

# Foundations of Complexity Theory

## Lecture 4: Foundational Relations III

November 4, 2026

# Next Dates

| LECTURE (WED 1:15PM)                                |
|-----------------------------------------------------|
| Nov 4th<br>Lecture 4: Fundamental relations III     |
| 11th<br>Lecture 5: Reductions & complete problems I |
| 18th<br>_____<br>(public holiday)                   |
| 25th<br>Lecture 6: Complete problems II             |
| Dec 2nd<br>_____<br>(dies academicus)               |
| 9th<br>Lecture 7: NP-complete problems              |
| 16th<br>Lecture 8: Complete problems III            |

| LECTURE (WED 1:15PM)                                                                             |
|--------------------------------------------------------------------------------------------------|
| Dec 23rd<br>_____<br>_____<br>30th<br>_____<br>_____<br>Jan 6th<br>Lecture 9: Boolean circuits I |
| 13th<br>Lecture 10: Boolean circuits II                                                          |
| 20th<br>Lecture 11: Razborov's theorem I                                                         |
| 27th<br>_____<br>_____<br>Feb 3rd<br>Lecture 12: Razborov's theorem II                           |

## Contents

- 1 Basic complexity classes & their relations
- 2 Reductions & complete problems
- 3 Boolean circuits & Razborov's theorem
- 4 Randomization & interactive proof systems
- 5 Polynomial primality test

# Complexity Classes

## Standard classes

$$\mathbf{L} = \mathbf{DSPACE}(\log x)$$

$$\mathbf{NL} = \mathbf{NSPACE}(\log x)$$

$$\mathbf{P} = \bigcup_{i \in \mathbb{N}_+} \mathbf{DTIME}(x^i)$$

$$\mathbf{NP} = \bigcup_{i \in \mathbb{N}_+} \mathbf{NTIME}(x^i)$$

$$\mathbf{PSPACE} = \bigcup_{i \in \mathbb{N}_+} \mathbf{DSPACE}(x^i) \stackrel{\dagger}{=} \bigcup_{i \in \mathbb{N}_+} \mathbf{NSPACE}(x^i)$$

$$\mathbf{EXPTIME} = \bigcup_{i \in \mathbb{N}_+} \mathbf{DTIME}(2^{x^i})$$

$$\mathbf{NEXPTIME} = \bigcup_{i \in \mathbb{N}_+} \mathbf{NTIME}(2^{x^i})$$

$\dagger$  = Savitch's theorem

# Savitch's Theorem

## §4.1 Theorem (Savitch's theorem)

Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq \log n$  for all  $n \in \mathbb{N}$ .

Then  $\text{NSPACE}(f) \subseteq \text{DSPACE}(f ; g)$  with  $g(n) = n^2$  for all  $n \in \mathbb{N}$

# Savitch's Theorem

## §4.1 Theorem (Savitch's theorem)

Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq \log n$  for all  $n \in \mathbb{N}$ .

Then  $\text{NSPACE}(f) \subseteq \text{DSPACE}(f; g)$  with  $g(n) = n^2$  for all  $n \in \mathbb{N}$

### Proof (1/2)

Let  $M = (Q, \Sigma, \Gamma, \Sigma', \delta, \square, q_0, q_+, q_-)$   $f$ -space-bounded  $k$ -TM and  $w \in \Sigma^*$  input of length  $n$ . Wlog. suppose  $M$  deletes its tapes and returns to initial position on input tape before accepting.

# Savitch's Theorem

## §4.1 Theorem (Savitch's theorem)

Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq \log n$  for all  $n \in \mathbb{N}$ .

Then  $\text{NSPACE}(f) \subseteq \text{DSPACE}(f; g)$  with  $g(n) = n^2$  for all  $n \in \mathbb{N}$

### Proof (1/2)

Let  $M = (Q, \Sigma, \Gamma, \Sigma', \delta, \square, q_0, q_+, q_-)$   $f$ -space-bounded  $k$ -TM and  $w \in \Sigma^*$  input of length  $n$ . Wlog. suppose  $M$  deletes its tapes and returns to initial position on input tape before accepting. Single accepting configuration is  $\top = (q_+, \square, \dots, \square, 1, \dots, 1)$  of length 1.

# Savitch's Theorem

## §4.1 Theorem (Savitch's theorem)

Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq \log n$  for all  $n \in \mathbb{N}$ .

Then  $\text{NSPACE}(f) \subseteq \text{DSPACE}(f ; g)$  with  $g(n) = n^2$  for all  $n \in \mathbb{N}$

### Proof (1/2)

Let  $M = (Q, \Sigma, \Gamma, \Sigma', \delta, \square, q_0, q_+, q_-)$   $f$ -space-bounded  $k$ -TM and  $w \in \Sigma^*$  input of length  $n$ . Wlog. suppose  $M$  deletes its tapes and returns to initial position on input tape before accepting. Single accepting configuration is  $\top = (q_+, \square, \dots, \square, 1, \dots, 1)$  of length 1.

By Lemma §1.7 exist  $a, b \in \mathbb{N}$  such that graph  $G$  of all configurations reachable from  $I_w$  (also length 1) contains  $\leq a \cdot n \cdot b^{f(n+2)}$  vertices (edges are derivation steps).

# Savitch's Theorem

## §4.1 Theorem (Savitch's theorem)

Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq \log n$  for all  $n \in \mathbb{N}$ .

Then  $\text{NSPACE}(f) \subseteq \text{DSPACE}(f ; g)$  with  $g(n) = n^2$  for all  $n \in \mathbb{N}$

### Proof (1/2)

Let  $M = (Q, \Sigma, \Gamma, \Sigma', \delta, \square, q_0, q_+, q_-)$   $f$ -space-bounded  $k$ -TM and  $w \in \Sigma^*$  input of length  $n$ . Wlog. suppose  $M$  deletes its tapes and returns to initial position on input tape before accepting. Single accepting configuration is  $\top = (q_+, \square, \dots, \square, 1, \dots, 1)$  of length 1.

By Lemma §1.7 exist  $a, b \in \mathbb{N}$  such that graph  $G$  of all configurations reachable from  $I_w$  (also length 1) contains  $\leq a \cdot n \cdot b^{f(n+2)}$  vertices (edges are derivation steps).

TM  $M$  accepts input  $w$  iff  $\top$  reachable from  $I_w$  in  $G$ .

# Savitch's Theorem

## Proof (2/2)

Test whether  $\top$  is reachable from  $l_w$  in  $G$ .

