem:

Fix an input/output alphabet $\Sigma$ (e.g., $\Sigma=\{0,1\}$). There exists a universal Turing machine $U$ (an “interpreter”), such that $U(\langle M\rangle,w)=M(w)$ for every machine $M$ with input alphabet $\Sigma$ and every word $w\in\Sigma^*$

This looks obvious, but is not completely obvious.

Notice that $U$ is a fixed machine, while $M$ may be arbitrarily large (many tapes, many states, large working alphabet)

# Universal machines

<u>Theorem:</u>

Fix an input/output alphabet $\Sigma$ (e.g., $\Sigma=\{0,1\}$). There exists a universal Turing machine $U$ (an "interpreter"), such that $U(\langle M\rangle,w)=M(w)$ for every machine $M$ with input alphabet $\Sigma$ and every word $w\in\Sigma^*$

<u>Proof</u>

Step 1: $U$ translates $M$ into an equivalent machine $M_2$ which uses only two working tapes, and such that the working alphabet is $\{0,1,\triangleright,\perp\}$

(now only the number of states of $M_2$ is larger than in $U$)

# Universal machines

Theorem:

Fix an input/output alphabet $\Sigma$ (e.g., $\Sigma=\{0,1\}$). There exists a universal Turing machine $U$ (an “interpreter”), such that $U(\langle M\rangle,w)=M(w)$ for every machine $M$ with input alphabet $\Sigma$ and every word $w\in\Sigma^*$

Proof

Step 1: $U$ translates $M$ into an equivalent machine $M_2$ which uses only two working tapes, and such that the working alphabet is $\{0,1,\triangleright,\perp\}$

(now only the number of states of $M_2$ is larger than in $U$)

Step 2: simulate $M_2$ on $w$

- input word $w$ (head as in $M_2$)
- first working tape of $M_2$
- second working tape of $M_2$
- state of $M_2$
- description of $M_2$
- output tape (as in $M_2$)

# Universal machines

Theorem:

Fix an input/output alphabet $\Sigma$ (e.g., $\Sigma=\{0,1\}$). There exists a universal Turing machine $U$ (an "interpreter"), such that $U(\langle M\rangle,w)=M(w)$ for every machine $M$ with input alphabet $\Sigma$ and every word $w\in\Sigma^*$

Proof

Step 2: simulate $M_2$ on $w$

| |
|---|
| input word $w$ (head as in $M_2$) |
| first working tape of $M_2$ |
| second working tape of $M_2$ |
| state of $M_2$ |
| description of $M_2$ |
| output tape (as in $M_2$) |

How fast is $U$? (when $M/M_2$ is fixed)

If $M_2$ works in time $T(|w|)$ and space $S(|w|)$,

then also $U$ works in time $O(T(|w|))$ and space $O(S(|w|))$.

(the length of the state of $M_2$ and of the description $M_2$ of is constant;

step 1 works in constant time/space)

# Universal machines

Theorem:

Fix an input/output alphabet $\Sigma$ (e.g., $\Sigma=\{0,1\}$). There exists a universal Turing machine $U$ (an "interpreter"), such that $U(\langle M\rangle,w)=M(w)$ for every machine $M$ with input alphabet $\Sigma$ and every word $w\in\Sigma^*$

Proof

Step 1: $U$ translates $M$ into an equivalent machine $M_2$ which uses only two working tapes, and such that the working alphabet is $\{0,1,\triangleright,\perp\}$

How fast is $M_2$? (comparing to $M$)

- If $M$ works in space $S(|w|)$, then also $M_2$ works in space $O(S(|w|))$.
- If $M$ works in time $T(|w|)$, then it is easy to create $M_2$ that works in time $O((T(|w|))^2)$ (we can even require that $M_2$ has only one tape)
- One can do better: if $M$ works in time $T(|w|)$, then we can create $M_2$ that works in time $O(T(|w|)\cdot log(T(|w|)))$

