Theory of Computer Science

G. Röger
Spring Term 2019

University of Basel
Computer Science

Exercise Sheet 12 — Solutions

Exercise 12.1 (LOOP programs, 1 mark)

Which function does the following program compute?

LOOP x_1 DO
  $x_1 := x_1 + 1$
END;
LOOP x_1 DO
  $x_1 := x_1 + 1$
END;
$x_0 := x_1$

Solution:

$f(x) = 4x$

Exercise 12.2 (LOOP-computability, 0.5 marks)

Consider the following function g that computes a modified modulo operation:

$$g(x, y) = \begin{cases} x \bmod y, & \text{if } y > 0 \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Is g LOOP-computable?

Solution:

No, only total functions can be LOOP-computable.

Exercise 12.3 (Alternative Definition of LOOP programs, 2 marks)

Show that with the following definition of LOOP' programs, we can compute exactly the same functions as with the definition of LOOP programs from the lecture:

LOOP' programs are inductively defined as follows:

- $x_i := x_j$ is a LOOP' program for every $i, j \in \mathbb{N}_0$ (*assignment*)
- $x_i := x_i + 1$ is a LOOP' program for every $i \in \mathbb{N}_0$ (*incrementation*)
- $x_i := x_i - 1$ is a LOOP' program for every $i \in \mathbb{N}_0$ (*modified decrementation*)
- If P_1 and P_2 are LOOP' programs, then so is $P_1; P_2$ (*composition*)
- If P is a LOOP' program, then so is
  LOOP x_i *DO* P *END*
  for every $i \in \mathbb{N}_0$ (*LOOP loop*)

Solution:

Composition and LOOP loops are present in both definitions, so it is sufficient to show that we can simulate assignment, incrementation and modified decrementation with LOOP programs (as defined in the lecture) and that we can simulate addition and modified subtraction with LOOP' programs.

We already have introduced assignments as syntactic sugar for LOOP programs in the lecture. Incrementation and decrementation can be simulated by addition and modified subtraction, using $i = j$ and $c = 1$.
For the other direction, we can simulate a LOOP statement $x_i := x_j + c$ with the following LOOP' program:

$x_i := x_j$;
$x_i := x_i + 1$;
$\vdots$ (overall c times)
$x_i := x_i + 1$

Analogously, we can use c decrement operations for a modified subtraction $x_i := x_j - c$.

Exercise 12.4 (Syntactic Sugar, 1.5 + 1.5 + 1.5 marks)
Simulate the following syntactical constructs for LOOP-programs (with obvious semantics) by using already known constructs. In addition to the base constructs of LOOP programs you may use the additional constructs introduced in chapter F1.

(a) **IF** $x_i > c$ **THEN** P **ELSE** P' **END**

Solution:

$x_k := x_i - c$;
IF $x_k \neq 0$ **THEN**
  P
END;
IF $x_k = 0$ **THEN**
  P'
END

Where x_k is a fresh variable.

(b) **IF** $x_i = x_j$ **THEN** P **END**

Solution:

$x_k := x_i - x_j$;
$x_l := x_j - x_i$;
$x_m := x_k + x_l$;
IF $x_m = 0$ **THEN**
  P
END

Where x_k, x_l and x_m are fresh variables.

(c) **FOR** $x_i = 1$ **TO** c **DO** P **END**

Solution:

$x_k := c$;
$x_i := 0$;
LOOP x_k **DO**
  $x_i := x_i + 1$;
  P
END

Where x_k is a fresh variable.

Exercise 12.5 (2 marks)

Specify a LOOP program that computes the exponentiation $f(x, y) = x^y$. You may use all syntactic sugar introduced in the lecture.

Solution:

$x_0 := 1;$
LOOP x_2 DO
  $x_3 := 0;$
  LOOP x_1 DO
    $x_3 := x_3 + x_0$
  END;
  $x_0 := x_3$
END