Typing Rule of Baremetalisp

Yuuki Takano

April 27, 2020

Table 1: Notation

x	variable
T	type variable
D	type name of user defined data
L	label of user defined data
$\mathcal{T}$	type
C	constraint
Γ	context
$L_{type} : L \to \mathcal{T}$	function from label to type
$L_{nth} : L \to \texttt{Int} \to \mathcal{T}$	function to n-th type of label L
$\mathcal{P}_{let}$	pattern of let expression
$\mathcal{T}_1 \equiv_\alpha \mathcal{T}_2$	$\mathcal{T}_1$ and $\mathcal{T}_2$ are α-equivalent
$\mathcal{S}$	substitution
$\mathcal{T}[\mathcal{S}]$	apply a substitution $\mathcal{S}$ to a type $\mathcal{T}$
$\{T\}$	set of T
$FV_{\mathcal{T}} : \mathcal{T} \to \{T\}$	function from a $\mathcal{T}$ to its free variables
$FV_{\Gamma} : \Gamma \to \{T\}$	function from a Γ to its free variables

$$
\begin{array}{rcll}
\Gamma & := & x : \mathcal{T}, \Gamma & \text{context} \\
& | & \varnothing & \\
\\
\mathcal{S} & := & T \to \mathcal{T}, \mathcal{S} & \text{substitution} \\
& | & \varnothing & \\
\\
\mathcal{C} & := & \mathcal{T} = \mathcal{T}, \mathcal{C} & \text{constraint} \\
& | & \varnothing & \\
\\
\mathcal{T} & := & \texttt{Int} & \\
& | & \texttt{Bool} & \\
& | & '(\mathcal{T}) & \text{list type} \\
& | & [\mathcal{T}+] & \text{tuple type} \\
& | & D & \text{user defined type} \\
& | & (D\ \mathcal{T}+) & \text{user defined type with type arguments} \\
& | & T & \text{type variable} \\
\\
\mathcal{P}_{let} & := & x & \\
& | & (L\ \mathcal{P}_{let}+) & \text{label with patterns} \\
& | & [\mathcal{P}_{let}+] & \text{tuple}
\end{array}
$$

Figure 1: Syntax

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \mid_{\varnothing} \{\} \quad \text{(T-True)} \qquad \Gamma \vdash \texttt{false} : \texttt{Bool} \mid_{\varnothing} \{\} \quad \text{(T-False)}$$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T \mid_{\{\}} \{\}} \quad \text{(T-Var)}$$

$$\frac{\begin{array}{l}\Gamma \vdash \mathcal{P}_{let} : T_0 \mid_{\mathcal{X}_0} C_0 \quad \Gamma \vdash e_1 : T_1 \mid_{\mathcal{X}_1} C_1 \quad \Gamma \vdash e_2 : T_2 \mid_{\mathcal{X}_2} C_2 \\ \mathcal{X}_0 \cap \mathcal{X}_1 \cap \mathcal{X}_2 = \varnothing \quad C = C_0 \cup C_1 \cup C_2 \cup \{T_0 = T_1\}\end{array}}{\Gamma \vdash (\texttt{let1}\ \mathcal{P}_{let}\ e_1\ e_2) : T_2 \mid_{\mathcal{X}_0 \cup \mathcal{X}_1 \cup \mathcal{X}_2} C} \quad \text{(T-Let1)}$$

$$\frac{\begin{array}{l}\Gamma \vdash e_1 : T_1 \mid_{\mathcal{X}_1} C_1 \quad \Gamma \vdash e_2 : T_2 \mid_{\mathcal{X}_2} C_2 \quad \Gamma \vdash e_3 : T_3 \mid_{\mathcal{X}_3} C_3 \\ \mathcal{X}_1 \cap \mathcal{X}_2 \cap \mathcal{X}_3 = \varnothing \quad C = C_1 \cup C_2 \cup C_3 \cup \{T_1 = \texttt{Bool}, T_2 = T_3\}\end{array}}{\Gamma \vdash (\texttt{if}\ e_1\ e_2\ e_3) : T_2 \mid_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3} C} \quad \text{(T-If)}$$

Figure 2: Typing rule

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T \mid_{\{\}} \{\}} \quad \text{(PL-Var)}$$

$$\frac{\begin{array}{l}\Gamma \vdash \mathcal{P}_{let_1} : T_1 \mid_{\mathcal{X}_1} C_1 \wedge \cdots \wedge \Gamma \vdash \mathcal{P}_{let_n} : T_n \mid_{\mathcal{X}_n} C_n \\ L_{type}(L)[\mathcal{S}] \equiv_\alpha T_0 \quad FV(T_0) \cap \mathcal{X}_1 \cap \cdots \cap \mathcal{X}_n = \varnothing \\ FV_{\mathcal{T}}(T_0) \cap FV_\Gamma(\Gamma) = \varnothing \quad \mathcal{X} = FV(T_0) \cup \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_n \\ C = \{L_{nth}(L, 1)[\mathcal{S}] = T_1, \cdots, L_{nth}(L, n)[\mathcal{S}] = T_n\} \cup C_1 \cup \cdots \cup C_n\end{array}}{\Gamma \vdash (L\ \mathcal{P}_{let_1}\ \cdots\ \mathcal{P}_{let_n}) : T_0 \mid_{\mathcal{X}} C} \quad \text{(PL-Label)}$$

$$\frac{\begin{array}{l}\Gamma \vdash \mathcal{P}_{let_1} : T_1 \mid_{\mathcal{X}_1} C_1 \wedge \cdots \wedge \Gamma \vdash \mathcal{P}_{let_n} : T_n \mid_{\mathcal{X}_n} C_n \\ \mathcal{X}_1 \cap \cdots \cap \mathcal{X}_n = \varnothing \quad \mathcal{X} = \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_n \quad C = C_1 \cup \cdots \cup C_n\end{array}}{\Gamma \vdash [\mathcal{P}_{let_1}\ \cdots\ \mathcal{P}_{let_n}] : [T_1 \cdots T_n] \mid_{\mathcal{X}} C} \quad \text{(PL-Tuple)}$$

Figure 3: Typing rule of let expression's pattern