

# F AE-cps – F AE with Continuation-Passing Style

## 1 INTRODUCTION

F AE-cps is a toy language for the [COSE212](#) course at Korea University. F AE-cps stands for the [F AE](#) language with the **continuation-passing style (CPS)**. Since it has the same syntax and semantics as F AE, it supports the following features:

- **integers**
- **basic arithmetic operators**: addition (+) and multiplication (\*)
- **first-class functions**

This document is the specification of F AE-cps. While it has the same syntax and semantics as F AE, Section 2 and Section 3 repeat the concrete and abstract syntax parts for completeness, respectively. Section 4 redefines the same semantics in a small-step operational (reduction) semantics style rather than a big-step style.

## 2 CONCRETE SYNTAX

The concrete syntax of F AE-cps 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-cps is defined as follows:

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

### 4 SEMANTICS

We use the following notations in the semantics:

|               |   |               |
|---------------|---|---------------|
| Values        | $\mathbb{V} \ni v ::= n$                                    | (NumV)        |
|               | $  \langle \lambda x. e, \sigma \rangle$                    | (CloV)        |
| Environments  | $\sigma \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}$ | (Env)         |
| Continuations | $\mathbb{K} \ni \kappa ::= \square$                         | (EmptyK)      |
|               | $  (\sigma \vdash e) :: \kappa$                             | (EvalK)       |
|               | $  (+) :: \kappa$   | (AddK)        |
|               | $  (\times) :: \kappa$                                      | (MulK)        |
|               | $  (@) :: \kappa$   | (AppK)        |
| Value Stacks  | $\mathbb{S} \ni s ::= \blacksquare \mid v :: s$             | (List[Value]) |

The small-step operational (reduction) semantics of FAE-cps is defined as follows:

|                  |  |  |
|------------------|--|--|
|                  | $\langle \kappa \parallel s \rangle \rightarrow \langle \kappa \parallel s \rangle$        |  |
| Num              | $\langle (\sigma \vdash n) :: \kappa \parallel s \rangle$                                  | $\rightarrow \langle \kappa \parallel n :: s \rangle$  |
| Add <sub>1</sub> | $\langle (\sigma \vdash e_1 + e_2) :: \kappa \parallel s \rangle$                          | $\rightarrow \langle (\sigma \vdash e_1) :: (\sigma \vdash e_2) :: (+) :: \kappa \parallel s \rangle$      |
| Add <sub>2</sub> | $\langle (+) :: \kappa \parallel n_2 :: n_1 :: s \rangle$                                  | $\rightarrow \langle \kappa \parallel (n_1 + n_2) :: s \rangle$  |
| Mul <sub>1</sub> | $\langle (\sigma \vdash e_1 \times e_2) :: \kappa \parallel s \rangle$                     | $\rightarrow \langle (\sigma \vdash e_1) :: (\sigma \vdash e_2) :: (\times) :: \kappa \parallel s \rangle$ |
| Mul <sub>2</sub> | $\langle (\times) :: \kappa \parallel n_2 :: n_1 :: s \rangle$                             | $\rightarrow \langle \kappa \parallel (n_1 \times n_2) :: s \rangle$                                       |
| Id               | $\langle (\sigma \vdash x) :: \kappa \parallel s \rangle$                                  | $\rightarrow \langle \kappa \parallel \sigma(x) :: s \rangle$  |
| Fun              | $\langle (\sigma \vdash \lambda x. e) :: \kappa \parallel s \rangle$                       | $\rightarrow \langle \kappa \parallel \langle \lambda x. e, \sigma \rangle :: s \rangle$                   |
| App <sub>1</sub> | $\langle (\sigma \vdash e_1(e_2)) :: \kappa \parallel s \rangle$                           | $\rightarrow \langle (\sigma \vdash e_2) :: (\sigma \vdash e_1) :: (@) :: \kappa \parallel s \rangle$      |
| App <sub>2</sub> | $\langle (@) :: \kappa \parallel v_2 :: \langle \lambda x. e, \sigma \rangle :: s \rangle$ | $\rightarrow \langle (\sigma[x \mapsto v_2] \vdash e) :: \kappa \parallel s \rangle$                       |

where  $\rightarrow^*$  is the reflexive-transitive closure of  $\rightarrow$  and denotes the repeated reduction:

$$\langle \kappa \parallel s \rangle \rightarrow^* \langle \kappa \parallel s \rangle$$

$$\frac{\langle \kappa \parallel s \rangle \rightarrow^* \langle \kappa' \parallel s' \rangle \quad \langle \kappa' \parallel s' \rangle \rightarrow \langle \kappa'' \parallel s'' \rangle}{\langle \kappa \parallel s \rangle \rightarrow^* \langle \kappa'' \parallel s'' \rangle}$$

The evaluation result of an expression  $e$  is the value  $v$  if

$$\langle (\emptyset \vdash e) :: \square \parallel \blacksquare \rangle \rightarrow^* \langle \square \parallel v :: \blacksquare \rangle$$