Formale Methoden der Informatik
Block 1: Computability and Complexity

Exercises 1-10
Submission deadline: 21.04.2014

SS 2014

Exercise 1 *Prove that the following problem is undecidable:*

> ***DOUBLE***
>
> *INSTANCE: A pair (Π, I), where I is a string and Π is a program that takes one string as input and outputs a string.*
> *QUESTION: Does the program Π on the string I as input return as output the string $I + I$? Here $+$ is the operator for string concatenation.*

*Prove the undecidability by providing a reduction from the **HALTING** problem to **DOUBLE**, and arguing that your reduction is correct.*

Solution. The proof proceeds by reduction from HALTING to DOUBLE.
Let(Π, I) be an arbitrary instance of the **HALTING** problem, i.e., Π is a program that takes one string and I is an Input for Π. From this, we construct an instance (Π', I) of **DOUBLE** by building (Π') as follows:

Listing 1: Π'

```
1 String Π' (String S) {
2   call Π(S);
3   return S+S;
4 }
```

It remains to show that the following equivalence holds:
Π halts on I $\Leftrightarrow$ Π' returns I+I
$\Rightarrow$
Assume Π halts on I. This means line three is reached and I+I is returned.
$\Leftarrow$
Assume Π' returns I+I. This implies, that the program must have passed line two. This implies that Π halts on I because it means running Π on I and returning from it. $\square$

Exercise 2 *Prove that **DOUBLE** is semi-decidable. To this end, provide a semi-decision procedure and justify your solution.*

Solution. We can write an interpreter program Π_i such that:

- Π_i takes as input a source code Π and an Input I for Π
- Π_i parses Π and simulates a run of Π on I.
- If the simulation of Π on I reaches the end and Π returns I+I, Π_i returns true
- If the simulation of Π on I reaches the end and Π doesn't return I+I, Π_i returns false
- If the simulation of Π on I doesn't reach the end, then, clearly Π_i cannot return any value

Exercise 3 *Give a formal proof that **SUBSET SUM** is in* NP*, i.e. define a certificate relation and discuss that it is polynomially balanced and polynomial-time decidable.*

SUBSET SUM:
INSTANCE: A finite set of integer numbers $S = \{a_1, a_2, \ldots, a_n\}$ and an integer number t.
QUESTION: Does there exist a subset $S' \subseteq S$, s.t. the sum of the elements in S' is equal to t, i.e. $\left(\sum_{a_i \in S'} a_i\right) = t$?

Solution. As proof I provide a certificate relation R and discuss why it is polynomially balanced and polynomial-time decidable:

$$R = \{((S,t), S') \mid S' \subseteq S \land \left(\sum_{a_i \in S'} a_i\right) = t\}$$

- R is certificate relation by construction: (S,t) is a positive instance of **SUBSET SUM** $\leftrightarrow$ There is a subset of S, S' for which if all the integers in the set are added provide a sum which is equal to t. $\leftrightarrow ((S,t), S') \in R$
- R is polynomially balanced because S' is a subset of S.
- R is polynomial-time decidable because adding numbers in set S' and then comparing the sum to d is in O($|S'| + 1$).

Exercise 4 *Formally prove that **PARTITION** is* NP*-complete. For this you may use the fact that **SUBSET SUM** is* NP*-complete.*

PARTITION:
INSTANCE: A finite set of n positive integers $P = \{p_1, p_2, \ldots, p_n\}$.
QUESTION: Can the set P be partitioned into two subsets P_1, P_2 such that the sum of the numbers in P_1 equals the sum of the numbers in P_2?

