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 (*)
- **immutable variable definitions** (val)
- **first-class functions** (=>)

This document is the specification of FAE. First, Section 2 describes the concrete syntax, and Section 3 describes the abstract syntax with the desugaring rules. 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>
<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	2
+	left	1

3 ABSTRACT SYNTAX

The abstract syntax of FAE is defined as follows:

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

The semantics of the remaining cases are defined with the following desugaring rules:

$$
\mathcal{D}[\![\text{val } x = e_1;\ e_2]\!] = (\lambda x.\mathcal{D}[\![e_2]\!])(\mathcal{D}[\![e_1]\!])
$$

The omitted cases recursively apply the desugaring rule to sub-expressions.

4 SEMANTICS

We use the following notations in the semantics:

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

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

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

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

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

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

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

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

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:

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