PTFAE – TFAE with Polymorphic Types

1 INTRODUCTION

PTFAE is a toy language for the COSE212 course at Korea University. PTFAE stands for an extension of the TFAE language with **polymorphic types**, and it supports the following features:

- **number (integer) values**
- **basic arithmetic operators**: addition (+) and multiplication (*)
- **first-class functions**
- **immutable variables** (`val`)
- **polymorphic types** (parametric polymorphism)
- **static type checking**

This document is the specification of PTFAE. First, Section 2 describes the concrete syntax, and Section 3 describes the abstract syntax. Then, Section 4 describes the type system. Finally, Section 5 describes the big-step operational (natural) semantics of PTFAE.

2 CONCRETE SYNTAX

The concrete syntax of PTFAE is written in a variant of the extended Backus–Naur form (EBNF). The notation `<nt>` denotes a nonterminal, and `"t"` denotes a terminal. We use `?` to denote an optional element and `+` (or `*`) to denote one or more (or zero or more) repetitions of the preceding element. We use `butnot` to denote a set difference to exclude some strings from a producible set of strings. We omit some obvious terminals using the ellipsis (...) notation.

```
// basic elements
<digit>    ::= "0" | "1" | "2" | ... | "9"
<number>   ::= "-"? <digit>+
<alphabet> ::= "A" | "B" | "C" | ... | "Z" | "a" | "b" | "c" | ... | "z"
<idstart>  ::= <alphabet> | "_"
<idcont>   ::= <alphabet> | "_" | <digit>
<keyword>  ::= "val" | "Number"
<id>       ::= <idstart> <idcont>* butnot <keyword>

// expressions
<expr>     ::= <number> | <expr> "+" <expr> | <expr> "*" <expr>
             | "(" <expr> ")" | "{" <expr> "}"
             | "val" <id> "=" <expr> ";"? <expr> | <id>
             | "(" <id> ":" <type> ")" "=>" <expr> | <expr> "(" <expr> ")"
             | "forall" "[" <type> "]" <expr> | <expr> "[" <type> "]"

// types
<type>     ::= "(" <type> ")" | "Number" | <id> | "forall" "[" <id> "]" <type>
```

For types, the arrow (=>) operator is right-associative. For expressions, the precedence and associativity of operators are defined as follows:

Operator	Associativity	Precedence
*	left	1
+	left	2

3 ABSTRACT SYNTAX

The abstract syntax of PTFAE is defined as follows:

Expressions	$\mathbb{E} \ni e ::= n$	(Num)	Numbers	$n \in \mathbb{Z}$	(BigInt)
	$\mid e + e$	(Add)	Identifiers	$x \in \mathbb{X}$	(String)
	$\mid e \times e$	(Mul)	Type Variables	$\alpha \in \mathbb{X}_\alpha$	(String)
	$\mid \texttt{val}\ x\texttt{=}e\texttt{;}\ e$	(Val)			
	$\mid x$	(Id)	Types	$\mathbb{T} \ni \tau ::= \mathsf{num}$	(NumT)
	$\mid \lambda x{:}\tau.e$	(Fun)		$\mid \tau \rightarrow \tau$	(ArrowT)
	$\mid e(e)$	(App)		$\mid \alpha$	(VarT)
	$\mid \forall\alpha.e$	(TypeAbs)		$\mid \forall\alpha.\tau$	(PolyT)
	$\mid e[\tau]$	(TypeApp)			

4 TYPE SYSTEM

This section explains type system of PTFAE, and we use the following notations:

$$\text{Type Environments} \quad \Gamma \;\in\; (\mathbb{X} \xrightarrow{\text{fin}} \mathbb{T}) \times \mathcal{P}(\mathbb{X}_\alpha) \quad (\mathsf{TypeEnv})$$

In the type system, type checking is defined with the following typing rules:

$$\boxed{\Gamma \vdash e : \tau}$$

$$\tau\text{–Num}\ \frac{}{\Gamma \vdash n : \mathsf{num}} \qquad \tau\text{–Add}\ \frac{\Gamma \vdash e_1 : \mathsf{num} \qquad \Gamma \vdash e_2 : \mathsf{num}}{\Gamma \vdash e_1 + e_2 : \mathsf{num}} \qquad \tau\text{–Mul}\ \frac{\Gamma \vdash e_1 : \mathsf{num} \qquad \Gamma \vdash e_2 : \mathsf{num}}{\Gamma \vdash e_1 \times e_2 : \mathsf{num}}$$

