µScheme

CS 456 - Programming Languages

Syntax

$$
\begin{array}{lcl}
\textit{literal} & ::= & \textit{integer} \mid \texttt{\#t} \mid \texttt{\#f} \mid {}'\textit{S-exp} \mid (\texttt{quote}\ \textit{S-exp}) \\
\textit{S-exp} & \mid & \textit{literal} \mid \textit{symbol-name} \mid (\{\textit{S-name}\}) \\
\textit{primitive} & \mid & + \mid - \mid \ldots \mid \texttt{print} \mid \texttt{error} \\
 & \mid & \texttt{car} \mid \texttt{cdr} \mid \texttt{cons} \mid \texttt{null?} \\
 & \mid & \texttt{number?} \mid \texttt{symbol?} \mid \texttt{pair?} \\
 & \mid & \texttt{boolean?} \mid \texttt{procedure?}
\end{array}
$$

Syntax

$$
\begin{array}{lcl}
\textit{def} & ::= & (\texttt{val}\ \textit{var-name}\ \textit{exp}) \\
 & | & \textit{exp} \\
 & | & (\texttt{define}\ \textit{func-name}\ (\textit{formals})\ \textit{exp}) \\
 & | & (\texttt{use}\ \textit{file-name}) \\
\textit{exp} & | & \textit{literal} \\
 & | & \textit{var-name} \\
 & | & (\texttt{set}\ \textit{var-name}) \\
 & | & (\texttt{if}\ \textit{exp}\ \textit{exp}\ \textit{exp}) \\
 & | & (\texttt{while}\ \textit{exp}\ \textit{exp}) \\
 & | & (\texttt{begin}\ \{\textit{exp}\}) \\
 & | & (\textit{exp}\ \{\textit{exp}\}) \\
 & | & (\texttt{let}\ (\{\textit{exp}\})\ \textit{exp}) \\
 & | & (\texttt{let*}\ (\{\textit{exp}\})\ \textit{exp}) \\
 & | & (\texttt{letrec}\ (\{\textit{exp}\})\ \textit{exp}) \\
 & | & (\texttt{lambda}\ (\textit{formals})\ \textit{exp}) \\
 & | & \textit{primitive}
\end{array}
$$

Abstract Syntax

$$
\begin{array}{llll}
\texttt{Def} & ::= & \texttt{VAL} & (\texttt{Name}, \texttt{Exp}) \\
 & | & \texttt{EXP} & (\texttt{Exp}) \\
 & | & \texttt{DEFINE} & (\texttt{Name}, \texttt{Lambda}) \\
 & | & \texttt{USE} & (\texttt{Name}) \\
 & & & \\
\texttt{Exp} & ::= & \texttt{LITERAL} & (\texttt{Value}) \\
 & | & \texttt{VAR} & (\texttt{Name}) \\
 & | & \texttt{SET} & (\texttt{Name}, \texttt{Exp}) \\
 & | & \texttt{IF} & (\texttt{Exp}, \texttt{Exp}, \texttt{Exp}) \\
 & | & \texttt{WHILE} & (\texttt{Exp}, \texttt{Exp}) \\
 & | & \texttt{BEGIN} & (\texttt{Explist}) \\
 & | & \texttt{APPLY} & (\texttt{Name}, \texttt{Explist}) \\
 & | & \texttt{LET} & (\texttt{LetKeyword}, \{\texttt{Name}\}, \{\texttt{Exp}\}, \texttt{Exp}) \\
 & | & \texttt{LAMBDA} & (\texttt{Lambda}) \\
 & & & \\
\texttt{LetKeyword} & ::= & \texttt{Let} \mid \texttt{Let}{*} \mid \texttt{LetRec} &
\end{array}
$$

Abstract Syntax

$$
\begin{array}{lcl}
\texttt{Lambda} & = & (\{\texttt{Name}\}, \texttt{Exp}) \\
\\
\texttt{Value} & ::= & \texttt{NIL} \\
 & | & \texttt{BOOL}\ (\texttt{int}) \\
 & | & \texttt{NUM}\ (\texttt{int}) \\
 & | & \texttt{SYM}\ (\texttt{Name}) \\
 & | & \texttt{PAIR}\ (\texttt{Value}\ {*}\texttt{car}, \texttt{Value}\ {*}\texttt{cdr}) \\
 & | & \texttt{CLOSURE}\ \ (\texttt{Lambda}, \texttt{Env}) \\
 & | & \texttt{PRIMITIVE}\ (\texttt{int tag}, \texttt{Primitive}\ {*}\texttt{function})
\end{array}
$$