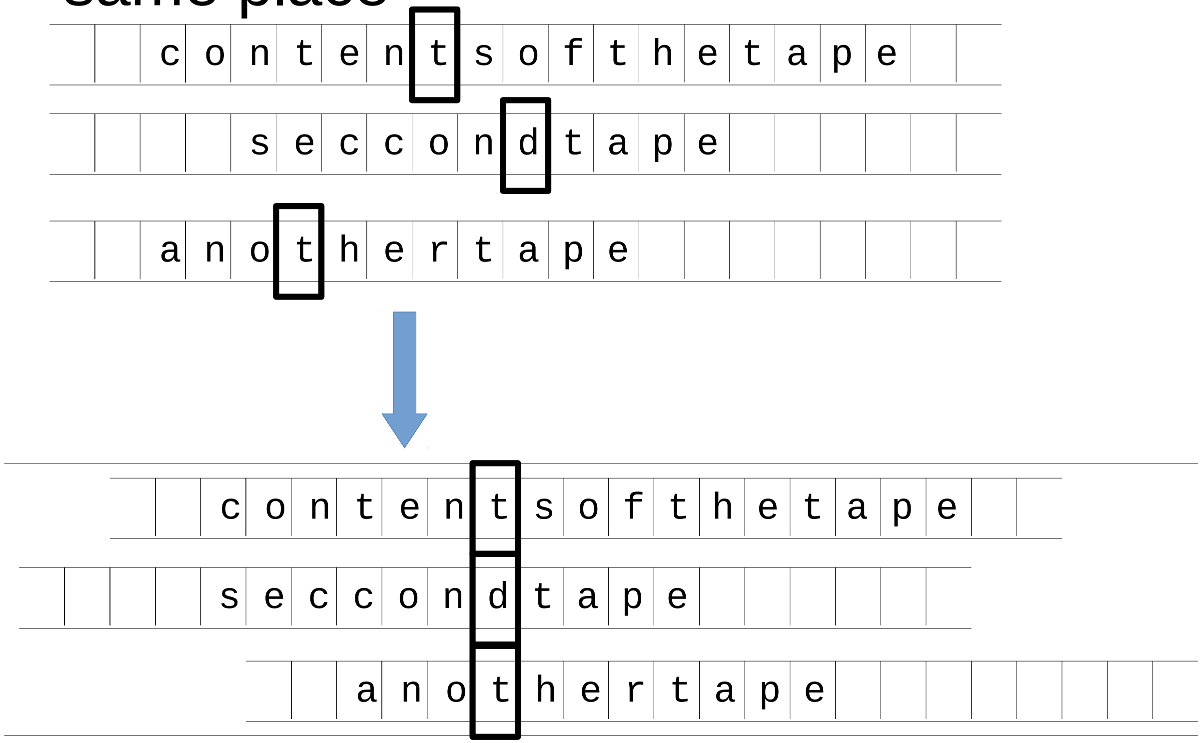
# Universal machines (✶)

## Lemma

One can simulate a multitape machine $M$ working in time $T(n)$ by a two-tape machine $M_2$ working in time $T(n)\cdot log(T(n))$.

