

# FVAE – VAE with First-Class Functions

## 1 INTRODUCTION

FVAE is a toy language for the [COSE212](#) course at Korea University. FVAE stands for an extension of the [VAE](#) language with **first-class functions**, 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** (=>)

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

## 2 CONCRETE SYNTAX

The concrete syntax of FVAE 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"
<id>         ::= <idstart> <idcont>* butnot <keyword>

// expressions
<expr>       ::= <number> | <expr> "+" <expr> | <expr> "*" <expr>
               | "(" <expr> ")" | "{" <expr> "}"
               | "val" <id> "=" <expr> ";" <expr> | <id>
               | <id> "=>" <expr> | <expr> "(" <expr> ")"
```

The precedence and associativity of operators are defined as follows:

Operator	Associativity	Precedence
*	left	1
+	left	2

### 3 ABSTRACT SYNTAX

The abstract syntax of FVAE is defined as follows:

Expressions	$e ::= n$	(Num)
	$  e + e$	(Add)
	$  e * e$	(Mul)
	$  \text{val } x = e; e$	(Val)
	$  x$	(Id)
	$  \lambda x. e$	(Fun)
	$  e(e)$	(App)

where

Numbers	$n \in \mathbb{Z}$	(BigInt)
Identifiers	$x \in \mathbb{X}$	(String)

### 4 SEMANTICS

We use the following notations in the semantics:

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

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

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

#### 4.1 Dynamic Scoping

The above semantics is defined with **static scoping** (or **lexical scoping**). We can augment it with **dynamic scoping** by changing the rule for function application as follows:

App $\frac{\sigma \vdash e_0 \Rightarrow \langle \lambda x. e_2, \sigma' \rangle \quad \sigma \vdash e_1 \Rightarrow v_1 \quad \sigma[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash e_0(e_1) \Rightarrow v_2}$
---