

# CMSC 631 – Homework 5

Mark Daly <mdaly@cs.umd.edu>

April 27, 2009

## Problem 4

**Lemma 1 (Progress)** *If  $A \vdash \langle s, e \rangle : t$  and  $e$  is a closed term (no free variables, though it may contain locations  $\ell$ ) then either  $e$  is a value or there exist  $s'$  and  $e'$  such that  $\langle s, e \rangle \rightarrow \langle s', e' \rangle$ .*

**Proof:** by induction on  $e$ .

- $x$  : By assumption ( $e$  is a closed term, and so no free variables can exist).
- $n$  :  $n$  is a value.
- $\lambda x : t.e$  :  $\lambda x : t.e$  is a value.
- $\ell$  :  $\ell$  is a value.
- $e_1 e_2$ 
  - If  $e_1$  is not a value, then by the IH it evaluates to some  $e'_1$ , and we can take a step  $\langle s, e_1 e_2 \rangle \rightarrow \langle s', e'_1 e_2 \rangle$  by C-APP-LEFT.
  - If  $e_2$  is not a value, then (similarly) by the IH it evaluates to some  $e'_2$ , and we can take a step  $\langle s, e_1 e_2 \rangle \rightarrow \langle s', e_1 e'_2 \rangle$  by C-APP-RIGHT.
  - Otherwise,  $e_1$  and  $e_2$  are values. Assuming  $A \vdash e_1 e_2$  is well typed, we know by inspection of the type rules (by T-APP) that  $A \vdash e_1 : t \rightarrow t'$  and  $A \vdash e_2 : t$ , and that (by T-LAM)  $\lambda x : t.e_x$  is the only value that can take type  $t \rightarrow t'$ ;  $e_2$  can be any value  $v$ . As such, we can take a step to  $e' = e_x[x \mapsto v]$ ,  $s' = s$  by S-APP.
- $ref\ e$ 
  - If  $e$  is not a value, then by the IH it evaluates to some  $e'$ , and we can take a step  $\langle s, ref\ e \rangle \rightarrow \langle s', ref\ e' \rangle$  by C-REF.
  - Otherwise,  $e$  is a value  $v$ . We can select a “fresh”  $\ell$  (i.e.  $\ell \notin dom(s)$ ) to represent the location of  $v$  in state  $s$  and expand  $s$  to include this new  $\ell$  (with value  $v$ ), creating  $s'$ . Given this  $\ell$  and  $s'$ , we can take a step to  $e' = \ell$ ,  $s' = s[\ell \mapsto v]$  by S-REF.
- $!e$ 
  - If  $e$  is not a value, then by the IH it evaluates to some  $e'$ , and we can take a step  $\langle s, !e \rangle \rightarrow \langle s', !(e') \rangle$  by C-DEREF.
  - Otherwise  $e$  is a value. Assuming  $A \vdash \langle s, !e \rangle$  is well typed, we know by inspection of the type rules (T-REF) that  $A \vdash e : t\ ref$ , and that (by T-LOC),  $\ell \mid \ell \in dom(A)$  is the only value that can take type  $t\ ref$ . By compatibility (again stemming from the fact that  $A \vdash \langle s, !e \rangle$  is well typed), we know that  $dom(A) = dom(s)$ , and so  $\ell \in dom(A) \Rightarrow \ell \in dom(s)$ . Therefore,  $e = \ell$ ,  $\ell \in dom(s)$ , and so we can take a step to  $e' = s(\ell)$ ,  $s' = s$  by S-DEREF.

$$\begin{aligned}
e &::= x \mid n \mid \lambda x : t.e \mid \ell \mid e e \mid \text{ref } e \mid !e \mid e := e \\
t &::= \text{int} \mid t \rightarrow t \mid t \text{ ref} \\
v &::= n \mid \lambda x : t.e \mid \ell \\
A &::= \emptyset \mid (A, x : t) \mid (A, \ell : t)
\end{aligned}$$