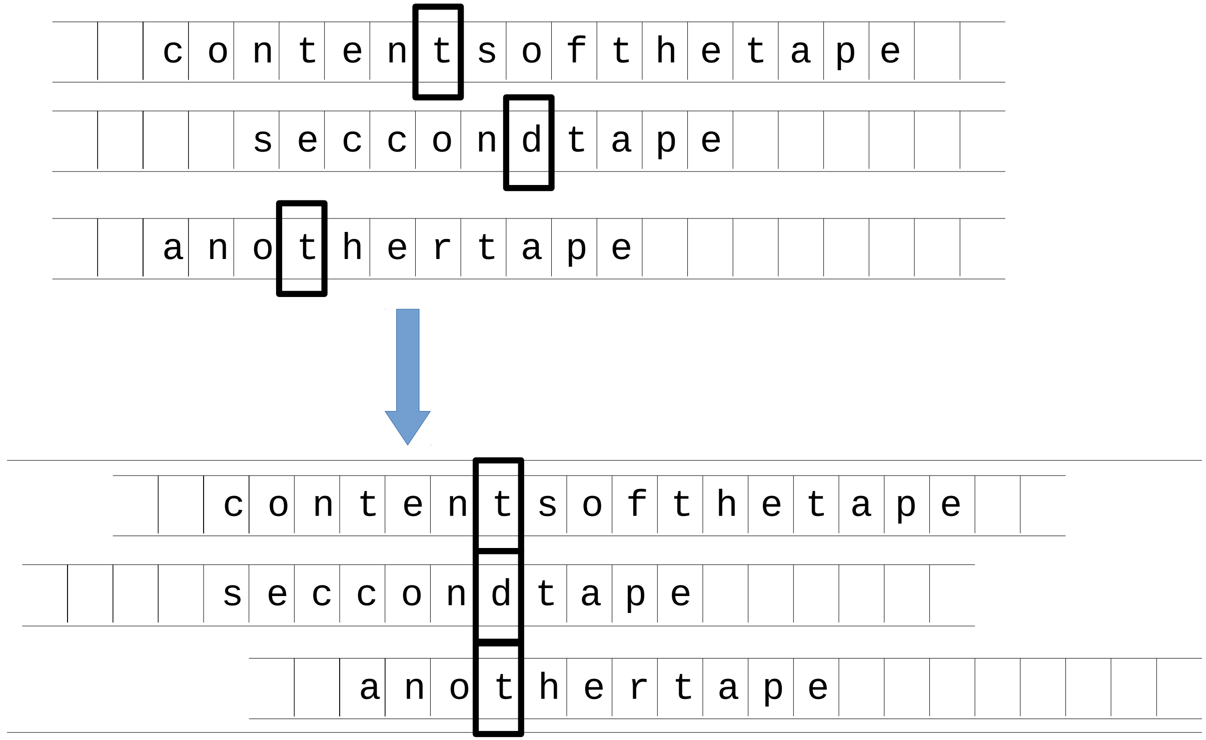
## Proof

- For simplicity: w.l.o.g. assume that tapes of $M$ & $M_2$ are infinite in both directions.
- Idea: keep all $k$ tapes in parallel, using alphabet $\Gamma^k$, with all heads in the same place



# Universal machines (✶)

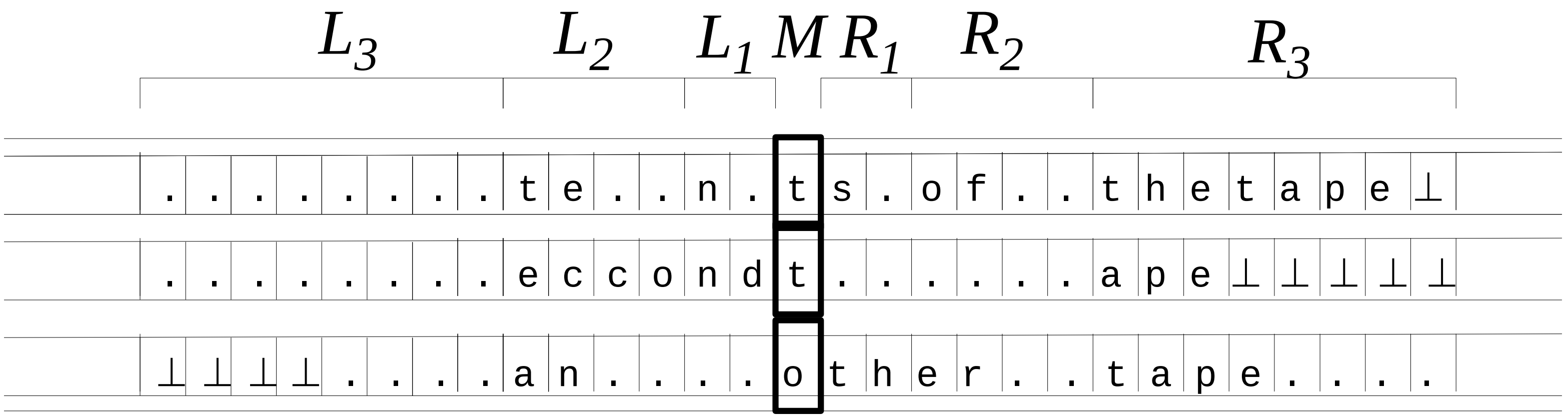
Idea: keep all $k$ tapes in parallel, using alphabet $\Gamma^k$, with all heads in the same place




