HW 10 – Type Checking and Evaluation

CS 421 – Spring 2008

Revision 1.0

Assigned April 10, 2008
Due April 17, 2008, submit in class
Extension due April 17, 5:00 pm, 20% penalty

Guideline:

- For problems 1 and 2, write your answers in the blanks right on page 1 and 2. For other problems, use the templates provided starting on page 3.
- If you're submitting late, hand in your homework either to Prof. Kamin in room 4237 or to Baris Aktemur in room 4307, by 5:00pm on April 17, Thursday. Note that 20% penalty applies.
- Submit hard-copy only. No electronic submission.

Problem 1. We gave three versions of the dynamic semantics of OCaml in class:
(Env) Environment-based definition, 4/3, slides 10–11
(Rew) Rewriting version (β-value reduction), 4/3, slides 12-13
(Env-σ) Environment, with state, 4/8, slides 4-5. (Note that we have added a rule for sequencing (semicolon) that wasn't in the original notes.)

For each one, give a semantics for the pair operation and the `fst` and `snd` operations. Assume an expression (e_1, e_2) produces a value that is written (v_1, v_2) (i.e. pairs are values).

(Env)
$$\frac{\qquad\qquad}{(e_1,e_2),\rho \Downarrow (v_1,v_2)}$$

$$\frac{\qquad\qquad}{\texttt{fst}\, e,\rho \Downarrow v} \qquad \frac{\qquad\qquad}{\texttt{snd}\, e,\rho \Downarrow v}$$

(Rew)
$$\frac{\qquad\qquad}{(e_1,e_2) \Downarrow (v_1,v_2)}$$

$$\frac{\qquad\qquad}{\texttt{fst}\, e \Downarrow v} \qquad \frac{\qquad\qquad}{\texttt{snd}\, e \Downarrow v}$$

(Env-σ)
$$\frac{\qquad\qquad}{\rho,\sigma \vdash (e_1,e_2) \Downarrow (v_1,v_2),\sigma'}$$

$$\frac{\qquad\qquad}{\rho,\sigma \vdash \texttt{fst}\, e \Downarrow v,\sigma'} \qquad \frac{\qquad\qquad}{\rho,\sigma \vdash \texttt{snd}\, e \Downarrow v,\sigma'}$$

Problem 2. For each of the semantic definitions, give a rule for this version of let: $\texttt{let}\ x = e\ \texttt{and}\ x' = e'\ \texttt{in}\ e''$:

(Env)
$$\frac{}{\texttt{let}\ x{=}e\ \texttt{and}\ x'{=}e'\ \texttt{in}\ e'',\rho \Downarrow v}$$

(Rew)
$$\frac{}{\texttt{let}\ x{=}e\ \texttt{and}\ x'{=}e'\ \texttt{in}\ e'' \Downarrow v}$$

(Env-σ)
$$\frac{}{\rho,\sigma \vdash \texttt{let}\ x{=}e\ \texttt{and}\ x'{=}e'\ \texttt{in}\ e'' \Downarrow v,\sigma'}$$

For the problems below, use the templates starting on page 3. Do not write your answers on this page.

Problem 3. Prove that the expression $e = ((\texttt{fun}\ x \texttt{->}\ \texttt{fun}\ y \texttt{->}\ x + y)3)4$ returns 7, using semantic definitions Env and Rew above. That is, using Env, prove: $\emptyset, e \Downarrow 7$ (where $\emptyset$ is the empty environment), and using Rew, prove $e \Downarrow 7$.

Problem 4. Use the type system presented on 4/8 (slides 11-14) to prove that expression e from problem 3 has type int.

Problem 5. Consider these two top-level expressions:

```
(1) let memcell = let cell = ref 0
                  in fun n -> let n'= !cell in (cell := n; n');;
(2) memcell 5;;
```

(a) Use the dynamic semantic definition Env-σ to evaluate the expression `let cell= ...` from expression (1), in the empty environment and state. That is, prove

$$\emptyset,\emptyset \vdash \texttt{let cell = ref 0 in} \ldots \texttt{cell:=n;n'} \Downarrow \underline{\qquad\qquad}, \underline{\qquad\qquad}$$

(Hint: the value is a closure.)
(Reminder: Use the template on page 5 to write your answer.)

