Principles of software composition 2017/18
Exam – July 25, 2018

[**Ex. 1**] Suppose you want to extend the syntax of IMP with the command **do** c **while** b.

1. Define the operational semantics of the new construct.
2. Extend the proof of determinacy taking into account the new construct.
3. Define the denotational semantics of the new construct, first as a recursive definition and then making explicit the function $\Psi_{c,b}$ of which the fixpoint must be computed.

[**Ex. 2**] Consider the HOFL term

$$t \stackrel{\text{def}}{=} \mathbf{rec}\ f.\ \lambda x.\ (\ x+1\ ,\ \mathbf{fst}(f\ x)\ )$$

1. Find the principal type of t.
2. Find the canonical form of the term $t\ 1$.
3. Compute the (lazy) denotational semantics of t.

[**Ex. 3**] Let us consider the CCS processes

$$\begin{array}{ll} p \stackrel{\text{def}}{=} \mathbf{rec}\ x.\alpha.\beta.x & \qquad r \stackrel{\text{def}}{=} (p|q)\backslash\beta \\ q \stackrel{\text{def}}{=} \mathbf{rec}\ y.\overline{\beta}.\alpha.y & \qquad s \stackrel{\text{def}}{=} \mathbf{rec}\ z.\alpha.\tau.z \end{array}$$

1. Draw the LTSs of the processes r and s.
2. Show that r and s are not strong bisimilar.
3. Prove that r and s are weak bisimilar.

[**Ex. 4**] Consider the atomic propositions:
$read_x$: holds when the variable x is being read;
upd_x: holds when the variable x is being updated.

1. Write the property "anytime the variable x is updated then it is eventually read" in LTL.
2. Write the property "the variable x is never updated and read at the same time" in CTL.
3. Write the property "the variable x is always in use (read or updated)" in the μ-calculus.