- ① Let  $i = 2$
- ② Let  $G_i$  graph of configurations of length  $\leq i$
- ③ Return **true** if configuration  $\top$  is reachable from  $l_w$  in  $G_i$

# Savitch's Theorem

## Proof (2/2)

Test whether  $\top$  is reachable from  $l_w$  in  $G$ .

- ① Let  $i = 2$
- ② Let  $G_i$  graph of configurations of length  $\leq i$
- ③ Return **true** if configuration  $\top$  is reachable from  $l_w$  in  $G_i$
- ④ Check reachability of some configuration of length  $i$ 
  - If reachable, then set  $i \leftarrow 2i$  and back to ②

# Savitch's Theorem

## Proof (2/2)

Test whether  $\top$  is reachable from  $l_w$  in  $G$ .

- ① Let  $i = 2$
- ② Let  $G_i$  graph of configurations of length  $\leq i$
- ③ Return **true** if configuration  $\top$  is reachable from  $l_w$  in  $G_i$
- ④ Check reachability of some configuration of length  $i$ 
  - ▶ If reachable, then set  $i \leftarrow 2i$  and back to ②
  - ▶ Otherwise return **false**

Obviously  $i \leq 2f(n+2)$ . Call  $\text{Reach}(l_w, \log(a \cdot n \cdot b^i), \top)$  of Savitch's algorithm (Theorem §3.6) requires space

$$\begin{aligned} O(\log^2(a \cdot x \cdot b^{2f(x+2)})) &= O((\log a + \log x + 2f(x+2) \cdot \log b)^2) \\ &\subseteq O(f(x+2)^2) \end{aligned}$$



# Theorem von Savitch

## Conclusions

- **PSPACE** =  $\bigcup_{i \in \mathbb{N}_+} \text{DSPACE}(x^i) = \bigcup_{i \in \mathbb{N}_+} \text{NSPACE}(x^i)$   
since  $\text{NSPACE}(x^i) \subseteq \text{DSPACE}(x^{2i})$  by Savitch's theorem

# Theorem von Savitch

## Conclusions

- $\mathbf{PSPACE} = \bigcup_{i \in \mathbb{N}_+} \text{DSPACE}(x^i) = \bigcup_{i \in \mathbb{N}_+} \text{NSPACE}(x^i)$   
since  $\text{NSPACE}(x^i) \subseteq \text{DSPACE}(x^{2i})$  by Savitch's theorem
- $\mathbf{L} \subsetneq \text{DSPACE}(\log^2 x) \subsetneq \text{DSPACE}(x) \subseteq \text{NSPACE}(x) \subsetneq \mathbf{PSPACE}$   
by space hierarchy theorem (Theorem §3.2)

# Theorem von Savitch

## Conclusions

- $\mathbf{PSPACE} = \bigcup_{i \in \mathbb{N}_+} \text{DSPACE}(x^i) = \bigcup_{i \in \mathbb{N}_+} \text{NSPACE}(x^i)$   
since  $\text{NSPACE}(x^i) \subseteq \text{DSPACE}(x^{2^i})$  by Savitch's theorem
- $\mathbf{L} \subsetneq \text{DSPACE}(\log^2 x) \subsetneq \text{DSPACE}(x) \subseteq \text{NSPACE}(x) \subsetneq \mathbf{PSPACE}$   
by space hierarchy theorem (Theorem §3.2)
- $\text{DTIME}(O(x)) \subsetneq \text{DTIME}(O(x^2)) \subsetneq \mathbf{P}$  and  
 $\mathbf{P} \subsetneq \text{DTIME}(O(2^x)) \subsetneq \text{DTIME}(O((2 + \epsilon)^x))$   
follows by time hierarchy theorem (Theorem §3.5)

# Translation Technique

## Motivation

- Extend inclusion to larger classes
- Contraposition: Extend non-inclusion to smaller classes
- Technique: Pad elements to desired length (Padding of language)

# Translation Technique

## Motivation

- Extend inclusion to larger classes
- Contraposition: Extend non-inclusion to smaller classes
- Technique: Pad elements to desired length (Padding of language)

## §4.2 Definition

Let  $L \subseteq \Sigma^*$ ,  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq n$  for all  $n \in \mathbb{N}$ , and  $\# \notin \Sigma$  new symbol

$$\text{Pad}_f^\#(L) = \{w\#^{f(|w|+2)-|w|} \mid w \in L\}$$

# Translation Technique

## Motivation

- Extend inclusion to larger classes
- Contraposition: Extend non-inclusion to smaller classes
- Technique: Pad elements to desired length (Padding of language)

## §4.2 Definition

Let  $L \subseteq \Sigma^*$ ,  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq n$  for all  $n \in \mathbb{N}$ , and  $\# \notin \Sigma$  new symbol

$$\text{Pad}_f^\#(L) = \{w\#^{f(|w|+2)-|w|} \mid w \in L\}$$

## Note

- Associates word of length  $f(|w| + 2)$  to each word  $w \in L$

## §4.3 Theorem (Translation theorem for time classes)

Let  $f, g: \mathbb{N} \rightarrow \mathbb{N}$  monotonic increasing with  $f(n) \geq n \leq g(n)$  for all  $n \in \mathbb{N}$ . Additionally, let  $g$  time-constructible and output  $\#^{f(n+2)}$  computable in time  $g(f(n+2) + 2)$  on input  $\#^n$ . Then for arbitrary  $L \subseteq \Sigma^*$  and  $\# \notin \Sigma$

- ①  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$  iff  $L \in \text{DTIME}(O(f; g))$
- ②  $\text{Pad}_f^\#(L) \in \text{NTIME}(O(g))$  iff  $L \in \text{NTIME}(O(f; g))$

## §4.3 Theorem (Translation theorem for time classes)

Let  $f, g: \mathbb{N} \rightarrow \mathbb{N}$  monotonic increasing with  $f(n) \geq n \leq g(n)$  for all  $n \in \mathbb{N}$ . Additionally, let  $g$  time-constructible and output  $\#^{f(n+2)}$  computable in time  $g(f(n+2) + 2)$  on input  $\#^n$ . Then for arbitrary  $L \subseteq \Sigma^*$  and  $\# \notin \Sigma$

- ①  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$  iff  $L \in \text{DTIME}(O(f ; g))$
- ②  $\text{Pad}_f^\#(L) \in \text{NTIME}(O(g))$  iff  $L \in \text{NTIME}(O(f ; g))$

