Prof. Dr. Titus Dose
Fakultät für Informatik und Mathematik
OTH Regensburg

Theoretische Informatik
Wintersemester 2024/2025
06.11.2024

Übungsblatt 5

(Besprechung am 14.11.2024)

1. **General halting problem**
   Prove that the general halting problem K is undecidable.

2. **Decidability**
   Let $f : \mathbb{N} \to \mathbb{N}$ be an arbitrary function. Prove the decidability of the following set.

$$A = \{m \in \mathbb{N} \mid \text{there exists an } n \in \mathbb{N} \text{ with } f(n) \text{ defined and } f(n) \geq m\}$$

3. **Decidability**
   Let $M_0, M_1, \ldots$ the Gödel numbering of all RAMs discussed in the lecture. Which of the following sets is decidable and which is not. Prove your statements.
   (a) $A = \{n \in \mathbb{N} \mid \text{there are 2 distinct primes } p, q \geq 2 \text{ and } i, j \geq 1 \text{ with } n = p^i q^j\}$.
   (b) $C = \{i \in \mathbb{N} \mid M_i \text{ does not halt on even inputs } n \in \mathbb{N}\}$.
   (c) $D = \{i \in \mathbb{N} \mid M_i \text{ computes the function } c_{K_0}\}$.
   (d) $E = \{i \in \mathbb{N} \mid M_i \text{ halts on at least 2024 inputs of } \mathbb{N}\}$.

4. **Differences of square numbers**
   Prove that the set $A = \{d \in \mathbb{N} \mid \exists x, y \in \mathbb{N}, d = x^2 - y^2\}$ is in REC.

Hints

Exercise 1:
You can show the undecidability with the help of the special halting problem K_0 and Property 2.75.

Exercise 2:
The function f is not necessarily total nor necessarily computable. This task can be solved without using algorithms.

Exercise 3:
The undecidability of sets can always be shown in this task using Rice's theorem. If you apply Rice's theorem, check whether the requirements of the theorem are met.
Exactly 2 of the sets are decidable.

Exercise 4:
Consider the distance of a square number $n_1 = x^2 \in \mathbb{N}$ to the next smaller square number $n_2 = (x + 1)^2$. Show that every square number x^2 for $x \geq 2$ is at least x away from every other square number.

Extra tasks

1. **RAM-Simulator**
   Write a Python program that can simulate general RAMs (cf. construction from Theorem 2.35). RAM programs are encoded as lists corresponding to the items 1 and 2 on page 219.

   Example:

Program:	Code as Python list:
`0 R3 <- 1`	$[[3, 3, 1, 0],$
`1 IF R1 = 0 GOTO 5`	$[7, 1, 5, 0],$
`2 R2 <- R2 + R0`	$[4, 2, 2, 0],$
`3 R1 <- R1 - R3`	$[5, 1, 1, 3],$
`4 GOTO 1`	$[6, 1, 0, 0],$
`5 R0 <- R2`	$[0, 0, 2, 0],$
`6 STOP`	$[9, 0, 0, 0]]$

   The Python program receives two lists as input. The first list contains the code of a RAM program, the second contains the arguments for its execution. Your Python program should return the function value computed by the RAM.

   NOTE: When having written the program, your program can —via the constructions from the proofs of Theorem 2.33 and Corollary 2.39— be converted into a RAM, which proves that there are universal RAMs.

Solutions

Solution for exercise 1:
Proof by reduction from K_0 to K: The function $f\colon \mathbb{N} \to \mathbb{N} \times \mathbb{N}$ with $f(x) = (x, x)$ is total and computable. Furthermore, $x \in K_0 \Leftrightarrow f(x) \in K$ for all $x \in \mathbb{N}$. So K_0 is reducible to K.
Then by Property 2.75: $K \in \text{REC} \Rightarrow K_0 \in \text{REC}$. As $K_0 \notin \text{REC}$, it also holds $K \notin \text{REC}$.

