

# Theory of Computer Science

## D2. Polynomial Reductions and NP-completeness

Gabriele Röger

University of Basel

May 5, 2021

# Theory of Computer Science

May 5, 2021 — D2. Polynomial Reductions and NP-completeness

## D2.1 Polynomial Reductions

## D2.2 NP-Hardness and NP-Completeness

## D2.3 Summary

# D2.1 Polynomial Reductions

# Polynomial Reductions: Idea

- ▶ **Reductions** are a common and powerful concept in computer science. We know them from Part C.
- ▶ The basic idea is that we solve a new problem by **reducing** it to a known problem.
- ▶ In complexity theory we want to use reductions that allow us to prove statements of the following kind:  
*Problem A can be solved efficiently  
if problem B can be solved efficiently.*
- ▶ For this, we need a reduction from  $A$  to  $B$  that can be computed efficiently itself (otherwise it would be useless for efficiently solving  $A$ ).

# Polynomial Reductions

## Definition (Polynomial Reduction)

Let  $A \subseteq \Sigma^*$  and  $B \subseteq \Gamma^*$  be decision problems.

We say that  $A$  can be polynomially reduced to  $B$ , written  $A \leq_p B$ , if there is a function  $f : \Sigma^* \rightarrow \Gamma^*$  such that:

- ▶  $f$  can be computed in polynomial time by a DTM
  - ▶ i. e., there is a polynomial  $p$  and a DTM  $M$  such that  $M$  computes  $f(w)$  in at most  $p(|w|)$  steps given input  $w \in \Sigma^*$
- ▶  $f$  reduces  $A$  to  $B$ 
  - ▶ i. e., for all  $w \in \Sigma^*$ :  $w \in A$  iff  $f(w) \in B$

$f$  is called a polynomial reduction from  $A$  to  $B$

**German:**  $A$  polynomiell auf  $B$  reduzierbar,  
polynomielle Reduktion von  $A$  auf  $B$

# Polynomial Reductions: Remarks

- ▶ Polynomial reductions are also called **Karp reductions** (after Richard Karp, who wrote a famous paper describing many such reductions in 1972).
- ▶ In practice, of course we do not have to specify a DTM for  $f$ : it just has to be clear that  $f$  can be computed in **polynomial time** by a **deterministic algorithm**.

# Polynomial Reductions: Example (1)

## Definition (HAMILTONCYCLE)

**HAMILTONCYCLE** is the following decision problem:

- ▶ **Given:** undirected graph  $G = \langle V, E \rangle$
- ▶ **Question:** Does  $G$  contain a Hamilton cycle?

Reminder:

## Definition (Hamilton Cycle)

A **Hamilton cycle** of  $G$  is a sequence of vertices in  $V$ ,  $\pi = \langle v_0, \dots, v_n \rangle$ , with the following properties:

- ▶  $\pi$  is a **path**: there is an edge from  $v_i$  to  $v_{i+1}$  for all  $0 \leq i < n$
- ▶  $\pi$  is a **cycle**:  $v_0 = v_n$
- ▶  $\pi$  is **simple**:  $v_i \neq v_j$  for all  $i \neq j$  with  $i, j < n$
- ▶  $\pi$  is **Hamiltonian**: all nodes of  $V$  are included in  $\pi$

## Polynomial Reductions: Example (2)

### Definition (TSP)

**TSP** (traveling salesperson problem) is the following decision problem:

- ▶ **Given:** finite set  $S \neq \emptyset$  of cities, symmetric cost function  $cost : S \times S \rightarrow \mathbb{N}_0$ , cost bound  $K \in \mathbb{N}_0$
- ▶ **Question:** Is there a tour with total cost at most  $K$ , i. e., a permutation  $\langle s_1, \dots, s_n \rangle$  of the cities with 
$$\sum_{i=1}^{n-1} cost(s_i, s_{i+1}) + cost(s_n, s_1) \leq K?$$

**German:** Problem der/des Handlungsreisenden



## Polynomial Reductions: Example (3)

Theorem ( $\text{HAMILTONCYCLE} \leq_p \text{TSP}$ )  
 $\text{HAMILTONCYCLE} \leq_p \text{TSP}$ .

Proof.

$\rightsquigarrow$  blackboard



## Exercise: Polynomial Reduction

### Definition (HAMILTONIANCOMPLETION)

**HAMILTONIANCOMPLETION** is the following decision problem:

- ▶ **Given:** undirected graph  $G = \langle V, E \rangle$ , number  $k \in \mathbb{N}_0$
- ▶ **Question:** Can  $G$  be extended with at most  $k$  edges such that the resulting graph has a Hamilton cycle?

Show that

$\text{HAMILTONCYCLE} \leq_p \text{HAMILTONIANCOMPLETION}$ .



