Formale Methoden der Informatik

Block 1: Computability and Complexity

Exercises (Sample Solutions)

Mantas Šimkus

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

SS 2013

Exercise 1

By providing a reduction from the **HALTING** problem to **REACHABLE-CODE**, prove that **REACHABLE-CODE** is undecidable.

Solution to Exercise 1

The reduction is defined as follows. Let (Π, I) be an arbitrary instance of **HALTING**. We build an instance (Π', n) of **REACHABLE-CODE** as follows. We let Π' be defined as

```
String Π'(String S)
Π(I); // Π and I are hardcoded, S is ignored
return 0;
```

We let n be the line number of "**return** 0;" in Π'.

In other words, for an instance $x = (\Pi, I)$, the instance $R(x)$ resulting from the reduction is (Π', n). To prove the correctness of the reduction we have to show:

(Π, I) is a positive instance of **HALTING** $\Leftrightarrow$ (Π', n) is a positive instance of **REACHABLE-CODE**.

Solution to Exercise 1 (continued)

"$\Rightarrow$" Assume (Π, I) is a positive instance of **HALTING**, i.e. Π terminates on I. Then the call $\Pi(I)$ in program Π' terminates on any input S to Π'. Thus the statment "**return** 0;" is reached on any input to Π'. Hence, (Π', n) is a positive instance of **REACHABLE-CODE**.

"$\Leftarrow$" Assume (Π', n) is a positive instance of **REACHABLE-CODE**, i.e. Π' has an input S on which it reaches the line number n. Since the code of line n comes *after* the call $\Pi(I)$, it must be the case that Π terminates on I, i.e. (Π, I) is a positive instance of **HALTING**.

Solution to Exercise 1: Why does it work?

Why does the reduction R prove the undecidability of **REACHABLE-CODE**?

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

Build a procedure Π_h, which takes instances of **HALTING**, as follows:

Bool Π_h(**String** Π,**String** I)
return $\Pi_{rc}(R((\Pi, I)))$;

It is easy to see that Π_h is a decision procedure for **HALTING**:

- $\Pi_h(\Pi, I)$ returns *true* if Π terminates on I
- $\Pi_h(\Pi, I)$ returns *false* if Π 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 (Π, I), where Π is a program that takes one string as input and returns a string, and I is a string.

  In a proof:

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

- INSTANCE: A program Π that takes one string as input and returns a string.

  In a proof:

  - Π, Π', Π_1, Π_2, are O.K.
  - (Π, I), (Π', I'), (Π, I, I'), (Π, I, k) are not O.K.

Exercise 2

By providing a semi-decision procedure, prove that **CORRECTNESS** is semi-decidable.

Solution to Exercise 2

We can write an interpreter Π_{int} that takes as input Π and I_1, I_2, i.e. an instance of **CORRECTNESS**, and simulates the run of Π on I_1:

- If the simulation reaches the point where the string I_2 is output, then Π_{int} returns *true*.
- If the simulation ends with an output I' such that $I' \neq I_2$, then Π_{int} returns *false*.

Solution to Exercise 2 (continued)

It can be seen as follows that such an interpreter Π_{int} is a semi-decision procedure for **CORRECTNESS**. We distinguish the following cases:

- Case 1. Suppose that (Π, I_1, I_2) is a positive instance, i.e., Π outputs I_2 on input I_1. Then the simulation in Π_{int} will encounter the output I_2 and return *true* by the construction of Π_{int}.
- Case 2.1. Suppose that (Π, I_1, I_2) is a negative instance and that Π halts on input I_1. Then Π halts with an output $I' \neq I_2$. Hence, the simulation in Π_{int} will detect that the output I' is not equal to I_2. Thus, Π_{int} returns *false* by the construction of Π_{int}.
- Case 2.2. Suppose that (Π, I_1, I_2) is a negative instance and that Π does not halt on input I_1. Then the simulation of this computation of Π on I_1 by the interpreter Π_{int} will not terminate either. Hence, Π_{int} will run forever on the negative instance (Π, I_1, I_2), which is a correct behavior for a semi-decision procedure.

Exercise 3

By providing a reduction from **CORRECTNESS** to **HALTING**, prove that **CORRECTNESS** is semi-decidable.

