# Formale Methoden der Informatik

## Block 1: Computability and Complexity

## Exercises (Examples)

Mantas Šimkus

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

WS 2013

## Exercise 1

By providing a reduction from the **HALTING** problem, prove that the following problem is undecidable:

**HELLO-WORLD**

INSTANCE: A string $I$ and a program $\Pi$ that takes one string as input and outputs a string.
QUESTION: Does the program $\Pi$ on input $I$ return the string "Hello world!" as output?

## Solution to Exercise 1

The reduction is defined as follows. Let $(\Pi, I)$ be an arbitrary instance of **HALTING**. We build an instance $(\Pi', I')$ of **HELLO-WORLD** by setting $I' = I$ and constructing $\Pi'$ as follows:

```
String Π′ (String S)
Type val = Π(S); /* Type is the type of the output of Π*/
return "Hello world!";
```

In other words, for an instance $x = (\Pi, I)$, the instance $R(x)$ resulting from the reduction is $(\Pi', I')$. To prove the correctness of the reduction we have to show: $(\Pi, I)$ is a positive instance of **HALTING** $\Leftrightarrow$ $(\Pi', I')$ is a positive instance of **HELLO-WORLD**.

"$\Rightarrow$ Assume $(\Pi, I)$ is positive, i.e. $\Pi$ terminates on $I$. Then by the construction of $(\Pi', I')$, we have that $\Pi'$ returns "Hello world!" on $I'$. It follows that $(\Pi', I')$ is a positive instance of **HELLO-WORLD**.

"$\Leftarrow$ Assume $(\Pi', I')$ is positive, i.e. $\Pi'$ returns "Hello world!" on the input $I'$. Since $\Pi'$ involves the call $\Pi(I')$ and since $I' = I$, we have that $\Pi$ terminates on $I$, i.e. that $(\Pi, I)$ is a positive instance of **HALTING**.

## Solution to Exercise 1: Why does it work?

Why does the reduction $R$ prove the undecidability of **HELLO-WORLD**?

Towards a contradiction, suppose **HELLO-WORLD** is decidable. Then there is an algorithm $\Pi_{hw}(\cdot)$ such that $\Pi_{hw}(x)$ returns *true* if $x$ is a positive instance of **HELLO-WORLD**, and returns *false* otherwise.

Build a procedure $\Pi_h$, which takes instances of **HALTING**, as follows:

```
Bool Π_h(String Π,String I)
return Π_hw(R((Π, I)));
```

It is easy to see that $\Pi_h$ is a decision procedure for **HALTING**:

- $\Pi_h(\Pi, I)$ returns *true* if $\Pi$ terminates on $I$
- $\Pi_h(\Pi, I)$ returns *false* if $\Pi$ does not terminate on $I$

We arrive at a contradiction: we know from the lecture that **HALTING** is undecidable.

## Sanity test

Check that the problem instances that you are using in your solutions are compatible with the definition of a given problem:

- INSTANCE: A pair $(\Pi, I)$, where $\Pi$ is a program that takes one string as input and returns a string, and $I$ is a string.

  In a proof:

  - $(\Pi, I)$, $(\Pi', I')$, $(\Pi, \text{"}hello\text{"})$ are O.K.
  - $(\Pi, I, I')$, $(\Pi, I, k)$, $\Pi$ are not O.K.

- INSTANCE: A program $\Pi$ that takes one string as input and returns a string.

  In a proof:

  - $\Pi$, $\Pi'$, $\Pi_1$, $\Pi_2$, are O.K.
  - $(\Pi, I)$, $(\Pi', I')$, $(\Pi, I, I')$, $(\Pi, I, k)$ are not O.K.

## Exercise 2

Prove that the **HELLO-WORLD** problem is semi-decidable.

## Solution to Exercise 2

Recall that a decision problem $P$ is called semi-decidable if we can build a program $\Pi$ such that:

- $\Pi$ takes as input instances $I$ of $P$;
- if $I$ is a "yes" instance, then $\Pi$ returns *true*;
- if $I$ is a "no" instance, then $\Pi$ returns *false* or does not terminate;

To show that **HELLO-WORLD** is semi-decidable we can provide a procedure $\Pi'$ that fulfills the above conditions:

- $\Pi'$ that takes as input an instance of **HELLO-WORLD**, i.e. a program $\Pi$ and an input $I$,
- $\Pi'$ simulates the run of $\Pi$ on $I$.
- If the simulation reaches output "Hello world!", then $\Pi'$ returns *true*.
- If the simulation ends without output "Hello world!", then $\Pi'$ returns *false*. (Observe: termination guaranteed on positive instances only).

