

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:

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

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

$$\mathcal{D}[\text{val } x = e; e'] = (\lambda x. \mathcal{D}[e']) (\mathcal{D}[e])$$

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

4 SEMANTICS

We use the following notations in the semantics:

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

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

$$\begin{array}{c}
 \boxed{\sigma \vdash e \Rightarrow v} \\
 \\
 \text{Num} \frac{}{\sigma \vdash n \Rightarrow n} \\
 \\
 \text{Add} \frac{\sigma \vdash e_1 \Rightarrow v_1 \quad v_1 \Downarrow n_1 \quad \sigma \vdash e_2 \Rightarrow v_2 \quad v_2 \Downarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2} \\
 \\
 \text{Mul} \frac{\sigma \vdash e_1 \Rightarrow v_1 \quad v_1 \Downarrow n_1 \quad \sigma \vdash e_2 \Rightarrow v_2 \quad v_2 \Downarrow n_2}{\sigma \vdash e_1 * e_2 \Rightarrow n_1 \times n_2} \\
 \\
 \text{Id} \frac{x \in \text{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)} \quad \text{Fun} \frac{}{\sigma \vdash \lambda x. e \Rightarrow \langle \lambda x. e, \sigma \rangle} \\
 \\
 \text{App} \frac{\sigma \vdash e_0 \Rightarrow v_0 \quad v_0 \Downarrow \langle \lambda x. e_2, \sigma' \rangle \quad \sigma'[x \mapsto \langle\langle e_1, \sigma \rangle\rangle] \vdash e_2 \Rightarrow v_2}{\sigma \vdash e_0(e_1) \Rightarrow v_2}
 \end{array}$$

with the following auxiliary function for strict evaluation:

$$\begin{array}{c}
 \boxed{v \Downarrow v} \\
 \\
 \text{StrictNum} \frac{}{n \Downarrow n} \quad \text{StrictClo} \frac{}{\langle \lambda x. e, \sigma \rangle \Downarrow \langle \lambda x. e, \sigma \rangle} \quad \text{StrictExpr} \frac{\sigma \vdash e \Rightarrow v \quad v \Downarrow v'}{\langle\langle e, \sigma \rangle\rangle \Downarrow v'}
 \end{array}$$