

# TFAE – FAE with Type System

## 1 INTRODUCTION

TFAE is a toy language for the [COSE212](#) course at Korea University. TFAE stands for an extension of the [FAE](#) language with a **type system**, and it supports the following features:

- **number (integer) values** (0, 1, -1, 2, -2, 3, -3, ...)
- **arithmetic operators**: addition (+) and multiplication (\*)
- **immutable variable definitions** (val)
- **first-class functions** (=>)
- **static type checking**

This document is the specification of TFAE. 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 TFAE.

## 2 CONCRETE SYNTAX

The concrete syntax of TFAE 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> ")"

// types
<type>     ::= "(" <type> ")" | "Number" | <type> "==" <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 TFAE is defined as follows:

Expressions	$\mathbb{E} \ni e ::= n$	(Num)	Numbers	$n \in \mathbb{Z}$	(BigInt)
	$  e + e$	(Add)			
	$  e * e$	(Mul)			
	$  \text{val } x = e; e$	(Val)			
	$  x$	(Id)			
	$  \lambda x : \tau. e$	(Fun)			
	$  e(e)$	(App)	Types	$\mathbb{T} \ni \tau ::= \text{num}$	(NumT)
				$  \tau \rightarrow \tau$	(ArrowT)

### 4 TYPE SYSTEM

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

$$\text{Type Environments} \quad \Gamma \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{T} \quad (\text{TypeEnv})$$

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

$$\begin{array}{c}
 \boxed{\Gamma \vdash e : \tau} \\
 \\
 \tau\text{-Num} \frac{}{\Gamma \vdash n : \text{num}} \\
 \\
 \tau\text{-Add} \frac{\Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num}}{\Gamma \vdash e_1 + e_2 : \text{num}} \quad \tau\text{-Mul} \frac{\Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num}}{\Gamma \vdash e_1 \times e_2 : \text{num}} \\
 \\
 \tau\text{-Val} \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma[x : \tau_1] \vdash e_2 : \tau_2}{\Gamma \vdash \text{val } x = e_1; e_2 : \tau_2} \quad \tau\text{-Id} \frac{x \in \text{Domain}(\Gamma)}{\Gamma \vdash x : \Gamma(x)} \\
 \\
 \tau\text{-Fun} \frac{\Gamma[x : \tau] \vdash e : \tau'}{\Gamma \vdash \lambda x : \tau. e : \tau \rightarrow \tau'} \quad \tau\text{-App} \frac{\Gamma \vdash e_0 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_1 : \tau_1}{\Gamma \vdash e_0(e_1) : \tau_2}
 \end{array}$$

### 5 SEMANTICS

We use the following notations in the semantics:

$$\begin{array}{ll}
 \text{Values} & \mathbb{V} \ni v ::= n \quad (\text{NumV}) \\
 & | \langle \lambda x. e, \sigma \rangle \quad (\text{CloV}) \\
 \text{Environments} & \sigma \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{V} \quad (\text{Env})
 \end{array}$$

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

$$\begin{array}{c}
 \boxed{\sigma \vdash e \Rightarrow v} \\
 \\
 \text{Num} \frac{}{\sigma \vdash n \Rightarrow n} \quad \text{Add} \frac{\sigma \vdash e_1 \Rightarrow n_1 \quad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2} \quad \text{Mul} \frac{\sigma \vdash e_1 \Rightarrow n_1 \quad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 * e_2 \Rightarrow n_1 \times n_2} \\
 \\
 \text{Val} \frac{\sigma \vdash e_1 \Rightarrow v_1 \quad \sigma[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash \text{val } x = e_1; e_2 \Rightarrow v_2} \quad \text{Fun} \frac{}{\sigma \vdash \lambda x : \tau. e \Rightarrow \langle \lambda x. e, \sigma \rangle} \\
 \\
 \text{Id} \frac{x \in \text{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)} \quad \text{App} \frac{\sigma \vdash e_0 \Rightarrow \langle \lambda x. e_2, \sigma' \rangle \quad \sigma \vdash e_1 \Rightarrow v_1 \quad \sigma'[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash e_0(e_1) \Rightarrow v_2}
 \end{array}$$