# 1 Operational Semantics

## 1.1 Environments

$E$ - a mapping from identifiers to values

$F$ - a mapping from identifiers to function definitions.

A function definition has three fields:

prog: a statement of the form $P$ (see context free grammar)

r: a statement of the form $e$ that comes after the ret keyword.

p_list: a list of identifiers that make up the parameter list of the function definition.

## 1.2 The Fundamental crumbL Statement

$$\frac{\begin{array}{c} E, F \vdash S_1 : E', F' \\ E', F' \vdash S_2 : E'', F'' \end{array}}{E, F \vdash S_1 \ S_2 : E'', F''}$$

## 1.3 Constants

$$\frac{\text{integer i}}{E, F \vdash \text{i} : \text{i}}$$

$$\frac{\text{String s}}{E, F \vdash \text{s} : \text{s}}$$

$$\frac{}{E, F \vdash \text{Nil} : \text{Nil}}$$

## 1.4 Arithmetic

$$\frac{\begin{array}{c} E, F \vdash e_1 : i_1(\text{integer}) \\ E, F \vdash e_2 : i_2(\text{integer}) \end{array}}{E, F \vdash e_1 \oplus e_2 : i_1 \oplus i_2}$$

where $\oplus \in \{+, -, *, \%\}$

$$\frac{\begin{array}{c} E, F \vdash e_1 : i_1(\text{integer}) \\ E, F \vdash e_2 : i_2(\text{integer}) \\ i_2 \neq 0 \end{array}}{E, F \vdash e_1/e_2 : i_1/i_2}$$

$$\frac{E, F \vdash e : i(\text{integer})}{E, F \vdash -e : -i}$$

$$\frac{E, F \vdash e_1 : s_1(\text{string}) \qquad E, F \vdash e_2 : s_2(\text{string})}{E, F \vdash e_1 :: e_2 : s_1 s_2}$$

### 1.5 Lists

$$\frac{E, F \vdash e_1 : v_1(\text{not a list}) \qquad E, F \vdash e_2 : v_2 \text{ (not Nil)}}{E, F \vdash e_1 @ e_2 : [v_1, v_2]}$$

$$\frac{E, F \vdash e_1 : v_1(\text{not a list}) \qquad E, F \vdash e_2 : \text{Nil}}{E, F \vdash e_1 @ e_2 : v_1}$$

$$\frac{E, F \vdash e : [v_1, v_2]}{E, F \vdash !e : v_1}$$

$$\frac{E, F \vdash e : [v_1, v_2]}{E, F \vdash \#e : v_2}$$

$$\frac{E, F \vdash e : v_1(\text{not a list})}{E, F \vdash !e : v_1}$$

$$\frac{E, F \vdash e : v_1(\text{not a list})}{E, F \vdash \#e : Nil}$$

### 1.6 Boolean Logic

$$\frac{E, F \vdash e_1 : v_1 \qquad E, F \vdash e_2 : v_2}{E, F \vdash e_1 \odot e_2 : v_1 \odot v_2}$$

where $v_1$ and $v_2$ are either both strings or both integers. If strings, comparisons are lexicgraphic.

$\odot \in \{<, >, <=, >=, ==, != \}$

Let **False** be "", 0, or Nil.

**True** is any expression that evaluates to something that is not **False**.

$$\frac{E, F \vdash e_1 : \text{True} \qquad E, F \vdash e_2 : \text{True}}{E, F \vdash e_1 \text{ and } e_2 : 1}$$

$$\frac{E, F \vdash e_1 : \text{False}}{E, F \vdash e_1 \text{ and } e_2 : 0}$$

$$\frac{\begin{array}{c} E, F \vdash e_1 : \text{True} \\ E, F \vdash e_2 : \text{False} \end{array}}{E, F \vdash e_1 \text{ and } e_2 : 0}$$

$$\frac{E, F \vdash e_1 : \text{True}}{E, F \vdash e_1 \text{ or } e_2 : 1}$$

$$\frac{\begin{array}{c} E, F \vdash e_1 : \text{False} \\ E, F \vdash e_2 : \text{False} \end{array}}{E, F \vdash e_1 \text{ or } e_2 : 0}$$

$$\frac{\begin{array}{c} E, F \vdash e_1 : \text{False} \\ E, F \vdash e_2 : \text{True} \end{array}}{E, F \vdash e_1 \text{ or } e_2 : 1}$$

