

Typing Rule of Baremetalisp

Yuuki Takano
ytakano@wide.ad.jp

May 31, 2020

1 Introduction

In this paper, I will formally describe the typing rule of Baremetalisp, which is a well typed Lisp for trusted execution environment.

2 Notation and Syntax

Table 1 and Fig. 1 shows notation used in this paper and syntax for the typing rule, respectively.

Listing 1: Example of variable and type

```
1 (defun add (a b) (Pure (-> (Int Int) Int))
2   (+ a b))
```

x is a variable. For example, $x \in \{a, b\}$ in Listing 1. \mathcal{T} is a type. For example, $\mathcal{T} \in \{\text{Int}, (\rightarrow (\text{Int Int}) \text{Int})\}$ in Listing 1. $(\rightarrow (\text{Int Int}) \text{Int})$ is a function type which takes 2 integer values and return 1 integer value. **Pure** in Listing 1 denotes the effect of the function but I just ignore it now. Function effects will be described in Sec. 4. \mathcal{T} can be other forms as described in Fig. 1 such as **Bool**, $'(\text{Int})$, $[\text{Bool Int}]$, $(\text{List } a)$, (List Int) . C is a type constraint, which is a set of pairs of types. For example, $C = \{(\rightarrow (t_1 t_2) t) = (\rightarrow (\text{Int Int}) \text{Int})\}$ deduced from Listing 1 means $(\rightarrow (t_1 t_2) t)$ and $(\rightarrow (\text{Int Int}) \text{Int})$ are semantically equal and every type variable in C , t_1, t_2, t , is thus **Int**. Γ is a map from variable and label to type. For example, $\Gamma = \{a : t_1, b : t_2, + : (\rightarrow (\text{Int Int}) \text{Int})\}$ in Listing 1. Γ is called context generally, thus I call Γ context in this paper.

Listing 2: Example of user defined data type

```
1 (data (List a)
2   (Cons a (List a))
3   Nil)
```

t is a type variable. For example, $t \in \{a\}$ in Listing 2. L is a label for user defined type. For example, $L \in \{\text{Cons}, \text{Nil}\}$ in Listing 2. D is user defined data. For example, $D \in \{\text{List}\}$ in Listing 2. Γ will hold mapping from labels in

Table 1: Notation

e	expression
z	integer literal such as 10, -34, 112
x	variable
t	type variable
D	type name of user defined data
L	label of user defined data
E	effect
$E_{\mathcal{T}} : T \rightarrow E$	effect of type
\mathcal{T}	type
C	type constraint
Γ	context
\mathcal{P}	pattern
\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
$\mathcal{S} : t \rightarrow \mathcal{T}$	substitution from type variable to type
$\mathcal{T} \cdot \mathcal{S}$	apply \mathcal{S} to \mathcal{T}
\mathcal{X}	set of t
$FV_{\mathcal{T}} : \mathcal{T} \rightarrow \mathcal{X}$	function from \mathcal{T} to its free variables
$FV_{\Gamma} : \Gamma \rightarrow \mathcal{X}$	function from Γ to its free variables
$Size : L \rightarrow \text{Int}$	the number of labels L 's type has
$\Gamma \vdash e : \mathcal{T} \mid_{\mathcal{X}} C$	e 's type is deduced as \mathcal{T} from Γ under constraint C and type variables \mathcal{X}

addition to variables. For example, $\Gamma = \{\text{Cons} : (\text{List } a), \text{Nil} : (\text{List } a), \text{Cons}_{1st} : a, \text{Cons}_{2nd} : (\text{List } a)\}$ in Listing 2.

$FV_{\mathcal{T}}$ and FV_{Γ} are functions, which take \mathcal{T} and Γ and return free variables. For example, $FV_{\mathcal{T}}((\rightarrow (t_1 t_2) t)) = \{t_1, t_2, t\}$ and

$$\begin{aligned}
& FV_{\Gamma}(\{a : t_1, b : t_1, + : (\rightarrow (\text{Int Int}) \text{Int})\}) \\
&= \{FV_{\mathcal{T}}(t_1), FV_{\mathcal{T}}(t_1), FV_{\mathcal{T}}((\rightarrow (\text{Int Int}) \text{Int}))\} \\
&= \{t_1, t_2\}.
\end{aligned}$$

