# F1VAE – VAE with First-Order Functions

## 1 INTRODUCTION

F1VAE is a toy language for the COSE212 course at Korea University. F1VAE stands for an extension of the VAE language with **first-order functions**, and it supports the following features:

- **integers**
- **basic arithmetic operators**: addition (+) and multiplication (*)
- **immutable variables** (`val`)
- **first-order functions** (`def`)

This document is the specification of F1VAE. 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 F1VAE.

## 2 CONCRETE SYNTAX

The concrete syntax of F1VAE 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" | "def"
<id>       ::= <idstart> <idcont>* butnot <keyword>

// programs
<program>  ::= <fdef>* <expr>

// function definitions
<fdef>     ::= "def" <id> "(" <id> ")" "=" <expr> ";"

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

The precedence and associativity of operators are defined as follows:

| Operator | Associativity | Precedence |
|---|---|---|
| * | left | 1 |
| + | left | 2 |

## 3 ABSTRACT SYNTAX

The abstract syntax of F1VAE is defined as follows:

$$
\begin{array}{llll}
\text{Programs} & \mathbb{P} \ni p & ::= f^* \; e & (\texttt{Program}) \\
\text{Function Definitions} & \mathbb{F} \ni f & ::= \texttt{def}\ x(x)\texttt{=}e & (\texttt{FunDef}) \\
\text{Expressions} & \mathbb{E} \ni e & ::= n & (\texttt{Num}) \\
 & & \mid e + e & (\texttt{Add}) \\
 & & \mid e \times e & (\texttt{Mul}) \\
 & & \mid \texttt{val}\ x\texttt{=}e\texttt{;}\ e & (\texttt{Val}) \\
 & & \mid x & (\texttt{Id}) \\
 & & \mid x(e) & (\texttt{App})
\end{array}
$$

where

$$
\begin{array}{lll}
\text{Environments} & \sigma \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{Z} & (\texttt{Env}) \\
\text{Function Environments} & \Lambda \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{F} & (\texttt{FEnv}) \\
\text{Integers} & n \in \mathbb{Z} & (\texttt{BigInt}) \\
\text{Identifiers} & x \in \mathbb{X} & (\texttt{String})
\end{array}
$$

## 4 SEMANTICS

For a given program $p = f^* \; e$, the initial function environment $\Lambda$ is constructed from $f^*$ by mapping each function name to its definition. We assume that there is no duplicate function name in $f^*$. The big-step operational (natural) semantics of F1VAE is defined as follows:

$$\boxed{\sigma, \Lambda \vdash e \Rightarrow n}$$

$$\texttt{Num}\ \frac{}{\sigma, \Lambda \vdash n \Rightarrow n}$$

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

$$\texttt{Val}\ \frac{\sigma, \Lambda \vdash e_1 \Rightarrow n_1 \qquad \sigma[x \mapsto n_1], \Lambda \vdash e_2 \Rightarrow n_2}{\sigma, \Lambda \vdash \texttt{val}\ x\texttt{=}e_1\texttt{;}\ e_2 \Rightarrow n_2} \qquad \texttt{Id}\ \frac{x \in \text{Domain}(\sigma)}{\sigma, \Lambda \vdash x \Rightarrow \sigma(x)}$$

$$\texttt{App}\ \frac{\sigma, \Lambda \vdash e_1 \Rightarrow n_1 \qquad x_0 \in \text{Domain}(\Lambda) \qquad \Lambda(x_0) = \texttt{def}\ x_0(x_1)\texttt{=}e_2 \qquad [x_1 \mapsto n_1], \Lambda \vdash e_2 \Rightarrow n_2}{\sigma, \Lambda \vdash x_0(e_1) \Rightarrow n_2}$$

### 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:

$$\texttt{App}\ \frac{\sigma, \Lambda \vdash e_1 \Rightarrow n_1 \qquad x_0 \in \text{Domain}(\Lambda) \qquad \Lambda(x_0) = \texttt{def}\ x_0(x_1)\texttt{=}e_2 \qquad \sigma[x_1 \mapsto n_1], \Lambda \vdash e_2 \Rightarrow n_2}{\sigma, \Lambda \vdash x_0(e_1) \Rightarrow n_2}$$