Solution. To formally prove that **PARTITION** is NP-complete we need to first show that it is in NP. Therefore we need to provide a certificate relation R and discuss why it is polynomially balanced and polynomial-time decidable:
$R = \{(P, P_1)) \mid P_1 \subseteq P \land \left(\sum_{a_i \in P_1} a_i\right) = \left(\sum_{a_i \in P \setminus P_1} a_i\right)\}$

- R is certificate relation by construction: (P,P_1) is a positive instance of **PARTITION** $\leftrightarrow$ There is a subset of P, P_1 for which if you partion P with P_1 you get a second subset P_2. If the integers in both sets are added up (seperately) they are equal. $\leftrightarrow ((P, P_1) \in R$

- R is polynomially balanced because P_1 is a subset of P.
- R is polynomial-time decidable because adding $|R|$ integers and then comparing two numbers is in P.

We proceed our proof by reducing **SUBSET SUM** to **PARTITION** in order to show hardness. Let an arbitrary instance of **SUBSET SUM** be given by a Set S and an integer t.
Then we construct the following instance of **PARTITION**:
b = $\left(\sum_{a_i \in S} a_i\right)$
$P = S \cup b + t \cup 2b - t$
To prove the correctness of the reduction, we must show that (S,t) has a subset S' where the sum of the elements equal to t $\leftrightarrow$ the resulting instance P of **PARTITION** is positive.
$\rightarrow$
We need to show that under the assumption of a positive instance (S,t) of **SUBSET SUM** the resulting instance P of **PARTITION**is positive:
Assume (S,t) has a subset S' where the sum of the elements equal to t. By our reduction $P = S \cup b + t \cup 2b - t$
Let $P_1 = S' \cup (2b - t)$ this implies that $P_2 = S \setminus S' \cup (b + t)$ by the definition of **PARTITION**. As t = $\left(\sum_{a_i \in S'} a_i\right)$ and $t + (2b - t) = 2b$ the sum of the elements in P_1 is equal to 2b. The sum of the elements in $S \setminus S'$ is equal to $b - t$ and $(b - t) + (b + t) = 2b$. Thus the sum of the elements in P_1 and P_2 are equal and we have a positive resulting instance P of **SUBSET SUM**.
$\leftarrow$
Suppose that P is a positive instance of **PARTITION**. Now we need to show, that (S,t) is a positive instance of **SUBSET SUM**. Because of the reduction $P = S \cup b + t \cup 2b - t$ and we can assume that there exist two sets P_1 and P_2 which partition P. The only way to partition P into two sets is to exactly split it into two sets where the sum of the elements add up to 2b, because the total sum of the elements in P is 4b. The only way to do this is to take some elements out of S which when summed up equal to the value t and add the element $(2b - t)$. (The other set is formed by $S \setminus S' \cup (b + t)$). Because we assumed that our set can be partioned we can be sure that there are elements of S which when summed up equal to the value t. Let the elements of S which when summed up equal to the value t form the Set S'. This is the definition of a positive instance of **SUBSET SUM**. $\square$

Exercise 5 *Provide a reduction from **VERTEX COVER** to **SET COVER**. Additionally, prove the "$\Rightarrow$" direction in the proof of correctness of the reduction, i.e. prove the following statement: if a **VERTEX COVER** instance is a yes instance then the created **SET COVER** instance is also a yes instance.*

VERTEX COVER:
INSTANCE: An undirected graph $G = (V, E)$ and integer k.
QUESTION: Does there exist a vertex cover N of size $\leq k$? i.e. $N \subseteq V$, s.t. for all $[i, j] \in E$, either $i \in N$ or $j \in N$?

SET COVER:
INSTANCE: A finite set X of elements, a collection of n subsets $S_i \subseteq X$, such that every element of X belongs to at least one subset S_i, and an integer m.
QUESTION: Does there exist a collection C of at most m of these subsets, such that the members of C cover all elements of X? i.e. $\bigcup_{S \in C} S = X$.

