The Whiley Language Specification

David J. Pearce School of Engineering and Computer Science Victoria University of Wellington, New Zealand djp@ecs.vuw.ac.nz

December 31, 2013

Contents

| 1 | Intr | oduction 3 |
|---|------|--------------------------------|
| | 1.1 | Overview |
| | 1.2 | Goals |
| | 1.3 | History |
| 2 | Lexi | ical Structure |
| | 2.1 | Indentation |
| | 2.2 | Blocks |
| | 2.3 | Whitespace |
| | 2.4 | Identifiers |
| 3 | Con | npilation Units |
| _ | 3.1 | Type Declarations |
| | 3.2 | Constant Declarations |
| | 3.3 | Function & Method Declarations |
| | 3.4 | Visibility Modifiers |
| | 3.5 | Packages |
| | 3.6 | Imports |
| 4 | Турс | es 6 |
| • | 4.1 | Overview |
| | 4.1 | Primitives |
| | 4.2 | |
| | | J J1 |
| | | 71 |
| | | 7.1 |
| | | 7 1 |
| | | 7 1 |
| | | 71 |
| | 4.3 | 71 |
| | 4.3 | V 1 |
| | | 7 1 |
| | | 1 71 |
| | 1 1 | 71 |
| | 4.4 | Union Types |
| | 4.5 | Intersection Types |
| | 4.6 | Negation Types |
| | 4.7 | Reference Types |
| | 4.8 | Subtyping |
| 5 | _ | ressions 10 |
| | 5 1 | Rinary Expressions |

| 6 | Statements | | | | | |
|---|------------|-----------------------|----|--|--|--|
| | | Variable Declarations | | | | |
| | 6.2 | Assign Statements | 12 | | | |
| | 6.3 | Return Statements | 12 | | | |
| | 6.4 | If/Else Statements | 12 | | | |
| | 6.5 | While Statements | 12 | | | |
| | 6.6 | Do/While Statements | 12 | | | |
| | 6.7 | For Statements | 12 | | | |
| | 6.8 | Switch Statements | 12 | | | |
| | 6.9 | Try/Catch Statements | 12 | | | |

Introduction

- 1.1 Overview
- 1.2 Goals
- 1.3 History

Lexical Structure

- 2.1 Indentation
- 2.2 Blocks
- 2.3 Whitespace
- 2.4 Identifiers

Compilation Units

- 3.1 Type Declarations
- 3.2 Constant Declarations
- 3.3 Function & Method Declarations
- 3.4 Visibility Modifiers
- 3.5 Packages
- 3.6 Imports

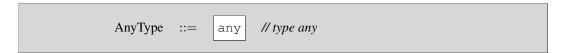
Types

4.1 Overview

Discuss syntactic versus semantic types.

4.2 Primitives

4.2.1 Any Type



Description.

Examples.

Semantics.

Notes.

4.2.2 Void Type



Description. The **void** type represents the type whose variables cannot exist! That is, they cannot hold any possible value. Void is used to represent the return type of a function which does not return anything. However, it is also used to represent the element type of an empty list of set.

Examples.

Semantics.

Notes. The void type is a subtype of everything; that is, it is bottom in the type lattice.

4.2.3 Null Type

| NullType ::= null // type null |
|--------------------------------|
|--------------------------------|

Description.

Examples.

Semantics.

Notes.

4.2.4 Bool Type

```
BoolType ::= bool // type bool
```

Description.

Examples.

Semantics.

Notes.

4.2.5 Char Type

Description.

Examples.

Semantics.

Notes.

4.2.6 Int Type

Description.

Examples.

Semantics. Notes. **Real Type** 4.2.7 RealType ::= real Description. Examples. Semantics. Notes. **Collection Types** 4.3 **Set Type** 4.3.1 { Type } SetType ::= Description. Examples. Semantics. Notes. Map Type 4.3.2 { | Type | => | Type | } MapType ::=Description. Examples. Semantics.

Notes.

4.3.3 List Type

| ListType ::= [Type] |
|-----------------------|
|-----------------------|

Description.

Examples.

Semantics.

Notes.

- 4.4 Union Types
- **4.5** Intersection Types
- **4.6** Negation Types
- 4.7 Reference Types
- 4.8 Subtyping

Discussion or present subtyping algorithm?

```
Cond [( | \&\& | | | + | |) Expr ]
   Expr
                                                   // Expressions
  Cond
                Append [ Cop Expr ]
                                                   // Condition Expressions
                Range\ [
                         ++ |Expr|
Append
                                                   // Append Expressions
                AddSub [ | ... | Expr ]
 Range
                                                   // Range Expressions
                MulDiv\ [\ (
AddSub
                                                   // Additive Expressions
                                                   // Multiplicative Expressions
MulDiv\\
                ???
  Index
                                                   // Index Expressions
```

Figure 5.1: Syntax for Binary Expressions

Expressions

5.1 Binary Expressions

```
// Terms
Term
        ::=
               Constant
                                                                                // Constant expressions
               Identifier \\
                                                                                // Identifier expressions
                             Expr_i)+
                                                                                // Tuple expressions
                   Expr
                                                                                // Bracketed expressions
                                                                                // Size expressions
                   Expr
                                [Expr_1(|,|Expr_i)^+]|)
               Identifier
                                                                                // Invocation expressions
                                                                                // Unary expressions
                new \mid Expr
                                                                                // Allocation expressions
                  |[Expr_1(|,|Expr_i)^*]|
                                                                                // Set expressions
                    |Expr_1| \Rightarrow |Expr_1'| \left( \mid, \mid Expr_i \mid \Rightarrow |Expr_i'|^* \right) | 
                                                                                // Map expressions
                                  Expr_i)*]|]
                                                                                // List expressions
                                     | , | n_i | : | Expr_i )^* ] | 
                                                                                // Record expressions
```

Figure 5.2: Syntax for Term Expressions

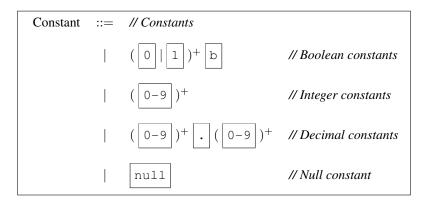


Figure 5.3: Syntax for Constant Expressions

Identifier ::=
$$(\begin{bmatrix} - \end{bmatrix} | \begin{bmatrix} a-z \end{bmatrix} | \begin{bmatrix} A-Z \end{bmatrix}) (\begin{bmatrix} - \end{bmatrix} | \begin{bmatrix} a-z \end{bmatrix} | \begin{bmatrix} A-Z \end{bmatrix} | \begin{bmatrix} 0-9 \end{bmatrix})^*$$
 // Identifiers

Figure 5.4: Syntax for Identifiers

Statements

- **6.1 Variable Declarations**
- 6.2 Assign Statements
- **6.3** Return Statements
- **6.4** If/Else Statements
- **6.5** While Statements
- 6.6 Do/While Statements
- **6.7** For Statements
- **6.8** Switch Statements
- **6.9** Try/Catch Statements