This does not yet work in $T \cdot log\ T$ – when one head moves, we have to shift contents of one tape, which can be of length $T$ (total time is $T^2$).

# Universal machines (∗)

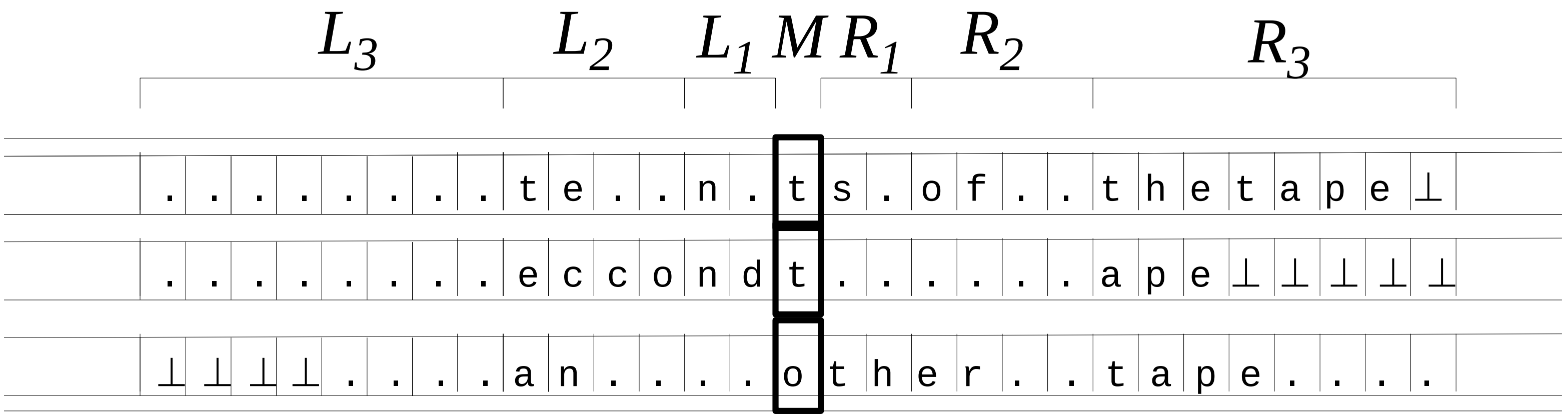
## Idea 2: add some “buffers”




- Split everything into zones ...,$L_3$,$L_2$,$L_1$,$M$,$R_1$,$R_2$,$R_3$,... ($O(\log T)$ zones) Zones $L_i/R_i$ have length $2^i$.
- Some cells are empty (contain “.”). Every zone is either empty, or full, or half-full. Zones $L_i$ and $R_i$ have together $2^i$ empty cells and $2^i$ full cells (where $\perp$ is treated as full).

