

5. Exercise sheet *Semantics and Verification of Software WS0809*

Due to Monday, 24 Nov. 2008, *before* the exercise course begins.

Exercise 5.1:

(1+1 points)

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

```
do b1 → c1
   b2 → c2
od
```

(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.

```
do x > y → x := x - y
   y > x → y := y - x
od
```

- (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

```
do b1 → c1
   b2 → c2
od
```

be defined as the least fixpoint of a mapping

$$\Phi : (\Sigma \rightarrow \Sigma) \rightarrow (\Sigma \rightarrow \Sigma)?$$

Solution

- (a) greatest common divisor: i.e. $\sigma_{12,9} \rightarrow \sigma_{3,9} \rightarrow \sigma_{3,6} \rightarrow \sigma_{3,3}$

Termination when $x = y$.

```
while x ≠ y do if x > y then x := x - y else y := y - x
```

- (b) In general: **while** b_1 xor b_2 **do if** b_1 **then** c_1 **else** c_2

Thus $\mathcal{C}[c] = \mathbf{fix}(\Phi)$ where

$$\Phi(f) = \mathbf{cond}(\mathfrak{B}[b_1 \text{ xor } b_2], f \circ \mathcal{C}[\mathbf{if } b_1 \text{ then } c_1 \text{ else } c_2], \mathbf{id}_\Sigma) = \mathbf{cond}(\mathfrak{B}[b_1 \text{ xor } b_2], f \circ \mathbf{cond}(\mathfrak{B}[b_1], \mathcal{C}[c_1], \mathcal{C}[c_2]), \mathbf{id}_\Sigma)$$

$$\text{For state } \sigma \in \Sigma: \Phi(f)(\sigma) = \begin{cases} f(\mathcal{C}[c_1]\sigma) & \text{if } \mathfrak{B}[b_1](\sigma) = \text{true and } \mathfrak{B}[b_2](\sigma) = \text{false} \\ f(\mathcal{C}[c_2]\sigma) & \text{if } \mathfrak{B}[b_2](\sigma) = \text{true and } \mathfrak{B}[b_1](\sigma) = \text{false} \\ \sigma & \text{otherwise} \end{cases}$$

Exercise 5.2:**(2+2+1 points)**

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

- Assume that at the beginning of a program evaluation, all variables have unknown values. To model this, extend the variable domain by \perp , and let σ_\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 σ_\perp .
- 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 $\neg(x = y) \wedge \textit{false}$ for σ_\perp .
- Define **cond** such that common evaluation results are preserved in case of an indefinite evaluation of the boolean expression. Evaluate **cond**($?, \mathfrak{C}[x := 2; y := 3; z := x + y], \mathfrak{C}[x := 3; y := 2; z := x + y]$) for σ_\perp .

Solution

$$\Sigma = \{\sigma \mid \sigma : \mathbf{Var} \rightarrow \mathbb{Z} \cup \{\perp\}\}$$

$$(a) \mathfrak{A}[\cdot] : \mathbf{AExp} \mapsto \Sigma \rightarrow \mathbb{Z} \cup \{\perp\}$$

- $\mathfrak{A}[z] \sigma := z$
- $\mathfrak{A}[x] \sigma := \sigma(x)$
- $\mathfrak{A}[a_1 + a_2] \sigma := \mathfrak{A}[a_1] \sigma \oplus \mathfrak{A}[a_2] \sigma$
 where $z_1 \oplus z_2 = \begin{cases} z_1 + z_2 & \text{if } z_1 \neq \perp \neq z_2 \\ \perp & \text{otherwise} \end{cases}$ for $z_1 = \mathfrak{A}[a_1], z_2 = \mathfrak{A}[a_2]$
- $\mathfrak{A}[a_1 * a_2] \sigma := \mathfrak{A}[a_1] \sigma \otimes \mathfrak{A}[a_2] \sigma$
 where $z_1 \otimes z_2 = \begin{cases} z_1 * z_2 & \text{if } z_i \neq \perp \text{ and } z_i \neq 0, i \in \{1, 2\} \\ 0 & \text{if } z_1 = 0 \text{ or } z_2 = 0 \\ \perp & \text{otherwise} \end{cases}$ for $z_1 = \mathfrak{A}[a_1], z_2 = \mathfrak{A}[a_2]$

Example evaluations:

