VU Formale Methoden der Informatik

Block 1: Computability and Complexity

Bernhard Urban

Matr.Nr.: 0725771

lewurm@gmail.com

May 21, 2011

Exercise 1: *Prove that the **HELLO-WORLD** problem is undecidable by providing a reduction from the **HALTING** problem.*

Let (Π_H, I_H) be an arbitrary instance of the **HALTING** problem, i.e. Π_H is a program that takes one string as input, and I_H is an input for Π_H.
From this, we construct an instance (Π, I) of **HELLO-WORLD** by setting $I = I_H$, and building Π from Π_H as follows:

```
String Π(String I) {
    call Π_H(I);
    return "Hello World";
}
```

It remains to show that the following equivalence holds:

$$\Pi_H \text{ halts on } I_H \Leftrightarrow \Pi \text{ returns “Hello World” on } I.$$

- “$\Rightarrow$” Suppose Π_H halts on I_H. Due to the construction of Π , Π also halts on the input I_H and returns “Hello World” as output.
- “$\Leftarrow$” Suppose Π returns “Hello World” on I. Since running Π on I involves running Π_H on I, we have that Π_H halts on I.

Thus, the undecideability of the **HELLO-WORLD** problem follows from the reduction of the **HALTING** problem.

Exercise 2: *Prove that the **HELLO-WORLD** problem is semi-decidable*

In order to prove that the **HELLO-WORLD** problem is semi-decidable, we construct a procedure Π' which returns `true` for every arbitrary positiv instance (Π, I) of the **HELLO-WORLD** problem. If (Π, I) is a negative instance, then Π' returns `false` or doesn't halt on I.

```
Bool Π'(I)
    return Π(I) == "Hello World";
```

Therefore, the **HELLO-WORLD** problem is semi-decidable.

Exercise 3: *Give a formal proof that the following problems are in* NP.

k-COLORABILITY for any $k \geq 1$

To prove that **k-COLORABILITY** $\in$ NP we have to define a polynomially balanced and polynomially decidable certificate relation for **k-COLORABILITY**. Simply let

$$R = \{(G, C) \mid G \text{ is valid k-colored graph under the color assignment } C\}$$

- R is a certificate relation by construction: G is a positive instance of **k-COLORABILITY** $\Leftrightarrow$ there exisits an color assignment C which makes G a valid graph $\Leftrightarrow$ $(G, C) \in R$.
- R is polynomially balanced because each color assignment C for a graph G can be represented as instance of $V(G)$ with linear size.
- R is polynomially decidable because a procedure to check a graph of the **k-COLORABILITY** problem, just needs to assign the colors to the vertices and compare each vertexcolor with the colors of its neigbours. Such an algorithm has an upper bound of $O(|V| + |E|)$.

NAESAT

$$R = \{(\rho, \nu) \mid \text{formula } \rho \text{ evaluates to true under the assignment } \nu\text{, such that } \textbf{not} \text{ all three literals in each clause of } \nu \text{ have the same truth value}\}$$

- R is a certificate relation by construction: ρ is a positive instance of the **NAESAT** problem $\Leftrightarrow$ there exists an assignment ν which makes ρ evaluate to `true` $\Leftrightarrow$ $(\rho, \mu) \in R$.
- R is polynomially balanced because each assignment ν for ρ can be represented as a subset of variables in ρ.
- R is polynomially decidable because evaluating the Boolean circuit $C(\rho)$ under μ takes only polynomial time and the additional check of the constraint by **NAESAT** is of constant time.

Exercise 4: *Formally prove that* ***HAMILTON-CYCLE*** *is* NP*-complete for directed graphs. You may use the well-known fact that* ***HAMILTON-CYCLE*** *for undirected graphs is* NP*-complete.*

Let an arbitrary instace with undirected **HAMILTON-CYCLE** be given by a graph $G = (V, E)$. Then $G' = (V', E')$ is an instance of the directed **HAMILTON-CYCLE** problem, defined as follows:

$$G' = \{(V', E') \mid V' = V,\ E' = \{(i, j), (j, i) \mid (i, j) \in E\}\}$$

Therefore, our reduction is:

$$G \text{ has a } \textbf{HAMILTON-CYCLE } \Leftrightarrow G' \text{ has a } \textbf{HAMILTON-CYCLE}$$