### Notes

- Slightly non-standard composition used; for all  $f, g: \mathbb{N} \rightarrow \mathbb{N}$  and  $n \in \mathbb{N}$

$$(f ; g)(n) = g(f(n) + 2)$$

- We only prove ① for DTIME
- Proof of ② analogous

## Proof (1/2)

We start with implication ( $\rightarrow$ ). Let  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$  and  $w \in \Sigma^*$  input of length  $n$ . We decide whether  $w \in L$  in time  $O(f; g)$  by

- 1 Write  $\#^{f(n+2)}$  onto 2nd tape in time  $g(f(n+2) + 2)$

## Proof (1/2)

We start with implication ( $\rightarrow$ ). Let  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$  and  $w \in \Sigma^*$  input of length  $n$ . We decide whether  $w \in L$  in time  $O(f; g)$  by

- 1 Write  $\#^{f(n+2)}$  onto 2nd tape in time  $g(f(n+2) + 2)$
- 2 Copy input and then padding from 2nd tape to extend  $w$  to  $w' = w\#^{f(n+2)-n}$  of length  $f(n+2)$  in time  $f(n+2)$

# Translation Technique

## Proof (1/2)

We start with implication ( $\rightarrow$ ). Let  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$  and  $w \in \Sigma^*$  input of length  $n$ . We decide whether  $w \in L$  in time  $O(f; g)$  by

- 1 Write  $\#^{f(n+2)}$  onto 2nd tape in time  $g(f(n+2) + 2)$
- 2 Copy input and then padding from 2nd tape to extend  $w$  to  $w' = w\#^{f(n+2)-n}$  of length  $f(n+2)$  in time  $f(n+2)$
- 3 Since  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$  we check whether  $w' \in \text{Pad}_f^\#(L)$  in time  $O(g(|w'| + 2)) = O(g(f(n+2) + 2))$  due to  $|w'| = f(n+2)$

## Proof (1/2)

We start with implication ( $\rightarrow$ ). Let  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$  and  $w \in \Sigma^*$  input of length  $n$ . We decide whether  $w \in L$  in time  $O(f; g)$  by

- 1 Write  $\#^{f(n+2)}$  onto 2nd tape in time  $g(f(n+2) + 2)$
- 2 Copy input and then padding from 2nd tape to extend  $w$  to  $w' = w\#^{f(n+2)-n}$  of length  $f(n+2)$  in time  $f(n+2)$
- 3 Since  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$  we check whether  $w' \in \text{Pad}_f^\#(L)$  in time  $O(g(|w'| + 2)) = O(g(f(n+2) + 2))$  due to  $|w'| = f(n+2)$

Obviously  $w' \in \text{Pad}_f(L)$  iff  $w \in L$ . Hence  $L \in \text{DTIME}(O(f; g))$ .

# Translation Technique

## Proof (2/2)

For implication ( $\leftarrow$ ) let  $L \in \text{DTIME}(O(f; g))$  and  $w' \in (\Sigma \cup \{\#\})^*$  input of length  $m$ . We decide whether  $w' \in \text{Pad}_f^\#(L)$  in time  $O(g)$  by

- ① Test in time  $m + 2 \leq g(m + 2)$   
whether  $w' = w\#^\ell$  for  $w \in \Sigma^*$  and  $\ell \in \mathbb{N}$  (Rejection other shapes)
- ② Let  $n = |w|$  and initialize counter by  $g(m + 2)$  on tape in time  $g(m + 2)$  ( $g$  time-constructible)

# Translation Technique

## Proof (2/2)

For implication ( $\leftarrow$ ) let  $L \in \text{DTIME}(O(f; g))$  and  $w' \in (\Sigma \cup \{\#\})^*$  input of length  $m$ . We decide whether  $w' \in \text{Pad}_f^\#(L)$  in time  $O(g)$  by

- ① Test in time  $m + 2 \leq g(m + 2)$   
whether  $w' = w\#^\ell$  for  $w \in \Sigma^*$  and  $\ell \in \mathbb{N}$  (Rejection other shapes)
- ② Let  $n = |w|$  and initialize counter by  $g(m + 2)$  on tape in time  $g(m + 2)$  ( $g$  time-constructible)
- ③ Start computation of  $\#^{f(n+2)}$  in time  $g(f(n + 2) + 2)$  additionally restricted to  $g(m + 2)$  steps

# Translation Technique

## Proof (2/2)

For implication ( $\leftarrow$ ) let  $L \in \text{DTIME}(O(f; g))$  and  $w' \in (\Sigma \cup \{\#\})^*$  input of length  $m$ . We decide whether  $w' \in \text{Pad}_f^\#(L)$  in time  $O(g)$  by

- ① Test in time  $m + 2 \leq g(m + 2)$   
whether  $w' = w\#^\ell$  for  $w \in \Sigma^*$  and  $\ell \in \mathbb{N}$  (Rejection other shapes)
- ② Let  $n = |w|$  and initialize counter by  $g(m + 2)$  on tape in time  $g(m + 2)$  ( $g$  time-constructible)
- ③ Start computation of  $\#^{f(n+2)}$  in time  $g(f(n + 2) + 2)$  additionally restricted to  $g(m + 2)$  steps
  - ▶ Computation unfinished: Then  $g(f(n + 2) + 2) > g(m + 2)$  and hence  $f(n + 2) > m = |w'|$ . Thus  $w' \notin \text{Pad}_f^\#(L)$  (wrong length) and we reject
  - ▶ Computation finished: If  $f(n + 2) \neq m$ , then we reject (wrong length).
- ④ Since  $L \in \text{DTIME}(O(f; g))$  we check whether  $w \in L$  in time  $O(g(f(n + 2) + 2)) = O(g(m + 2))$  due to  $m = f(n + 2)$ .

# Translation Technique

## Proof (2/2)

For implication ( $\leftarrow$ ) let  $L \in \text{DTIME}(O(f; g))$  and  $w' \in (\Sigma \cup \{\#\})^*$  input of length  $m$ . We decide whether  $w' \in \text{Pad}_f^\#(L)$  in time  $O(g)$  by

- ① Test in time  $m + 2 \leq g(m + 2)$   
whether  $w' = w\#^\ell$  for  $w \in \Sigma^*$  and  $\ell \in \mathbb{N}$  (Rejection other shapes)
- ② Let  $n = |w|$  and initialize counter by  $g(m + 2)$  on tape in time  $g(m + 2)$  ( $g$  time-constructible)
- ③ Start computation of  $\#^{f(n+2)}$  in time  $g(f(n + 2) + 2)$  additionally restricted to  $g(m + 2)$  steps
  - ▶ Computation unfinished: Then  $g(f(n + 2) + 2) > g(m + 2)$  and hence  $f(n + 2) > m = |w'|$ . Thus  $w' \notin \text{Pad}_f^\#(L)$  (wrong length) and we reject
  - ▶ Computation finished: If  $f(n + 2) \neq m$ , then we reject (wrong length).
- ④ Since  $L \in \text{DTIME}(O(f; g))$  we check whether  $w \in L$  in time  $O(g(f(n + 2) + 2)) = O(g(m + 2))$  due to  $m = f(n + 2)$ .

