Software-Modellierung und Verifikation
Informatik 2
Prof. J.-P. Katoen
RWTH Aachen

Priv.-Doz. T. Noll noll@cs.rwth-aachen.de
D. Willems willems@cs.rwth-aachen.de

5. Exercise sheet *Semantics and Verification of Software 2007*

Due to Wed., 16 May 2007, *before* the exercise course begins.

Exercise 5.1:

Dijkstra's *guarded commands* are essentially of the form

$$\begin{array}{l}\textbf{do } b_1 \to c_1 \\ \quad\; b_2 \to c_2 \\ \textbf{od}\end{array}$$

(where $b_1, b_2 \in \mathbf{BExp}$ and $c_1, c_2 \in \mathbf{Cmd}$). They form a natural generalisation of the **while** loop:

> While at least one of the tests is true, the corresponding statement is executed. Here the satisfaction of both tests results in a non–deterministic choice of the command. The computation terminates as soon as neither of the tests is true.

(a) Which function on the natural numbers is computed by the following statement? Transform it to a *WHILE* statement.

$$\begin{array}{l}\textbf{do } x > y \to x := x - y \\ \quad\; y > x \to y := y - x \\ \textbf{od}\end{array}$$

(b) Let $b_1, b_2 \in \mathbf{BExp}$ be two mutually excluding tests (i.e., in no state both b_1 and b_2 are true) and $c_1, c_2 \in \mathbf{Cmd}$. How can the semantics of

$$\begin{array}{l}\textbf{do } b_1 \to c_1 \\ \quad\; b_2 \to c_2 \\ \textbf{od}\end{array}$$

be defined as the least fixpoint of a mapping

$$\Phi : (\Sigma \dashrightarrow \Sigma) \to (\Sigma \dashrightarrow \Sigma)?$$

Exercise 5.2:

Define a three-valued denotational semantics for the *WHILE* language as follows:

(a) Assume that at the beginning of a programm evaluation, all variables have unknown values. To model this, extend the variable domain by $\perp$, and let $\sigma_\perp$ with $\sigma(x) = \perp$ for all $x \in \mathbf{Var}$ be the initial *state* of all programs. Define $\mathfrak{A}[\![\cdot]\!]$ in analogy to Definition 4.6 and evaluate $3 + x$ and $0 * x$ for $\sigma_\perp$.

(b) In addition to true and false, a third truth-value ? is needed to express uncertainty about the result of a boolean expression, i.e. $x > 0$ may hold or not, depending on how x is initialized, and thus it should evaluate to ?. Define $\mathfrak{B}[\![\cdot]\!]$ in analogy to Definition 4.7 and evaluate $\neg(x = y) \wedge \mathit{false}$ for $\sigma_\perp$.

(c) Define **cond** such that common evaluation results are preserved in case of an indefinite evaluation of the boolean expression. Evaluate $\mathbf{cond}(?, \mathfrak{C}[\![x := 2; y := 3; z := x + y]\!], \mathfrak{C}[\![x := 3; y := 2; z := x + y]\!])$ for $\sigma_\perp$.

Exercise 5.3:

(a) Give an assertion $A \in \mathbf{Assn}$ with a logical variable $i \in \mathbf{LVar}$ which expresses that i is a prime number. More concretely, for every $\sigma \in \Sigma$ and every $I \in \mathbf{Int}$,

$$\sigma \models^I A$$

should be valid iff i is a prime number.

(b) Give an assertion $A \in \mathbf{Assn}$ with logical variables $i, j, k \in \mathbf{LVar}$ which expresses that k is the least common multiple of i and j.