Figure 1: Grammar definition for  $\lambda$ -calculus with references

$$\begin{array}{c}
\text{S-APP} \qquad \qquad \qquad \text{C-APP-LEFT} \qquad \qquad \qquad \text{C-APP-RIGHT} \\
\frac{}{\langle s, (\lambda x.e) v \rangle \rightarrow \langle s, e[x \mapsto v] \rangle} \quad \frac{\langle s, e_1 \rangle \rightarrow \langle s', e'_1 \rangle}{\langle s, e_1 e_2 \rangle \rightarrow \langle s', e'_1 e_2 \rangle} \quad \frac{\langle s, e_2 \rangle \rightarrow \langle s', e'_2 \rangle}{\langle s, v e_2 \rangle \rightarrow \langle s', v e'_2 \rangle} \\
\\
\text{S-REF} \qquad \qquad \qquad \text{C-REF} \\
\frac{\ell \notin \text{dom}(s)}{\langle s, \text{ref } v \rangle \rightarrow \langle s[\ell \mapsto v], \ell \rangle} \quad \frac{\langle s, e \rangle \rightarrow \langle s', e' \rangle}{\langle s, \text{ref } e \rangle \rightarrow \langle s', \text{ref } e' \rangle} \\
\\
\text{S-DEREF} \qquad \qquad \qquad \text{C-DEREF} \\
\frac{\ell \in \text{dom}(s)}{\langle s, !\ell \rangle \rightarrow \langle s, s(\ell) \rangle} \quad \frac{\langle s, e \rangle \rightarrow \langle s', e' \rangle}{\langle s, !e \rangle \rightarrow \langle s', !(e') \rangle} \\
\\
\text{S-ASSIGN} \qquad \qquad \qquad \text{C-ASSIGN-LEFT} \qquad \qquad \qquad \text{C-ASSIGN-RIGHT} \\
\frac{\ell \in \text{dom}(s)}{\langle s, \ell := v \rangle \rightarrow \langle s[\ell \mapsto v], v \rangle} \quad \frac{\langle s, e_1 \rangle \rightarrow \langle s', e'_1 \rangle}{\langle s, e_1 := e_2 \rangle \rightarrow \langle s', e'_1 := e_2 \rangle} \quad \frac{\langle s, e_2 \rangle \rightarrow \langle s', e'_2 \rangle}{\langle s, \ell := e_2 \rangle \rightarrow \langle s', \ell := e'_2 \rangle}
\end{array}$$

Figure 2: (Small-step) Operational semantics with stores

$$\begin{array}{c}
\text{T-VAR} \qquad \text{T-LOC} \qquad \text{T-INT} \qquad \text{T-LAM} \qquad \text{T-APP} \\
\frac{x \in \text{dom}(A)}{A \vdash x : A(x)} \quad \frac{\ell \in \text{dom}(A)}{A \vdash \ell : A(\ell)} \quad \frac{}{A \vdash n : \text{int}} \quad \frac{A, x : t \vdash e : t'}{A \vdash \lambda x : t.e : t \rightarrow t'} \quad \frac{A \vdash e_1 : t \rightarrow t' \quad A \vdash e_2 : t}{A \vdash e_1 e_2 : t'} \\
\\
\text{T-REF} \qquad \text{T-DEREF} \qquad \text{T-ASSIGN} \\
\frac{A \vdash e : t}{A \vdash \text{ref } e : t \text{ ref}} \quad \frac{A \vdash e : t \text{ ref}}{A \vdash !e : t} \quad \frac{A \vdash e_1 : t \text{ ref} \quad A \vdash e_2 : t}{A \vdash e_1 := e_2 : t}
\end{array}$$

Figure 3: Type rules with references

- $e_1 := e_2$

- If  $e_1$  is not a value, then by the IH it evaluates to some  $e'_1$ , and we can take a step  $\langle s, e_1 := e_2 \rangle \rightarrow \langle s', e'_1 := e_2 \rangle$  by C-ASSIGN-LEFT.
- If  $e_2$  is not a value, then (similarly) by the IH it evaluates to some  $e'_2$ , and we can take a step  $\langle s, e_1 := e_2 \rangle \rightarrow \langle s', e_1 := e'_2 \rangle$  by C-ASSIGN-RIGHT.
- Otherwise,  $e_1$  and  $e_2$  are values. Assuming  $A \vdash \langle s, e_1 := e_2 \rangle$  is well typed, by inspection of the type rules (T-ASSIGN) we know that  $e_1 : t \text{ ref}$  and  $e_2 : t$ . As in the value case for  $!e$ , the type rules and compatibility tell us that  $\ell \mid \ell \in \text{dom}(A)$  is the only value that can take type  $t \text{ ref}$ , and so  $e_1 = \ell, \ell \in \text{dom}(s)$ ;  $e_2$  can be any value  $v$ . Therefore, we can take a step to  $e' = v, s' = s[\ell \mapsto v]$  by S-ASSIGN.