Hence  $w' \in \text{Pad}_f^\#(L)$  iff  $w \in L$  and thus  $\text{Pad}_f^\#(L) \in \text{DTIME}(O(g))$ . □

## §4.4 Theorem (Translation theorem for space classes)

Let  $f, g: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq n$  and  $g(n) \geq \log n$  for all  $n \in \mathbb{N}$  and  $g$  space-constructible. Let binary representation of  $f(n+2)$  be computable in space  $g(f(n+2) + 2)$  on input  $\#^n$ . For arbitrary  $L \subseteq \Sigma^*$  and  $\# \notin \Sigma$

- 1  $\text{Pad}_f^\#(L) \in \text{DSPACE}(g)$  iff  $L \in \text{DSPACE}(f ; g)$
- 2  $\text{Pad}_f^\#(L) \in \text{NSPACE}(g)$  iff  $L \in \text{NSPACE}(f ; g)$

## §4.4 Theorem (Translation theorem for space classes)

Let  $f, g: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) \geq n$  and  $g(n) \geq \log n$  for all  $n \in \mathbb{N}$  and  $g$  space-constructible. Let binary representation of  $f(n+2)$  be computable in space  $g(f(n+2) + 2)$  on input  $\#^n$ . For arbitrary  $L \subseteq \Sigma^*$  and  $\# \notin \Sigma$

- 1  $\text{Pad}_f^\#(L) \in \text{DSPACE}(g)$  iff  $L \in \text{DSPACE}(f ; g)$
- 2  $\text{Pad}_f^\#(L) \in \text{NSPACE}(g)$  iff  $L \in \text{NSPACE}(f ; g)$

## Consequences

- Collapse of hierarchy most likely on larger classes
- Separation results most likely for smaller classes

## §4.5 Corollary

**DCSL** =  $\text{DSPACE}(x) \neq \text{NSPACE}(x) = \text{CSL}$  implies  $\text{L} \neq \text{NL}$

## §4.5 Corollary

**DCSL** =  $\text{DSPACE}(x) \neq \text{NSPACE}(x) = \text{CSL}$  implies  $\mathbf{L} \neq \mathbf{NL}$

Proof (by contraposition)

Let  $\mathbf{L} = \mathbf{NL}$  and  $L \in \text{NSPACE}(x)$  with  $L \subseteq \Sigma^*$ .

## §4.5 Corollary

**DCSL** =  $\text{DSPACE}(x) \neq \text{NSPACE}(x) = \text{CSL}$  implies  $\mathbf{L} \neq \mathbf{NL}$

### Proof (by contraposition)

Let  $\mathbf{L} = \mathbf{NL}$  and  $L \in \text{NSPACE}(x)$  with  $L \subseteq \Sigma^*$ . Let  $\text{exp}: \mathbb{N} \rightarrow \mathbb{N}$  be such that  $\text{exp}(n) = 2^n$  for all  $n \in \mathbb{N}$  and  $\# \notin \Sigma$ . Then

$$\text{Pad}_{\text{exp}}^{\#}(L) \in \text{NSPACE}(\log x) = \mathbf{NL} = \mathbf{L} = \text{DSPACE}(\log x)$$

## §4.5 Corollary

**DCSL** =  $\text{DSPACE}(x) \neq \text{NSPACE}(x) = \text{CSL}$  implies  $\mathbf{L} \neq \mathbf{NL}$

### Proof (by contraposition)

Let  $\mathbf{L} = \mathbf{NL}$  and  $L \in \text{NSPACE}(x)$  with  $L \subseteq \Sigma^*$ . Let  $\text{exp}: \mathbb{N} \rightarrow \mathbb{N}$  be such that  $\text{exp}(n) = 2^n$  for all  $n \in \mathbb{N}$  and  $\# \notin \Sigma$ . Then

$$\text{Pad}_{\text{exp}}^{\#}(L) \in \text{NSPACE}(\log x) = \mathbf{NL} = \mathbf{L} = \text{DSPACE}(\log x)$$

Hence by translation theorem for space classes (Theorem §4.4) we have  $L \in \text{DSPACE}(\text{exp}; \log) = \text{DSPACE}(x)$ . □

## §4.6 Corollary

$P \neq DSPACE(x) = DCSL$

## §4.6 Corollary

$\mathbf{P} \neq \mathbf{DSPACE}(x) = \mathbf{DCSL}$

Proof (by contradiction)

Let  $\mathbf{P} = \mathbf{DSPACE}(x)$ . Additionally, let  $L \in \mathbf{DSPACE}(x^2) \setminus \mathbf{DSPACE}(x)$  (different by space hierarchy) and  $f: \mathbb{N} \rightarrow \mathbb{N}$  wit  $f(n) = n^2$  for all  $n \in \mathbb{N}$ . Even further, let  $L \subseteq \Sigma^*$  and  $\# \notin \Sigma$ .

## §4.6 Corollary

$$\mathbf{P} \neq \mathbf{DSPACE}(x) = \mathbf{DCSL}$$

Proof (by contradiction)

Let  $\mathbf{P} = \mathbf{DSPACE}(x)$ . Additionally, let  $L \in \mathbf{DSPACE}(x^2) \setminus \mathbf{DSPACE}(x)$  (different by space hierarchy) and  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) = n^2$  for all  $n \in \mathbb{N}$ . Even further, let  $L \subseteq \Sigma^*$  and  $\# \notin \Sigma$ . Then

$$\text{Pad}_f^\#(L) \in \mathbf{DSPACE}(x) = \mathbf{P}$$

and thus  $\text{Pad}_f^\#(L) \in \mathbf{DTIME}(x^i)$  for some  $i \in \mathbb{N}_+$ .