Solution for exercise 2:
$A = \{m \in \mathbb{N} \mid \text{there exists a } y \in W_f \text{ with } m \leq y\}$. If W_f is finite, then A is also finite and therefore decidable according to Theorem 2.72. If W_f is not finite, then W_f is unbounded and therefore $A = \mathbb{N}$. Then the set is decidable according to Theorem 2.72.

Solution for exercise 3:

1. A is decidable. The following Python program computes c_A, where `divisors` is a Python program for the function from Example 2.22.

   ```
   def cA(n):
     r = 0
     for p in range(2, n):
       if divisiors(p) == 2 and n % p == 0:
         r += 1
     if r == 2:
       return 1
     return 0
   ```

   The program uses that if $p, q \geq 2$ and $p + q = n$, then $p, q \leq n$.

2. C is undecidable. Define $S = \{f : \mathbb{N} \to \mathbb{N} \mid f \text{ computable and } f(2n) = n.d. \text{ for all } n \in \mathbb{N}\}$. Due to the nowhere defined function S is not empty and the function $c_{\mathbb{N}}$ is computable, but not in S, which shows that S is a proper subset of the set of all computable functions. Thus, Rice's theorem can be applied and $I(S)$ is undecidable.
   Due to $I(S) = \{i \in \mathbb{N} \mid \text{the function computed by } M_i \text{ is in } S\} = \{i \in \mathbb{N} \mid \text{the function computed by } M_i \text{ has no even numbers in}$
   C the set C is also undecidable.

3. D is decidable. As no RAM computes c_{K_0}, the set D is empty.

4. E is undecidable. Define $S = \{f : \mathbb{N} \to \mathbb{N} \mid f \text{ computable and } |D_f| \geq 2024\}$. S is not empty due to $f(x) = x$ and is a proper subset of the set of all computable functions as the nowhere defined function is computable, but not in S. Thus, Rice's theorem is applicable and $I(S)$ is undecidable.
   Due to $I(S) = \{i \in \mathbb{N} \mid \text{the function computed by } M_i \text{ is in } S\} = \{i \in \mathbb{N} \mid \text{the function computed by } M_i \text{ has at least 2024 numbe}$
   E the set E is also undecidable.

Solution for exercise 4:
The distance of a square number Der Abstand einer Quadratzahl $n_1 = x^2 \in \mathbb{N}$ from the next smaller square number $n_2 = (x-1)^2$ is:

$$a(n_1) = n_1 - n_2 = x^2 - (x^2 - 2x + 1) = 2x - 1.$$

A square x^2 for $x \geq 2$ has thus a distance of at least $\geq 2x - 1 > x$ from every other square (here we silently use that the square function grows monotonically). Thus, a number $x \geq 2$ can be represented as a difference of two squares if and only if there are $y, z < x$ with $x = y^2 - z^2$.
This leads to the following decision algorithm for A

1. Input $x \in \mathbb{N}$
2. For $i = 0, \ldots, x - 2$:
   For $j = i, \ldots, x - 1$:
   If $x = j^2 - i^2$, then return 1
3. Return 0.

Alternative proof: We show that $A = \mathbb{N} - \{4n + 2 \mid n \in \mathbb{N}\}$. Thus A is decidable.
$\subseteq$: This is equivalent to $A \cap \{4n+2 \mid n \in \mathbb{N}\} = \emptyset$. Assume that $n \in \mathbb{N}$ exists, so that there is $x, y \in \mathbb{N}$ with $4n+2 = x^2 - y^2$. Then $4n + 2 = (x - y) \cdot (x + y)$ holds. Since $4n + 2$ is even, one of the factors $(x - y)$ and $x + y$ must be even. If $(x - y)$ is even, then $(x + y)$ is also even. If $x + y$ is even, then $x - y$ is also even. So both factors are even and therefore $4n + 2$ is divisible by 4, a contradiction.
$\supseteq$: Let $x \in \mathbb{N} - \{4n+2 \mid n \in \mathbb{N}\}$. If x is odd, i.e. $x = 2x' + 1$ for an $x' \in \mathbb{N}$, then $x = (x'+1)^2 - x'^2$. It therefore remains to consider the case that x is divisible by 4, i.e. $x = 4x'$. Without loss of generality, $x' > 0$ (otherwise $x = x' = 0^2 - 0^2$). Now choose $a = (x' + 1) - (x' - 1)$ and $b = (x' + 1) + (x' - 1)$. Then $a \cdot b = (x' + 1)^2 - (x' - 1)^2$ and at the same time $a \cdot b = 2 \cdot 2x' = 4x' = x$.

