

Principles for software composition 2021/22

05 - HOFL

[Ex. 1] Determine the type of the HOFL term

$$t \stackrel{\text{def}}{=} \mathbf{rec} \ x. ((\lambda y. \mathbf{if} \ y \ \mathbf{then} \ 0 \ \mathbf{else} \ 0) \ x).$$

Then compute its (lazy) canonical form.

[Ex. 2] Determine the type of the HOFL term

$$\mathit{map} \stackrel{\text{def}}{=} \lambda f. \lambda x. ((f \ \mathbf{fst}(x)), (f \ \mathbf{snd}(x)))$$

Then, compute the (lazy) canonical forms of the terms

$$t_1 \stackrel{\text{def}}{=} \mathit{map} \ (\lambda z. 2 \times z) \ (1, 2) \qquad t_2 \stackrel{\text{def}}{=} \mathbf{fst} \ (\mathit{map} \ (\lambda z. 2 \times z) \ (1, 2))$$

[Ex. 3] Let (D, \sqsubseteq_D) be a CPO and $f : D \rightarrow D$ be a continuous function. Prove that the set of fixpoints of f is itself a CPO (ordered by \sqsubseteq_D).

[Ex. 4] (Test for convergence) We would like to modify the denotational semantics of HOFL assigning to the construct

$$\mathbf{if} \ t \ \mathbf{then} \ t_0 \ \mathbf{else} \ t_1$$

- the semantics of t_1 if the semantics of t is $\perp_{\mathbb{Z}_\perp}$, and
- the semantics of t_0 otherwise.

Is it possible? If not, why?

[Ex. 5] (Strict conditional) Modify the operational semantics of HOFL by taking the following rules for conditionals:

$$\frac{t \rightarrow 0 \quad t_0 \rightarrow c_0 \quad t_1 \rightarrow c_1}{\mathbf{if} \ t \ \mathbf{then} \ t_0 \ \mathbf{else} \ t_1 \rightarrow c_0} \qquad \frac{t \rightarrow n \quad n \neq 0 \quad t_0 \rightarrow c_0 \quad t_1 \rightarrow c_1}{\mathbf{if} \ t \ \mathbf{then} \ t_0 \ \mathbf{else} \ t_1 \rightarrow c_1}.$$

Without changing the denotational semantics, prove that:

1. for any term t and canonical form c , we have $t \rightarrow c \Rightarrow \forall \rho. \llbracket t \rrbracket \rho = \llbracket c \rrbracket \rho$;
2. in general $t \Downarrow \not\equiv t \downarrow$ (exhibit a counterexample).

[Ex. 6] Determine the type of the HOFL term

$$t \stackrel{\text{def}}{=} \mathbf{rec} \ f. (\lambda x. 1, \mathbf{fst}(f) \ 0)$$

Then, compute the (lazy) denotational semantics of t .