Practice Homework

CS 421 – Summer 2009
Revision 1.0

Assigned July 30, 2009
Due N/A
Total points N/A

1 Change Log

1.0 Initial release.

2 Overview

This is a practice homework designed to help you master proof trees, type systems, operational semantics, and Hoare logic.

3 Collaboration

Collaboration among any number of students is allowed.

4 What to submit

Nothing! This is purely for your benefit.

5 Instructions

Solve the problems below. Refer to relevant lecture slides for axioms, inference rules, *etc*.

6 Problems

1. Fill in the blanks in the derivation for the type judgment $\emptyset \vdash \text{fun x} \rightarrow \text{(+ x) 1} : \text{int} \rightarrow \text{int}$.

$$
\dfrac{\dfrac{\dfrac{}{\emptyset[\mathrm{x} : \mathrm{int}] \vdash + : \underline{\qquad\qquad}} \qquad \dfrac{}{\emptyset[\mathrm{x} : \mathrm{int}] \vdash \mathrm{x} : \underline{\qquad\qquad}}}{\emptyset[\mathrm{x} : \mathrm{int}] \vdash +\mathrm{x} : \underline{\qquad\qquad}} \qquad \dfrac{}{\underline{\qquad\qquad} \vdash 1 : \mathrm{int}}}{\dfrac{\emptyset[\mathrm{x} : \mathrm{int}] \vdash (+\mathrm{x})\ 1 : \underline{\qquad\qquad}}{\emptyset \vdash \mathrm{fun\ x} \rightarrow (+\mathrm{x})\ 1 : \mathrm{int} \rightarrow \mathrm{int}}}
$$

2. Write the full proof tree for the type judgment below. You can use the abbreviation Γ_f for $\emptyset[\mathrm{f} : (\mathrm{int} \rightarrow \mathrm{int}) \rightarrow \mathrm{int}]$.

$$
\dfrac{}{\emptyset \vdash \mathrm{fun\ f} \rightarrow \mathrm{f}\ (+1) : ((\mathrm{int} \rightarrow \mathrm{int}) \rightarrow \mathrm{int}) \rightarrow \mathrm{int}}
$$

3. The proof tree for the evaluation judgment (fun f $\rightarrow$ f (f 1))(fun y $\rightarrow$ y * 2) $\Downarrow$ 4 begins with:

$$
\dfrac{\dfrac{}{(\mathrm{fun\ f} \rightarrow \mathrm{f}\ (\mathrm{f}\ 1)) \Downarrow (\mathrm{fun\ f} \rightarrow \mathrm{f}\ (\mathrm{f}\ 1))} \qquad \dfrac{}{(\mathrm{fun\ y} \rightarrow \mathrm{y} * 2) \Downarrow (\mathrm{fun\ y} \rightarrow \mathrm{y} * 2)} \qquad \dfrac{?}{(\mathrm{fun\ y} \rightarrow \mathrm{y} * 2)((\mathrm{fun\ y} \rightarrow \mathrm{y} * 2)\ 1) \Downarrow 4}}{(\mathrm{fun\ f} \rightarrow \mathrm{f}\ (\mathrm{f}\ 1))\ (\mathrm{fun\ y} \rightarrow \mathrm{y} * 2) \Downarrow 4}
$$

Write the proof tree that should be filled in for ? to complete the derivation. You can break it into sections if it gets too wide to fit on a page.

4. Using the typing rules for $\mathrm{T}_{\mathrm{OCaml}}$, give the derivation tree for the judgment

$$
\dfrac{}{\emptyset \vdash \mathrm{let\ f} = \mathrm{fun\ x} \rightarrow \mathrm{x\ in\ (f\ f)}\ 1 : \mathrm{int}}
$$

5. Using the evaluation rules for $\mathrm{OS}_{\mathrm{clo}}$, give the derivation tree for the judgment

$$
\dfrac{}{\emptyset, \emptyset \vdash (\mathrm{fun\ f} \rightarrow \mathrm{f}\ (\mathrm{f}\ 2))\ (\mathrm{fun\ y} \rightarrow \mathrm{y} + 1) \Downarrow 4}
$$

6. Recall that $\mathrm{OS}_{\mathrm{subst}}$ and $\mathrm{OS}_{\mathrm{clo}}$ are evaluation models for the same language where the former uses substitution and the latter uses closures. $\mathrm{OS}_{\mathrm{state}}$ extends $\mathrm{OS}_{\mathrm{clo}}$ with state to handle references, dereferencing and assignment. Let $\mathrm{OS}_{\mathrm{ss}}$ be the set of evaluation rules for the same language that still has state, but uses substitution instead of closures. Give the definition of the Application rule in $\mathrm{OS}_{\mathrm{ss}}$. (Technically, this requires that locations are considered as expressions in order for a substitution to be well-defined. You can assume that this extension has been made.)

   Below are the definitions of the (δ) and (Abstr) rules for your reference.

$$(\delta)\frac{\sigma \vdash e_1 \Downarrow v_1, \sigma_1 \qquad \sigma_1 \vdash e_2 \Downarrow v_2, \sigma_2 \qquad v = v_1 \oplus v_2}{\sigma \vdash e_1 \oplus e_2 \Downarrow v, \sigma_2}$$

$$(\text{Abstr})\frac{}{\sigma \vdash (\text{fun x} \to e) \Downarrow (\text{fun x} \to e), \sigma}$$

   Here is the outline of the (App) rule. Fill in the blanks.

$$(\text{App})\frac{\_\_ \vdash e_1 \Downarrow \_\_\_\_\_\_\_\_\_\_, \_\_ \qquad \_\_ \vdash e_2 \Downarrow \_\_\_\_, \_\_ \qquad \_\_ \vdash \_\_\_\_\_\_\_\_\_\_ \Downarrow \_\_\_\_, \_\_}{\sigma \vdash e_1 e_2 \Downarrow v, \_\_\_\_}$$

7. On slide 19 of Lecture 23 (Hoare logic), a proof of the absolute value code is almost completely given. Complete it.

8. Write the proof tree for the gcd algorithm on slide 23 of the same lecture. Hint: the invariant of the loop is "gcd(a,b) = gcd(a0,b0)".

9. The following program calculates the number of positive elements in b:

```
c := 0; a := b;
while (a != []) {
    if (hd a > 0)
        then c := c+1;
    a := tl a;
}
```

   (a) Give a loop invariant that could be used to prove the partial correctness of the program.

   (b) Give a function Φ that could be used to prove the termination of the program.

10. Write the λ-calculus version of the and and or operators. You will need the following definitions:

$$\begin{gathered}\mathrm{True} \equiv \lambda \mathrm{x}.\lambda \mathrm{y}.\mathrm{x} \\ \mathrm{False} \equiv \lambda \mathrm{x}.\lambda \mathrm{y}.\mathrm{y} \\ \mathrm{if\_then\_else\_} \equiv \lambda \mathrm{b\ c\ d.b\ c\ d}\end{gathered}$$

11. Using the definition of plus for Church numerals, show the result of adding the Church numeral 2 to itself—show all β-reductions, and use lazy evaluation order. You will need the following definitions:

$$\begin{gathered}2 \equiv \lambda \mathrm{f\ x.f\ (f\ x)} \\ \mathrm{f} \circ \mathrm{g} \equiv \lambda \mathrm{x.f\ (g\ x)}\ \mathrm{i} + \mathrm{j} \equiv \lambda \mathrm{f.(i\ f)} \circ \mathrm{(j\ f)}\end{gathered}$$