**Lemma 2 (Preservation)** *If  $A \vdash \langle s, e \rangle : t$  and  $\langle s, e \rangle \rightarrow \langle s', e' \rangle$  then there exist  $A', s'$ , and  $e'$  such that  $A' \vdash \langle s', e' \rangle : t$  and  $A' \upharpoonright_{\text{dom}(A)} = A$ .*

**Proof:** by induction on  $e \rightarrow e'$ .

- S-APP, C-APP-LEFT, C-APP-RIGHT

We assume that  $e = e_1 e_2$  and that  $A \vdash \langle s, e \rangle : t$  is well typed.

- If  $e_1 \rightarrow e'_1$ , then by the IH  $A \vdash \langle s, e'_1 \rangle : t' \rightarrow t$ , and so  $A' \vdash \langle s', e'_1 e_2 \rangle : t$ .
- Similarly, if  $e_2 \rightarrow e'_2$ , then by the IH  $A \vdash \langle s, e_2 \rangle : t'$ , and so  $A' \vdash \langle s', e_1 e'_2 \rangle : t$ .
- Otherwise, both  $e_1$  and  $e_2$  are values. We know by inspection of the type rules (by T-APP) that  $A \vdash e_1 : t' \rightarrow t$  and  $A \vdash e_2 : t'$ , and that (by T-LAM)  $\lambda x : t'. e$  is the only value that can take type  $t' \rightarrow t$ ; as such,  $e_1$  must be of the form  $\lambda x : t'. e_x$ .  $e_2$  can be any value  $v : t$ . We have  $A, x : t' \vdash \langle s, e_x \rangle : t$  and  $A \vdash \langle s, v \rangle$ , and by lemma 3 (substitution), we have  $A \vdash \langle s, e_x[x \mapsto v] \rangle : t$ . Thus, preservation holds for this case.

- S-REF, C-REF

- If  $e \rightarrow e'$  and  $A \vdash \langle s, \text{ref } e \rangle : t$ , then by the IH  $A' \vdash \langle s', \text{ref } e' \rangle : t$ .
- Otherwise,  $e$  is a value  $v$ . By inspection of the type rules (T-REF), we know that  $A \vdash \text{ref } e : t \text{ ref}$ . Selecting a “fresh”  $\ell$  (i.e.  $\ell \notin \text{dom}(s)$ ), we wish to provide  $A', s'$ , and  $e'$  that satisfy  $A' \vdash \langle s', e' \rangle : t$  and  $A' \upharpoonright_{\text{dom}(A)} = A$ . Using S-REF, we (carefully) choose  $A' = (A, \ell : t \text{ ref})$ ,  $s' = s[\ell \mapsto v]$ , and  $e' = \ell$ . By lemmas 4 and 5, we have  $A' \sim s'$  and  $A' \upharpoonright_{\text{dom}(A)} = A$ ; for  $A' \vdash \langle s', e' \rangle : t$ , we need now only prove that  $A' \vdash e' : t$ . However, this follows directly from our selection of  $A'$  and  $e'$ :  $A, \ell : t \text{ ref} \vdash \ell : t \text{ ref}$ . Thus,  $A' \vdash \langle s', e' \rangle : t$  satisfies configuration typing and so is well typed, and preservation holds for this case.

- S-DEREF, C-DEREF

- If  $e \rightarrow e'$  and  $A \vdash \langle s, !e \rangle : t$ , then by the IH  $A' \vdash \langle s', !(e') \rangle : t$ .
- Otherwise,  $e$  is a value. By inspection of the type rules (T-DEREF), we know that because  $A \vdash !e : t$ ,  $A \vdash e : t \text{ ref}$ . Also by the typing rules (T-LOC), the only value that can assume such a type is  $\ell \mid \ell \in \text{dom}(A)$ . By configuration typing,  $\ell \in \text{dom}(A) \Rightarrow \ell \in \text{dom}(s)$  and  $A \vdash s(\ell) : t$ . Therefore, we use S-DEREF to choose  $A' = A, s = s$ , and  $e' = s(\ell)$ , and preservation holds for this case.

- S-ASSIGN, C-ASSIGN-LEFT, C-ASSIGN-RIGHT

We assume that  $e = e_1 := e_2$ .

- If  $e_1 \rightarrow e'_1$ , then by the IH  $A \vdash \langle s, e'_1 \rangle : t \text{ ref}$ , and so  $A' \vdash \langle s', e'_1 := e_2 \rangle : t$ .
- Similarly, if  $e_2 \rightarrow e'_2$ , then by the IH  $A \vdash \langle s, e_2 \rangle : t$ , and so  $A' \vdash \langle s', e_1 := e'_2 \rangle : t$ .