Exercise Sheet 9: Non-determinism and Parallelism

Due date: July 1st. You can hand in your solutions at the start of the exercise class.

Exercise 1 (Axiomatic/Predicate Transformers Semantics of Non-determinism) **30%**

Consider an extension of the WHILE programming language with a non-deterministic operator $c_1 \,\Box\, c_2$. There are two possible semantical models for this operator. In the *demonic* model to establish a postcondition Q one requires that every possible program execution (induced by its non-deterministic choices) establishes Q, while on the *angelic* model one requires that at least one execution establishes Q.

(a) [5%] Extend the Hoare logic proof system to model demonic non-determinism.

(b) [2.5%] Give an inductive definition of $\mathtt{wp}[c_1 \,\Box\, c_2]$ in the demonic model.

(c) [7.5%] Decide whether statement $\mathtt{wp}[c](Q_1 \vee Q_2) = \mathtt{wp}[c](Q_1) \vee \mathtt{wp}[c](Q_2)$ holds or not under a demonic model of non-determinism. (If so, prove it; if not, provide a counterexample.)

(d) [5%] Extend the Hoare logic proof system to model angelic non-determinism.

(e) [2.5%] Give an inductive definition of $\mathtt{wp}[c_1 \,\Box\, c_2]$ in the angelic model.

(f) [7.5%] Decide whether statement $\mathtt{wp}[c](Q_1 \wedge Q_2) = \mathtt{wp}[c](Q_1) \wedge \mathtt{wp}[c](Q_2)$ holds or not under an angelic model of non-determinism. (If so, prove it; if not, provide a counterexample.)

Exercise 2 (Connection between Denotational and Axiomatic Semantics for ND) **20%**

Assume we extend the WHILE programming language with the non-deterministic operator $c_1 \,\Box\, c_2$ and define function

$$[\![c]\!] \colon \Sigma \to \mathcal{P}(\Sigma_\perp)$$

that maps each initial state to the set of possible final states, where $\perp$ represents divergence. For instance for the following programs

$c_1\colon\; x := 1 \,\Box\, x := 2$

$c_2\colon\; x := 1 \,\Box\, x := 2;$
$\quad \mathsf{while\ true\ do\ skip}$

$c_3\colon\; b := \mathsf{true};\, n := 0$
$\quad \mathsf{while}\ b\ \mathsf{do}$
$\qquad n := n{+}1;$
$\qquad \mathsf{skip} \,\Box\, b := \mathsf{false}$

we have

$$[\![c_1]\!](\sigma) = \{\sigma[x \mapsto 1], \sigma[x \mapsto 2]\}$$
$$[\![c_2]\!](\sigma) = \{\perp\}$$
$$[\![c_3]\!](\sigma) = \{\perp, \sigma[b, n \mapsto \mathsf{false}, 1], \sigma[b, n \mapsto \mathsf{false}, 2], \ldots\}$$

For simplicity assume that P and Q contain no logical variables. Moreover assume the standard demonic model of non-determinism.

(a) [10%] Characterise the validity of triple $\{P\}\, c\, \{\Downarrow Q\}$ in terms of $[\![c]\!]$.

(b) [10%] Characterise the validity of triple $\{P\}\, c\, \{Q\}$ in terms of $[\![c]\!]$.

Exercise 3 (Equivalence of Statements in ParWHILE) $\boxed{\textbf{30\%}}$

Two statements $c_1, c_2 \in \mathsf{Cmd}$ are *equivalent*, written $c_1 \approx c_2$, if and only if

$$\forall \sigma, \sigma' \in \Sigma : \langle \sigma, c_1 \rangle \rightarrow^\star \sigma' \Leftrightarrow \langle \sigma, c_2 \rangle \rightarrow^\star \sigma'.$$

In previous exercises, we occasionally made use of the fact that program statements can be replaced by equivalent ones without changing the programs behavior, i.e. $P[c \mapsto c_1] \approx P[c \mapsto c_2]$ holds for all `WHILE` programs $P \in \mathsf{Cmd}$ containing a statement $c \in \mathsf{Cmd}$ and $c_1 \approx c_2$. Prove or disprove that ParWHILE programs have the same property.

Exercise 4 (Fairness in CSP) $\boxed{\textbf{20\%}}$

Prove or disprove each of the following statements on executions of programs written in CSP.

(a) [10%] Every strongly unfair execution is weakly unfair.

(b) [10%] Every weakly unfair execution is strongly unfair.