
CMSC 22100/32100: Programming Languages

Homework 5

M. Blume

Due: November 4, 2008

Election day — if you are eligible, don't forget to vote!

Recall the definition of MinML from the textbook. Here is the notation that we shall use:

variables : $x ::= \dots$
numbers : $n ::= 0, 1, -1, 2, -2, \dots$
operations : $\oplus ::= + \mid * \mid - \mid = \mid <$
types : $\tau ::= \mathbf{int} \mid \mathbf{bool} \mid \tau \rightarrow \tau$
expressions : $e ::= x \mid n \mid \mathbf{true} \mid \mathbf{false} \mid$ variables and constants
 $e \oplus e \mid$ primitive operations
 $\mathbf{if } e \mathbf{ then } e \mathbf{ else } e \mid$ conditional expression
 $\mathbf{fun } f(x : \tau) : \tau \mathbf{ is } e \mid$ recursive function abstraction
 $e e$ application

The typing rules are:

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \text{VAR} \quad \frac{}{\Gamma \vdash n : \mathbf{int}} \text{NUM} \quad \frac{}{\Gamma \vdash \mathbf{true} : \mathbf{bool}} \text{TRUE} \quad \frac{}{\Gamma \vdash \mathbf{false} : \mathbf{bool}} \text{FALSE}$$
$$\frac{\oplus \in \{+, *, -\} \quad \Gamma \vdash e_1 : \mathbf{int} \quad \Gamma \vdash e_2 : \mathbf{int}}{\Gamma \vdash e_1 \oplus e_2 : \mathbf{int}} \text{ARITH}$$
$$\frac{\oplus \in \{=, <\}}{\Gamma \vdash e_1 \oplus e_2 : \mathbf{bool}} \text{CMP}$$
$$\frac{\Gamma \vdash e_1 : \mathbf{bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \mathbf{if } e_1 \mathbf{ then } e_2 \mathbf{ else } e_3 : \tau} \text{IF} \quad \frac{\Gamma[f \mapsto \tau_1 \rightarrow \tau_2, x \mapsto \tau_1] \vdash e : \tau_2}{\Gamma \vdash \mathbf{fun } f(x : \tau_1) : \tau_2 \mathbf{ is } e : \tau_1 \rightarrow \tau_2} \text{FUN}$$
$$\frac{\Gamma \vdash e_1 : \tau_2 \rightarrow \tau \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1 e_2 : \tau} \text{APP}$$

1. Suppose we added a **let**-form, in the style of the “arithmetic plus **let**” language discussed earlier in the course:

$e ::= \dots \mid \mathbf{let } x = e \mathbf{ in } e$

- (a) Give the typing rule for the **let**-form.

- (b) Consider a small-step structural operational semantics for the language, *i.e.*, a set of rules that lets us derive judgments of the form $e \mapsto^1 e'$ indicating that expression e “steps to” expression e' . Show the rule(s) that must be added to handle the new **let**-form.
 - (c) Explain the part of a proof of *progress* that deals with **let**.
 - (d) Explain the part of a proof of *preservation* that deals with **let**.
 - (e) How does the C-machine (section 11.1 in the Harper text) have to be extended to handle **let**? Show any additional frames and any additional transitions.
2. Do the cases in the proofs of Theorem 11.1 (p. 73 in the Harper text), parts 1 and 2, involving **if**-expressions.