Natural Semantics Specification Of Evaluation for Project 4

Throughout this document, ρ is an environment mapping variable names to values. The term $\rho(\mathtt{x})$ is the value assigned to the variable `x`. It is assumed that the environment is represented as an ordered list of variable name/value pairs, and that it is searched from the front with the first match returned. Similarly, σ is an environment storing the definitions of functions. The term $\sigma(\mathtt{f})$ is the definition of the function `f`, represented as a pair of the list of formal parameters and the expression forming the body of the function.

Evaluation of literals and variables:

In these three rules, `n` is an integer literal, `r` is a floating point literal, and `x` is an identifier.

$$\frac{}{(\mathtt{n}, \rho, \sigma) \downarrow \texttt{(Int } n\texttt{)}} \qquad \frac{}{(\mathtt{r}, \rho, \sigma) \downarrow \texttt{(Real } r\texttt{)}} \qquad \frac{}{(\mathtt{x}, \rho, \sigma) \downarrow \rho(\mathtt{x})}$$

Evaluation of built-in operators:

In the following four rules, $\bullet$ on the left of an arrow is any one of `+`, `-`, or `*`; on the right of an arrow it is the corresponding one of $+$, $-$, or $*$.

$$\frac{(e_1, \rho, \sigma) \downarrow \texttt{(Int } i_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Int } i_2\texttt{)}}{(\texttt{(}\bullet\ e_1\ e_2\texttt{)}, \rho, \sigma) \downarrow \texttt{(Int } (i_1 \bullet i_2)\texttt{)}} \qquad \frac{(e_1, \rho, \sigma) \downarrow \texttt{(Real } r_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Real } r_2\texttt{)}}{(\texttt{(}\bullet\ e_1\ e_2\texttt{)}, \rho, \sigma) \downarrow \texttt{(Real } (r_1 \bullet r_2)\texttt{)}}$$

$$\frac{(e_1, \rho, \sigma) \downarrow \texttt{(Real } r_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Int } i_2\texttt{)}}{(\texttt{(}\bullet\ e_1\ e_2\texttt{)}, \rho, \sigma) \downarrow \texttt{(Real } (r_1 \bullet (i_2 \text{ as real}))\texttt{)}} \qquad \frac{(e_1, \rho, \sigma) \downarrow \texttt{(Int } i_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Real } r_2\texttt{)}}{(\texttt{(}\bullet\ e_1\ e_2\texttt{)}, \rho, \sigma) \downarrow \texttt{(Real } ((i_1 \text{ as real}) \bullet r_2)\texttt{)}}$$

$$\frac{(e_1, \rho, \sigma) \downarrow \texttt{(Int } i_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Int } i_2\texttt{)} \quad i_1\ mod\ i_2 = 0}{(\texttt{(/ } e_1\ e_2\texttt{)}, \rho, \sigma) \downarrow \texttt{(Int } (i_1\ div\ i_2)\texttt{)}}$$

$$\frac{(e_1, \rho, \sigma) \downarrow \texttt{(Int } i_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Int } i_2\texttt{)} \quad i_1\ mod\ i_2 \neq 0}{(\texttt{(/ } e_1\ e_2\texttt{)}, \rho, \sigma) \downarrow \texttt{(Real } ((i_1 \text{ as real})/(i_2 \text{ as real}))\texttt{)}}$$

$$\frac{(e_1, \rho, \sigma) \downarrow \texttt{(Real } r_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Real } r_2\texttt{)}}{(\texttt{(/ } e_1\ e_2\texttt{)}, \rho, \sigma) \downarrow \texttt{(Real } (r_1/r_2)\texttt{)}}$$

$$\frac{(e_1, \rho, \sigma) \downarrow \texttt{(Real } r_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Int } i_2\texttt{)}}{(\texttt{(/ } e_1\ e_2\texttt{)}, \rho, \sigma) \downarrow \texttt{(Real } (r_1/(i_2 \text{ as real}))\texttt{)}} \qquad \frac{(e_1, \rho, \sigma) \downarrow \texttt{(Int } i_1\texttt{)} \quad (e_2, \rho, \sigma) \downarrow \texttt{(Real } r_2\texttt{)}}{(\texttt{(/ } e_1\ e_2\texttt{)}, \rho) \downarrow \texttt{(Real } ((i_1 \text{ as real})/r_2)\texttt{)}}$$

Evaluation of user-defined operators:

In the following rule, @ is the list append function, and *zip* is the function which, given two lists of the same length, returns a list of pairs of corresponding elements.

$$\frac{(e_1, \rho, \sigma) \downarrow v_1 \quad \cdots \quad (e_n, \rho, \sigma) \downarrow v_n \quad (2^{nd}(\sigma(\mathtt{f})), zip(1^{st}(\sigma(\mathtt{f})), [v_1, \ldots, v_n])@\rho, \sigma) \downarrow v}{(\mathtt{(f}\ e_1 \cdots e_n\mathtt{)}, \rho, \sigma) \downarrow v}$$

Notet that if there is a user-defined operator with the same name as one of the built-in operators, the user definition takes precedence.

Evaluation of Commands:

While expressions (in the context of variable and function definitions) evaluate to values, commands evaluate to new contexts.

$$\frac{}{(\mathtt{(exit)}, \rho, \sigma) \downarrow \bot} \qquad \text{(The system shuts down.)}$$

$$\frac{}{(\mathtt{(bindings)}, \rho, \sigma) \downarrow (\rho, \sigma)} \qquad \text{(The current bindings are displayed.)}$$

$$\frac{(exp, \rho, \sigma) \downarrow v}{(\mathtt{(define}\ \ var\ exp\mathtt{)}, \rho, \sigma) \downarrow (((var, v) :: \rho), \sigma)}$$

$$\frac{}{(\mathtt{(defun}\ \ \mathtt{(}(opr :: vars)\mathtt{)}\ exp\mathtt{)}, \rho, \sigma) \downarrow (\rho, (opr, vars, exp) :: \sigma))}$$

Composition of Commands:

$$\frac{(c_1, \rho, \sigma) \downarrow (\rho', \sigma') \quad (c_2, \rho', \sigma') \downarrow (\rho'', \sigma'')}{((c_1; c_2), \rho, \sigma) \downarrow (\rho'', \sigma''))}$$