1 Type

This document describes how type checking is performed on a Clafer model. Type checking pertains only to constraints within the model. A constraint is a tree of expressions and each expression must be one of the following types:

- integer
- real
- string
- boolean
- clafer

An expression's type may depend on its call site.

```
abstract Y
    y : integer
    x
    [ y > 0 ]
```

The expression y in the constraint resolves to the type $integer$.

```
abstract Y
    y : integer
    [ #(y ++ x) ]
```

The expression y in the constraint resolves to the type $clafer$.

If a Clafer model has duplicate Clafer names, then there is a corresponding model with unique names. The remainder of the document assumes that each Clafer is uniquely named.

2 Notation

This section explains the notation used to describe the type rules.

2.1 Symbols

:: is shorthand for "type of".

$\vdash$ is shorthand for "entails".

x is a Clafer reference.

E, F, G are expressions.

```
abstract Car : string
    speed : integer
    [ speed > 0 ]
```

In the constraint, *speed* is a Clafer reference.

There are two leaf expressions: "*speed*" and "0"; and one super expression "*super* > 0".

$isNumeric$ is a predicate that maps $Type \rightarrow boolean$.

$$isNumeric(type) = \begin{cases} true & type \in \{integer, real\} \\ false & \text{otherwise} \end{cases}$$

$\mathbb{INTEGER}$ is any integer constant such as 1234 or -5678.

$\mathbb{REAL}$ is any real constant such as 3.14 or 2.718.

$\mathbb{STRING}$ is any string constant such as "this is a string".

τ, υ are type variables. A type variable is a placeholder for a type.

2.2 Type environment Γ

The type environment is a data structure for holding the types of Clafers. It contains a mapping from $Clafer \rightarrow Type$.

For example:

```
abstract Y : string
    y : integer
    x
    [ #(y ++ x) ]

X : Y
```

will have the type environment:

$$\Gamma = \{Y :: string,\ y :: integer,\ x :: clafer,\ X :: string\}$$

2.3 Type rule

The type system is specified in a series of formal rules.

$$\textsc{name of rule}\ \frac{statementA}{statementB}$$

The above rule says that if A holds, then B follows. Multiple statements are allowed above the bar, separated visually by a gap.

$$\textsc{name of rule}\ \frac{statementA \qquad statementB}{statementC}$$

The rule states that if A *and* B holds, then C follows.

Here are few examples how these rules can describe the type system.

$$\textsc{value}\ \frac{(x :: \tau) \in \Gamma}{\Gamma \vdash x :: \tau}$$

The *value* rule says: if $x :: \tau$ is in the type environment, then x resolves to type τ given Γ.

$$\textsc{eq}\ \frac{\Gamma \vdash E :: \tau \qquad \Gamma \vdash F :: \tau}{\Gamma \vdash E = F :: boolean}$$

The *eq* rule says: if we can prove that E *and* F type check to the same type given Γ then the expression $E = F$ type checks to *boolean* given Γ.

An expression is type correct iff we can find a tree of rule applications that prove it correct. See the last section for examples.

3 Clafer Type Rules

A clafer reference can always be treated as a clafer.

$$\textsc{clafer}\ \frac{}{\Gamma \vdash x :: clafer}$$

A clafer reference can be of type stored in the type environment.

$$\textsc{value}\ \frac{(x :: \tau) \in \Gamma}{\Gamma \vdash x :: \tau}$$

Constants.

$$\textsc{intconst}\ \frac{}{\Gamma \vdash \mathbb{INTEGER} :: integer}$$

$$\textsc{realconst}\ \frac{}{\Gamma \vdash \mathbb{REAL} :: real}$$

$$\textsc{strconst}\ \frac{}{\Gamma \vdash \mathbb{STRING} :: string}$$

Unary functions.

$$\textsc{not}\ \frac{\Gamma \vdash E :: boolean}{\Gamma \vdash !E :: boolean}$$

$$\textsc{cset}\ \frac{\Gamma \vdash E :: clafer}{\Gamma \vdash \#E :: integer}$$

$$\textsc{min}\ \frac{\Gamma \vdash E :: \tau \qquad isNumeric(\tau)}{\Gamma \vdash -E :: \tau}$$

Binary functions

$$\textsc{join}\ \frac{\Gamma \vdash E :: clafer \qquad \Gamma \vdash F :: \tau}{\Gamma \vdash E.F :: \tau}$$

$$\textsc{binbool}\ \frac{\Gamma \vdash E :: boolean \qquad \Gamma \vdash F :: boolean}{\Gamma \vdash E \odot F :: boolean} \quad \odot \in \{<=>, =>, ||, \&\&, xor\}$$

$$\textsc{eq}\ \frac{\Gamma \vdash E :: \tau \qquad \Gamma \vdash F :: \tau}{\Gamma \vdash E \oplus F :: boolean} \quad \oplus \in \{=, !=\}$$

$$\textsc{eqcast1}\ \frac{\Gamma \vdash E :: real \qquad \Gamma \vdash F :: integer}{\Gamma \vdash E \oplus F :: boolean} \quad \oplus \in \{=, !=\}$$

$$\textsc{eqcast2}\ \frac{\Gamma \vdash E :: integer \qquad \Gamma \vdash F :: real}{\Gamma \vdash E \oplus F :: boolean} \quad \oplus \in \{=, !=\}$$