μScheme: SOS Judgment

Single Environment: names -> locations

$$\rho(x) = \ell$$

$$\langle e, \rho, \sigma \rangle \Downarrow \langle v, \sigma' \rangle$$

Store: locations -> value

$$\sigma(\ell) = v$$

Evaluating *expressions* cannot change the environment

Definitions $\langle e, \rho, \sigma \rangle \rightarrow \langle \rho', \sigma' \rangle$

μScheme: SOS

$$\textsc{Var}\quad \frac{x \in \mathsf{dom}(\rho) \qquad \rho(x) = \ell \qquad \ell \in \mathsf{dom}(\sigma) \qquad \sigma(\ell) = v}{\langle \mathtt{VAR}(x), \rho, \sigma \rangle \Downarrow \langle v, \sigma \rangle}$$

$$\textsc{Assign}\quad \frac{x \in \mathsf{dom}(\rho) \qquad \rho(x) = \ell \qquad \langle e, \rho, \sigma \rangle \Downarrow \langle v, \sigma' \rangle}{\langle \mathtt{SET}(x, e), \rho, \sigma \rangle \Downarrow \langle v, \sigma'\{\ell \mapsto v\} \rangle}$$

μScheme: SOS

LET

$$\frac{(\forall\ i\ j, x_i \neq x_j) \qquad l_0, \ldots, l_n \notin \mathsf{dom}(\sigma) \qquad \langle e_i, \rho, \sigma_i \rangle \Downarrow \langle v_i, \sigma_{i+1} \rangle \qquad \langle e, \rho\{x_i \mapsto v_i\}, \sigma_n\{\ell_i \mapsto v_i\} \rangle \Downarrow \langle v, \sigma' \rangle}{\langle \mathtt{LET}(\{x_0 \ldots x_n\}, \{e_0 \ldots e_n\}, e), \rho, \sigma_0 \rangle \Downarrow \langle v, \sigma' \rangle}$$

LET*

$$\frac{(\forall\ i\ j, x_i \neq x_j) \qquad l_0, \ldots, l_n \notin \mathsf{dom}(\sigma) \qquad \langle e_0, \rho_0, \sigma_0 \rangle \Downarrow \langle v_1, \sigma_1 \rangle \qquad \langle e_i, \underbrace{\rho_{i-1}\{x_{i-1} \mapsto \ell_{i-1}\}}_{\rho_i}, \sigma_i\{\ell_{i-1} \mapsto v_{i-1}\} \rangle \Downarrow \langle v_i, \sigma_i \rangle \qquad \langle e, \rho_{n+1}, \sigma_{n+1}\{\ell_n \mapsto v_n\} \rangle \Downarrow \langle v, \sigma' \rangle}{\langle \mathtt{LET*}(\{x_0 \ldots x_n\}, \{e_0 \ldots e_n\}, e), \rho_0, \sigma_0 \rangle \Downarrow \langle v, \sigma' \rangle}$$

μScheme: SOS

$$\frac{\begin{array}{c}\textsc{Letrec} \\ (\forall\ i\ j, x_i \neq x_j) \qquad l_0, \ldots, l_n \notin \mathsf{dom}(\sigma) \\ \rho' = \rho\{x_0 \mapsto \ell_0, \ldots, x_n \mapsto \ell_n\} \qquad \sigma_0 = \sigma\{\ell_0 \mapsto \bot, \ldots, \ell_n \mapsto \bot\} \\ \langle e_i, \rho', \sigma_i\rangle \Downarrow \langle v_i, \sigma_{i+1}\rangle \qquad \langle e, \rho', \sigma_{n+1}\{\ell_n \mapsto v_n\}\rangle \Downarrow \langle v, \sigma'\rangle\end{array}}{\langle \mathtt{LETREC}(\{x_0 \ldots x_n\}, \{e_0 \ldots e_n\}, e), \rho, \sigma\rangle \Downarrow \langle v, \sigma'}$$