Solution to Exercise 3

The reduction is defined as follows. Let (Π, I_1, I_2) be an arbitrary instance of **CORRECTNESS**. We build an instance (Π', I') of **HALTING** by setting $I' = I_1$ and constructing Π' as follows:

String Π' (**String** S)
$OUT = \Pi(S)$; // Π is hardcoded in Π'
if $OUT = I_2$ **then return** 0
else while True do $\{\}$

To prove the correctness of the reduction we have to show:

(Π, I_1, I_2) is a positive instance of **CORRECTNESS** $\Leftrightarrow$ (Π', I') is a positive instance of **HALTING**.

Solution to Exercise 3 (continued)

"$\Rightarrow$" Assume (Π, I_1, I_2) is a positive instance of **CORRECTNESS**, i.e. Π returns I_2 on input I_1. Then $OUT = I_2$ when I_1 is input to Π'. Then Π' terminates with output 0 on input I_1. Hence (Π', I') is a positive instance of **HALTING**.

"$\Leftarrow$" Assume (Π', I') is a positive instance of **HALTING**, i.e. Π' terminates on I'. Then the call $\Pi(S)$ in program Π' terminates on $S = I'$. This means that the "if" statement is reached by Π' on input I'. Since Π' terminates on I', it must be the case that $OUT = I_2$. Hence, we have the fact that Π returns I_2 on input I', where $I' = I_1$ by problem reduction, i.e. (Π', I_1, I_2) is a positive instance of **CORRECTNESS**.

Exercise 4

Prove that the following problem is undecidable:

> **ALL-FALSE**
>
> INSTANCE: A program Π that takes as input a natural number and returns *true* or *false*. It is guaranteed that Π terminates on any input.
>
> QUESTION: $\Pi(k) = \mathit{false}$ for all natural numbers k?

Hint: For your proof you may assume the availability of an interpreter for instances of **HALTING**. In particular, you have available a decision procedure Π_{int} that does the following:

1. Π_{int} takes as input a program Π, a string I, and a natural number n.
2. Π_{int} emulates the first n steps of the run of Π on I. If Π terminates on I within n steps, then Π_{int} returns *true*. Otherwise, Π_{int} returns *false*.

Solution to Exercise 4

We provide a reduction from **co-HALTING**, which is known to be undecidable. Let (Π, I) be an arbitrary instance of **co-HALTING**. We build an instance Π' of **ALL-FALSE** by constructing Π' as follows:

String Π' (**Int** n)
return $\Pi_{int}(\Pi, I, n)$ // Π and I are 'hard-coded' in Π'

To prove the correctness of the reduction we have to show:

(Π, I) is a positive instance of **co-HALTING** $\Leftrightarrow$ Π' is a positive instance of **ALL-FALSE**.

Solution to Exercise 4 (continued)

"$\Rightarrow$" Assume (Π, I) is a positive instance of **co-HALTING**, i.e. Π does not terminate on I. In particular, for any n, Π does not terminate on I within n steps. Hence, for any n, $\Pi_{int}(\Pi, I, n) = \mathit{false}$ by definition of Π_{int} and $\Pi'(n) = \mathit{false}$ definition of Π'. That is, $\Pi'(n) = \mathit{false}$ for any natural number n. Thus Π' is a positive instance of **ALL-FALSE**.

"$\Leftarrow$" Assume Π' is a positive instance of **ALL-FALSE**, i.e. $\Pi'(n) = \mathit{false}$ for all natural numbers n. By definition of Π', $\Pi_{int}(\Pi, I, n) = \mathit{false}$ for all n. That is, there is no number n such that $\Pi_{int}(\Pi, I, n) = \mathit{true}$, i.e. such that Π terminates on I within n steps. Thus (Π, I) is a positive instance of **co-HALTING**.

Problem

Suppose you have n processes, where some processes may need to communicate with each other. Suppose you also have m computers, where some of them are connected by a (fast) direct network connection. Each computer has a limit on the number of processes it can run. Your problem is to assign processes to computers so that the limits are obeyed and all the processes that need to communicate can communicate. This can be formalized as follows:

ASSIGNMENT

INSTANCE: A pair $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of undirected graphs, and a function *limit* that assigns to each $v \in V_2$ an integer. It is assumed that G_2 is reflexive, i.e. for every $v \in V_2$, $[v, v] \in E_2$.