$\mathcal{T}_1 \equiv_{\alpha} \mathcal{T}_2$ denotes that \mathcal{T}_1 and \mathcal{T}_2 are α -equivalent, which means \mathcal{T}_1 and \mathcal{T}_2 are semantically equal. For example, $(\rightarrow (t_1 t_2) t) \equiv_{\alpha} (\rightarrow (t_{10} t_{11}) t_{12})$. \mathcal{S} is a substitution, which is a map from type variable to type, and it can be applied to \mathcal{T} as $\mathcal{T} \cdot \mathcal{S}$. For example, if $\mathcal{S}(t_1) = [\text{Bool Int}]$, $\mathcal{S}(t_2) = (\text{List } t_3)$ then $(\rightarrow (t_1 t_2) t) \cdot \mathcal{S} = (\rightarrow ([\text{Bool Int}] (\text{List } t_3)) t)$.

Listing 3: Example of pattern matching

```

1 (data Dim2 (Dim2 Int Int))
2
3 (data (Maybe t)
4   (Just t)
5   Nothing)
6
```

```

7 (defun match-let (a) (Pure (-> ((Maybe Dim2)) Int))
8   (match a
9     ((Just val)
10      (let ((Dim2 x y) val))
11          (+ x y)))
12   (Nothing
13    0)))

```

\mathcal{P} and \mathcal{P}_{let} are pattern in match and let expressions. For example, in listings 3, $(Just\ val)$ and $Nothing$ at line 9 and 12 are from \mathcal{P} and $(Dim2\ x\ y)$ at line 10 is from \mathcal{P}_{let} . $Size$ is a function which takes a label and return the number of labels the label's type has. For example, $Size(Just) = Size(Nothing) = 2$ because $Maybe$ type has 2 labels and $Size(Dim2) = 1$ because $Dim2$ type has 1 label in listings 3.

3 Typing Rule

In this section, I will introduce the typing rule of Baremetalisp. Before describing the rule, I introduce an assumption that there is no variable shadowing to make it simple. This means that every variable should be properly α -converted by using the De Bruijn index technique or variable shadowing should be handled when implementing the type inference algorithm.

Fig. 2 and 3 are the typing rule of expressions and function definitions.

4 Effect

\mathcal{C}	$:=$	$\mathcal{T} = \mathcal{T}, \mathcal{C}$ \emptyset	type constraint
Γ	$:=$	$x : \mathcal{T}, \Gamma$ $L : \mathcal{T}, \Gamma$ $L_{nth} : \mathcal{T}, \Gamma$ \emptyset	context type of variable type of label n-th type of label's element
E	$:=$	Pure IO	effect
\mathcal{T}	$:=$	Int Bool $'(\mathcal{T})$ $[\mathcal{T}+]$ D $(D \ \mathcal{T}+)$ $(E \ (\rightarrow \ (\mathcal{T}^*) \ \mathcal{T}))$ t	type list type tuple type user defined type user defined type with type arguments function type type variable
\mathcal{P}	$:=$	x L $(L \ \mathcal{P}+)$ $'()$ $[\mathcal{P}+]$	pattern variable label label with patterns empty list tuple
\mathcal{P}_{let}	$:=$	x $(L \ \mathcal{P}_{let}+)$ $[\mathcal{P}_{let}+]$	patten for let variable label with patterns tuple

