Theoretical Computer Science – Exercise 10

SS 2022
Jochen Schmidt

Technische Hochschule Rosenheim

Please prepare the following exercises at home prior to the tutorial:

Exercise 1

Addition and multiplication of two natural numbers belong to the class of primitive-recursive functions (and may be used in the questions). Show that the following functions are also primitive-recursive:

a) The predecessor function:
   $v(x) = x - 1$ if $x \geq 1$
   $v(x) = 0$ if $x = 0$
b) The modified subtraction (as for LOOP-programs):
   $f(m, n) = m - n$ if $m \geq n$
   $f(m, n) = 0$ else
c) The exponential function $\exp(x_1, x_2) = x_1^{x_2}$
d) The factorial n!, defined by:
   $n! = n\,(n - 1)!$
   $0! = 1$

Exercise 2

An important function regarding computability in theoretical computer science is the Ackermann function a(x, y). It is defined recursively as follows:

a(0, y) = y + 1
a(x + 1, 0) = a(x, 1)
a(x + 1, y + 1) = a(x, a(x + 1, y))

a) Calculate: a(0,0), a(0,1), a(1,0), and a(1,1).

b) Calculate a(2,3). Re-use the results from question (a).
   Hint: It is convenient to write down the contents of the stack on which the parameters are stored instead of the nested functions, e.g.:
   Instead of: a(2, 3) = a(1, a(2, 2)) = a(1, a(1, a(2, 1))) = …
   use: a(2, 3) = 2, 3 = 1, 2, 2 = 1, 1, 2, 1 = …
   Here, the right side corresponds to "top" on the stack. In each step the two top values are popped from the stack and replaced either by the result or by the parameters for calculating the next Ackermann function.

c) Based on the results from (a) and (b), try to find equivalent closed-form formulas (i.e., non-recursive) for the following recursive functions f_0, f_1, and f_2. Proof by induction that your formulas are correct.
   $f_0(x) = a(0, x)$
   $f_1(x) = a(1, x)$
   $f_2(x) = a(2, x)$

d) Is the Ackermann function primitive-recursive or μ-recursive?

e) Is the Ackermann function LOOP-computable?

Exercise 3

Implement functions to calculate the Ackermann function in C or Java:

a) recursive (directly from the definition of the function),
b) iterative. Hint: Implement the stack calculation as described in question (2b).

Exercise 4

There exists a closed-form formula for a(3, y), which is $a(3, y) = 2^{y+3} - 3$

a) Calculate the values a(3, 6), a(4, 0), a(4, 1), a(4, 2), a(4, 3), and a(5, 0)
b) Check the values with your C/Java implementation. Does everything work as expected?

We will do the following exercises together during the tutorial:

Exercise 5

a) Show that the function $f(n) = \frac{n!}{3!(n-3)!}$ is primitive-recursive for $n \geq 3$. In addition to the definition of primitive recursion, you can use that the following functions are also primitively recursive:
   - Multiplication: $m(x, y) = xy$
   - Division: $d(x, y) = \frac{x}{y}$
   - Predecessor: $v(x) = x - 1$
b) Specify a LOOP program that computes $f(n)$. Again, the multiplication and division operations may be assumed as given LOOP-computable functions.