# 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**

1.1 1.2 1.3 <b>Lexi</b> 2.1 2.2 2.3 2.4	Overview Goals History  kical Structure Indentation Blocks	
1.3 Lexi 2.1 2.2 2.3	History	
Lexi 2.1 2.2 2.3	kical Structure Indentation	•
2.1 2.2 2.3	Indentation	
2.2 2.3	Blocks	
2.3		
	XXXII.	
2.4	Whitespace	
	Identifiers	
Con	mpilation Units	
3.1		
3.2	Constant Declarations	
3.3		
3.4		
3.5		
3.6		
Type	oes	
4.1		(
4.2		
	· · · · ·	
	· · · · · · · · · · · · · · · · · · ·	
	<b>₹ ≜</b>	
	· · · · · · · · · · · · · · · · · · ·	
	<b>71</b>	
	· · · · · · · · · · · · · · · · · · ·	
4.3	* •	
	* *	
	* **	
4.4	· · · · · · · · · · · · · · · · · · ·	
4.5		
4.6	<b>₹ ≜</b>	
4.7		
4.8	**	
Evn	pressions	1
	3.1 3.2 3.3 3.4 3.5 3.6 <b>Tyl</b> 4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8	3.2 Constant Declarations 3.3 Function & Method Declarations 3.4 Visibility Modifiers 3.5 Packages 3.6 Imports  Types 4.1 Overview 4.2 Primitives 4.2.1 Any Type 4.2.2 Void Type 4.2.3 Null Type 4.2.4 Bool Type 4.2.5 Char Type 4.2.6 Int Type 4.2.7 Real Type 4.2.7 Real Type 4.3.1 Set Type 4.3.1 Set Type 4.3.2 Map Type 4.3.2 Map Type 4.3.3 List Type 4.4.4 Union Types 4.5 Intersection Types 4.5 Intersection Types 4.6 Negation Types 4.7 Reference Types 4.8 Subtyping  Expressions

	ements	
	Variable Declarations	
6.2	Assign Statements	13
6.3	Return Statements	13
6.4	If/Else Statements	13
6.5	While Statements	13
6.6	Do/While Statements	13
	For Statements	
6.8	Switch Statements	13
6.9	Try/Catch Statements	13

## 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

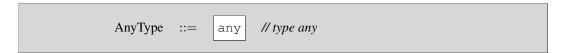
## **Types**

#### 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

UnionType ::=	IntersectionType (   IntersectionType )+
---------------	--

Description.

Examples.

Semantics.

Notes.

### 4.5 Intersection Types

```
IntersectionType ::= TermType ( & TermType )+
```

Description.

Examples.

Semantics.

Notes.

### 4.6 Negation Types



Description.

Examples.

Semantics.

Notes.

### 4.7 Reference Types

ReferenceType ::= & Type
--------------------------

Description.

Examples.

Semantics.

Notes.

### 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\ [\ (
                                                   // Additive Expressions
AddSub
                                                   // 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

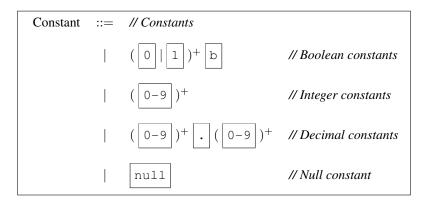


Figure 5.3: Syntax for Constant Expressions



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