Solution for extra task 1:

```
def write(r, i, v):
    while(len(r) <= i):
        r.append(0)
    r[i] = v
    return r

def read(r, i):
    if (len(r) <= i):
        return 0
    return r[i]

def simulateRAM_(prog, r):
    # Programm in Zeile 0 starten...
    br = 0
    while(1):
        # Abbrechen, falls br auf falsche Adresse zeigt
        if (br < 0 or br >= len(prog)):
            return read(u, v, 0)
        #
        # Aktuellen Befehl merken
        [opcode, i, j, k] = prog[br]
        #
        # Falscher Befehlscode?
        if (opcode < 0 or opcode > 9):
            print("ERROR, ABORTING.")
            return -1
        #
        # Befehlszeiger erhoehen; Sprungbefehle ueberschreiben
        # dieses Verhalten weiter unten
        br += 1
        #
        # Je nach Befehlscode den zugehourigen Befehl abarbeiten...
        if (opcode == 0):
            r = write(r, i, read(r, j))
        if (opcode == 1):
            r = write(r, i, read(r, read(r, j)))
        if (opcode == 2):
            r = write(r, read(r, i), read(r, j))
        if (opcode == 3):
            r = write(r, i, j)
        if (opcode == 4):
            r = write(r, i, read(r, j) + read(r, k))
        if (opcode == 5):
            r = write(r, i, read(r, j) - read(r, k))
            if(r[i] < 0):
              r[i] = 0
        if (opcode == 6):
            br = i
        if ((opcode == 7 and read(r, i) == 0) or (opcode == 8 and read(r, i) > 0)):
            br = j
        if (opcode == 9):
            return read(u, v, 0)

#----------------------- TEST -----------------------
import importlib
userfile = importlib.import_module("03-Z01")
simulateRAM = userfile.simulateRAM


def test(x = 7, y = 7, z = 7):
  #
  # 0 R3 <- RR2
  # 1 R3 <- R3 + R2
  # 2 R2 <- R2 + R1
  # 3 RR2 <- R3
  # 4 R0 <- R0 - R1
  # 5 IF R0 > 0 GOTO 0
```

```
  # 6 R0 <- RR2
  # 7 STOP
  #
  prog1 = [[1,3,2,0], [4,3,3,2], [4,2,2,1], [2,2,3,0], [5,0,0,1], [8,0,0,0], [1,0,2,0], [9,0,0,0]]
  #
  # 0 R4 <- 1
  # 1 IF R4 > 0 GOTO 3
  # 2 STOP
  # 3 GOTO 5
  # 4 STOP
  # 5 R0 <- R1
  #
  prog2 = [[3,4,1,0], [8,4,3,0], [9,0,0,0], [6,5,0,0], [9,0,0,0], [0,0,1,0]]
  log = dict()
  for i in range(1,x):
    for j in range(1,y):
      for k in range(1,z):
        print("Try :" + str(i) + " " + str(j) + " " + str(k))
        if simulateRAM(prog1, [i,j,k]) !=  simulateRAM_(prog1, [i,j,k]):
          log[(2,i,j,k)] = ["Programm 1", "Ist: " + str(simulateRAM(prog1, [i,j,k])), "Soll: " + str(simulateRAM_(
g1, [i,j,k]))]
        if simulateRAM(prog2, [i,j,k]) !=  simulateRAM_(prog2, [i,j,k]):
          log[(1,i,j,k)] = ["Programm 2", "Ist: " + str(simulateRAM(prog2, [i,j,k])), "Soll: " + str(simulateRAM_(
g2, [i,j,k]))]
  if len(log)>0:
    print("Fehler gefunden. Folgendes sind die Fehelr:")
    print(log)
  else:
    print("Keine Fehler gefunden")


if __name__ == "__main__":
  test()
```