$$\tau\text{–Val}\ \frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma[x : \tau_1] \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{val}\ x\texttt{=}e_1\texttt{;}\ e_2 : \tau_2} \qquad \tau\text{–Id}\ \frac{x \in \mathrm{Domain}(\Gamma)}{\Gamma \vdash x : \Gamma(x)}$$

$$\tau\text{–Fun}\ \frac{\Gamma[x : \tau] \vdash e : \tau'}{\Gamma \vdash \lambda x{:}\tau.e : \tau \rightarrow \tau'} \qquad \tau\text{–App}\ \frac{\Gamma \vdash e_0 : \tau_1 \rightarrow \tau_2 \qquad \Gamma \vdash e_1 : \tau_3 \qquad \tau_1 \equiv \tau_3}{\Gamma \vdash e_0(e_1) : \tau_2}$$

$$\tau\text{–TypeAbs}\ \frac{\alpha \notin \mathrm{Domain}(\Gamma) \qquad \Gamma[\alpha] \vdash e : \tau}{\Gamma \vdash \forall\alpha.e : \forall\alpha.\tau} \qquad \tau\text{–TypeApp}\ \frac{\Gamma \vdash \tau \qquad \Gamma \vdash e : \forall\alpha.\tau'}{\Gamma \vdash e[\tau] : \tau'[\alpha \leftarrow \tau]}$$

the following rules for well-formedness of types:

$$\boxed{\Gamma \vdash \tau}$$

$$\frac{}{\Gamma \vdash \mathsf{num}} \qquad \frac{\Gamma \vdash \tau \qquad \Gamma \vdash \tau'}{\Gamma \vdash \tau \rightarrow \tau'} \qquad \frac{\alpha \in \mathrm{Domain}(\Gamma)}{\Gamma \vdash \alpha} \qquad \frac{\Gamma[\alpha] \vdash \tau}{\Gamma \vdash \forall\alpha.\tau}$$

and the following rules for type equivalence:

$$\boxed{\tau \equiv \tau}$$

$$\frac{}{\mathsf{num} \equiv \mathsf{num}} \qquad \frac{\tau_1 \equiv \tau_1' \qquad \tau_2 \equiv \tau_2'}{(\tau_1 \rightarrow \tau_2) \equiv (\tau_1' \rightarrow \tau_2')} \qquad \frac{}{\alpha \equiv \alpha} \qquad \frac{\tau \equiv \tau'[\alpha' \leftarrow \alpha]}{\forall\alpha.\tau \equiv \forall\alpha'.\tau'}$$

5 SEMANTICS

We use the following notations in the semantics:

$$
\begin{array}{llll}
\text{Values} & \mathbb{V} \ni v ::= n & (\texttt{NumV}) \\
& \quad\;\; \mid \langle \lambda x.e, \sigma \rangle & (\texttt{CloV}) \\
& \quad\;\; \mid \langle \forall \alpha.e, \sigma \rangle & (\texttt{TypeAbsV})
\end{array}
\qquad
\text{Environments} \quad \sigma \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{V} \quad (\texttt{Env})
$$

The big-step operational (natural) semantics of PTFAE is defined as follows:

$$\boxed{\sigma \vdash e \Rightarrow v}$$

$$
\texttt{Num}\ \frac{}{\sigma \vdash n \Rightarrow n}
\qquad
\texttt{Add}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2}
\qquad
\texttt{Mul}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 \times e_2 \Rightarrow n_1 \times n_2}
$$

$$
\texttt{Val}\ \frac{\sigma \vdash e_1 \Rightarrow v_1 \qquad \sigma[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash \texttt{val}\ x{=}e_1;\ e_2 \Rightarrow v_2}
\qquad
\texttt{Fun}\ \frac{}{\sigma \vdash \lambda x{:}\tau.e \Rightarrow \langle \lambda x.e, \sigma \rangle}
$$

$$
\texttt{Id}\ \frac{x \in \text{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}
\qquad
\texttt{App}\ \frac{\sigma \vdash e_0 \Rightarrow \langle \lambda x.e_2, \sigma' \rangle \qquad \sigma \vdash e_1 \Rightarrow v_1 \qquad \sigma'[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash e_0(e_1) \Rightarrow v_2}
$$

$$
\texttt{TypeAbs}\ \frac{}{\sigma \vdash \forall \alpha.e \Rightarrow \langle \forall \alpha.e, \sigma \rangle}
\qquad
\texttt{TypeApp}\ \frac{\sigma \vdash e \Rightarrow \langle \forall \alpha.e', \sigma' \rangle \qquad \sigma' \vdash e' \Rightarrow v}{\Gamma \vdash e[\tau] : v}
$$