# Universal machines (✶)

## Idea 2: add some “buffers”

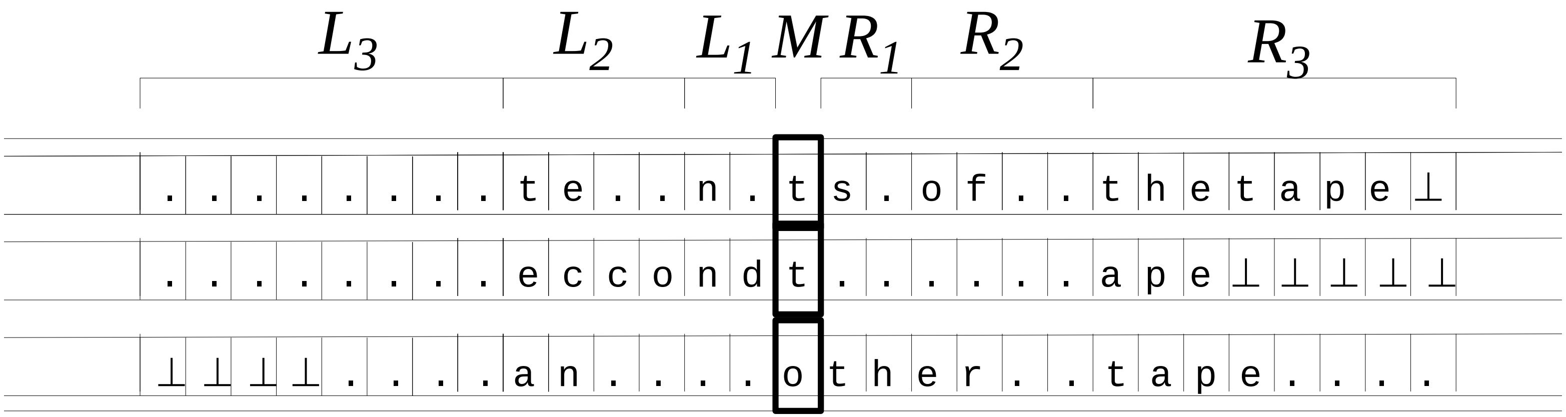



How do we move the head (right):

- Find the smallest $R_i$ that is nonempty
- Move first $2^{i-1}$ symbols from $R_i$ to $M,R_1,\dots,R_{i-1}$ (so that they become half-full). Symmetrically proceed with $L_i,L_{i-1},\dots,L_1,M$.

# Universal machines (∗)

## Idea 2: add some “buffers”




How do we move the head (right):

- Find the smallest $R_i$ that is nonempty
- Move first $2^{i-1}$ symbols from $R_i$ to $M,R_1,\dots,R_{i-1}$ (so that they become half-full). Symmetrically proceed with $L_i,L_{i-1},\dots,L_1,M$.
- The cost is $O(2^i)$ (we use the second tape while copying symbols)
- After this operation, zones $L_{i-1},\dots,L_1,M,R_1,\dots,R_{i-1}$ are half-full.
- Thus zone $L_i$ will not be touched during the next $2^{i-1}$ steps.
- For every $i$ the running time accumulates to constant / step.
- This gives $O(T\cdot \log T)$ in total.

# Universal machines

Theorem:

There exists a universal Turing machine $U$ (an “interpreter”), such that $U(\langle M\rangle,w)=M(w)$. If $M$ works in time $T(|w|)$ and space $S(|w|)$, then $U$ works in time $O(T(|w|)\cdot log(T(|w|)))$ and space $O(S(|w|))$.

# Universal machines

<u>Theorem:</u>

There exists a universal Turing machine $U$ (an “interpreter”), such that $U(\langle M\rangle,w)=M(w)$. If $M$ works in time $T(|w|)$ and space $S(|w|)$, then $U$ works in time $O(T(|w|)\cdot log(T(|w|)))$ and space $O(S(|w|))$.

Two possible definitions of time / space complexity:

- $T_1/S_1$ using machines (“there exists a machine...”)
- $T_2/S_2$ using programs for the universal machine (“there exists a program...”)

Relation between them:

- $T_1 \leq T_2 \leq T_1 \cdot log\ T_1$
- $S_1 = S_2$

only small difference!
we use the definition with machines

# Hierarchy theorems

Are there problems that require very large time / space to be solved?
(Maybe every problem can be solved e.g. in polynomial time?)