(b) Suppose the evaluation in step (a) returns a value v and changes the state to σ. Then give the evaluation of expression (2) in the environment $\rho_0 = \{\texttt{memcell} \rightarrow v\}$ and state σ.

Problem 6. Use the polymorphic type system from 4/10 to infer:

$$\emptyset \vdash \texttt{let}\ i = \texttt{fun}\ x \texttt{->} (x, x)\ \texttt{in}\ (i\ 3,\ i\ \text{“3”}) : (int * int) * (string * string)$$

Problem 3.a. Env

You may use the following abbreviations to save space in your proof tree.

$e = ((\texttt{fun}\ x \texttt{->} \texttt{fun}\ y \texttt{->} x + y)3)4$

$e' = \texttt{fun}\ x \texttt{->} \texttt{fun}\ y \texttt{->} x + y$

$e'' = \texttt{fun}\ y \texttt{->} x + y$

$\rho_x = \{x \to 3\}$

$\rho_{xy} = \{x \to 3, y \to 4\}$

$\emptyset, 3 \Downarrow 3$

$\emptyset, e'\ 3 \Downarrow \langle y, x + y, \rho_x \rangle$

$\emptyset, e \Downarrow 7$

Problem 3.b. Rew

You may use the following abbreviation in addition to those above to save space in your proof tree.

$e_3 = \texttt{fun}\ y \texttt{->} 3 + y$

$e \Downarrow 7$

Problem 4

$$\emptyset \vdash ((\texttt{fun}\ x \texttt{->} \texttt{fun}\ y \texttt{->} x + y)3) \ :$$

$$\emptyset \vdash ((\texttt{fun}\ x \texttt{->} \texttt{fun}\ y \texttt{->} x + y)3)4 \ : \ int$$

Problem 5.a

You may use the following abbreviations to save space in the proof tree.

$\sigma = \{m \rightarrow 0\}$
$\rho = \{\texttt{cell} \rightarrow m\}$

$$\dfrac{\dfrac{\dfrac{}{\quad\quad\quad}}{\emptyset, \emptyset \vdash \texttt{ref 0} \Downarrow m, \sigma} \qquad \dfrac{}{\quad\quad\quad\quad\quad\quad}}{\emptyset, \emptyset \vdash \texttt{let cell=ref 0 in} \ldots \Downarrow \underline{\qquad\qquad\qquad\qquad}, \underline{\qquad\qquad}}$$

Problem 5.b

You may use the following abbreviations in addition to those in part (a) to save space in the proof tree.

$\rho_0 = \{\texttt{memcell} \rightarrow \text{value from part a}\}$
$\rho_1 = \rho_0[\texttt{n} \rightarrow 5]$
$\rho_2 = \rho_1[\texttt{n}' \rightarrow 0]$
$\sigma_1 = \{m \rightarrow 5\}$

TREE-A

$\rho_1, \sigma \vdash \texttt{let n}' = \texttt{!cell in} \ldots \Downarrow$ __, __

$\rho_0, \sigma \vdash \texttt{memcell 5} \Downarrow 0, \sigma_1$

where TREE-A is:

____, ____ $\vdash \texttt{cell:=n;n}' \Downarrow$ ____, ____ Problem 6

You may use the following abbreviation to save space in the proof tree.

$\rho = \{i \rightarrow$ ____________________ $\}$

$$\dfrac{\emptyset \vdash \texttt{fun}\ x \texttt{ -> } (x, x)\ :\ \alpha \rightarrow \alpha * \alpha \qquad \text{TREE-B}}{\emptyset \vdash \texttt{let}\ i = \texttt{fun}\ x \texttt{ -> } (x, x)\ \texttt{in}\ (i\ 3, i\ \text{“3”})\ :\ (int * int) * (string * string)}$$

where TREE-B is:

$$\dfrac{\rho \vdash 3\ :\ int \qquad \rho \vdash \text{“3”}\ :\ string}{\rho \vdash (i\ 3, i\ \text{“3”})\ :\ (int * int) * (string * string)}$$