## §4.6 Corollary

$$\mathbf{P} \neq \mathbf{DSPACE}(x) = \mathbf{DCSL}$$

Proof (by contradiction)

Let  $\mathbf{P} = \mathbf{DSPACE}(x)$ . Additionally, let  $L \in \mathbf{DSPACE}(x^2) \setminus \mathbf{DSPACE}(x)$  (different by space hierarchy) and  $f: \mathbb{N} \rightarrow \mathbb{N}$  with  $f(n) = n^2$  for all  $n \in \mathbb{N}$ . Even further, let  $L \subseteq \Sigma^*$  and  $\# \notin \Sigma$ . Then

$$\text{Pad}_f^\#(L) \in \mathbf{DSPACE}(x) = \mathbf{P}$$

and thus  $\text{Pad}_f^\#(L) \in \mathbf{DTIME}(x^i)$  for some  $i \in \mathbb{N}_+$ . Hence by the translation theorem for time classes (Theorem §4.3)

$$L \in \mathbf{DTIME}(O(f; x^i)) = \mathbf{DTIME}(O(x^{2i})) \subseteq \mathbf{P} = \mathbf{DSPACE}(x)$$

which shows the contradiction. □

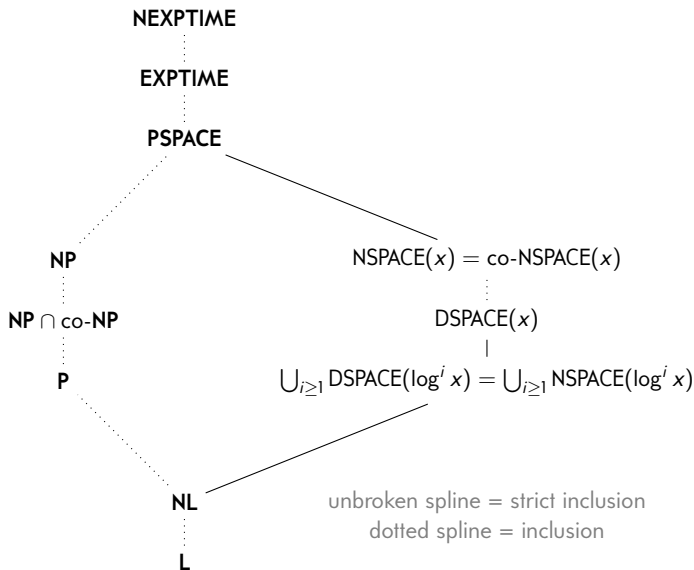
## Notes

- $P \neq DSPACE(x) = \mathbf{DCSL}$   
(**DCSL** = class of deterministic context-sensitive languages)

## Notes

- $P \neq DSPACE(x) = DCSL$   
(**DCSL** = class of deterministic context-sensitive languages)
- Following relations are possible
  - ▶  $P \subsetneq DSPACE(x)$
  - ▶  $DSPACE(x) \subsetneq P$
  - ▶  $P = DSPACE(\log x) = L$

# Summary (with additional results)



# Complements of Nondeterministic Space Classes

## Neil Immerman (\* 1953)

- US computer scientist
- Founder of descriptive complexity theory
- Received Gödel award



## Róbert Szelepcsényi (\* 1966)

- Slovak mathematician & computer scientist
- Independently achieved same result
- Received Gödel award

## §4.7 Theorem (Immerman-Szelepcsényi theorem, 1988)

Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  monotonic increasing with  $f(n) \geq \log n$  for all  $n \in \mathbb{N}$ .  
Then  $\text{NSPACE}(f) = \text{co-NSPACE}(f)$

## §4.7 Theorem (Immerman-Szelepcsényi theorem, 1988)

Let  $f: \mathbb{N} \rightarrow \mathbb{N}$  monotonic increasing with  $f(n) \geq \log n$  for all  $n \in \mathbb{N}$ .  
Then  $\text{NSPACE}(f) = \text{co-NSPACE}(f)$

### Notes

- Question open for 20 years
- Utilize inductive counting in proof

# Complements of Nondeterministic Space Classes

## Proof (1/6)

Let  $M$   $f$ -space-bounded 1-TM and  $w \in \Sigma^*$  input of length  $n$ .

Wlog. suppose that  $M$  empties tape before acceptance. Hence only configuration  $\top = (q_+, \square, 1, 1)$  of length 1 is accepting.

We need to nondeterministically check in space  $O(f)$  whether  $w \in L(M)$ .

# Complements of Nondeterministic Space Classes

## Proof (1/6)

Let  $M$   $f$ -space-bounded 1-TM and  $w \in \Sigma^*$  input of length  $n$ .

Wlog. suppose that  $M$  empties tape before acceptance. Hence only configuration  $\top = (q_+, \square, 1, 1)$  of length 1 is accepting.

We need to nondeterministically check in space  $O(f)$  whether  $w \in L(M)$ .

We start by defining total order  $\preceq$  on configurations

$\kappa = (q, w, i, k)$  and  $\kappa' = (q', w', i', k')$  by  $\kappa \prec \kappa'$  iff

- $|w| < |w'|$  (first length)

# Complements of Nondeterministic Space Classes

## Proof (1/6)

Let  $M$   $f$ -space-bounded 1-TM and  $w \in \Sigma^*$  input of length  $n$ .

Wlog. suppose that  $M$  empties tape before acceptance. Hence only configuration  $\top = (q_+, \square, 1, 1)$  of length 1 is accepting.

We need to nondeterministically check in space  $O(f)$  whether  $w \in L(M)$ .

We start by defining total order  $\preceq$  on configurations

$\kappa = (q, w, i, k)$  and  $\kappa' = (q', w', i', k')$  by  $\kappa \prec \kappa'$  iff

- $|w| < |w'|$  (first length)
- $|w| = |w'|$  and  $w <_{\text{lex}} w'$  (lexicographically with  $\square$  minimal)

# Complements of Nondeterministic Space Classes

## Proof (1/6)

Let  $M$   $f$ -space-bounded 1-TM and  $w \in \Sigma^*$  input of length  $n$ .

Wlog. suppose that  $M$  empties tape before acceptance. Hence only configuration  $\top = (q_+, \square, 1, 1)$  of length 1 is accepting.

We need to nondeterministically check in space  $O(f)$  whether  $w \in L(M)$ .