## Reminder: P and NP

**P**: class of languages that are decidable in polynomial time  
by a deterministic Turing machine

**NP**: class of languages that are decidable in polynomial time  
by a non-deterministic Turing machine

# Properties of Polynomial Reductions (1)

## Theorem (Properties of Polynomial Reductions)

*Let  $A$ ,  $B$  and  $C$  decision problems.*

- ❶ *If  $A \leq_p B$  and  $B \in P$ , then  $A \in P$ .*
- ❷ *If  $A \leq_p B$  and  $B \in NP$ , then  $A \in NP$ .*
- ❸ *If  $A \leq_p B$  and  $A \notin P$ , then  $B \notin P$ .*
- ❹ *If  $A \leq_p B$  and  $A \notin NP$ , then  $B \notin NP$ .*
- ❺ *If  $A \leq_p B$  and  $B \leq_p C$ , then  $A \leq_p C$ .*

## Properties of Polynomial Reductions (2)

Proof.

for 1.:

We must show that there is a DTM accepting  $A$  in polynomial time.

We know:

- ▶ There is a DTM  $M_B$  that accepts  $B$  in time  $p$ , where  $p$  is a polynomial.
- ▶ There is a DTM  $M_f$  that computes a reduction from  $A$  to  $B$  in time  $q$ , where  $q$  is a polynomial.

...

## Properties of Polynomial Reductions (3)

### Proof (continued).

Consider the machine  $M$  that first behaves like  $M_f$ , and then (after  $M_f$  stops) behaves like  $M_B$  on the output of  $M_f$ .

$M$  accepts  $A$ :

- ▶  $M$  behaves on input  $w$  as  $M_B$  does on input  $f(w)$ , so it accepts  $w$  if and only if  $f(w) \in B$ .
- ▶ Because  $f$  is a reduction,  $w \in A$  iff  $f(w) \in B$ .

...

# Properties of Polynomial Reductions (4)

Proof (continued).

Computation time of  $M$  on input  $w$ :

- ▶ first  $M_f$  runs on input  $w$ :  $\leq q(|w|)$  steps
- ▶ then  $M_B$  runs on input  $f(w)$ :  $\leq p(|f(w)|)$  steps
- ▶  $|f(w)| \leq |w| + q(|w|)$  because in  $q(|w|)$  steps,  $M_f$  can write at most  $q(|w|)$  additional symbols onto the tape
- ↪ total computation time  $\leq q(|w|) + p(|f(w)|)$   
 $\leq q(|w|) + p(|w| + q(|w|))$
- ↪ this is polynomial in  $|w| \rightsquigarrow A \in \text{P}$ .

...

# Properties of Polynomial Reductions (5)

Proof (continued).

for 2.:

analogous to 1., only that  $M_B$  and  $M$  are NTMs

of 3.+4.:

equivalent formulations of 1.+2. (contraposition)

of 5.:

Let  $A \leq_p B$  with reduction  $f$  and  $B \leq_p C$  with reduction  $g$ .  
Then  $g \circ f$  is a reduction of  $A$  to  $C$ .

The computation time of the two computations in sequence  
is polynomial by the same argument used in the proof for 1. □



# D2.2 NP-Hardness and NP-Completeness

# NP-Hardness and NP-Completeness

## Definition (NP-Hard, NP-Complete)

Let  $B$  be a decision problem.

$B$  is called **NP-hard** if  $A \leq_p B$  for **all** problems  $A \in \text{NP}$ .

$B$  is called **NP-complete** if  $B \in \text{NP}$  and  $B$  is NP-hard.

**German:** NP-hart (selten: NP-schwer), NP-vollständig

# NP-Complete Problems: Meaning

- ▶ NP-hard problems are “at least as difficult” as all problems in NP.
- ▶ NP-complete problems are “the most difficult” problems in NP: **all** problems in NP can be reduced to them.
- ▶ If  $A \in P$  for **any** NP-complete problem, then  $P = NP$ . (**Why?**)
- ▶ That means that either there are efficient algorithms for **all** NP-complete problems or for **none** of them.
- ▶ **Do NP-complete problems actually exist?**

## D2.3 Summary

# Summary

- ▶ **polynomial reductions:**  $A \leq_p B$  if there is a total function  $f$  computable in polynomial time, such that for all words  $w$ :  $w \in A$  iff  $f(w) \in B$
- ▶  $A \leq_p B$  implies that  $A$  is “at most as difficult” as  $B$
- ▶ polynomial reductions are **transitive**
- ▶ **NP-hard** problems  $B$ :  $A \leq_p B$  for **all**  $A \in \text{NP}$
- ▶ **NP-complete** problems  $B$ :  $B \in \text{NP}$  and  $B$  is NP-hard