- "$\Rightarrow$" Suppose that $G = (V, E)$ has a **HAMILTON-CYCLE** C. Then C must be a subset of $E \Rightarrow C \subseteq E$. This must also be true for E', since for every undirected edge a directed one exists in G' by construction.
- "$\Leftarrow$" Suppose that $G' = (V', E')$ has a **HAMILTON-CYCLE** C'. Then C' must be a subset of $E' \Rightarrow C' \subseteq E'$. However, this must be also true for E, since every vertex in C' is visited only *once* ($V = V'$), i.e. there also exists a **HAMILTON-CYCLE** C' in G.

Hence, **HAMILTON-CYCLE** for directed graphs is also NP-complete.

Exercise 5: *Prove the "$\Rightarrow$" direction of the correctness of the reduction, i.e. prove the following statement: if G is 3-colorable, then φ_G is satisfiable.*

We have to show the following:

$$G = (V, E) \text{ is 3-colorable} \Rightarrow \varphi_G \text{ is satisfiable under a truth assignment } \mu$$

- φ_1: Let $i \in G$ be an arbitrary vertex and its corresponding clause $\varphi_1^i = (c_i^0 \lor c_i^0 \lor c_i^2)$. There exists at least one variable c_i^m for i which is set to true under μ and therefore φ_1^i is true for every $i \in G$, i.e. φ_1 evaluates to true.
- φ_2: Let $i \in G$ be an arbitrary vertex and its corresponding clause $\varphi_2^i = (\neg c_i^0 \lor \neg c_i^1) \land (\neg c_i^0 \lor \neg c_i^2) \land (\neg c_i^1 \lor \neg c_i^2)$. We know that only one c_i^m can be true, since i is assigned exactly to one color. Thus φ_2^i evaluates to true for every $i \in G$, therefore φ_2 evaluates to true under μ too.
- φ_3: Let $(a_i, a_j) \in E$ and its corresponding clause in $\varphi_3^{(a_i,a_j)} = (\neg c_i^0 \lor \neg c_j^0) \land (\neg c_i^1 \lor \neg c_j^1) \land (\neg c_i^2 \lor \neg c_j^2)$. We know that one c_x^m for each $x \in \{i, j\}$ must be true, while the following condition holds: $\forall i, j : c_i^m \neq c_j^m \mid m \in \{0, 1, 2\}$
  Therefore, $\varphi_3^{a_i,a_j}$ evaluates to true, i.e. φ_3 evaluates to true too.

Thus, G is 3-colorable then φ_G is satisfiable, since $\varphi_G = \varphi_1 \land \varphi_2 \land \varphi_3$,

Exercise 6: *Prove the "⇐" direction of the correctness of the reduction in Exercise 5, i.e. prove the following statement: if φ_G is satisfiable, then G is 3-colorable.*

We have to show the following:

$$\varphi_G \text{ is satisfiable under a truth assignment } \mu \Rightarrow G = (V, E) \text{ is 3-colorable}$$

It remains to show that if φ_G is satisfiable:

- Suppose that φ_1 evaluates to true, then at least one c_i^m, for every Vertex $i \in V$, has to be true, i.e. every vertex has at least one color.
- Suppose that φ_2 evaluates to true, then two out of three c_i^m, for every Vertex $i \in V$, has to be false, i.e. every vertex has *exactly one* color.
- Suppose that φ_3 evaluates to true, then $c_i^m \neq c_j^m$ holds for every $(i, j) \in V$, i.e. every vertices i and j connected to each other have a different color.

Exercise 7: *Provide a reduction from* ***FAIR-3-COLORING*** *to* ***SAT***.

