TRFAE – RFAE with Type System

1 INTRODUCTION

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

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

This document is the specification of TRFAE. First, Section 2 describes the concrete syntax, and Section 3 describes the abstract syntax with the desugaring rules. Then, Section 4 describes the type system. Finally, Section 5 describes the big-step operational (natural) semantics of TRFAE.

2 CONCRETE SYNTAX

The concrete syntax of TRFAE 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" | "Number" | "Boolean"
<id>       ::= <idstart> <idcont>* butnot <keyword>

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

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

// types
<type> ::= "(" <type> ")" | "Number" | "Boolean" | <type> "=>" <type>
```

For types, the arrow (=>) operator is right-associative. For expressions, 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 TRFAE is defined as follows:

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

$$
\begin{array}{lrll}
\text{Types} & \mathbb{T} \ni \tau ::= & \texttt{num} & (\texttt{NumT}) \\
& \mid & \texttt{bool} & (\texttt{BoolT}) \\
& \mid & \tau \to \tau & (\texttt{ArrowT})
\end{array}
$$

$$
\begin{array}{lcll}
\text{Numbers} & n & \in \mathbb{Z} & (\texttt{BigInt}) \\
\text{Identifiers} & x & \in \mathbb{X} & (\texttt{String}) \\
\text{Booleans} & b & \in \mathbb{B} = \{\texttt{true}, \texttt{false}\} & (\texttt{Boolean})
\end{array}
$$

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

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

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

4 TYPE SYSTEM

This section explains type system of TRFAE, and we use the following notations:

$$\text{Type Environments} \quad \Gamma \in \mathbb{X} \xrightarrow{\text{fin}} \mathbb{T} \quad (\texttt{TypeEnv})$$

In the type system, type checking is defined with the following typing rules:

$$\boxed{\Gamma \vdash e : \tau}$$

$$\tau\texttt{-Num}\ \frac{}{\Gamma \vdash n : \texttt{num}} \qquad \tau\texttt{-Bool}\ \frac{}{\Gamma \vdash b : \texttt{bool}}$$

$$\tau\texttt{-Add}\ \frac{\Gamma \vdash e_1 : \texttt{num} \qquad \Gamma \vdash e_2 : \texttt{num}}{\Gamma \vdash e_1 + e_2 : \texttt{num}} \qquad \tau\texttt{-Mul}\ \frac{\Gamma \vdash e_1 : \texttt{num} \qquad \Gamma \vdash e_2 : \texttt{num}}{\Gamma \vdash e_1 \times e_2 : \texttt{num}}$$

$$\tau\texttt{-Div}\ \frac{\Gamma \vdash e_1 : \texttt{num} \qquad \Gamma \vdash e_2 : \texttt{num}}{\Gamma \vdash e_1 \div e_2 : \texttt{num}} \qquad \tau\texttt{-Mod}\ \frac{\Gamma \vdash e_1 : \texttt{num} \qquad \Gamma \vdash e_2 : \texttt{num}}{\Gamma \vdash e_1 \ \texttt{mod}\ e_2 : \texttt{num}}$$

$$\tau\texttt{-Eq}\ \frac{\Gamma \vdash e_1 : \texttt{num} \qquad \Gamma \vdash e_2 : \texttt{num}}{\Gamma \vdash e_1 = e_2 : \texttt{bool}} \qquad \tau\texttt{-Lt}\ \frac{\Gamma \vdash e_1 : \texttt{num} \qquad \Gamma \vdash e_2 : \texttt{num}}{\Gamma \vdash e_1 < e_2 : \texttt{bool}}$$

$$\tau\texttt{-Val}\ \frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma[x : \tau_1] \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{val}\ x = e_1;\ e_2 : \tau_2} \qquad \tau\texttt{-Id}\ \frac{x \in \text{Domain}(\Gamma)}{\Gamma \vdash x : \Gamma(x)}$$

$$\tau\texttt{-Fun}\ \frac{\Gamma[x : \tau] \vdash e : \tau'}{\Gamma \vdash \lambda x{:}\tau.e : \tau \to \tau'} \qquad \tau\texttt{-App}\ \frac{\Gamma \vdash e_0 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_1 : \tau_1}{\Gamma \vdash e_0(e_1) : \tau_2}$$

$$\tau\texttt{-Rec}\ \frac{\Gamma[x_0 : \tau_1 \to \tau_2, x_1 : \tau_1] \vdash e_2 : \tau_2 \qquad \Gamma[x_0 : \tau_1 \to \tau_2] \vdash e_3 : \tau_3}{\Gamma \vdash \texttt{def}\ x_0(x_1{:}\tau_1){:}\tau_2 = e_2;\ e_3 : \tau_3}$$

$$\tau\texttt{-If}\ \frac{\Gamma \vdash e_0 : \texttt{bool} \qquad \Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \texttt{if}\ (e_0)\ e_1\ \texttt{else}\ e_2 : \tau}$$

5 SEMANTICS

We use the following notations in the semantics:

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

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

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

$$\texttt{Num}\ \frac{}{\sigma \vdash n \Rightarrow n} \qquad \texttt{Bool}\ \frac{}{\sigma \vdash b \Rightarrow b}$$

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

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

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

$$\texttt{Val}\ \frac{\sigma \vdash e_1 \Rightarrow v_1 \qquad \sigma[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash \texttt{val}\ x \texttt{ = } e_1\texttt{;}\ e_2 \Rightarrow v_2} \qquad \texttt{Id}\ \frac{x \in \mathrm{Domain}(\sigma)}{\sigma \vdash x \Rightarrow \sigma(x)}$$

$$\texttt{Fun}\ \frac{}{\sigma \vdash \lambda x.e \Rightarrow \langle \lambda x.e, \sigma \rangle} \qquad \texttt{Rec}\ \frac{\sigma' = \sigma[x_0 \mapsto \langle \lambda x_1.e_2, \sigma' \rangle] \qquad \sigma' \vdash e_3 \Rightarrow v_3}{\sigma \vdash \texttt{def}\ x_0(x_1 \texttt{:} \tau_1)\texttt{:}\tau_2 \texttt{ = } e_2\texttt{;}\ e_3 \Rightarrow v_3}$$

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

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