# Hierarchy theorems

Are there problems that require very large time / space to be solved?
(Maybe every problem can be solved e.g. in polynomial time?)

Space hierarchy theorem:

If:

- function $g(n)$ is space-constructible, and
- $f(n)=o(g(n))$

then DSPACE($f(n)$)≠DSPACE($g(n)$)

Time hierarchy theorem – similar

definition: $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

# Hierarchy theorems

<u>Space hierarchy theorem:</u>

If:

- function $g(n)$ is space-constructible, and
- <u>$f(n)=o(g(n))$</u>

then <u>DSPACE($f(n)$)$\neq$DSPACE($g(n)$)</u>

<u>Proof:</u>

- Consider the language

$$L = \{(\langle M\rangle,w) \mid \text{tape alphabet of } M \text{ is } \{0,1,\triangleright,\perp\}, \text{ and } |\langle M\rangle| \leq g(|(\langle M\rangle,w)|), \text{ and } M \text{ rejects } (\langle M\rangle,w) \text{ in space } g(|(\langle M\rangle,w)|)\}$$

# Hierarchy theorems

$L = \{(\langle M\rangle,w) \mid$ tape alphabet of $M$ is $\{0,1,\triangleright,\perp\}$, and $|\langle M\rangle|\leq g(|(\langle M\rangle,w)|)$, and $M$ rejects $(\langle M\rangle,w)$ in space $g(|(\langle M\rangle,w)|)\}$

<u>Part 1</u> – $L\notin$DSPACE($f(n)$)

Suppose that $L\in$DSPACE($f(n)$). Then there is $M$ with tape alphabet $\{0,1,\triangleright,\perp\}$, which recognizes $L$ in space $O(f(n))$.

Because $f(n)=o(g(n))$, for some long word $w$ machine $M$ works on $(\langle M\rangle,w)$ in space $g(|(\langle M\rangle,w)|)$, and $|\langle M\rangle|\leq g(|(\langle M\rangle,w)|)$

We have a contradiction:

($M$ accepts $(\langle M\rangle,w)$) $\Leftrightarrow$ $(\langle M\rangle,w)\in L$ $\Leftrightarrow$ ($M$ rejects $(\langle M\rangle,w)$)

Remark – for the language

$$L' = \{((\langle M\rangle,w) \mid M \text{ rejects } (\langle M\rangle,w)\}$$

the same argument gives undecidability.

# Hierarchy theorems

$L = \{(\langle M\rangle,w) \mid$ tape alphabet of $M$ is $\{0,1,\triangleright,\perp\}$, and $|\langle M\rangle| \leq g(|(\langle M\rangle,w)|)$, and $M$ rejects $(\langle M\rangle,w)$ in space $g(|(\langle M\rangle,w)|)\}$

Part 2: $L \in$ DSPACE($g(n)$) – i.e., $L$ can be recognized in space $O(g(n))$.

- Generally: simulate the run of $M$ on $(\langle M\rangle,w)$

# Hierarchy theorems

$L = \{(\langle M\rangle,w) \mid$ tape alphabet of $M$ is $\{0,1,\triangleright,\perp\}$, and $|\langle M\rangle| \leq g(|(\langle M\rangle,w)|)$, and $M$ rejects $(\langle M\rangle,w)$ in space $g(|(\langle M\rangle,w)|)\}$

Part 2: $L \in$ DSPACE($g(n)$) – i.e., $L$ can be recognized in space $O(g(n))$.

- Generally: simulate the run of $M$ on $(\langle M\rangle,w)$
- Reserve working space $g(n)$ (where $n$ = length of input)
  - space $O(g(n))$ is enough (by assumption $g$ is space-constructible)

# Hierarchy theorems

$L = \{(\langle M\rangle,w)$ | tape alphabet of $M$ is $\{0,1,\triangleright,\perp\}$, and $|\langle M\rangle|\leq g(|(\langle M\rangle,w)|)$, and $M$ rejects $(\langle M\rangle,w)$ in space $g(|(\langle M\rangle,w)|)\}$

Part 2: $L\in$DSPACE($g(n)$) – i.e., $L$ can be recognized in space $O(g(n))$.