$$\frac{E, F \vdash e_1 : \text{True}}{E, F \vdash \text{not } e_1 : 0}$$

$$\frac{E, F \vdash e_1 : \text{False}}{E, F \vdash \text{not } e_1 : 1}$$

$$\frac{E, F \vdash e_1 : Nil}{E, F \vdash \text{isNil } e_1 : 1}$$

$$\frac{E, F \vdash e_1 : \text{not Nil}}{E, F \vdash \text{isNil } e_1 : 0}$$

## 1.7 Conditional Statements

$$\frac{\begin{array}{c} E, F \vdash C : \text{True} \\ E, F \vdash S_1 : E', F' \end{array}}{E, F \vdash \text{if } (C) \text{ then } S_1 \text{ else } S_2 \text{ fi } : E', F'}$$

$$\frac{\begin{array}{c} E, F \vdash C : \text{False} \\ E, F \vdash S_2 : E', F' \end{array}}{E, F \vdash \text{if } (C) \text{ then } S_1 \text{ else } S_2 \text{ fi } : E', F'}$$

$$\frac{\begin{array}{c} E, F \vdash C : \text{True} \\ E, F \vdash S : E', F' \end{array}}{E, F \vdash \text{if } (C) \text{ then } S \text{ fi } : E', F'}$$

$$\frac{E, F \vdash C : \text{False}}{E, F \vdash \text{if } (C) \text{ then } S \text{ fi } : E, F}$$

$$\frac{E, F \vdash C : \text{False}}{E, F \vdash \text{while}(C) \text{ do } S \text{ ob } : E, F}$$

$$\frac{\begin{array}{c} E, F \vdash C : \text{True} \\ E, F \vdash S : E', F \\ E', F \vdash \text{while}(C) \text{ do } S \text{ ob } : E'', F \end{array}}{E, F \vdash while(C) \text{ do } S \text{ ob } : E'', F}$$

Note: Under this rule, the statement S must not change the function environment.

## 1.8 Identifiers and Functions

$$\frac{E, F \vdash e : v}{E, F \vdash id = e; : E[id \leftarrow v], F}$$

$$\frac{}{E, F \vdash \text{lazy } id = e; : E[id \leftarrow e], F}$$

$$\frac{\begin{array}{c} e = E[id] \\ E, F \vdash e : v \\ E' = E[id \leftarrow v] \end{array}}{E, F \vdash id : v, E', F}$$

$$\frac{\begin{array}{c} fentry = \{\text{prog: P, r: e, p\_list: p\_list}\} \\ F' = F[\mathbf{fname} \leftarrow fentry] \end{array}}{E, F \vdash \text{func } \mathbf{fname}(\text{p\_list}) \; P \text{ ret } e; \text{ cnuf } : E, F'}$$

$$\frac{\begin{array}{c} fentry = F[\mathbf{fname}] \\ E' = \text{apply}(fentry.\text{p\_list}, \text{ call\_list}) \\ p = fentry.\text{prog} \\ E', F \vdash p : E'' \\ E'', F \vdash fentry.\text{r} : v \end{array}}{E, F \vdash \mathbf{fname}(\text{call\_list}) : v}$$

Note: apply is just an operational semantics subroutine to construct a new environment for a called function, and is not useable from source code.

$$\frac{\begin{array}{c} E, F \vdash \text{p\_list} = [p_1, R_1] \\ E, F \vdash \text{call\_list} = [e_1, R_2] \\ E, F \vdash e_1 : v_1 \\ E, F \vdash \text{apply}(R_1, R_2) : E' \\ E'' = E'[p_1 \leftarrow v_1] \end{array}}{E, F \vdash \text{apply}(\text{p\_list}, \text{call\_list}) : E''}$$

$$\frac{\begin{array}{c} E, F \vdash \text{p\_list} = [\text{lazy } p_1, R_1] \\ E, F \vdash \text{call\_list} = [e_1, R_2] \\ E, F \vdash \text{apply}(R_1, R_2) : E' \\ E'' = E'[p_1 \leftarrow e_1] \end{array}}{E, F \vdash \text{apply}(\text{p\_list}, \text{call\_list}) : E''}$$

$$\frac{}{\text{apply}(\epsilon, \epsilon) : \emptyset}$$

## 1.9 I/O

$$\frac{E, F \vdash e : v}{E, F \vdash \text{print}(e); : E, F, \text{ print out } v}$$

$$\frac{\begin{array}{c} T \vdash e_1 : \alpha_1 \\ T \vdash e_2 : \alpha_2 \\ \alpha_2 = \alpha_1 \text{ or } \alpha_2 = \alpha_1 List \end{array}}{T \vdash e_1 @ e_2 : \alpha_1 List}$$