- $\mathfrak{A}[3 + x] \sigma_\perp = \mathfrak{A}[3] \sigma_\perp \oplus \mathfrak{A}[x] \sigma_\perp = 3 \oplus \perp = \perp$
- $\mathfrak{A}[0 * x] \sigma_\perp = \mathfrak{A}[0] \sigma_\perp \otimes \mathfrak{A}[x] \sigma_\perp = 0 \otimes \perp = 0$

$$(b) \mathfrak{B}[\cdot] : \mathbf{BExp} \mapsto \Sigma \rightarrow \{\textit{true}, \textit{false}, ?\}$$

- $\mathfrak{B}[t] \sigma := t$
- $\mathfrak{B}[a_1 = a_2] \sigma := \begin{cases} \textit{true} & \text{if } \mathfrak{A}[a_1] \sigma = \mathfrak{A}[a_2] \sigma \neq \perp \\ \textit{false} & \text{if } \perp \neq \mathfrak{A}[a_1] \sigma \neq \mathfrak{A}[a_2] \sigma \neq \perp \\ ? & \text{otherwise} \end{cases}$
- for $>$ analogously
- $\mathfrak{B}[\neg b] \sigma := \begin{cases} \textit{true} & \text{if } \mathfrak{B}[b] \sigma = \textit{false} \\ \textit{false} & \text{if } \mathfrak{B}[b] \sigma = \textit{true} \\ ? & \text{otherwise} \end{cases}$
- $\mathfrak{B}[b_1 \wedge b_2] \sigma := \begin{cases} \textit{true} & \text{if } \mathfrak{B}[b_1] \sigma = \textit{true} \text{ and } \mathfrak{B}[b_2] \sigma = \textit{true} \\ \textit{false} & \text{if } \mathfrak{B}[b_1] \sigma = \textit{false} \text{ or } \mathfrak{B}[b_2] \sigma = \textit{false} \\ ? & \text{otherwise} \end{cases}$
- for \vee analogously

Example evaluations

- $\mathfrak{B}[[x = y]]\sigma_{\perp} = \perp, \sigma_{\perp}(x) = \perp, \sigma_{\perp}(y) = 3$
- $\mathfrak{B}[[\neg(x = y) \wedge \text{false}]]\sigma_{\perp} = \text{false}$

$$(c) \text{ cond}(p, f, g)(\sigma) := \begin{cases} f(\sigma) & \text{if } p(\sigma) = \text{true} \\ g(\sigma) & \text{if } p(\sigma) = \text{false} \\ f(\sigma) \sqcup g(\sigma) & \text{otherwise} \end{cases}$$

$$\text{where } (\sigma_1 \sqcup \sigma_2)(x) = \begin{cases} \sigma_1(x) & \text{if } \sigma_1(x) = \sigma_2(x) \\ \perp & \text{otherwise} \end{cases}$$

$$\begin{aligned} & \text{cond}(\text{?}, \mathfrak{C}[[x := 2; y := 3; z := x + y]], \mathfrak{C}[[x := 3; y := 2; z := x + y]])\sigma_{\perp} \\ &= \sigma_{\perp}[x \mapsto 2, y \mapsto 3, z \mapsto 5] \sqcup \sigma_{\perp}[x \mapsto 3, y \mapsto 2, z \mapsto 5] \\ &= \sigma_{\perp}[z \mapsto 5] \end{aligned}$$

□

Exercise 5.3:

(1+1 points)

- (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 .

Solution

- (a) i prime, iff $i > 1$ and the only divisors of i are i and 1 iff $i > 1 \wedge \forall j : (j|i \Rightarrow j = 1 \vee j = i)$

$$j|i \text{ iff } i \text{ is a multiple of } j \text{ iff } \exists k : k \geq 1 \wedge i = j * k$$

$$\text{Altogether: } i > 1 \wedge \forall j : ((\exists k : k \geq 1 \wedge i = j * k) \Rightarrow j = 1 \vee j = i)$$

- (b) k is a multiple of i iff $M(k, i) := \exists i' : i' \geq 1 \wedge k = i * i'$

$$k \text{ is least multiple of } i \text{ and } j, \text{ iff } M(k, i) \wedge M(k, j) \wedge \forall k' : (M(k', i) \wedge M(k', j) \Rightarrow k' \geq k)$$

□

Exercise 5.4:

(1+1 points)

- (a) Prove by structural induction on expressions $a \in \mathbf{LExp}$ and $n \in \mathbb{N}$ that

$$\mathfrak{L}[[a]]I[n/i]\sigma = \mathfrak{L}[[a[n/i]]]I\sigma.$$

- (b) By using the fact above, for any $A \in \mathbf{Assn}$, prove

$$\sigma \models^I \forall i. A \text{ iff } \sigma \models^I A[n/i] \text{ for all } n \in \mathbb{N}.$$