We start by defining total order  $\preceq$  on configurations

$\kappa = (q, w, i, k)$  and  $\kappa' = (q', w', i', k')$  by  $\kappa \prec \kappa'$  iff

- $|w| < |w'|$  (first length)
- $|w| = |w'|$  and  $w <_{\text{lex}} w'$  (lexicographically with  $\square$  minimal)
- $w = w'$  and  $k < k'$  (then tape position)

# Complements of Nondeterministic Space Classes

## Proof (1/6)

Let  $M$   $f$ -space-bounded 1-TM and  $w \in \Sigma^*$  input of length  $n$ .

Wlog. suppose that  $M$  empties tape before acceptance. Hence only configuration  $\top = (q_+, \square, 1, 1)$  of length 1 is accepting.

We need to nondeterministically check in space  $O(f)$  whether  $w \in L(M)$ .

We start by defining total order  $\preceq$  on configurations

$\kappa = (q, w, i, k)$  and  $\kappa' = (q', w', i', k')$  by  $\kappa \prec \kappa'$  iff

- $|w| < |w'|$  (first length)
- $|w| = |w'|$  and  $w <_{\text{lex}} w'$  (lexicographically with  $\square$  minimal)
- $w = w'$  and  $k < k'$  (then tape position)
- $w = w'$ ,  $k = k'$ , and  $i < i'$  (then input position)

# Complements of Nondeterministic Space Classes

## Proof (1/6)

Let  $M$   $f$ -space-bounded 1-TM and  $w \in \Sigma^*$  input of length  $n$ .

Wlog. suppose that  $M$  empties tape before acceptance. Hence only configuration  $\top = (q_+, \square, 1, 1)$  of length 1 is accepting.

We need to nondeterministically check in space  $O(f)$  whether  $w \in L(M)$ .

We start by defining total order  $\preceq$  on configurations

$\kappa = (q, w, i, k)$  and  $\kappa' = (q', w', i', k')$  by  $\kappa \prec \kappa'$  iff

- $|w| < |w'|$  (first length)
- $|w| = |w'|$  and  $w <_{\text{lex}} w'$  (lexicographically with  $\square$  minimal)
- $w = w'$  and  $k < k'$  (then tape position)
- $w = w', k = k',$  and  $i < i'$  (then input position)
- $w = w', k = k', i = i',$  and  $q < q'$  (finally by state)

for arbitrary total order  $\leq$  on  $Q$  such that  $q_+ \leq q$  for all  $q \in Q$

# Complements of Nondeterministic Space Classes

## Proof (2/6)

Additionally for all  $k \in \mathbb{N}$  let

$$R(k) = \{\kappa \mid \exists i \leq k: I_w \Rightarrow_M^i \kappa\} \quad (\text{conf. reachable in } \leq k \text{ steps})$$

$$r(k) = |R(k)| \quad (\text{number of conf. reachable in } \leq k \text{ steps})$$

$$\ell(k) = \max\{|\kappa| \mid \kappa \in R(k)\} \quad (\text{max. length of configurations of } R(k))$$

$$\hat{r} = \max\{r(k) \mid k \in \mathbb{N}\} \quad (\text{number of reachable conf.})$$

Obviously  $r(k) \leq \hat{r}$  and there exists  $g \in 2^{O(f)}$  such that  $\hat{r} \leq g(n+2)$  by Lemma §1.7.

# Complements of Nondeterministic Space Classes

## Proof (2/6)

Additionally for all  $k \in \mathbb{N}$  let

$$R(k) = \{\kappa \mid \exists i \leq k: I_w \Rightarrow_M^i \kappa\} \quad (\text{conf. reachable in } \leq k \text{ steps})$$

$$r(k) = |R(k)| \quad (\text{number of conf. reachable in } \leq k \text{ steps})$$

$$\ell(k) = \max\{|\kappa| \mid \kappa \in R(k)\} \quad (\text{max. length of configurations of } R(k))$$

$$\hat{r} = \max\{r(k) \mid k \in \mathbb{N}\} \quad (\text{number of reachable conf.})$$

Obviously  $r(k) \leq \hat{r}$  and there exists  $g \in 2^{O(f)}$  such that  $\hat{r} \leq g(n+2)$  by Lemma §1.7. We construct  $O(f)$ -space-bounded TM that

- Computes  $\hat{r}$  if  $w \notin L(M)$ ; single accepting computation sufficient
- Has no accepting computations if  $w \in L(M)$

# Complements of Nondeterministic Space Classes

## Proof (3/6)

We suppose that  $r(k+1)$  can be computed from  $r = r(k)$  with the help of  $\text{compute-}r(k+1, r)$  in space  $O(f)$

### Main program

- ① Let  $k = 0$  and  $r = 1$  (only 1 conf. reachable)

# Complements of Nondeterministic Space Classes

## Proof (3/6)

We suppose that  $r(k+1)$  can be computed from  $r = r(k)$  with the help of  $\text{compute-}r(k+1, r)$  in space  $O(f)$

### Main program

- ① Let  $k = 0$  and  $r = 1$  (only 1 conf. reachable)
- ② Repeat forever
  - ①  $r' = \text{compute-}r(k+1, r)$  (compute  $r(k+1)$ )

# Complements of Nondeterministic Space Classes

## Proof (3/6)

We suppose that  $r(k+1)$  can be computed from  $r = r(k)$  with the help of  $\text{compute-}r(k+1, r)$  in space  $O(f)$

### Main program

- ① Let  $k = 0$  and  $r = 1$  (only 1 conf. reachable)
- ② Repeat forever
  - ①  $r' = \text{compute-}r(k+1, r)$  (compute  $r(k+1)$ )
  - ② If  $r' = r$  then return  $r$  (no further conf.)

# Complements of Nondeterministic Space Classes

## Proof (3/6)

We suppose that  $r(k+1)$  can be computed from  $r = r(k)$  with the help of  $\text{compute-}r(k+1, r)$  in space  $O(f)$

### Main program

- ① Let  $k = 0$  and  $r = 1$  (only 1 conf. reachable)
- ② Repeat forever
  - ①  $r' = \text{compute-}r(k+1, r)$  (compute  $r(k+1)$ )
  - ② If  $r' = r$  then return  $r$  (no further conf.)
  - ③ Otherwise let  $k \leftarrow k+1$  and  $r \leftarrow r'$  (next iteration)

Obviously we require space  $O(f)$  for this part since  $\hat{r} \leq g(n+2)$

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

- 1 Let  $\kappa = \top$  and  $\ell = 1$  (only length 1 reachable)
- 2 Repeat  $r$  times

