

BFAE – FAE with Mutable Boxes

1 INTRODUCTION

BFAE is a toy language for the [COSE212](#) course at Korea University. BFAE stands for an extension of the [FAE](#) language with **mutable boxes**, 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** (=>)
- **mutable boxes** (Box)
- **box operations**: get (get) and set (set)
- **sequences** (;)

This document is the specification of BFAE. 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 BFAE.

2 CONCRETE SYNTAX

The concrete syntax of BFAE 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>    ::= "Box" | "val"
<id>         ::= <idstart> <idcont>* butnot <keyword>

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

The precedence and associativity of operators are defined as follows:

Description	Operator	Precedence	Associativity
Multiplicative	*	2	left
Additive	+	1	

3 ABSTRACT SYNTAX

The abstract syntax of BFAE is defined as follows:

Expressions	$\mathbb{E} \ni e ::= n$	(Num)	$\text{Box}(e)$	(NewBox)
	$e + e$	(Add)	$e.\text{get}$	(GetBox)
	$e * e$	(Mul)	$e.\text{set}(e)$	(SetBox)
	x	(Id)	$e; e$	(Seq)
	$\lambda x.e$	(Fun)		
	$e(e)$	(App)		

where

Numbers	$n \in \mathbb{Z}$	(BigInt)	Identifiers	$x \in \mathbb{X}$	(String)
---------	--------------------	----------	-------------	--------------------	----------

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

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

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)	Environments	$\sigma \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}$	(Env)	
	$ a$	(BoxV)		Addresses	$a \in \mathbb{A}$	(Addr)
	$ \langle \lambda x.e, \sigma \rangle$	(CloV)		Memories	$M \in \mathbb{A} \xrightarrow{\text{fin}} \mathbb{V}$	(Mem)

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

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

$$\begin{array}{c}
 \text{Num} \frac{}{\sigma, M \vdash n \Rightarrow n, M} \quad \text{Add} \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow n_2, M_2}{\sigma, M \vdash e_1 + e_2 \Rightarrow n_1 + n_2, M_2} \\
 \\
 \text{Mul} \frac{\sigma, M \vdash e_1 \Rightarrow n_1, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow n_2, M_2}{\sigma, M \vdash e_1 * e_2 \Rightarrow n_1 \times n_2, M_2} \\
 \\
 \text{Id} \frac{x \in \text{Domain}(\sigma)}{\sigma, M \vdash x \Rightarrow \sigma(x), M} \quad \text{Fun} \frac{}{\sigma, M \vdash \lambda x.e \Rightarrow \langle \lambda x.e, \sigma \rangle, M} \\
 \\
 \text{App} \frac{\sigma, M \vdash e_1 \Rightarrow \langle \lambda x.e_3, \sigma' \rangle, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow v_2, M_2 \quad \sigma'[x \mapsto v_2], M_2 \vdash e_3 \Rightarrow v_3, M_3}{\sigma, M \vdash e_1(e_2) \Rightarrow v_3, M_3} \\
 \\
 \text{NewBox} \frac{\sigma, M \vdash e \Rightarrow v, M_1 \quad a \notin \text{Domain}(M_1)}{\sigma, M \vdash \text{Box}(e) \Rightarrow a, M_1[a \mapsto v]} \\
 \\
 \text{GetBox} \frac{\sigma, M \vdash e \Rightarrow a, M_1}{\sigma, M \vdash e.\text{get} \Rightarrow M_1(a), M_1} \quad \text{SetBox} \frac{\sigma, M \vdash e_1 \Rightarrow a, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow v, M_2}{\sigma, M \vdash e_1.\text{set}(e_2) \Rightarrow v, M_2[a \mapsto v]} \\
 \\
 \text{Seq} \frac{\sigma, M \vdash e_1 \Rightarrow _, M_1 \quad \sigma, M_1 \vdash e_2 \Rightarrow v_2, M_2}{\sigma, M \vdash e_1; e_2 \Rightarrow v_2, M_2}
 \end{array}$$