HW 12 – Proof Systems

CS 421 – Spring 2009 Revision 1.1

Assigned Thursday, April 16, 2009 **Due** Wednesday, April 22, in class **Extension** 48 hours (20% penalty) **Total points** 50

1 Change Log

- **1.1** Hints and proof tree layouts added.
- 1.0 Initial Release.

2 Overview

After completing this MP, you should have a better understanding of proof systems for OCaml typing and operational semantics.

3 Collaboration

Collaboration is NOT allowed on this assignment.

4 Instructions

Submit in hard-copy at the beginning of class.

5 Problems (50 pts)

For problems 1-4, use the proof tree layouts given at the end.

1. (10 pts) Using the typing rules for T_{OCaml}, give the derivation tree for the judgment

$$\emptyset \vdash \mathtt{let} \; \mathtt{rec} \; \mathtt{x} = \mathtt{fun} \; \mathtt{y} \to \mathtt{x}(\mathtt{y}) + \mathtt{1} \; \mathtt{in} \; \mathtt{x}(\mathtt{1}) : \mathtt{int}$$

2. (10 pts) Using the typing rules for T_{OCaml}, give the derivation tree for the judgment

$$\emptyset \vdash \mathtt{let} \ \mathtt{f} = \mathtt{fun} \ \mathtt{x} \to \mathtt{x} \ \mathtt{in} \ (\mathtt{f} \ \mathtt{f})\mathtt{1} : \mathtt{int}$$

3. (10 pts) Using the evaluation rules for OS_{clo}, give the derivation tree for the judgment

$$\emptyset$$
, (fun f \rightarrow f (f 2))(fun y \rightarrow y + 1) \Downarrow 4

4. (10 pts) Using the evaluation rules for OS_{state}, give the derivation tree for the judgment

$$\emptyset, \emptyset \vdash (\text{fun } x \rightarrow x := !x + 1)(\text{ref } 0) \Downarrow (), \{\ell \mapsto 1\}$$

5. (10 pts) Recall that OS_{subst} and OS_{clo} are evaluation models for the same language where the former uses substitution and the latter uses closures. OS_{state} extends OS_{clo} with state to handle references, dereferencing and assignment. Let OS_{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 OS_{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.

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

(Abstr)
$$\sigma \vdash (\operatorname{fun} \mathbf{x} \to e) \Downarrow (\operatorname{fun} \mathbf{x} \to e), \sigma$$

Fill in the blanks in the (App) rule below.

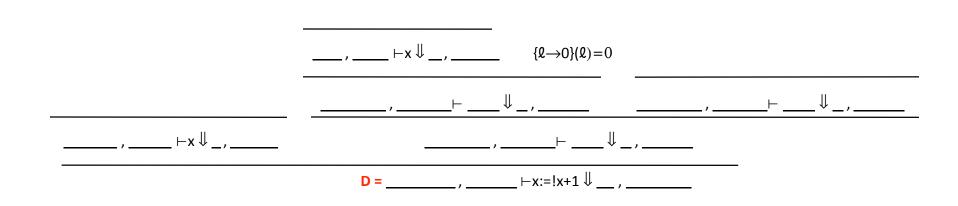
$\frac{\{x:\alpha\rightarrow \text{int, }y:\alpha\}\vdash x:\alpha\rightarrow \text{int}}{\{x:\alpha\rightarrow \text{int, }y:\alpha\}\vdash x(y):\text{int}}$		$\overline{\{x:\alpha\rightarrow int,y:\alpha\}}\vdash y:\alpha$						
		$\{x:\alpha \rightarrow \text{int, } y:\alpha\} \vdash 1: \text{ int}$						
	{	}} ⊢x(y)+1:	-		{	}}⊢x:	{	}}⊢1: int
	{	} 	_:	- -	{		}⊢x(1):	-
		Ø⊢let rec	x = fun y →>	x(y)+1 in x(1) : int	t			
2)								
-,								
							_	
		{}	<u>⊢</u> :		{}	}\:		
	⊢x:			{	}}⊢	:	{	}}⊢1: int
	⊢fun x →x :			{}		}}⊢:		

 \emptyset Het f = fun x \rightarrow x in (f f)1 : int

 \emptyset ,(fun f \rightarrow f(f 2))(fun y \rightarrow y+1) \downarrow 4

To reduce notational clutter, use η as an abbreviation for the environment $\{f: \langle fun\ y \rightarrow y+1 \rangle, \varnothing \}$.

4)



 \varnothing , \varnothing \vdash (fun x \rightarrow x:=!x+1)(ref 0) \Downarrow (),{ ℓ \rightarrow 1}