QUESTION: Does there exist an assignment μ that assigns to each vertex in V_1 a vertex in V_2 such that:

(A) if $[v, v'] \in E_1$, then also $[\mu(v), \mu(v')] \in E_2$, and

(B) for every vertex v in V_2, no more than $\mathit{limit}(v)$ nodes of V_1 are assigned to v.

Exercise 5

Give a proof that **ASSIGNMENT** is in NP, i.e. define a certificate relation and briefly discuss that it is polynomially balanced and polynomial-time decidable.

Solution to Exercise 5

Define the relation

$$R = \{\langle (G_1, G_2, \mathit{limit}), \mu \rangle \mid \mu \text{ is a correct assignment from } V_1 \text{ to } V_2 \},$$

where "correct" means that it satisfies the conditions (A) and (B).

Clearly, R is a certificate relation for **ASSIGNMENT**, since the following equivalences hold: $(G_1, G_2, \mathit{limit})$ is a positive instance of **ASSIGNMENT** $\Leftrightarrow$ there exists an assignment μ that it satisfies the conditions (A) and (B) $\Leftrightarrow \langle (G_1, G_2, \mathit{limit}), \mu \rangle \in R$.

R is polynomially balanced because any assignment μ can be represented in space that is linear in the size of G_1 and G_2. E.g. by a list of vertex pairs of length $\leq |V_1|$.

Finally R is decidable in polynomial time because, given $(G_1, G_2, \mathit{limit})$ and a candidate assignment μ one can check in polynomial time w.r.t. the size of $(G_1, G_2, \mathit{limit})$ and μ whether μ satisfies the conditions (A) and (B).

Exercise 6

Define a polynomial-time reduction from **CLIQUE** to **ASSIGNMENT**.

Note: the result of Exercise 5 together with the reduction show that **ASSIGNMENT** is NP-complete.

Solution to Exercise 6

Assume an instance (G, k) of **CLIQUE**. We next define an instance $(G_1, G_2, \mathit{limit})$ of **ASSIGNMENT** as follows.

1. We let G_1 be a graph with $V_1 = \{1, \ldots, k\}$ and such that there is an edge between every pair of distinct vertices in V_1. I.e. G_1 is a complete graph with k vertices.
2. We obtain G_2 by adding self-loops to G. That is $V_2 = V$ and there is an edge $[v, v'] \in E_2$ iff $[v, v'] \in E$ or $v = v'$. This makes sure that G_2 is reflexive.
3. We let $\mathit{limit}(v) = 1$ for every $v \in V_2$.

It is not difficult to see that:

G has a clique of size $\geq k \Leftrightarrow$ there exists an assignment μ from vertices in G_1 to G_2 that satisfies the conditions (A) and (B).

Solution to Exercise 6 (Cnt'd)

($\Rightarrow$) Suppose G has a clique of size $\geq k$. Then G also has a clique $C = \{v_1, \ldots, v_k\}$ of size $= k$. Define an assignment $\mu : V_1 \to V_2$ as follows:

$$\mu(i) = v_i, \quad i \in \{1, \ldots, k\}.$$

We check that μ satisfies (A) and (B):

(A): Take any pair $[i, j] \in E_1$. Since C is a clique in G, we know that C is also a clique in G_2. Since $\mu(i), \mu(j) \in C$ by the construction of μ, we get $[\mu(i), \mu(j)] \in E_2$.

(B): For every vertex v of V_2 there are only two options:

- No vertex is assigned to v.
- $v = v_i$ for some $v_i \in C$. In this case, only the vertex i assigned to v.

Solution to Exercise 6 (Cnt'd)

($\Leftarrow$) Suppose there exists an assignment μ from vertices in G_1 to G_2 that satisfies the conditions (A) and (B).

Take the following set:

$$C = \{v \in V_2 \mid \exists i \in V_1 : \mu(i) = v\}.$$

In other words, C is the range of the function μ. Clearly, due to the selection of *limit*, we get $|C| = k$.