- Generally: simulate the run of $M$ on $(\langle M\rangle,w)$
- Reserve working space $g(n)$ (where $n$ = length of input)
  - space $O(g(n))$ is enough (by assumption $g$ is space-constructible)
- Check that the input is of the form $(\langle M\rangle,w)$, that the alphabet is $\{0,1,\triangleright,\perp\}$, and that $|\langle M\rangle|\leq g(n)$
  - space $O(g(n))$ is enough

# Hierarchy theorems

$L = \{(\langle M\rangle,w) \mid$ tape alphabet of $M$ is $\{0,1,\triangleright,\perp\}$, and $|\langle M\rangle| \leq g(|(\langle M\rangle,w)|)$, and $M$ rejects $(\langle M\rangle,w)$ in space $g(|(\langle M\rangle,w)|)\}$

Part 2: $L \in$ DSPACE($g(n)$) – i.e., $L$ can be recognized in space $O(g(n))$.

- Generally: simulate the run of $M$ on $(\langle M\rangle,w)$
- Reserve working space $g(n)$ (where $n$ = length of input)
  - space $O(g(n))$ is enough (by assumption $g$ is space-constructible)
- Check that the input is of the form $(\langle M\rangle,w)$, that the alphabet is $\{0,1,\triangleright,\perp\}$, and that $|\langle M\rangle| \leq g(n)$
  - space $O(g(n))$ is enough
- Use the Sipser's theorem (or assume that $g(n)=\Omega(log(n))$, and use the approach with a counter), and check whether $M$ rejects $(\langle M\rangle,w)$ in reserved space $g(n)$.
  - when $M$ rejects → we accept
  - when $M$ accepts or loops or exceeds space → we reject
  - space $O(g(n))$ is enough

# Hierarchy theorems

<u>Space hierarchy theorem:</u>

If:

- function $g(n)$ is space-constructible, and
- <u>$f(n)=o(g(n))$</u>

then <u>DSPACE($f(n)$)≠DSPACE($g(n)$)</u>

<u>Time hierarchy theorem:</u>

If:

- function $g(n)$ is time-constructible,
- <u>$f(n)=o(g(n))$</u>

then <u>DTIME($f(n)$)≠DTIME($g(n)log(g(n))$)</u>

# Hierarchy theorems

<u>Time hierarchy theorem:</u>

If:

- function $g(n)$ is time-constructible,
- <u>$f(n)=o(g(n))$</u>

then <u>DTIME($f(n)$)$\neq$DTIME($g(n)log(g(n))$)</u>

<u>Proof</u>

- Consider the language

$L = \{(\langle M\rangle,w) \mid$ tape alphabet of $M$ is $\{0,1,\triangleright,\perp\}$, and $|\langle M\rangle| \leq log(|(\langle M\rangle,w)|)$, and $M$ rejects $(\langle M\rangle,w)$ in <u>time</u> $g(|(\langle M\rangle,w)|)\}$

- <u>Part 1</u> – $L\notin$DTIME($f(n)$) $\rightarrow$ exactly as previously

# Hierarchy theorems

$L = \{(\langle M\rangle,w) \mid$ tape alphabet of $M$ is $\{0,1,\triangleright,\perp\}$, and $|\langle M\rangle| \leq log(|(\langle M\rangle,w)|)$, and

$M$ rejects $(\langle M\rangle,w)$ in time $g(|(\langle M\rangle,w)|)\}$

Part 2 – $L \in$ DTIME($g(n)log(g(n))$) – i.e., $L$ can be recognized in time $O(g(n)log(g(n)))$

