Typechecking rules

Martin Bråthen

November 30, 2019

x	Variable
$T(\_)$	Type of _
e_i	Expression
N	Integer type
B	Boolean type

Table 1: Notation

1 Arithmetic

$$\frac{T(e_0) \longrightarrow N, T(e_1) \longrightarrow N}{T(e_0 + e_1) \longrightarrow N} \quad \text{[add]}$$

$$\frac{T(e_0) \longrightarrow N, T(e_1) \longrightarrow N}{T(e_0 * e_1) \longrightarrow N} \quad \text{[mul]}$$

$$\frac{T(e_0) \longrightarrow N, T(e_1) \longrightarrow N}{T(e_0 / e_1) \longrightarrow N} \quad \text{[div]}$$

$$\frac{T(e_0) \longrightarrow N, T(e_1) \longrightarrow N}{T(e_0 \;\%\; e_1) \longrightarrow N} \quad \text{[mod]}$$

$$\frac{T(e) \longrightarrow N}{T(-e) \longrightarrow N} \quad \text{[neg]}$$

1.1 Examples

1.1.1 add

`1 + 2` is OK because both operands are integers. Result is also an integer.

1.1.2 mul

`5 * false` is not OK because both operands are not integers. Result would have been an integer.

1.1.3 div

`1 / 0` is OK because both operands are integers. Result would have been an integer.

1.1.4 mod

`true % 23` is not OK because both operands are not integers. Result would have been an integer.

1.1.5 neg

`-true` is not OK because the operand is not an integer. Result would have been an integer.

2 Boolean

$$\frac{T(e_0) \longrightarrow B, T(e_1) \longrightarrow B}{T(e_0\ AND\ e_1) \longrightarrow B} \quad \text{[and]}$$

$$\frac{T(e_0) \longrightarrow B, T(e_1) \longrightarrow B}{T(e_0\ OR\ e_1) \longrightarrow B} \quad \text{[or]}$$

$$\frac{T(e) \longrightarrow B}{T(NOT\ e) \longrightarrow B} \quad \text{[not]}$$

2.1 Examples

2.1.1 and

`true AND false` is OK because both operands are booleans. Result is also a boolean.

2.2 or

`1 OR false` is not OK because both operands are not booleans. Result would have been a boolean.

2.2.1 not

`!true` is OK because the operand is a boolean. Result is alo a boolean.

3 Comparison

$$\frac{T(e_0) \longrightarrow B, T(e_1) \longrightarrow B}{T(e_0 == e_1) \longrightarrow B} \qquad \text{[EQ1]}$$

$$\frac{T(e_0) \longrightarrow N, T(e_1) \longrightarrow N}{T(e_0 == e_1) \longrightarrow B} \qquad \text{[EQ2]}$$

$$\frac{T(e_0) \longrightarrow B, T(e_1) \longrightarrow B}{T(e_0 \; != e_1) \longrightarrow B} \qquad \text{[NEQ1]}$$

$$\frac{T(e_0) \longrightarrow N, T(e_1) \longrightarrow N}{T(e_0 \; != e_1) \longrightarrow B} \qquad \text{[NEQ2]}$$

3.1 Examples

3.1.1 EQ

- `true == false` is OK because both operands are booleans. Result is also a boolean.
- `15 == 15` is OK because both operands are integers. Result is a boolean.
- `15 == true` is not OK because the operands are of different types. Result is a boolean.

3.1.2 NEQ

- `true != false` is OK because both operands are booleans. Result is also a boolean.
- `15 != 15` is OK because both operands are integers. Result is a boolean.
- `15 != true` is not OK because the operands are of different types. Result is a boolean.

4 Statements

$$\frac{T(x) \longrightarrow N, T(e_1) \longrightarrow N}{T(x) \longrightarrow N} \qquad \text{[assign1]}$$

$$\frac{T(x) \longrightarrow B, T(e_1) \longrightarrow B}{T(x) \longrightarrow B} \qquad \text{[assign2]}$$

$$T(e) \longrightarrow B \qquad \text{[if]}$$

$$T(e) \longrightarrow B \qquad \text{[while]}$$

4.1 Examples

4.1.1 assign

```
let a : i32;
a = true;
```

Is not OK becuase the type of the variable is not the same as the assigned value.

4.1.2 if

```
if 1 + 2 {
    ...
}
```

Is not OK since the type of the expression is not boolean.

4.1.3 while

```
while 1 == 2 {
    ...
}
```

Is OK since the type of the expression is boolean.

5 Requirements

- Function definitions
- Commands (let, assignment, if then (else), while)
- Expressions (includig function calls)
- Primitive types (boolean, i32) and their literals
- Explicit types everywhere
- Explicit return(s)
- Your type checker should reject ill-typed programs according to your typing rules.

All of the above requirements have been met. Implementation was done by me.