## Solution to Exercise 2 (continued)

We argue that such an interpreter $\Pi'$ is a semi-decision procedure for **HELLO-WORLD**. We distinguish the following cases:

- Case 1. Suppose that $(\Pi, I)$ is a positive instance, i.e., $\Pi$ on the string $I$ as input returns "Hello world". Then the simulation in $\Pi'$ will encounter the call to print "Hello world" and return *true* by the construction of $\Pi'$.
- Case 2.1. Suppose that $(\Pi, I)$ is a negative instance and that $\Pi$ halts on input $I$. Then $\Pi$ halts with output different from "Hello world". Hence, the simulation in $\Pi'$ will detect that the output is different from "Hello world". Thus, $\Pi'$ returns *false* by the construction of $\Pi'$.
- Case 2.2. Suppose that $(\Pi, I)$ is a negative instance and that $\Pi$ does not halt on input $I$. Then the simulation of this computation of $\Pi$ on $I$ by the interpreter $\Pi'$ will not terminate either. Hence, $\Pi'$ will run forever on the negative instance $(\Pi, I)$, which is a correct behavior for a semi-decision procedure.

## Exercise 3

Give a formal proof that **3-COLORABILITY** is in NP, i.e. define the required certificate relation and discuss its polynomial balance and polynomial decidability.

## Solution to Exercise 3

Define the relation

$$R = \{(G, \mu) \mid \mu \text{ is a 3-coloring for the graph } G\}.$$

- We argue that $R$ is a certificate relation for **3-COLORABILITY**. Indeed, a graph $G$ is a positive instance of **3-COLORABILITY**$\Leftrightarrow$ there exists a proper 3-coloring $\mu$ of $G \Leftrightarrow (G, \mu) \in R$.
- $R$ is polynomially balanced because a color assignment $\mu$ can be represented in space that is linear in the size of $G$.
- $R$ is polynomially decidable. Indeed, given a graph $G$ and an assignment $\mu$ of colors, checking if $\mu$ is a $k$-coloring of $G$ is feasible in polynomial time in the size of $G$ and $\mu$.

## Exercise 4

Formally prove that **VERTEX COVER** is NP-hard. For this you may use the well-known fact that **INDEPENDENT SET** is NP-complete.

## Solution to Exercise 4

We reduce **INDEPENDENT SET** to **VERTEX COVER**.

Let $(G, k)$ be an arbitrary instance of **INDEPENDENT SET**, i.e., $G$ is an undirected graph and $k$ is an integer. Let $G = (V, E)$. We construct the instance $(G, |V| - k)$ of **VERTEX COVER**. Clearly, this reduction is feasible in polynomial time.

It remains to prove the correctness. Indeed, by the proposition on page 13 in the lecture "4. NP-Completeness", vertex set $I$ is an independent set in $G$ iff $V \setminus I$ is a vertex cover in $G$. It follows that $(G, k)$ is a positive instance of **INDEPENDENT SET** iff $(G, |V| - k)$ is a positive instance of **VERTEX COVER**.

## Exercise 5

We consider a polynomial time reduction from the **3-COLORABILITY** problem for directed graphs to **SAT**. Let $G = (V, E)$ be a directed graph (i.e. an instance of **3-COLORABILITY**) with $V = \{a_1, \ldots, a_n\}$. We construct a formula $\varphi_G$ (i.e. an instance of **SAT**) as follows.
We use propositional variables $c_i^m$, where $1 \leq i \leq n$ and $m \in \{0, 1, 2\}$, to indicate that the vertex $a_i$ of $G$ is colored with color $m$. Then $\varphi_G$ is defined as $\varphi_G = \varphi_1 \wedge \varphi_2 \wedge \varphi_3$, where

$$\varphi_1 = \bigwedge_{1 \leq i \leq n} (c_i^0 \vee c_i^1 \vee c_i^2),$$

$$\varphi_2 = \bigwedge_{1 \leq i \leq n} \left(\neg(c_i^0 \wedge c_i^1) \wedge \neg(c_i^0 \wedge c_i^2) \wedge \neg(c_i^1 \wedge c_i^2)\right),$$

$$\varphi_3 = \bigwedge_{(a_i, a_j) \in E} \left(\neg(c_i^0 \wedge c_j^0) \wedge \neg(c_i^1 \wedge c_j^1) \wedge \neg(c_i^2 \wedge c_j^2)\right).$$

Prove the "$\Rightarrow$" direction of the correctness of the reduction, i.e. prove the following statement: if $G$ is 3-colorable, then $\varphi_G$ is satisfiable.