Figure 1: Syntax

$$\begin{array}{c}
\Gamma \vdash \mathbf{true} : \mathbf{Bool} \mid_{\emptyset} \emptyset \quad (\text{T-True}) \qquad \Gamma \vdash \mathbf{false} : \mathbf{Bool} \mid_{\emptyset} \emptyset \quad (\text{T-False}) \\
\\
\frac{x : T \in \Gamma}{\Gamma \vdash x : T \mid_{\emptyset} \emptyset} \quad (\text{T-Var}) \qquad \Gamma \vdash z : \mathbf{Int} \mid_{\emptyset} \emptyset \quad (\text{T-Num}) \\
\\
\frac{x : T' \in \Gamma \quad T' \cdot S \equiv_{\alpha} T}{\Gamma \vdash x : T \mid_{FV_{\mathcal{T}}(T)} \emptyset} \quad (\text{T-VarPoly}) \\
\\
\frac{\begin{array}{l} \Gamma \vdash \mathcal{P}_{let} : \mathcal{T}_0 \mid_{\mathcal{X}_0} C_0 \quad \Gamma \vdash e_1 : \mathcal{T}_1 \mid_{\mathcal{X}_1} C_1 \quad \Gamma \vdash e_2 : \mathcal{T}_2 \mid_{\mathcal{X}_2} C_2 \\ \mathcal{X}_0 \cap \mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset \quad C = C_0 \cup C_1 \cup C_2 \cup \{\mathcal{T}_0 = \mathcal{T}_1\} \end{array}}{\Gamma \vdash (\mathbf{let1} \mathcal{P}_{let} e_1 e_2) : \mathcal{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 : \mathcal{T}_1 \mid_{\mathcal{X}_1} C_1 \quad \Gamma \vdash e_2 : \mathcal{T}_2 \mid_{\mathcal{X}_2} C_2 \quad \Gamma \vdash e_3 : \mathcal{T}_3 \mid_{\mathcal{X}_3} C_3 \\ \mathcal{X}_1 \cap \mathcal{X}_2 \cap \mathcal{X}_3 = \emptyset \quad C = C_1 \cup C_2 \cup C_3 \cup \{\mathcal{T}_1 = \mathbf{Bool}, \mathcal{T}_2 = \mathcal{T}_3\} \end{array}}{\Gamma \vdash (\mathbf{if} e_1 e_2 e_3) : \mathcal{T}_2 \mid_{\mathcal{X}_1 \cup \mathcal{X}_2 \cup \mathcal{X}_3} C} \quad (\text{T-If}) \\
\\
\frac{\begin{array}{l} \Gamma \vdash e_1 : \mathcal{T}_1 \mid_{\mathcal{X}_1} C_1 \quad \Gamma \vdash e_2 : \mathcal{T}_2 \mid_{\mathcal{X}_2} C_2 \wedge \dots \wedge \Gamma \vdash e_n : \mathcal{T}_n \mid_{\mathcal{X}_n} C_n \\ \{t\} \cap FV_{\Gamma}(\Gamma) = \emptyset \quad \{t\} \cap \mathcal{X}_1 \cap \dots \cap \mathcal{X}_n = \emptyset \\ \mathcal{X} = \{t\} \cup \mathcal{X}_1 \cup \dots \cup \mathcal{X}_n \quad E = E_{\mathcal{T}}(\mathcal{T}_1) \\ C = C_1 \cup \dots \cup C_n \cup \{\mathcal{T}_1 = (E \rightarrow (\mathcal{T}_2 \dots \mathcal{T}_n) t)\} \end{array}}{\Gamma \vdash (e_1 e_2 \dots e_n) : t \mid_{\mathcal{X}} C} \quad (\text{T-App}) \\
\\
\frac{\begin{array}{l} \Gamma \vdash e_0 : \mathcal{T}_0 \mid_{\mathcal{X}_0} C_0 \\ \Gamma \vdash e_1 : \mathcal{T}_{e1} \mid_{\mathcal{X}_{e1}} C_{e1} \wedge \dots \wedge \Gamma \vdash e_n : \mathcal{T}_{en} \mid_{\mathcal{X}_{en}} C_{en} \\ \Gamma \vdash \mathcal{P}_1 : \mathcal{T}_{p1} \mid_{\mathcal{X}_{p1}} C_{p1} \wedge \dots \wedge \Gamma \vdash \mathcal{P}_{pn} : \mathcal{T}_{pn} \mid_{\mathcal{X}_{pn}} C_{pn} \\ \mathcal{X}_0 \cap \mathcal{X}_{e1} \cap \dots \cap \mathcal{X}_{en} \cap \mathcal{X}_{p1} \cap \dots \cap \mathcal{X}_{pn} = \emptyset \\ \mathcal{X} = \mathcal{X}_0 \cup \mathcal{X}_{e1} \cup \dots \cup \mathcal{X}_{en} \cup \mathcal{X}_{p1} \cup \dots \cup \mathcal{X}_{pn} \\ C = C_0 \cup C_{e1} \cup \dots \cup C_{en} \cup C_{p1} \cup \dots \cup C_{pn} \cup \\ \{\mathcal{T}_0 = \mathcal{T}_{p1}, \dots, \mathcal{T}_0 = \mathcal{T}_{pn}\} \cup \{\mathcal{T}_{e1} = \mathcal{T}_{e2}, \dots, \mathcal{T}_{e1} = \mathcal{T}_{en}\} \end{array}}{\Gamma \vdash (\mathbf{match} e_0 (\mathcal{P}_1 e_1) \dots (\mathcal{P}_n e_n)) : T_{e1} \mid_{\mathcal{X}} C} \quad (\text{T-Match})
\end{array}$$