We have already proven that the reduction from **3-COLORING** to **3-SAT**. In order to provide a reduction from **FAIR-3-COLORING** we add μ_4 to the conditions, which ensures that each color has been assigned at least once to a vertex. μ_4 is defined as follows:

$$\mu_F = \mu_G \ \wedge \ \mu_4 \qquad \mu_4 = \bigwedge_{0 \le k \le 2} \ \bigvee_{1 \le i \le n} c_i^k$$

It remains to show that:

$$G \text{ is fair-3-colorable } \Leftrightarrow \varphi_F \text{ is satisfiable under a truth assignment } \mu$$

- "⇒": φ_G has already been proven (a fair-3-color assignment is a 3-color assignment too)
- "⇒": There exists by construction at least one vertex with each color. Therefore, one literal c_i^k in each clause must be true, i.e. φ_4 evaluates to true under μ too.
- "⇐": φ_G has already been proven.
- "⇐": Suppose that φ_4 evaluates to true, then at least one literal must be true for $0 \le m \le 2$ in μ. Therefore, each color is assigned at least once to a vertex in G.

Exercise 8: *Give a problem reduction from* ***VERTEX COVER*** *to* ***CLIQUE****. Prove the correctness of the reduction.*

From the lecture we know that for an arbitrary undirected Graph $G = (V, E)$ with $I \subseteq V$ and a complement graph $\overline{G} = (V, \overline{E})$, i.e. $[i, j] \in E \Leftrightarrow [i, j] \notin \overline{E}$:

$$I \text{ is an } \textbf{INDEPENDENT SET} \text{ in } G \Leftrightarrow I \text{ is a } \textbf{CLIQUE} \text{ in } \overline{G}$$

holds. Furthermore we could say for the same graph G and subgraph I:

$$I \text{ is an } \textbf{INDEPENDENT SET} \text{ in } G \Leftrightarrow N = V \setminus I \text{ is a } \textbf{VERTEX COVER} \text{ in } G$$

Ultimately, we can reduce **VERTEX COVER** to **CLIQUE** via **INDEPENDENT SET** and reduce it as follows:

$$N = V \setminus I \text{ is a } \textbf{VERTEX COVER} \text{ in } G \Leftrightarrow I \text{ is a } \textbf{CLIQUE} \text{ in } \overline{G}$$

Exercise 9: *Provide a logarithmic space algorithm for solving* ***SELECT-3RD****. Argue why it uses only logarithmic space.*

```
Bool SELECT-3RD(n, L) {
    Bool foundN = False;
    Int i = 0;
    ListElem p = L.first();

    while ((i < 3) && (L.last() != p)) {
        if (p.value() == n) {
            foundN = True;
        } else if (p.value() > n) {
            i++;
        }
        p = p.next();
    }
    return (foundN && (i == 2));
}
```

The algorithm uses a reference, a counter and a flag to compute the result for any given List, i.e. for any given instance of the **SELECT-3RD** problem, therefore

$$\log |L| + d$$

while d is the space used by the additional variables (d is constant). Thus, the algorithm uses logarithmic space.

Exercise 10: *Let* $L = \{w \in \{0,1\}^* | \ \ |w| \text{ is even}\}$*, 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*. Additionally, provide a high-level description of* M*.*

Let $M = (K, \Sigma, \delta, \textsf{even})$ with $K = \{\textsf{even}, \textsf{odd}\}$, $\Sigma = \{0, 1, \sqcup, \rhd\}$ and a transition function δ defined as follows in the table. The Turing machine M consists of two states:

even By reaching the end of the input string $\sqcup$ in this state, we output "Yes". Otherwise, if we get a "0" or "1" we make a transition to odd and move the pointer to the next element of the input string. Also, this is the initial state, because an empty string is by definition even.

odd Quite similar as even, we output "No" on $\sqcup$ and otherwise (i.e. on "0" and "1") we make a transition to even and move the pointer to the next element of the input string.

$p \in k$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
even	$\rhd$	$(\textsf{even}, \rhd, \rightarrow)$
even	$\sqcup$	$(\text{"Yes"}, \sqcup, -)$
even	0	$(\textsf{odd}, 0, \rightarrow)$
even	1	$(\textsf{odd}, 1, \rightarrow)$
odd	$\sqcup$	$(\text{"No"}, \sqcup, -)$
odd	0	$(\textsf{even}, 0, \rightarrow)$
odd	1	$(\textsf{even}, 1, \rightarrow)$