Solution

- (a) Semantics for $\mathcal{L}[a]I[n/i]\sigma$, $a \in \mathbf{LExp}$ and $n \in \mathbb{N}$:
Base case:

$$\begin{aligned}\mathcal{L}[n]I[n/i]\sigma &= n, \text{ for } n \in \mathbb{N} \\ \mathcal{L}[x]I[n/i]\sigma &= \sigma(x), \text{ for } x \in \mathbf{Var} \\ \mathcal{L}[j]I[n/i]\sigma &= \begin{cases} n & , i \equiv j \\ I(j) & , \text{otherwise} \end{cases}\end{aligned}$$

Induction hypothesis: Holds for a_0 and a_1 .
Induction step:

$$\begin{aligned}\mathcal{L}[a_0 + a_1]I[n/i]\sigma &= \mathcal{L}[a_0]I[n/i]\sigma + \mathcal{L}[a_1]I[n/i]\sigma \\ \mathcal{L}[a_0 - a_1]I[n/i]\sigma &= \mathcal{L}[a_0]I[n/i]\sigma - \mathcal{L}[a_1]I[n/i]\sigma \\ \mathcal{L}[a_0 * a_1]I[n/i]\sigma &= \mathcal{L}[a_0]I[n/i]\sigma * \mathcal{L}[a_1]I[n/i]\sigma\end{aligned}$$

Semantics for $\mathcal{L}[a[n/i]]I\sigma$, $a \in \mathbf{LExp}$ and $n \in \mathbb{N}$:
Base case:

$$\begin{aligned}\mathcal{L}[n[n/i]]I\sigma &= \mathcal{L}[n]I\sigma = n, \text{ for } n \in \mathbb{N} \\ \mathcal{L}[x[n/i]]I\sigma &= \mathcal{L}[x]I\sigma = \sigma(x), \text{ for } x \in \mathbf{Var} \\ \mathcal{L}[j[n/i]]I\sigma &= \begin{cases} \mathcal{L}[n]I\sigma & , i \equiv j \\ \mathcal{L}[j]I\sigma & , \text{otherwise} \end{cases} = \begin{cases} n & , i \equiv j \\ I(j) & , \text{otherwise} \end{cases}\end{aligned}$$

Induction hypothesis: Holds for a_0 and a_1 .
Induction step:

$$\begin{aligned}\mathcal{L}[(a_0 + a_1)[n/i]]I\sigma &= \mathcal{L}[a_0[n/i]]I\sigma + \mathcal{L}[a_1[n/i]]I\sigma \\ \mathcal{L}[(a_0 - a_1)[n/i]]I\sigma &= \mathcal{L}[a_0[n/i]]I\sigma - \mathcal{L}[a_1[n/i]]I\sigma \\ \mathcal{L}[(a_0 * a_1)[n/i]]I\sigma &= \mathcal{L}[a_0[n/i]]I\sigma * \mathcal{L}[a_1[n/i]]I\sigma\end{aligned}$$

One can easily see that $\mathcal{L}[a]I[n/i]\sigma = \mathcal{L}[a[n/i]]I\sigma$.

- (b) The proof is by structural induction on $A \in \mathbf{Assn}$. We consider only the case when $A := (a_0 = a_1)$. For the rest of the cases the proof follows the same lines.

$$\begin{aligned}\sigma \models^I \forall i. A &\Leftrightarrow \sigma \models^{I[n/i]} A \Leftrightarrow \mathcal{L}[a_0]I[n/i]\sigma = \mathcal{L}[a_1]I[n/i]\sigma, \text{ for all } n \in \mathbb{N} \\ \sigma \models^I A[n/i] &\Leftrightarrow \mathcal{L}[a_0[n/i]]I\sigma = \mathcal{L}[a_1[n/i]]I\sigma, \text{ for all } n \in \mathbb{N}\end{aligned}$$

From (a) we know that $\mathcal{L}[a_0[n/i]]I\sigma = \mathcal{L}[a_1[n/i]]I\sigma \Leftrightarrow \mathcal{L}[a_0]I[n/i]\sigma = \mathcal{L}[a_1]I[n/i]\sigma$ which means that $\sigma \models^I \forall i. A \Leftrightarrow \sigma \models^I A[n/i]$ for all $n \in \mathbb{N}$.

□