## Solution to Exercise 5

To prove "⇒", assume $G = (V, E)$ is 3-colorable. Let $\mu : V \to \{0, 1, 2\}$ be an assignment of colors that shows 3-colorability of $G$.

To show that $\varphi_G$ is satisfiable, we have to define a truth assinment $T$ that makes $\varphi_G$ evaluate to *true*.

We take the truth assignment $T$ as follows:

$$T(c_i^m) = \begin{cases} \mathit{true} & \text{if } \mu(a_i) = m, \\ \mathit{false} & \text{if } \mu(a_i) \neq m. \end{cases}$$

To show that $\varphi_G$ evaluates to *true* under $T$, it suffices to make sure that $\varphi_1, \varphi_2, \varphi_3$ evaluate to *true* under $T$.

- $\mu$ assigns at least one color to each vertex of $G$. Then due to the construction of $T$, the formula $\varphi_1$ evaluates to *true* under $T$.
- $\mu$ assigns no more than one color to each vertex of $G$. Then due to the construction of $T$, also $\varphi_2$ evaluates to *true* under $T$.
- We deal with $\varphi_3$ next...

## Solution to Exercise 5

- We know $\mu$ is a proper 3-coloring of $G$, i.e. for each $(a_i, a_j) \in E$, we have $(*)$ $\mu(a_i) \neq \mu(a_j)$. Towards a contradiction, suppose $\varphi_3$ evaluates to *false* under $T$. Then there must exists $(a_i, a_j) \in E$ and and some $m \in \{0, 1, 2\}$ such that the subformula $\neg(c_i^m \wedge c_j^m)$ evaluates to *false* under $T$. Hence, $(c_i^m \wedge c_j^m)$ evaluates to *true* under $T$. It follows that $T(c_i^m) = \mathit{true}$ and $T(c_j^m) = \mathit{true}$. Then due to the construction of $T$, we have $\mu(a_i) = m$ and $\mu(a_j) = m$. This contradicts $(*)$, and thus $\varphi_3$ evaluates to *true* under $T$.

## Exercise 6

Argue that the following problem is solvable in logarithmic space:

**SAME-DIGITS**

INSTANCE: A pair $L_1, L_2$ of lists, where each list contains some digits from $0, \ldots, 9$.

QUESTION: Is the set of digits occurring in $L_1$ equal to the set of digits occurring in $L_2$?

## Solution to Exercise 6

We go through each element $e$ of $L_1$ (1 pointer) and try to find $e$ in $L_2$ (1 pointer). In the second step, we go through each element $e$ of $L_2$ (1 pointer) and try to find $e$ in $L_1$ (1 pointer). We can reuse the pointers from the first step for the second step. Thus in total we need only 2 (constantly many) pointers.

## Exercise 7

Let $L = \{w \in \{0,1\}^* \mid |w| \text{ is even}\}$, i.e. $L$ is the set of all strings $w$ such that (a) $w$ is built using symbols 0 and 1, and (b) $w$ is of even length. Define a Turing machine $M$ that decides $L$, i.e. define a tuple $M = (K, \Sigma, \delta, s)$ such that, for all $w \in \{0,1\}^*$, we have:

- if $w \in L$, then $M(w) = "yes"$;
- if $w \notin L$, then $M(w) = "no"$.

Additionally, provide a high-level description of $M$.

## Solution to Exercise 7

$M = (K, \Sigma, \delta, s)$ with $K = \{s, q\}$, $\Sigma = \{0, 1, \sqcup, \triangleright\}$ and a transition function $\delta$ defined as follows:

| $p \in K$ | $\sigma \in \Sigma$ | $\delta(p, \sigma)$ |
|---|---|---|
| $s$ | $\triangleright$ | $(s, \triangleright, \rightarrow)$ |
| $s$ | $0$ | $(q, 0, \rightarrow)$ |
| $s$ | $1$ | $(q, 1, \rightarrow)$ |
| $s$ | $\sqcup$ | $(\text{“yes”}, \sqcup, -)$ |
| $q$ | $0$ | $(s, 0, \rightarrow)$ |
| $q$ | $1$ | $(s, 1, \rightarrow)$ |
| $q$ | $\sqcup$ | $(\text{“no”}, \sqcup, -)$ |

## Solution to Exercise 7 (continued)

High-level description of $M$:

- In state $s$: If the symbol is $\triangleright$, then move the head to the right without changing the state. If the symbol is 0 or 1, then move the head to the right and change the state to $q$. If the symbol is $\sqcup$, then stop in the state "yes".
- In state $q$: If the symbol is 0 or 1, then move the head to the right and change the state to $s$. If the symbol is $\sqcup$, then stop in the state "no".