- Generally: simulate the run of $M$ on $(\langle M\rangle,w)$
- Check that the input is of the form $(\langle M\rangle,w)$, that the alphabet is $\{0,1,\triangleright,\perp\}$, and that $|\langle M\rangle| \leq log(n)$ (where $n$ = length of input)
  - running time: $O(n)$
- Reserve a unary counter of length $g(n)$, on a separate tape
  - $g$ is time constructible
  - running time: $O(g(n))$
- Simulate $M$ on word $(\langle M\rangle,w)$, like the universal machine; increase the counter after every step.
  - running time: $O(g(n)\cdot(log\ g(n)+|\langle M\rangle|)) = O(g(n)log(g(n)))$

simulating tapes

reading the description of $M$, modifying state

# Hierarchy theorems

$L = \{(\langle M\rangle,w) \mid$ tape alphabet of $M$ is $\{0,1,\triangleright,\perp\}$, and $|\langle M\rangle| \leq log(|(\langle M\rangle,w)|)$, and $M$ rejects $(\langle M\rangle,w)$ in time $g(|(\langle M\rangle,w)|)\}$

<u>Part 2</u> – $L \in$DTIME($g(n)log(g(n))$) – i.e., $L$ can be recognized in time $O(g(n)log(g(n)))$

- Generally: simulate the run of $M$ on $(\langle M\rangle,w)$
- Check that the input is of the form $(\langle M\rangle,w)$, that the alphabet is $\{0,1,\triangleright,\perp\}$, and that $|\langle M\rangle| \leq log(n)$ (where $n$ = length of input)
  - running time: $O(n)$
- Reserve a unary counter of length $g(n)$, on a separate tape
  - $g$ is time constructible
  - running time: $O(g(n))$
- Simulate $M$ on word $(\langle M\rangle,w)$, like the universal machine; increase the counter after every step.
  - running time: $O(g(n)\cdot(log\ g(n)+|\langle M\rangle|)) = O(g(n)log(g(n)))$
  - when $M$ rejects $\rightarrow$ we accept
  - when $M$ accepts or exceeds time $\rightarrow$ we reject

# Hierarchy theorems

Are there problems that require very large time / space to be solved?
(Maybe every problem can be solved e.g. in polynomial time?)

Corollary from hierarchy theorems

- DTIME($n^k$)≠DTIME($n^{k+1}$), DSPACE($n^k$)≠DSPACE($n^{k+1}$)
- L≠PSPACE, P≠EXPTIME

because P⊆DTIME($2^n$)≠DTIME($4^n$)⊆EXPTIME

# Hierarchy theorems

Are there problems that require very large time / space to be solved?
(Maybe every problem can be solved e.g. in polynomial time?)

<u>Corollary</u> from hierarchy theorems

- DTIME($n^k$)≠DTIME($n^{k+1}$), DSPACE($n^k$)≠DSPACE($n^{k+1}$)
- L≠PSPACE, P≠EXPTIME

because P⊆DTIME($2^n$)≠DTIME($4^n$)⊆EXPTIME

If a machine $M$ works in time / space precisely $f(n)$, then there exists a problem requiring more time / space to be solved

- e.g. $2^{f(n)}$ *or* $f(n)^2$ – for time & space
- e.g. $f(n)\cdot log(log(n))$ – for space
- Moreover, functions being complexities of problems are distributed "quite densely", especially for space

# Gap theorems

- Functions being complexities of problems are distributed “quite densely”
- Simultaneously, we have the following gap theorems:

There is a computable function $f(n) \geq n$ such that DTIME($f(n)$)=DTIME($2^{f(n)}$).

There is a computable function $f(n)$ such that DSPACE($f(n)$)
=DSPACE($2^{f(n)}$).

A contradiction with hierarchy theorems?

No – the function $f$ will not be constructible (it can be computed, but in a larger time / space)

At the same time: we see that in the hierarchy theorems the assumption about constructability is really needed