RFAE – FAE with Recursion and Conditionals

1 INTRODUCTION

RFAE is a toy language for the COSE212 course at Korea University. RFAE stands for an extension of the FAE language with **recursion** and **conditionals**, and it supports the following features:

- **integers**
- **arithmetic operators**: negation (`-`), addition (`+`), subtraction (`-`), multiplication (`*`), division (`/`), and modulo (`%`)
- **first-class functions**
- **recursive functions** (`def`)
- **conditionals** (`if-else`)
- **boolean values** (`true` and `false`)
- **arithmetic comparison operators**: equality (`==` and `!=`) and relational (`<`, `>`, `<=`, and `>=`)
- **logical operators**: conjunction (`&&`), disjunction (`||`), and negation (`!`)

This document is the specification of RFAE. 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 RFAE.

2 CONCRETE SYNTAX

The concrete syntax of RFAE 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>  ::= "true" | "false" | "def" | "if" | "else"
<id>       ::= <idstart> <idcont>* butnot <keyword>

// expressions
<expr>     ::= <id> | <number> | "true" | "false"
             | <uop> <expr> | <expr> <bop> <expr>
             | "(" <expr> ")" | "{" <expr> "}"
             | <id> "=>" <expr> | <expr> "(" <expr> ")"
             | "def" <id> "(" <id> ")" "=" <expr> ";" <expr>
             | "if" "(" <expr> ")" <expr> "else" <expr>

// operators
<uop>      ::= "-" | "!"
<bop>      ::= "+" | "-" | "*" | "/" | "%" | "&&" | "||"
             | "==" | "!=" | "<" | "<=" | ">" | ">="
```

The precedence and associativity of operators are defined as follows:

Description	Operator	Precedence	Associativity
Unary	`-`, `!`	1	right
Multiplicative	`*`, `/`, `%`	2	left
Additive	`+`, `-`	3	
Relational	`<`, `<=`, `>`, `>=`	4	
Equality	`==`, `!=`	5	
Logical Conjunction	`&&`	6	
Logical Disjunction	`||`	7	

3 ABSTRACT SYNTAX

The abstract syntax of RFAE is defined as follows:

$$
\begin{array}{lll}
\text{Expressions} & \mathbb{E} \ni e ::= n & \texttt{(Num)} \\
& \mid b & \texttt{(Bool)} \\
& \mid x & \texttt{(Id)} \\
& \mid e + e & \texttt{(Add)} \\
& \mid e \times e & \texttt{(Mul)} \\
& \mid e \div e & \texttt{(Div)} \\
& \mid e \ \texttt{mod}\ e & \texttt{(Mod)} \\
& \mid e = e & \texttt{(Eq)} \\
& \mid e < e & \texttt{(Lt)} \\
& \mid \lambda x.e & \texttt{(Fun)} \\
& \mid \texttt{def}\ x(x) = e; e & \texttt{(Rec)} \\
& \mid e(e) & \texttt{(App)} \\
& \mid \texttt{if}\ (e)\ e\ \texttt{else}\ e & \texttt{(If)}
\end{array}
$$

$$
\begin{array}{llll}
\text{where} & \text{Integers} & n \in \mathbb{Z} & \texttt{(BigInt)} \\
& \text{Booleans} & b \in \mathbb{B} & \texttt{(Boolean)} \\
& \text{Identifiers} & x \in \mathbb{X} & \texttt{(String)}
\end{array}
$$

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

$$
\begin{array}{ll}
\mathcal{D}[\![\texttt{-}\ e]\!] & = \mathcal{D}[\![e]\!] \ \texttt{*}\ \texttt{(-1)} \\
\mathcal{D}[\![\texttt{!}\ e]\!] & = \texttt{if}\ (\mathcal{D}[\![e]\!])\ \texttt{false else true} \\
\mathcal{D}[\![e_1\ \texttt{-}\ e_2]\!] & = \mathcal{D}[\![e_1]\!] + \mathcal{D}[\![\texttt{-}\ e_2]\!] \\
\mathcal{D}[\![e_1\ \texttt{\&\&}\ e_2]\!] & = \texttt{if}\ (\mathcal{D}[\![e_1]\!])\ \mathcal{D}[\![e_2]\!]\ \texttt{else false} \\
\mathcal{D}[\![e_1\ \texttt{||}\ e_2]\!] & = \texttt{if}\ (\mathcal{D}[\![e_1]\!])\ \texttt{true else}\ \mathcal{D}[\![e_2]\!]
\end{array}
$$

$$
\begin{array}{ll}
\mathcal{D}[\![e_1\ \texttt{!=}\ e_2]\!] & = \mathcal{D}[\![\texttt{!}\ (e_1\ \texttt{==}\ e_2)]\!] \\
\mathcal{D}[\![e_1\ \texttt{<=}\ e_2]\!] & = \mathcal{D}[\![(e_1\ \texttt{<}\ e_2)\ \texttt{||}\ (e_1\ \texttt{==}\ e_2)]\!] \\
\mathcal{D}[\![e_1\ \texttt{>}\ e_2]\!] & = \mathcal{D}[\![\texttt{!}\ (e_1\ \texttt{<=}\ e_2)]\!] \\
\mathcal{D}[\![e_1\ \texttt{>=}\ e_2]\!] & = \mathcal{D}[\![\texttt{!}\ (e_1\ \texttt{<}\ e_2)]\!]
\end{array}
$$

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

4 SEMANTICS

We use the following notations in the semantics:

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

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

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

$$\texttt{Num}\ \frac{}{\sigma \vdash n \Rightarrow n} \qquad \texttt{Bool}\ \frac{}{\sigma \vdash b \Rightarrow b} \qquad \texttt{Id}\ \frac{x \in \mathrm{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}$$

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

$$\texttt{Div}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2 \qquad n_2 \neq 0}{\sigma \vdash e_1 \div e_2 \Rightarrow n_1 \div n_2} \qquad \texttt{Mod}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2 \qquad n_2 \neq 0}{\sigma \vdash e_1\ \texttt{mod}\ e_2 \Rightarrow n_1\ \texttt{mod}\ n_2}$$

$$\texttt{Eq}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 = e_2 \Rightarrow n_1 = n_2} \qquad \texttt{Lt}\ \frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 < e_2 \Rightarrow n_1 < n_2}$$

$$\texttt{Fun}\ \frac{}{\sigma \vdash \lambda x.e \Rightarrow \langle \lambda x.e, \sigma \rangle} \qquad \texttt{Rec}\ \frac{\sigma' = \sigma[x_0 \mapsto \langle \lambda x_1.e_0, \sigma' \rangle] \qquad \sigma' \vdash e_1 \Rightarrow v_1}{\sigma \vdash \texttt{def}\ x_0(x_1) = e_0; e_1 \Rightarrow v_1}$$

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

$$\texttt{If}_T\ \frac{\sigma \vdash e_0 \Rightarrow \texttt{true} \qquad \sigma \vdash e_1 \Rightarrow v_1}{\sigma \vdash \texttt{if}\ (e_0)\ e_1\ \texttt{else}\ e_2 \Rightarrow v_1} \qquad \texttt{If}_F\ \frac{\sigma \vdash e_0 \Rightarrow \texttt{false} \qquad \sigma \vdash e_2 \Rightarrow v_2}{\sigma \vdash \texttt{if}\ (e_0)\ e_1\ \texttt{else}\ e_2 \Rightarrow v_2}$$