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

- ① Let  $\kappa = \top$  and  $\ell = 1$  (only length 1 reachable)
- ② Repeat  $r$  times
  - ① Compute arbitrary  $\kappa' \in R(k)$

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

- ① Let  $\kappa = \top$  and  $\ell = 1$  (only length 1 reachable)
- ② Repeat  $r$  times
  - ① Compute arbitrary  $\kappa' \in R(k)$
  - ② If  $\kappa \prec \kappa'$  then set  $\kappa \leftarrow \kappa'$  and  $\ell \leftarrow |\kappa'|$  (next iteration)

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

- ① Let  $\kappa = \top$  and  $\ell = 1$  (only length 1 reachable)
- ② Repeat  $r$  times
  - ① Compute arbitrary  $\kappa' \in R(k)$
  - ② If  $\kappa \prec \kappa'$  then set  $\kappa \leftarrow \kappa'$  and  $\ell \leftarrow |\kappa'|$  (next iteration)
  - ③ Otherwise fail computation (error, computation unsuccessful)
- ③ Return  $\ell$

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

- ① Let  $\kappa = \top$  and  $\ell = 1$  (only length 1 reachable)
- ② Repeat  $r$  times
  - ① Compute arbitrary  $\kappa' \in R(k)$
  - ② If  $\kappa \prec \kappa'$  then set  $\kappa \leftarrow \kappa'$  and  $\ell \leftarrow |\kappa'|$  (next iteration)
  - ③ Otherwise fail computation (error, computation unsuccessful)
- ③ Return  $\ell$

Obviously we also need only space  $O(f)$  for this part since  $M$  is  $f$ -space-bounded.

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

- ① Let  $\kappa = \top$  and  $\ell = 1$  (only length 1 reachable)
- ② Repeat  $r$  times
  - ① Compute arbitrary  $\kappa' \in R(k)$
  - ② If  $\kappa \prec \kappa'$  then set  $\kappa \leftarrow \kappa'$  and  $\ell \leftarrow |\kappa'|$  (next iteration)
  - ③ Otherwise fail computation (error, computation unsuccessful)
- ③ Return  $\ell$

Obviously we also need only space  $O(f)$  for this part since  $M$  is  $f$ -space-bounded. If  $\top \in R(k)$  then we cannot explore  $r = r(k)$  strictly larger configurations. Hence program fails in this case.

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

- ① Let  $\kappa = \top$  and  $\ell = 1$  (only length 1 reachable)
- ② Repeat  $r$  times
  - ① Compute arbitrary  $\kappa' \in R(k)$
  - ② If  $\kappa \prec \kappa'$  then set  $\kappa \leftarrow \kappa'$  and  $\ell \leftarrow |\kappa'|$  (next iteration)
  - ③ Otherwise fail computation (error, computation unsuccessful)
- ③ Return  $\ell$

Obviously we also need only space  $O(f)$  for this part since  $M$  is  $f$ -space-bounded. If  $\top \in R(k)$  then we cannot explore  $r = r(k)$  strictly larger configurations. Hence program fails in this case. For  $w \in L(M)$  there exists  $k \in \mathbb{N}$  such that  $\top \in R(k)$  and this will fail when computing  $\hat{r}$ .

# Complements of Nondeterministic Space Classes

## Proof (4/6)

Next we implement `compute-r`. We need function `compute- $\ell(k, r)$`  to compute  $\ell(k)$  based on  $r = r(k)$

- ① Let  $\kappa = \top$  and  $\ell = 1$  (only length 1 reachable)
- ② Repeat  $r$  times
  - ① Compute arbitrary  $\kappa' \in R(k)$
  - ② If  $\kappa \prec \kappa'$  then set  $\kappa \leftarrow \kappa'$  and  $\ell \leftarrow |\kappa'|$  (next iteration)
  - ③ Otherwise fail computation (error, computation unsuccessful)
- ③ Return  $\ell$

Obviously we also need only space  $O(f)$  for this part since  $M$  is  $f$ -space-bounded. If  $\top \in R(k)$  then we cannot explore  $r = r(k)$  strictly larger configurations. Hence program fails in this case. For  $w \in L(M)$  there exists  $k \in \mathbb{N}$  such that  $\top \in R(k)$  and this will fail when computing  $\hat{r}$ . If  $\top \notin R(k)$  we can explore configurations in correct order and thus  $\ell = \ell(k)$  afterwards.

# Complements of Nondeterministic Space Classes

## Proof (5/6)

Now we implement  $\text{compute-r}(k+1, r)$  based on  $r = r(k)$

- ① Let  $r' = 0$  and  $\ell = \text{compute-}\ell(k, r)$  (compute max. length for  $k$ )
- ② For all configurations  $\kappa$  with  $|\kappa| \leq \ell + 1$

# Complements of Nondeterministic Space Classes

## Proof (5/6)

Now we implement  $\text{compute-r}(k+1, r)$  based on  $r = r(k)$

- ① Let  $r' = 0$  and  $\ell = \text{compute-}\ell(k, r)$  (compute max. length for  $k$ )
- ② For all configurations  $\kappa$  with  $|\kappa| \leq \ell + 1$ 
  - ① If  $\text{Reach}(r, k+1, \kappa)$  then set  $r' \leftarrow r' + 1$  (next iteration)

# Complements of Nondeterministic Space Classes

## Proof (5/6)

Now we implement  $\text{compute-r}(k+1, r)$  based on  $r = r(k)$

- ① Let  $r' = 0$  and  $\ell = \text{compute-}\ell(k, r)$  (compute max. length for  $k$ )
- ② For all configurations  $\kappa$  with  $|\kappa| \leq \ell + 1$ 
  - ① If  $\text{Reach}(r, k+1, \kappa)$  then set  $r' \leftarrow r' + 1$  (next iteration)
- ③ Return  $r'$

# Complements of Nondeterministic Space Classes

## Proof (5/6)

Now we implement  $\text{compute-r}(k+1, r)$  based on  $r = r(k)$

- ① Let  $r' = 0$  and  $\ell = \text{compute-}\ell(k, r)$  (compute max. length for  $k$ )
- ② For all configurations  $\kappa$  with  $|\kappa| \leq \ell + 1$ 
  - ① If  $\text{Reach}(r, k+1, \kappa)$  then set  $r' \leftarrow r' + 1$  (next iteration)
- ③ Return  $r'$

Obviously also this part requires only space  $O(f)$  and the length restriction is sensible since  $\ell(k+1) \leq \ell(k) + 1$ .