µScheme: SOS

Lambda

$$\frac{(\forall i\ j, i \neq j)}{\langle \texttt{LAMBDA}(\{x_0, \ldots, x_n\}, e), \rho, \sigma\rangle \Downarrow \langle (\!|\texttt{LAMBDA}(\{x_0, \ldots, x_n\}, e), \rho|\!), \sigma\rangle}$$

ApplyClosure

$$\frac{\begin{array}{cc} l_0, \ldots, l_n \notin \mathsf{dom}(\sigma) & \langle e, \rho, \sigma\rangle \Downarrow \langle (\!|\texttt{LAMBDA}(\{x_0, \ldots, x_n\}, e_c), \rho_c|\!), \sigma_0\rangle \\ \langle e_i, \rho, \sigma_i\rangle \Downarrow \langle v_i, \sigma_{i+1}\rangle & \langle e_c, \rho_c\{x_i \mapsto \ell_i\}, \sigma_{n+1}\{\ell_i \mapsto v_i\}\rangle \Downarrow \langle v, \sigma'\rangle \end{array}}{\langle \texttt{APPLY}(e, e_0, \ldots, e_n), \rho, \sigma\rangle \Downarrow \langle v, \sigma'\rangle}$$

μScheme: SOS

$$\textsc{Cons}\quad \frac{\langle e, \rho, \sigma \rangle \Downarrow \langle \texttt{PRIMITIVE}(\texttt{cons}), \sigma_0 \rangle \\ \langle e_0, \rho, \sigma_0 \rangle \Downarrow \langle v_0, \sigma_1 \rangle \qquad \langle e_1, \rho, \sigma_1 \rangle \Downarrow \langle v_1, \sigma' \rangle \qquad l_0, l_1 \notin \mathsf{dom}(\sigma')}{\langle \texttt{APPLY}(e, e_0, e_1), \rho, \sigma \rangle \Downarrow \langle \sigma'(\ell_1), \sigma' \rangle}$$

$$\textsc{Car}\quad \frac{\langle e, \rho, \sigma \rangle \Downarrow \langle \texttt{PRIMITIVE}(\texttt{car}), \sigma_1 \rangle \qquad \langle e_0, \rho, \sigma_1 \rangle \Downarrow \langle \texttt{CONS}(\ell_1, \ell_2), \sigma' \rangle}{\langle \texttt{APPLY}(e, e_0), \rho, \sigma \rangle \Downarrow \langle \sigma'(\ell_1), \sigma' \rangle}$$

$$\textsc{Cdr}\quad \frac{\langle e, \rho, \sigma \rangle \Downarrow \langle \texttt{PRIMITIVE}(\texttt{cdr}), \sigma_1 \rangle \qquad \langle e_0, \rho, \sigma_1 \rangle \Downarrow \langle \texttt{CONS}(\ell_1, \ell_2), \sigma' \rangle}{\langle \texttt{APPLY}(e, e_0), \rho, \sigma \rangle \Downarrow \langle \sigma'(\ell_2), \sigma' \rangle}$$

μScheme: SOS

$$\frac{\textsc{ResetGlobal} \quad x \in \mathsf{dom}(\rho) \qquad \langle e, \rho, \sigma \rangle}{\langle \mathtt{VAL}(x, e), \rho, \sigma \rangle \to \langle \rho, \sigma'\{\rho(x) \mapsto v\} \rangle}$$

$$\frac{\textsc{Global} \quad x \notin \mathsf{dom}(\rho) \qquad \ell \notin \mathsf{dom}(\sigma) \qquad \langle \mathtt{SET}(x, e), \rho\{x \mapsto \ell\}, \sigma\{\ell \mapsto \bot\} \rangle \Downarrow \langle v, \sigma' \rangle}{\langle \mathtt{VAL}(x, e), \rho, \sigma \rangle \to \langle \rho\{x \mapsto \ell\}, \sigma' \rangle}$$

$$\frac{\textsc{Function} \quad \langle \mathtt{VAL}(f, \mathtt{LAMBDA}(\{x_0, \ldots, x_n\}, e)), \rho\{x \mapsto \ell\}, \sigma \rangle \to \langle \rho', \sigma' \rangle}{\langle \mathtt{DEFINE}(f, \{x_0, \ldots, x_n\}, e), \rho, \sigma \rangle \to \langle \rho', \sigma' \rangle}$$