It remains to see that C is a clique in G. Take any pair $v, v' \in C$ with $v \neq v'$. Then there exist $i \neq i' \in V_1$ s.t. $\mu(i) = v$ and $\mu(i') = v'$. By the construction of E_1, we know $[i, i'] \in E_1$. Since μ satisfies (A), we get $[v, v'] \in E_2$. Since $v \neq v'$, we must have $[v, v'] \in G$ by the construction of G_2 from G.

Exercise 7

We consider a polynomial-time reduction from **INDEPENDENT SET** to **SAT**. Let an arbitrary instance of **INDEPENDENT SET** be given by the undirected graph $G = (V, E)$ and integer k. Let V be of the form $V = \{b_1, \ldots, b_m\}$. We construct a propositional formula $\varphi_{G,k}$ (i.e. an instance of **SAT**) as follows. First of all, we use the following propositional variables:

- M_{i,b_j} for each $1 \leq i \leq k$ and $1 \leq j \leq m$ (intended meaning: M_{i,b_j} is set to *true* in a model of $\varphi_{G,k}$ if and only if the number i is assigned to the node b_j of G).

Exercise 7 (continued)

Then the formula $\varphi_{G,k}$ is defined as $\varphi_{G,k} = \alpha_1 \wedge \alpha_2 \wedge \alpha_3$, where

$$\alpha_1 = \bigwedge_{1 \leq i \leq k} \Big(\bigvee_{1 \leq j \leq m} M_{i,b_j} \Big)$$

$$\alpha_2 = \bigwedge_{(1 \leq n \leq m) \wedge (1 \leq i,j \leq k) \wedge (i \neq j)} \neg (M_{i,b_n} \wedge M_{j,b_n})$$

$$\alpha_3 = \bigwedge_{[v_1,v_2] \in E} \bigwedge_{1 \leq i,j \leq k} \neg (M_{i,v_1} \wedge M_{j,v_2})$$

Exercise 7 (continued)

Informal explanation of the reduction. Intuitively (!), the formulae $\alpha_1, \alpha_2, \alpha_3$ can be explained as follows.

- The formula α_1 expresses the condition that each number $i \in \{1, \ldots k, \}$ must be assigned to at least one node from G.
- The formula α_2 makes sure that each number $i \in \{1, \ldots, k\}$ is assigned to a separate node in G. Thus α_1 together with α_2 make sure that at least k nodes from G have numbers "assigned".
- The formula α_3 ensures that for every edge $[v_1, v_2]$ of G we have not assigned numbers to both v_1 and v_2.

Remark. All the above comments are *explanations* of the intuition of the problem reduction. They are *not proofs*!! When you are requested to prove the correctness of the problem reduction, you are not allowed to refer to these explanations. *Your proofs have to be self-contained*!

Exercise 7 (continued)

Prove formally the "⇒" direction of the correctness of the reduction, i.e. prove the following statement: if G has an independent set I of size $\geq k$, then there exists a truth assignment T that makes $\varphi_{G,k}$ evaluate to *true*.

Solution to Exercise 7

Suppose G has an independent set $I' = \{b_{j_1}, \ldots b_{j_l}\}$ of size $\geq k$. Then, in particular, G has an independent set $I = \{b_{j_1}, \ldots, b_{j_k}\}$ of size $= k$. We define an assignment T that makes $\varphi_{G,k}$ evaluate to **true**, and thus shows the satisfiability of $\varphi_{G,k}$. T is defined by setting the following variables to **true**:

- $M_{i,b_{j_i}}$ for all $i \in \{1, \ldots, k\}$.

The remaining variables are set to **false**. To see that $\varphi_{G,k}$ evaluates to **true**, we have to show that each of the subformulas $\alpha_1, \alpha_2, \alpha_3$ of $\varphi_{G,k}$ evaluates to **true**:

Solution to Exercise 7 (continued)

