LFAE – FAE with Lazy Evaluation

1 INTRODUCTION

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

This document is the specification of LFAE. 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 LFAE.

2 CONCRETE SYNTAX

The concrete syntax of LFAE 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:

Description	Operator	Precedence	Associativity
Multiplicative	`*`	1	left
Additive	`+`	2	

3 ABSTRACT SYNTAX

The abstract syntax of LFAE is defined as follows:

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

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

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

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

4 SEMANTICS

We use the following notations in the semantics:

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

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

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

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

$$\texttt{Add}\ \frac{\sigma \vdash e_1 \Rightarrow v_1 \qquad v_1 \Downarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow v_2 \qquad v_2 \Downarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2}$$

$$\texttt{Mul}\ \frac{\sigma \vdash e_1 \Rightarrow v_1 \qquad v_1 \Downarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow v_2 \qquad v_2 \Downarrow n_2}{\sigma \vdash e_1 * e_2 \Rightarrow n_1 \times n_2}$$

$$\texttt{Id}\ \frac{x \in \text{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)} \qquad \texttt{Fun}\ \frac{}{\sigma \vdash \lambda x.e \Rightarrow \langle \lambda x.e, \sigma \rangle}$$

$$\texttt{App}\ \frac{\sigma \vdash e_0 \Rightarrow v_0 \qquad v_0 \Downarrow \langle \lambda x.e_2, \sigma' \rangle \qquad \sigma'[x \mapsto \langle\!\langle e_1, \sigma \rangle\!\rangle] \vdash e_2 \Rightarrow v_2}{\sigma \vdash e_0(e_1) \Rightarrow v_2}$$

with the following auxiliary function for strict evaluation:

$$\boxed{v \Downarrow v}$$

$$\texttt{StrictNum}\ \frac{}{n \Downarrow n} \qquad \texttt{StrictClo}\ \frac{}{\langle \lambda x.e, \sigma \rangle \Downarrow \langle \lambda x.e, \sigma \rangle} \qquad \texttt{StrictExpr}\ \frac{\sigma \vdash e \Rightarrow v \qquad v \Downarrow v'}{\langle\!\langle e, \sigma \rangle\!\rangle \Downarrow v'}$$