# Complements of Nondeterministic Space Classes

## Proof (5/6)

Now we implement  $\text{compute-r}(k+1, r)$  based on  $r = r(k)$

- ① Let  $r' = 0$  and  $\ell = \text{compute-}\ell(k, r)$  (compute max. length for  $k$ )
- ② For all configurations  $\kappa$  with  $|\kappa| \leq \ell + 1$ 
  - ① If  $\text{Reach}(r, k+1, \kappa)$  then set  $r' \leftarrow r' + 1$  (next iteration)
- ③ Return  $r'$

Obviously also this part requires only space  $O(f)$  and the length restriction is sensible since  $\ell(k+1) \leq \ell(k) + 1$ . If  $w \in L(M)$  then there exists minimal  $k \in \mathbb{N}$  such that  $\top \in R(k)$ .

# Complements of Nondeterministic Space Classes

## Proof (5/6)

Now we implement  $\text{compute-r}(k+1, r)$  based on  $r = r(k)$

- ① Let  $r' = 0$  and  $\ell = \text{compute-}\ell(k, r)$  (compute max. length for  $k$ )
- ② For all configurations  $\kappa$  with  $|\kappa| \leq \ell + 1$ 
  - ① If  $\text{Reach}(r, k+1, \kappa)$  then set  $r' \leftarrow r' + 1$  (next iteration)
- ③ Return  $r'$

Obviously also this part requires only space  $O(f)$  and the length restriction is sensible since  $\ell(k+1) \leq \ell(k) + 1$ . If  $w \in L(M)$  then there exists minimal  $k \in \mathbb{N}$  such that  $\top \in R(k)$ . During computation of  $\text{compute-r}(k+1, r)$  the call  $\text{compute-}\ell(k, r)$  will fail, which yields that no accepting computation can exist.

# Complements of Nondeterministic Space Classes

## Proof (5/6)

Now we implement  $\text{compute-r}(k+1, r)$  based on  $r = r(k)$

- ① Let  $r' = 0$  and  $\ell = \text{compute-}\ell(k, r)$  (compute max. length for  $k$ )
- ② For all configurations  $\kappa$  with  $|\kappa| \leq \ell + 1$ 
  - ① If  $\text{Reach}(r, k+1, \kappa)$  then set  $r' \leftarrow r' + 1$  (next iteration)
- ③ Return  $r'$

Obviously also this part requires only space  $O(f)$  and the length restriction is sensible since  $\ell(k+1) \leq \ell(k) + 1$ . If  $w \in L(M)$  then there exists minimal  $k \in \mathbb{N}$  such that  $\top \in R(k)$ . During computation of  $\text{compute-r}(k+1, r)$  the call  $\text{compute-}\ell(k, r)$  will fail, which yields that no accepting computation can exist. An accepting computation of  $\hat{r}$  is thus possible iff  $w \notin L(M)$ . If computation of  $\hat{r}$  succeeds, then we accept the input  $w$ .

## Proof (6/6)

Only  $\text{Reach}(r, k+1, \kappa)$  is missing, which for configuration  $\kappa$  checks whether  $\kappa \in R(k+1)$  based on  $r = r(k)$

- 1 Let  $\kappa' = \top$
- 2 Repeat  $r$  times

# Complements of Nondeterministic Space Classes

## Proof (6/6)

Only  $\text{Reach}(r, k+1, \kappa)$  is missing, which for configuration  $\kappa$  checks whether  $\kappa \in R(k+1)$  based on  $r = r(k)$

- ① Let  $\kappa' = \top$
- ② Repeat  $r$  times
  - ① Compute configuration  $\kappa'' \in R(k)$
  - ② If  $\kappa'' \preceq \kappa'$  then fail computation
  - ③ Otherwise if  $\kappa'' = \kappa$  or  $\kappa'' \Rightarrow_M \kappa$ , then return **true**
  - ④ Otherwise let  $\kappa' \leftarrow \kappa''$  (next iteration)

# Complements of Nondeterministic Space Classes

## Proof (6/6)

Only  $\text{Reach}(r, k+1, \kappa)$  is missing, which for configuration  $\kappa$  checks whether  $\kappa \in R(k+1)$  based on  $r = r(k)$

- ① Let  $\kappa' = \top$
- ② Repeat  $r$  times
  - ① Compute configuration  $\kappa'' \in R(k)$
  - ② If  $\kappa'' \preceq \kappa'$  then fail computation
  - ③ Otherwise if  $\kappa'' = \kappa$  or  $\kappa'' \Rightarrow_M \kappa$ , then return **true**
  - ④ Otherwise let  $\kappa' \leftarrow \kappa''$  (next iteration)
- ③ Return **false**

# Complements of Nondeterministic Space Classes

## Proof (6/6)

Only  $\text{Reach}(r, k+1, \kappa)$  is missing, which for configuration  $\kappa$  checks whether  $\kappa \in R(k+1)$  based on  $r = r(k)$

- ① Let  $\kappa' = \top$
- ② Repeat  $r$  times
  - ① Compute configuration  $\kappa'' \in R(k)$
  - ② If  $\kappa'' \preceq \kappa'$  then fail computation
  - ③ Otherwise if  $\kappa'' = \kappa$  or  $\kappa'' \Rightarrow_M \kappa$ , then return **true**
  - ④ Otherwise let  $\kappa' \leftarrow \kappa''$
- ③ Return **false**

(next iteration)

Once again only space  $O(f)$  is needed.

# Complements of Nondeterministic Space Classes

## Proof (6/6)

Only  $\text{Reach}(r, k+1, \kappa)$  is missing, which for configuration  $\kappa$  checks whether  $\kappa \in R(k+1)$  based on  $r = r(k)$

- ① Let  $\kappa' = \top$
- ② Repeat  $r$  times
  - ① Compute configuration  $\kappa'' \in R(k)$
  - ② If  $\kappa'' \preceq \kappa'$  then fail computation
  - ③ Otherwise if  $\kappa'' = \kappa$  or  $\kappa'' \Rightarrow_M \kappa$ , then return **true**
  - ④ Otherwise let  $\kappa' \leftarrow \kappa''$  (next iteration)
- ③ Return **false**

Once again only space  $O(f)$  is needed. If the computation does not fail, then all configurations of  $R(k)$  are correctly enumerated and it is correctly checked whether  $\kappa$  belongs or can be derived in single step.  $\square$

# Summary

- Savitch's theorem
- Translation theorems & applications
- Immerman-Szelepcsényi theorem

Fourth exercise sheet available in Moodle