Figure 2: Typing rule (1/2)

$$\Gamma \vdash '() : '(T) \mid_{\{T\}} \emptyset \quad (\text{T-Nil}) \quad \frac{L : \mathcal{T}' \in \Gamma \quad \mathcal{T}' \cdot \mathcal{S} \equiv_{\alpha} \mathcal{T}}{\Gamma \vdash L : \mathcal{T} \mid_{FV_{\mathcal{T}}(\mathcal{T})} \emptyset} \quad (\text{T-Label0})$$

$$\frac{\Gamma \vdash e_1 : T_1 \mid_{\mathcal{X}_1} C_1 \wedge \dots \wedge \Gamma \vdash e_n : T_n \mid_{\mathcal{X}_n} C_n \quad \mathcal{X}_1 \cap \dots \cap \mathcal{X}_n = \emptyset \quad \mathcal{X} = \mathcal{X}_1 \cup \dots \cup \mathcal{X}_n \quad C = C_1 \cup \dots \cup C_n}{\Gamma \vdash [e_1 \ \dots \ e_n] : [T_1 \ \dots \ T_n] \mid_{\mathcal{X}} C} \quad (\text{T-Tuple})$$

$$\frac{\Gamma \vdash e_1 : T_1 \mid_{\mathcal{X}_1} C_1 \wedge \dots \wedge \Gamma \vdash e_n : T_n \mid_{\mathcal{X}_n} C_n \quad \mathcal{X}_1 \cap \dots \cap \mathcal{X}_n = \emptyset \quad \mathcal{X} = \mathcal{X}_1 \cup \dots \cup \mathcal{X}_n \quad C = C_1 \cup \dots \cup C_n \cup \{T_1 = T_2, \dots, T_1 = T_n\}}{\Gamma \vdash '(e_1 \ \dots \ e_n) : '(T_1) \mid_{\mathcal{X}} C} \quad (\text{T-List})$$

$$\frac{\begin{array}{l} \Gamma \vdash e_1 : \mathcal{T}_1 \mid_{\mathcal{X}_1} C_1 \wedge \dots \wedge \Gamma \vdash e_n : \mathcal{T}_n \mid_{\mathcal{X}_n} C_n \\ L : \mathcal{T}'_0 \in \Gamma \quad \mathcal{T}'_0 \cdot \mathcal{S} \equiv_{\alpha} \mathcal{T}_0 \quad FV(\mathcal{T}_0) \cap \mathcal{X}_1 \cap \dots \cap \mathcal{X}_n = \emptyset \\ FV_{\mathcal{T}}(\mathcal{T}_0) \cap FV_{\Gamma}(\Gamma) = \emptyset \quad \mathcal{X} = FV(\mathcal{T}_0) \cup \mathcal{X}_1 \cup \dots \cup \mathcal{X}_n \\ L_{1st} : \mathcal{T}'_1 \in \Gamma \wedge \dots \wedge L_{nth} : \mathcal{T}'_n \in \Gamma \\ C = C_1 \cup \dots \cup C_n \cup \{\mathcal{T}'_1 \cdot \mathcal{S} = \mathcal{T}_1, \dots, \mathcal{T}'_n \cdot \mathcal{S} = \mathcal{T}_n\} \end{array}}{\Gamma \vdash (L \ e_1 \ \dots \ e_n) : \mathcal{T}_0 \mid_{\mathcal{X}} C} \quad (\text{T-Label})$$

$$\frac{\begin{array}{l} \Gamma \vdash x_1 : t_1 \mid_{\emptyset} \emptyset \wedge \dots \wedge \Gamma \vdash x_n : t_n \mid_{\emptyset} \emptyset \\ \Gamma \vdash e : \mathcal{T}_0 \mid_{\mathcal{X}} C_0 \\ C = \{\mathcal{T} = (\text{Pure } (\rightarrow (t_1 \ \dots \ t_n) \mathcal{T}_0))\} \cup C_0 \end{array}}{\Gamma \vdash (\text{lambda } (x_1 \ \dots \ x_n) e) : \mathcal{T} \mid_{\mathcal{X}} C} \quad (\text{T-Lambda})$$

