
CMSC 22100/32100: Programming Languages

Homework 4

M. Blume

Due: October 28, 2008

1. Consider the following version of the Simply Typed λ -Calculus, where we only have one base type **nat**:

$$\begin{array}{ll} \tau ::= \mathbf{nat} \mid \tau \rightarrow \tau & \text{types} \\ v ::= x \mid n \mid \lambda x : \tau. e & \text{values} \\ e ::= v \mid \mathbf{succ}(e) \mid \mathbf{pred}(e) \mid \mathbf{if0} \ e \ \mathbf{then} \ e \ \mathbf{else} \ e \mid e \ e & \text{expressions} \end{array}$$

The typing rules and evaluation rules are the same as we have discussed in class—with the minor exception of **if0**. Here are the four rules of interest for **if0**—one typing rule and three rules for the small-step semantics:

$$\frac{\Gamma \vdash e_1 : \mathbf{nat} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \mathbf{if0} \ e_1 \ \mathbf{then} \ e_2 \ \mathbf{else} \ e_3 : \tau}$$
$$\frac{e_1 \mapsto^1 e'_1}{\mathbf{if0} \ e_1 \ \mathbf{then} \ e_2 \ \mathbf{else} \ e_3 \mapsto^1 \mathbf{if0} \ e'_1 \ \mathbf{then} \ e_2 \ \mathbf{else} \ e_3}$$
$$\frac{}{\mathbf{if0} \ 0 \ \mathbf{then} \ e_2 \ \mathbf{else} \ e_3 \mapsto^1 e_2} \quad \frac{n \neq 0}{\mathbf{if0} \ n \ \mathbf{then} \ e_2 \ \mathbf{else} \ e_3 \mapsto^1 e_3}$$

Questions:

- (a) Define the substitution function $\{v/x\}e$ where v is assumed to be closed.
 - (b) Formally state the substitution lemma for this language. (Informally: Substituting a value of the “right” type into a well-typed expression keeps that expression well-typed.)
 - (c) Prove your substitution lemma.
2. Canonical forms and progress:
- (a) For the same language as in question 1, state and prove the *canonical forms lemma*. (See Lemma 10.2 on page 61 of the textbook for reference. Your proof should be spelled out in more detail than the one-liner in the textbook.)
 - (b) State the progress lemma for this language.

(c) Show the proof of the progress lemma for the case of application ($e_1 e_2$).

3. Consider the following alternative language design:

$$\begin{array}{lll} \tau & ::= & \mathbf{nat} \mid \tau \rightarrow \tau & \text{types} \\ v & ::= & x \mid n \mid \lambda x : \tau. e \mid \mathbf{pred} \mid \mathbf{succ} & \text{values} \\ e & ::= & v \mid \mathbf{if0} \ e \ \mathbf{then} \ e \ \mathbf{else} \ e \mid e \ e & \text{expressions} \end{array}$$

Here the predecessor and successor operations do not have their own syntax. Instead, there are two “built-in” constants by the names of **pred** and **succ**. These constants denote functions that—when applied (using the usual application form $e_1 e_2$)—produce the predecessor or successor of their arguments.

Questions:

- (a) Show the typing rules for this language.
- (b) Show the small-step evaluation rules for the language. (You only have to show the rules for function application. Make sure you show *all* the rules that deal with function application.)
- (c) State the canonical forms lemma for this language. (You do not need to prove it.)
- (d) Explain the proof of the function application case of the progress lemma for this language. In particular, contrast this with your answer to question 2c.