EECS 762 — Programming Language Foundation

Final Exam – Spring 2025

Due 11:59pm May 14, 2025

For all problems following, assume the simply-typed λ-calculus, $\lambda_\rightarrow$ extended with Bool, Nat, and Unit types along with if, pred, succ, iszero, and sequence operators.

Problem 1 *In class we discussed the sequence operator, $t;t$, that allows one term to be executed after another. A* while *loop is a kind of sequence operator that executes the same term repeatedly while a condition is true. Assume the following form for a* while *loop:*

$$\mathsf{while}\ t_c\ t_b$$

where the loop body t_b is evaluated repeatedly until the condition value t_c is 0. This version of while *does not use a Boolean condition, but decrements a numeric counter until it is zero. Assume the following inference rules define evaluation of the* while *term:*

$$\textit{E-While}\ \frac{t_c \rightarrow t_c'}{\mathsf{while}\ t_c\ t_b \rightarrow \mathsf{while}\ t_c'\ t_b}$$

$$\textit{E-WhileZero}\ \frac{}{\mathsf{while}\ 0\ t_b \rightarrow t_b}$$

$$\textit{E-WhileNotZero}\ \frac{}{\mathsf{while}\ \mathsf{succ}\ v\ t_b \rightarrow t_b; \mathsf{while}\ v\ t_b}$$

$$\textit{T-While}\ \frac{\Gamma \vdash t_c : \mathsf{Nat} \qquad \Gamma \vdash t_b : \mathsf{Unit}}{\Gamma \vdash \mathsf{while}\ t_c\ t_b : \mathsf{Unit}}$$

In the following, do not redo the proof steps from Pierce's proofs that do not need to change. Simply specify new cases for your solution.

1. *Write the type inversion lemma for* while *(DO NOT PROVE)*

2. *Prove or disprove that* while *exhibits progress*

3. *Prove or disprove that* while *exhibits preservation*

4. *Prove or disprove that* while *always terminates*

5. *Can you use a derived form and elaboration to accomplish the language extension for* while*? If so, do it. If not, explain why.*

6. *What would happen if you replaced the numeric counter with a general Boolean condition? Specifically, do you lose any nice properties?*

Problem 2 *Many functional languages with pairs define a* with *expression of the following form:*

$$t_1 \ \mathsf{with}\ \ell := t_2$$

where t_1 *is a pair,* ℓ *is a label taken from* $\{l, r\}$ *and* t_2 *is an expression. The purpose of the* with *expression is to replace the value associated with* ℓ *in the pair* t_1 *with the value obtained by evaluating* t_2*. The other value in the pair remains the same.*

For example $\{1, 2\}$ with $r := 1 + 2$ *would evaluate to* $\{1, 3\}$*. We get a new pair with the* r *value replaced with the result of evaluating* $1 + 2$*.*

Assume the $\lambda_{\rightarrow}$ *language extended with pairs as defined in our text and in class. You will extend this language to include the* with *expression.*

1. *Define one or more evaluation rules for* with

2. *Define one or more type rules for* with

3. *Can you use a derived form and elaboration to accomplish the language extension for* with*? If so, do it. If not, explain why.*

4. *Assuming* with *can be defined using a derived form, what would you prove to show the derived form is correct? (DO NOT PROVE)*

Problem 3 *Answer the following questions:*

1. *The untyped* Ω*-combinator has the form* $(\lambda x.xx)(\lambda x.xx)$*. Write a typed version in the simply-typed* λ*-calculus or explain why you can't.*

2. *What is the purpose of a canonical forms lemma for a language?*

3. *What does a type inversion lemma tell us about a language?*

4. *Why is it important to have only one type rule for each term in a language?*

5. *What does it mean for a typing relation relation to be deterministic? How would you prove it?*