$$\textsc{ineq}\ \frac{\Gamma \vdash E :: \tau \qquad \Gamma \vdash F :: \upsilon \qquad isNumeric(\tau) \qquad isNumeric(\upsilon)}{\Gamma \vdash E \otimes F :: boolean} \quad \otimes \in \{<, <=, >, >=\}$$

$$\textsc{in}\ \frac{\Gamma \vdash E :: clafer \qquad \Gamma \vdash F :: clafer}{\Gamma \vdash E \ominus F :: boolean} \quad \ominus \in \{in, not\ in\}$$

$$\textsc{setops}\ \frac{\Gamma \vdash E :: clafer \qquad \Gamma \vdash F :: clafer}{\Gamma \vdash E \oslash F :: clafer} \quad \oslash \in \{++, --, \&\}$$

$$\textsc{domain}\ \frac{\Gamma \vdash E :: clafer \qquad \Gamma \vdash F :: \tau}{\Gamma \vdash E <: F :: \tau}$$

$$\textsc{range}\ \frac{\Gamma \vdash E :: \tau \qquad \Gamma \vdash F :: clafer}{\Gamma \vdash E :> F :: \tau}$$

$$\textsc{numops}\ \frac{\Gamma \vdash E :: \tau \qquad \Gamma \vdash F :: \tau \qquad isNumeric(\tau)}{\Gamma \vdash E \diamond F :: \tau} \quad \diamond \in \{+, -, *, /\}$$

$$\textsc{numopscast1}\ \frac{\Gamma \vdash E :: real \qquad \Gamma \vdash F :: integer}{\Gamma \vdash E \diamond F :: real} \quad \diamond \in \{+, -, *, /\}$$

$$\textsc{numopscast2}\ \frac{\Gamma \vdash E :: integer \qquad \Gamma \vdash F :: real}{\Gamma \vdash E \diamond F :: real} \qquad \diamond \in \{+, -, *, /\}$$

$$\textsc{strconcat}\ \frac{\Gamma \vdash E :: string \qquad \Gamma \vdash F :: string}{\Gamma \vdash E + F :: string}$$

Ternary functions.

$$\textsc{ifthenelse}\ \frac{\Gamma \vdash E :: boolean \qquad \Gamma \vdash F :: \tau \qquad \Gamma \vdash G :: \tau}{\Gamma \vdash E => F\ else\ G :: \tau}$$

$$\textsc{ifthenelsecast1}\ \frac{\Gamma \vdash E :: boolean \qquad \Gamma \vdash F :: real \qquad \Gamma \vdash G :: integer}{\Gamma \vdash E => F\ else\ G :: real}$$

$$\textsc{ifthenelsecast2}\ \frac{\Gamma \vdash E :: boolean \qquad \Gamma \vdash F :: integer \qquad \Gamma \vdash G :: real}{\Gamma \vdash E => F\ else\ G :: real}$$

Quantified expressions create a new local type environment with the local names binded to the type. See the examples in the next section if this rule is unclear.

$$\textsc{quant}\ \frac{\Gamma \vdash E :: \tau \qquad \Gamma, a :: \tau, b :: \tau, ..., z :: \tau \vdash F :: \upsilon}{\Gamma \vdash \star\ a\ b\ ...\ z : E \mid F :: boolean} \qquad \star \in \{no, lone, one, some, all\}$$

4 Examples

4.1 Example one

Prove that the constraint in the following model is type correct.

```
car
    speed:integer
    [speed > 0]
```

$$\Gamma = \{car :: clafer, speed :: integer\}$$

$$\textsc{ineq}\ \frac{\textsc{value}\ \dfrac{(speed :: integer) \in \Gamma}{\Gamma \vdash speed :: integer} \qquad \textsc{intconst}\ \dfrac{}{\Gamma \vdash 0 :: integer} \qquad isNumeric(integer)}{\Gamma \vdash speed > 0 :: boolean}$$

4.2 Example two

Prove that the constraint in the following model is type correct.

```
car
	speed:integer
	[#speed = 0]
```

$$\Gamma = \{car :: clafer, speed :: integer\}$$

$$\textsc{eq}\ \dfrac{\textsc{cset}\ \dfrac{\textsc{clafer}\ \dfrac{}{\Gamma \vdash speed :: clafer}}{\Gamma \vdash \#speed :: integer} \qquad \textsc{intconst}\ \dfrac{}{\Gamma \vdash 0 :: integer}}{\Gamma \vdash \#speed = 0 :: boolean}$$

4.3 Example three

Prove that the constraint in the following model is type correct.

```
car
	speed:integer
	[some a : speed | a = 3]
```

$$\Gamma = \{car :: clafer, speed :: integer\}$$

$$\textsc{quant}\ \dfrac{\textsc{value}\ \dfrac{(speed :: integer) \in \Gamma}{\Gamma \vdash speed :: integer} \qquad \textsc{eq}\ \dfrac{\textsc{value}\ \dfrac{(a :: integer) \in \Gamma \cup \{a :: integer\}}{\Gamma \cup \{a :: integer\} \vdash a :: integer} \qquad \dfrac{}{\Gamma \cup \{a :: integer\} \vdash 3 :: integer}\ \textsc{intconst}}{\Gamma \cup \{a :: integer\} \vdash a = 3 :: boolean}}{\Gamma \vdash some\ a : speed \mid a = 3 :: boolean}$$