- α_1: By construction, the formula α_1 evaluates to *true* if for every $i \in \{1, \ldots, k\}$, there exists $1 \leq j \leq m$ such that M_{i,b_j} is set to **true** in T. This is indeed true: for every $i \in \{1, \ldots, k\}$, we have that j_i is such that $M_{i,b_{j_i}}$ is set to **true** in T, by the definition of T.
- α_2: For the formula α_2 to evaluate to **true** in T there should not exist distinct integers $1 \leq c \neq d \leq k$ and a vertex $b_n \in I$ such that M_{c,b_n} and M_{d,b_n} are set to **true** in T. Suppose towards a contradiction that such c, d, b_n exist. Then by the definition of T, $b_{j_c} = b_{j_d} = b_n$. This contradicts the assumption that I of size k.
- α_3: Consider arbitrary indices i_1, i_2, j_1, j_2 with $(1 \leq i_1, i_2 \leq k)$ and $(1 \leq j_1, j_2 \leq m)$ such that $[b_{j_1}, b_{j_2}] \in E$. We have to show that $\neg M_{i_1,b_{j_1}} \vee \neg M_{i_2,b_{j_2}}$ is **true** in T. Suppose it is not the case, i.e. $M_{i_1,b_{j_1}} \wedge M_{i_2,b_{j_2}}$ is **true** in T. Then by the construction of T, $\{b_{j_1}, b_{j_2}\}] \subseteq I$. This contradicts the assumption that I is an independent set in G.

Exercise 8

Prove the "$\Leftarrow$" direction of the correctness of the reduction in Exercise 7, i.e. prove the following statement: if $\varphi_{G,k}$ is satisfiable, then there exists some independent set I in G of size $\geq k$.

Solution to Exercise 8

Suppose $\varphi_{G,k}$ is satisfiable, i.e. there is a truth assignment T under which all subformulae $\alpha_1, \alpha_2, \alpha_3$ evaluate to **true**. We have to show that G has an independent set of size $\geq k$. We construct a set I of vertices as follows:

$$I = \{b_j \mid \exists i : T(M_{i,b_j}) = \textbf{true}\}.$$

It remains to show that I is an independent set in G of size $\geq k$. Since α_1 and α_2 evaluate to **true** in T, we have that $|I| \geq k$.

To see that I is an independent set, assume an arbitrary pair $v_1, v_2 \in I$. We must show that G has no edge $[v_1, v_2]$.

Towards a contradiction, assume $[v_1, v_2] \in E$. First note, by construction of I, there are i, j such that M_{i,v_1} and M_{j,v_2} are **true** in T, i.e. such that $(M_{i,v_1} \wedge M_{j,v_2})$ is **true** in T. Since T makes α_3 evaluate to **true**, we have have that $\bigwedge_{1 \leq i,j \leq k} \neg(M_{i,v_1} \wedge M_{j,v_2})$ evaluates to **true**. That is, $\neg(M_{i,v_1} \wedge M_{j,v_2})$ evaluates to **true** for any $1 \leq i, j \leq k$. Contradiction.

Exercise 9

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 9

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 10

Let $L = \{w \in \{1\}^* \mid w \text{ has length } 3k \text{ for some integer } k \geq 0\}$, i.e. L is the set of all strings w such that (a) w is built using the symbol 1, and (b) the length of w is a multiple of 3. Define a Turing machine M that decides L, i.e. define a tuple $M = (K, \Sigma, \delta, s)$ such that, for all $w \in \{1\}^*$, we have:

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

Additionally, provide a high-level description of M.

Solution to Exercise 10

$M = (K, \Sigma, \delta, s_0)$ with $K = \{s_0, s_1, s_2\}$, $\Sigma = \{1, \sqcup, \triangleright\}$ and a transition function δ defined as follows:

$p \in K$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
s_0	$\triangleright$	$(s_0, \triangleright, \rightarrow)$
s_0	1	$(s_1, 1, \rightarrow)$
s_0	$\sqcup$	$(\text{“yes”}, \sqcup, -)$
s_1	1	$(s_2, 1, \rightarrow)$
s_1	$\sqcup$	$(\text{“no”}, \sqcup, -)$
s_2	1	$(s_0, 1, \rightarrow)$
s_2	$\sqcup$	$(\text{“no”}, \sqcup, -)$

(note: $\delta(s_1, \triangleright)$ and $\delta(s_2, \triangleright)$ can be arbitrary)

Solution to Exercise 10 (continued)

High-level description of M: The machine reads the input from left to right. Whenever it reads the symbol 1, it switches the state from s_0 to s_1, from s_1 to s_2, or from s_2 to s_0. It rejects the input if it reads $\sqcup$ not being in the state s_0.