Example: The following **Set Cover** *instance: $X = \{1, 2, 3, 4, 5\}, S_1 = \{1, 2, 3\}, S_2 = \{3, 4\}$, $S_3 = \{1, 2, 5\}, S_4 = \{4, 5\}$ and $m = 2$, is a yes instance, because there exists a collection C with two subsets that cover all elements of X: $C = \{S_1, S_4\}$.*

Exercise 6 *Provide a reduction from a simplified **EMPLOYEE SCHEDULING** problem to **SAT**.*

EMPLOYEE SCHEDULING:
INSTANCE:

- *Number of employees: n.*
- *Set $A = \{D, A, N, -\}$ where $D =$ "day shift", $A =$ "afternoon shift", $N =$ "night shift", and $- =$ "day-off".*
- *w: length of schedule. Suppose that $w = 7$.*
- *Temporal requirements: The requirement matrix R (3×7), where each element $r_{i,j}$ of the requirement matrix R shows the required number of employees for shift i during day j.*
- *Constraints:*
  - *Sequences of shifts not permitted to be assigned to employees. Suppose that these sequences are not permitted: $(N\ D)$, $(A\ D)$, and $(N\ A)$. For example, the sequence $(N\ D)$ (Night Day) indicates that after working in the night shift, it is not allowed to work the next day in the day shift.*
  - *Maximum and minimum length of blocks of workdays: $MAXW$, $MINW$. A Work block is a sequence consisting only from the working shifts (without day-off in beetwen). Suppose that $MAXW = 5$ and $MINW = 2$.*

QUESTION: Find a schedule (assignment of shifts to employees) that satisfies the requirement matrix, and the two given constraints.

A schedule is represented by an $n \times w$ matrix $S \in A^{nw}$. Each element $s_{i,j}$ of matrix S corresponds to one shift. Element $s_{i,j}$ shows on which shift employee i works during day j or whether the employee has day-off.

EXAMPLE: Suppose that we are given an instance of ***EMPLOYEE SCHEDULING*** *with $n = 9$ and the following requirement matrix:*

$$R_{3,7} = \begin{pmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{pmatrix}$$

In Table 1 an employee schedule is presented that satisfies the requirement matrix, and the two given constraints in the problem definition. This schedule describes explicitly the working schedule of 9 employees during one week. The first employee works from Monday until Friday in a day shift (D) and during Saturday and Sunday has days-off. The second employee has days-off on Monday and Tuesday and works in a day shift during the rest of the week. Further, the last employee works from Monday until Wednesday in a night shift (N), on Thursday and Friday has days-off, and on Saturday and Sunday works in the day shift. Each row of this table represents the weekly schedule of one employee.

Table 1: A typical week schedule for 9 employees

Emp./day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	D	D	D	D	D	-	-
2	-	-	D	D	D	D	D
3	-	-	-	N	N	N	N
4	D	D	-	-	A	A	A
5	A	A	A	A	-	-	-
6	N	N	N	N	N	-	-
7	-	-	A	A	A	A	A
8	A	A	-	-	-	N	N
9	N	N	N	-	-	D	D

Exercise 7 *Give a proof sketch of the correctness of your reduction in the previous exercise. Does this reduction imply that* ***EMPLOYEE SCHEDULING*** *is an* NP*-complete problem? Argue your answer.*

Exercise 8 *Fomally prove that logical entailment in propositional logic is co−*NP*-complete.*

ENTAILMENT ($\models$):
INSTANCE: Propositional logic sentences α and β.
QUESTION: Does the sentence α entails the sentence β ($\alpha \models \beta$)?

$\alpha \models \beta$ if and only if, in every truth assignment in which α is true, β is also true.

Exercise 9 *Consider the following problem:*

SELECT-3RD
INSTANCE: An integer n, and a list $L = (n_1, n_2, \ldots, n_m)$ of integers, where no integer occurs twice in L (i.e. L represents a set). *QUESTION: Is n the third biggest element in L?*

*Provide a logarithmic space algorithm for solving **SELECT-3RD** (use pseudo-code notation). Argue why it uses only logarithmic space.*

Hint. Each element in L can be addressed using a pointer. Each pointer requires only logarithmic space.

Exercise 10 *Design a Turing machine that increments by one a positive value represented by a string of 0s and 1s.*