

# Homework # 13

## due Tuesday, May 3rd, 3:30 PM

### 1 Small-Step Semantics

A small-step semantics work by returning the program with one step of evaluation done. Function calls and let's work by substituting the value for the variable into the body. The output is always an AST. To evaluate fully, one applies the small-step rules until no other rule is possible, which happens when there is a type error, or we end up with a *value* such as `const(42)`. There are two kinds of values for Language Three: constants and functions.

The small-step semantics for Language Three is as follows, where  $e$  means any expression, and  $v$  means the expression must be a value, and  $n$  means the expression must be an integer constant.

$$\begin{array}{c}
 \frac{e_1 \rightarrow e'_1}{e_1 + e_2 \rightarrow e'_1 + e_2} \qquad \frac{e_2 \rightarrow e'_2}{v_1 + e_2 \rightarrow v_1 + e'_2} \qquad \frac{}{n_1 + n_2 \rightarrow n_1 + n_2} \\
 \\
 \frac{e_1 \rightarrow e'_1}{e_1 * e_2 \rightarrow e'_1 * e_2} \qquad \frac{e_2 \rightarrow e'_2}{v_1 * e_2 \rightarrow v_1 * e'_2} \qquad \frac{}{n_1 * n_2 \rightarrow n_1 n_2} \\
 \\
 \frac{e_1 \rightarrow e'_1}{\text{let } x=e_1 \text{ in } e_2 \text{ end} \rightarrow \text{let } x=e'_1 \text{ in } e_2 \text{ end}} \qquad \frac{}{\text{let } x=v_1 \text{ in } e_2 \text{ end} \rightarrow e_2[x \rightarrow v_1]} \\
 \\
 \frac{e_1 \rightarrow e'_1}{e_1 e_2 \rightarrow e'_1 e_2} \qquad \frac{e_2 \rightarrow e'_2}{v_1 e_2 \rightarrow v_1 e'_2} \qquad \frac{}{(\text{fn } x \Rightarrow e) v \rightarrow e[x \rightarrow v]}
 \end{array}$$

In lecture, we implement these rules in Prolog.

For this Homework, you will add conditionals and recursion.

- We add the following new syntactic forms for conditionals `if  $e_0$  then  $e_1$  else  $e_2$`  (written `if(E0,E1,E2)` in Prolog),  `$e_0 < e_1$`  (written `less(E0,E1)` in Prolog), `true` and `false`.
- We add a form for recursive functions: `fun  $f$   $x \Rightarrow e$`  (written `fun(F,X,E)`).
- Add the following new evaluation rules to the Prolog implementation

$$\begin{array}{c}
 \frac{e_1 \rightarrow e'_1}{e_1 < e_2 \rightarrow e'_1 < e_2} \qquad \frac{e_2 \rightarrow e'_2}{v_1 < e_2 \rightarrow v_1 < e'_2} \qquad \frac{}{n_1 < n_2 \rightarrow \begin{cases} \text{true} & \text{if } n_1 < n_2 \\ \text{false} & \text{otherwise} \end{cases}} \\
 \\
 \frac{e_0 \rightarrow e'_0}{\text{if } e_0 \text{ then } e_1 \text{ else } e_2 \rightarrow \text{if } e'_0 \text{ then } e_1 \text{ else } e_2} \qquad \frac{}{\text{if true then } e_1 \text{ else } e_2 \rightarrow e_1} \\
 \\
 \frac{}{\text{if false then } e_1 \text{ else } e_2 \rightarrow e_2} \qquad \frac{}{\text{fun } f \ x \Rightarrow e \rightarrow \text{fn } x \Rightarrow e[f \rightarrow \text{fun } f \ x \Rightarrow e]}
 \end{array}$$

- Write a factorial program and check that it can be used to compute 4!

Put all your Prolog in `homework13.pl` in your AFS volume.