$$\frac{\begin{array}{l} \Gamma \vdash x_1 : \mathcal{T}_1 \mid_{\emptyset} \emptyset \wedge \dots \wedge \Gamma \vdash x_n : \mathcal{T}_n \mid_{\emptyset} \emptyset \\ \Gamma \vdash e : \mathcal{T}_0 \mid_{\mathcal{X}} C_0 \quad E = E_{\mathcal{T}}(\mathcal{T}) \\ C = C_0 \cup \{\mathcal{T} = (E (\rightarrow (\mathcal{T}_1 \ \dots \ \mathcal{T}_n) \mathcal{T}_0))\} \end{array}}{\Gamma \vdash (\text{defun name } (x_1 \ \dots \ x_n) \mathcal{T} e) : \mathcal{T} \mid_{\mathcal{X}} C} \quad (\text{T-Defun})$$

Figure 3: Typing rule (2/2)

$$\begin{array}{c}
\Gamma \vdash \mathbf{true} : \mathbf{Bool} \mid_{\emptyset} \emptyset \quad (\text{P-True}) \qquad \Gamma \vdash \mathbf{false} : \mathbf{Bool} \mid_{\emptyset} \emptyset \quad (\text{P-False}) \\
\\
\frac{x : T \in \Gamma}{\Gamma \vdash x : T \mid_{\emptyset} \emptyset} \quad (\text{P-Var}) \qquad \Gamma \vdash z : \mathbf{Int} \mid_{\emptyset} \emptyset \quad (\text{P-Num}) \\
\\
\Gamma \vdash '() : '(T) \mid_{\{T\}} \emptyset \quad (\text{P-Nil}) \qquad \frac{L : \mathcal{T}' \in \Gamma \quad \mathcal{T}' \cdot \mathcal{S} \equiv_{\alpha} \mathcal{T}}{\Gamma \vdash L : \mathcal{T} \mid_{FV_{\mathcal{T}}(\mathcal{T})} \emptyset} \quad (\text{P-Label0}) \\
\\
\frac{\begin{array}{l}
\Gamma \vdash \mathcal{P}_1 : \mathcal{T}_1 \mid_{\mathcal{X}_1} C_1 \wedge \dots \wedge \Gamma \vdash \mathcal{P}_n : \mathcal{T}_n \mid_{\mathcal{X}_n} C_n \\
L : \mathcal{T}'_0 \in \Gamma \quad \mathcal{T}'_0 \cdot \mathcal{S} \equiv_{\alpha} \mathcal{T}_0 \quad FV(\mathcal{T}_0) \cap \mathcal{X}_1 \cap \dots \cap \mathcal{X}_n = \emptyset \\
FV_{\mathcal{T}}(\mathcal{T}_0) \cap FV_{\Gamma}(\Gamma) = \emptyset \quad \mathcal{X} = FV(\mathcal{T}_0) \cup \mathcal{X}_1 \cup \dots \cup \mathcal{X}_n \\
L_{1st} : \mathcal{T}'_1 \in \Gamma \wedge \dots \wedge L_{nth} : \mathcal{T}'_n \in \Gamma \\
C = C_1 \cup \dots \cup C_n \cup \{\mathcal{T}'_1 \cdot \mathcal{S} = \mathcal{T}_1, \dots, \mathcal{T}'_n \cdot \mathcal{S} = \mathcal{T}_n\} \\
Size(L) = 1 \text{ for only } P_{let}
\end{array}}{\Gamma \vdash (L \mathcal{P}_1 \dots \mathcal{P}_n) : \mathcal{T}_0 \mid_{\mathcal{X}} C} \quad (\text{P-Label}) \\
\\
\frac{\begin{array}{l}
\Gamma \vdash \mathcal{P}_1 : \mathcal{T}_1 \mid_{\mathcal{X}_1} C_1 \wedge \dots \wedge \Gamma \vdash \mathcal{P}_n : \mathcal{T}_n \mid_{\mathcal{X}_n} C_n \\
\mathcal{X}_1 \cap \dots \cap \mathcal{X}_n = \emptyset \quad \mathcal{X} = \mathcal{X}_1 \cup \dots \cup \mathcal{X}_n \quad C = C_1 \cup \dots \cup C_n
\end{array}}{\Gamma \vdash [\mathcal{P}_1 \dots \mathcal{P}_n] : [\mathcal{T}_1 \dots \mathcal{T}_n] \mid_{\mathcal{X}} C} \quad (\text{P-Tuple})
\end{array}$$

Figure 4: Typing rule of pattern