FAE – AE with First-Class Functions

1 INTRODUCTION

FAE is a toy language for the COSE212 course at Korea University. FAE stands for an extension of the AE 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 (`*`)
- **first-class functions** (`=>`)

This document is the specification of FAE. 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 FAE.

2 CONCRETE SYNTAX

The concrete syntax of FAE 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>
<id>       ::= <idstart> <idcont>*

// expressions
<expr>     ::= <number> | <expr> "+" <expr> | <expr> "*" <expr> | <id>
             | "(" <expr> ")" | "{" <expr> "}"
             | <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 FAE is defined as follows:

$$
\begin{array}{llll}
\text{Expressions} & \mathbb{E} \ni e ::= n & (\texttt{Num}) & \\
& \quad\mid e + e & (\texttt{Add}) & \\
& \quad\mid e * e & (\texttt{Mul}) & \text{where} \quad \begin{array}{lll} \text{Numbers} & n \in \mathbb{Z} & (\texttt{BigInt}) \\ \text{Identifiers} & x \in \mathbb{X} & (\texttt{String}) \end{array} \\
& \quad\mid x & (\texttt{Id}) & \\
& \quad\mid \lambda x.e & (\texttt{Fun}) & \\
& \quad\mid e(e) & (\texttt{App}) &
\end{array}
$$

4 SEMANTICS

We use the following notations in the semantics:

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

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

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

$$\mathsf{Num}\ \frac{}{\sigma \vdash n \Rightarrow n}$$

$$\mathsf{Add}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2} \qquad \mathtt{Mul}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 * e_2 \Rightarrow n_1 \times n_2}$$

$$\mathtt{Id}\ \frac{x \in \text{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}$$

$$\mathsf{Fun}\ \frac{}{\sigma \vdash \lambda x.e \Rightarrow \langle \lambda x.e, \sigma \rangle}$$

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

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:

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