Advanced Programming Languages — Assignment 6

CS 6110 S16 — Due Friday, 11 May 2016

Instructions: *You may complete this assignment with one other CS 6110 student. You must register with your partner on CMS before you begin working together on the assignment. With the exception of your CMS-registered partner you should not give or receive assistance on this assignment. If slip days are used, they must be used by both partners. In addition, please limit use of outside resources to the lecture notes and textbooks for this course. If you have any questions about what is allowed and what is not allowed, please contact the course staff.*

1. Recursive Types

   Consider the following OCaml datatype definitions, which represent even and odd natural numbers:

```
type even = Zero of unit | OSucc of odd
and odd = ESucc of even
```

   (a) Show to encode both OCaml datatypes using the following types:

$$\tau ::= \mathsf{unit} \mid \tau_1 \times \tau_2 \mid \tau_1 + \tau_2 \mid \alpha \mid \mu\alpha.\tau$$

   Be sure to express the mutual recursion using μ!

   (b) Show (briefly) that the unfoldings of your types satisfy their respective recursive equations.

2. Algorithmic Subtyping

   In this problem you will develop an algorithmic version of the subtyping relation. The key advantage of this algorithmic relation is that its rules are syntax directed, so implementing it efficiently is simple. Consider the types generated by the following grammar,

$$\tau ::= \top \mid \tau_1 \to \tau_2 \mid \tau_1 \times \tau_2$$

   with the subtyping relation given by the following inference rules:

$$\frac{}{\vdash \tau \leq \tau} \qquad \frac{\vdash \tau_1 \leq \tau_2 \qquad \vdash \tau_2 \leq \tau_3}{\vdash \tau_1 \leq \tau_3} \qquad \frac{}{\vdash \tau \leq \top}$$

$$\frac{\vdash \tau_1' \leq \tau_1 \qquad \vdash \tau_2 \leq \tau_2'}{\vdash \tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'} \qquad \frac{\vdash \tau_1 \leq \tau_1' \qquad \vdash \tau_2 \leq \tau_2'}{\vdash \tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'}$$

   (a) Prove that for every type τ we have $\vdash \tau \leq \tau$ without using the reflexivity rule.

   (b) Prove that for every pair of types τ_1 and τ_2, if $\vdash \tau_1 \leq \tau_2$, then there is a derivation of the same fact without using the transitivity rule.

   (c) Now consider the following algorithmic subtyping relation:

$$\frac{}{\Vdash \tau \leq \top} \qquad \frac{\Vdash \tau_1' \leq \tau_1 \qquad \Vdash \tau_2 \leq \tau_2'}{\Vdash \tau_1 \to \tau_2 \leq \tau_1' \to \tau_2'} \qquad \frac{\Vdash \tau_1 \leq \tau_1' \qquad \Vdash \tau_2 \leq \tau_2'}{\Vdash \tau_1 \times \tau_2 \leq \tau_1' \times \tau_2'}$$

   Prove that $\vdash \tau \leq \tau'$ implies $\Vdash \tau \leq \tau'$. You should clearly state the induction principle you are using and any uses of the induction hypothesis. We will consider the quality and clarity of your proof during grading.

3. Propositions as Types

   (a) Show that each of the following formulas is intuitionistically valid by finding a closed System F term with the corresponding type:
      - $\forall \alpha, \beta, \gamma.\ (\alpha \Rightarrow \beta \Rightarrow \gamma) \Rightarrow \beta \Rightarrow \alpha \Rightarrow \gamma$
      - $\forall \alpha, \beta, \gamma.\ (\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \gamma) \Rightarrow \alpha \Rightarrow \gamma$
      - $\forall \alpha.\ \alpha \Rightarrow \alpha \Rightarrow \alpha$

   (b) Some early versions of ML included functions outleft and outright, with the following types,

$$\frac{}{\Gamma \vdash \mathsf{outleft} : \tau_1 + \tau_2 \to \tau_1} \qquad \frac{}{\Gamma \vdash \mathsf{outright} : \tau_1 + \tau_2 \to \tau_2}$$

   and operational semantics rules:

$$\mathsf{outleft}\ (\mathsf{inL}_{\tau_1+\tau_2}\ v_1) \to v_1 \qquad \mathsf{outright}\ (\mathsf{inR}_{\tau_1+\tau_2}\ v_2) \to v_2$$

   It is not difficult to see that these operators are not type sound. Which property do they break? Progress, preservation, or both?

   (c) Briefly explain how the propositions-as-types principle provides yet more evidence that outleft and outright are unnatural.

4. Encoding Sums as Products

   In this exercise, we will show prove that sum types $\tau + \sigma$ can be encoded using product and universal types in System F. The propositions-as-type principle will help in constructing the appropriate translations. In particular, we will use one of De Morgan's laws: $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$.

   As the source language, we will use the simply-typed λ-calculus extended with sums:

$$\begin{array}{rcl} e & ::= & () \mid x \mid e_1\, e_2 \mid \lambda x : \tau .\, e \mid \mathsf{inL}_{\tau_1+\tau_2} e \mid \mathsf{inR}_{\tau_1+\tau_2} e \mid \mathsf{case}\ e_0\ \mathsf{of}\ e_1 \,|\, e_2 \\ \tau & ::= & \mathsf{unit} \mid \tau_1 \to \tau_2 \mid \tau_1 + \tau_2 \end{array}$$

   As the target language, we will use System F extended with products (but not sums!):

$$\begin{array}{rcl} e & ::= & () \mid x \mid e_1\, e_2 \mid \lambda x : \tau .\, e \mid (e_1, e_2) \mid \#1\, e \mid \#2\, e \mid \Lambda \alpha.\ e \mid e\, \tau \\ \tau & ::= & \mathsf{unit} \mid \tau_1 \to \tau_2 \mid \tau_1 \times \tau_2 \mid \alpha \mid \forall \alpha . \tau \end{array}$$

   (a) Give a formula that is equivalent to $\varphi \vee \psi$, but which only contains logical operators for which there are corresponding types in the fragment of System F above. (Hint: use a universal type to encode "negation".)

   (b) Define a translation $\mathcal{T}[\![\cdot]\!]$ on types that takes a type in the fragment of simply-typed λ-calculus listed above and produces a type in the fragment of System F extended with products.

   (c) Define the corresponding translation $\mathcal{E}[\![\cdot]\!]$ on expressions in the fragment of simply-typed λ calculus listed above. You only need to give the cases for $\mathsf{inL}_{\tau_1+\tau_2}$, $\mathsf{inR}_{\tau_1+\tau_2}$, and $\mathsf{case}\ e_0\ \mathsf{of}\ e_1 \,|\, e_2$.

   You may find it helpful to think about types. In particular, your translations should be type preserving in the sense that they should map well-typed terms to well-typed terms. However, you do not need to prove this property.

5. Debriefing

   (a) How many hours did you spend on this assignment?

   (b) Would you rate it as easy, moderate, or difficult?

   (c) Did everyone in your study group participate?

   (d) How deeply do you feel you understand the material it covers (0%–100%)?

   (e) If you have any other comments, we would like to hear them! Please write them here or send email to `jnfoster@cs.cornell.edu`.