

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 $c_1 \sqcap c_2$. 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 $\mathbf{wp}[c_1 \sqcap c_2]$ in the demonic model.
- (c) [7.5%] Decide whether statement $\mathbf{wp}[c](Q_1 \vee Q_2) = \mathbf{wp}[c](Q_1) \vee \mathbf{wp}[c](Q_2)$ 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 $\mathbf{wp}[c_1 \sqcap c_2]$ in the angelic model.
- (f) [7.5%] Decide whether statement $\mathbf{wp}[c](Q_1 \wedge Q_2) = \mathbf{wp}[c](Q_1) \wedge \mathbf{wp}[c](Q_2)$ holds or not under an angelic model of non-determinism. (If so, prove it; if not, provide a counterexample.)

Solution:

$$(a). \frac{\{P\} c_1 \{Q\} \quad \{P\} c_2 \{Q\}}{\{P\} c_1 \sqcap c_2 \{Q\}}$$

$$(b). \mathbf{wp}[c_1 \sqcap c_2](Q) = \mathbf{wp}[c_1](Q) \wedge \mathbf{wp}[c_2](Q)$$

(c). Statement does not hold. For

$$\begin{aligned} c &:= x := 0 \sqcap x := 1 \\ Q_1 &:= x = 0 \\ Q_2 &:= x = 1 \end{aligned}$$

we have $\mathbf{wp}[c](Q_1 \vee Q_2) = \text{true}$, while $\mathbf{wp}[c](Q_1) = \mathbf{wp}[c](Q_2) = \text{false}$.

$$(d). \frac{\{P\} c_1 \{Q\}}{\{P\} c_1 \sqcap c_2 \{Q\}} \quad \frac{\{P\} c_2 \{Q\}}{\{P\} c_1 \sqcap c_2 \{Q\}}$$

$$(e). \mathbf{wp}[c_1 \sqcap c_2](Q) = \mathbf{wp}[c_1](Q) \vee \mathbf{wp}[c_2](Q)$$

(f). Statement does not hold. For

$$\begin{aligned} c &:= x := 0 \sqcap x := 1 \\ Q_1 &:= x = 0 \\ Q_2 &:= x = 1 \end{aligned}$$

we have $\mathbf{wp}[c](Q_1 \wedge Q_2) = \text{false}$, while $\mathbf{wp}[c](Q_1) = \mathbf{wp}[c](Q_2) = \text{true}$.

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

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

$$\llbracket c \rrbracket : \Sigma \rightarrow \mathcal{P}(\Sigma_{\perp})$$

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

$c_1: \quad x := 1 \sqcap x := 2$	$c_2: \quad x := 1 \sqcap x := 2;$ while true do skip	$c_3: \quad b := \text{true}; n := 0$ while b do $n := n+1;$ skip $\sqcap b := \text{false}$
-----------------------------------	--	---

we have

$$\llbracket c_1 \rrbracket(\sigma) = \{\sigma[x \mapsto 1], \sigma[x \mapsto 2]\}$$

$$\llbracket c_2 \rrbracket(\sigma) = \{\perp\}$$

$$\llbracket c_3 \rrbracket(\sigma) = \{\perp, \sigma[b, n \mapsto \text{false}, 1], \sigma[b, n \mapsto \text{false}, 2], \dots\}$$

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

Solution:

$$(a). \quad \{P\} c \{\Downarrow Q\} \quad \text{iff} \quad \forall \sigma \bullet \sigma \in P \implies \perp \notin \llbracket c \rrbracket(\sigma) \wedge \llbracket c \rrbracket(\sigma) \subseteq Q.$$

$$(b). \quad \{P\} c \{Q\} \quad \text{iff} \quad \forall \sigma \bullet \sigma \in P \implies (\llbracket c \rrbracket(\sigma) \setminus \perp) \subseteq Q.$$

Exercise 3 (Equivalence of Statements in ParWHILE)**30%**

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

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

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 $P \in \text{Cmd}$ containing a statement $c \in \text{Cmd}$ and $c_1 \approx c_2$. Prove or disprove that ParWHILE programs have the same property.

Solution: Consider the following counterexample:

$$c_1 \equiv x := 1; x := x + 1$$

$$c_2 \equiv x := 2$$

$$c_3 \equiv x := 3$$

Obviously, $c_1 \approx c_2$, but $c_1 || c_3 \not\approx c_2 || c_3$, because $\langle c_1 || c_3, \sigma \rangle \rightarrow^* \sigma[x \mapsto i]$ where $i \in \{2, 3, 4\}$ and $\langle c_2 || c_3, \sigma \rangle \rightarrow^* \sigma[x \mapsto j]$ where $j \in \{2, 3\}$ for all $\sigma \in \Sigma$.

Exercise 4 (Fairness in CSP)**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.

Solution: Yes, strong unfairness implies weak fairness. Let $\kappa = \kappa_1 \rightarrow \kappa_2 \rightarrow \dots$ be a strongly unfair execution with $\kappa_i = \langle c_1^i || \dots || c_n^i, \sigma_i \rangle$ as defined in the lecture. Then there exists some $i \in [1, n]$ and $k_0 \in \mathbb{N}$ such that c_i^k is enabled for all $k \geq k_0$. Since κ is infinite, this implies that c_i^k is enabled for infinitely many $k \geq k_0$.

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

Solution: Clearly, this is wrong. Consider for instance, example 16.7 (3) in the lecture where $\alpha?y$ is enabled infinitely often (in every third configuration), but there exists no $k_0 \in \mathbb{N}$ such that $\alpha?y$ is enabled for all $k \geq k_0$.

As a reminder, the example mentioned above is

$$\begin{aligned} & \langle \text{do } \alpha!1 \rightarrow \text{skip od} || \text{do } \alpha?x \rightarrow \text{skip od} || \alpha?y \dots \rangle \\ & \rightarrow \langle \text{skip}; \text{do } \alpha!1 \rightarrow \text{skip od} || \text{skip}; \text{do } \alpha?x \rightarrow \text{skip od} || \alpha?y \dots \rangle \\ & \rightarrow \langle \text{skip}; \text{do } \alpha!1 \rightarrow \text{skip od} || \text{do } \alpha?x \rightarrow \text{skip od} || \alpha?y \dots \rangle \\ & \rightarrow \langle \text{do } \alpha!1 \rightarrow \text{skip od} || \text{do } \alpha?x \rightarrow \text{skip od} || \alpha?y \dots \rangle \\ & \rightarrow \dots \end{aligned}$$