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

Chapter 1

Introduction

1.1 Overview

1.2 Goals

1.3 History

Chapter 2

Lexical Structure

2.1 Indentation

2.2 Blocks

2.3 Whitespace

2.4 Identifiers

Chapter 3

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

Chapter 4

Types

4.1 Overview

Discuss syntactic versus semantic types.

4.2 Primitives

4.2.1 Any Type

AnyType ::= `any` *// type any*

Description.

Examples.

Semantics.

Notes.

4.2.2 Void Type

VoidType ::= `void` *// type void*

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

CharType ::= `char`

Description.

Examples.

Semantics.

Notes.

4.2.6 Int Type

IntType ::= `int`

Description.

Examples.

Semantics.

Notes.

4.2.7 Real Type

RealType ::= `real`

Description.

Examples.

Semantics.

Notes.

4.3 Collection Types

4.3.1 Set Type

SetType ::= { Type }

Description.

Examples.

Semantics.

Notes.

4.3.2 Map Type

MapType ::= { Type => Type }

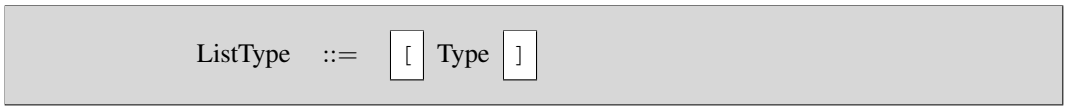
Description.

Examples.

Semantics.

Notes.

4.3.3 List 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?

Expr	::=	*Cond* [(`&&` | `||`) *Expr*]	*// Expressions*
Cond	::=	*Append* [*Cop Expr*]	*// Condition Expressions*
Append	::=	*Range* [`++` *Expr*]	*// Append Expressions*
Range	::=	*AddSub* [`..` *Expr*]	*// Range Expressions*
AddSub	::=	*MulDiv* [(`+` | `-`) *Expr*]	*// Additive Expressions*
MulDiv	::=	*Index* [(`*` | `/` | `%`) *Expr*]	*// Multiplicative Expressions*
Index	::=	???	*// Index Expressions*

Figure 5.1: Syntax for Binary Expressions

Chapter 5

Expressions

5.1 Binary Expressions

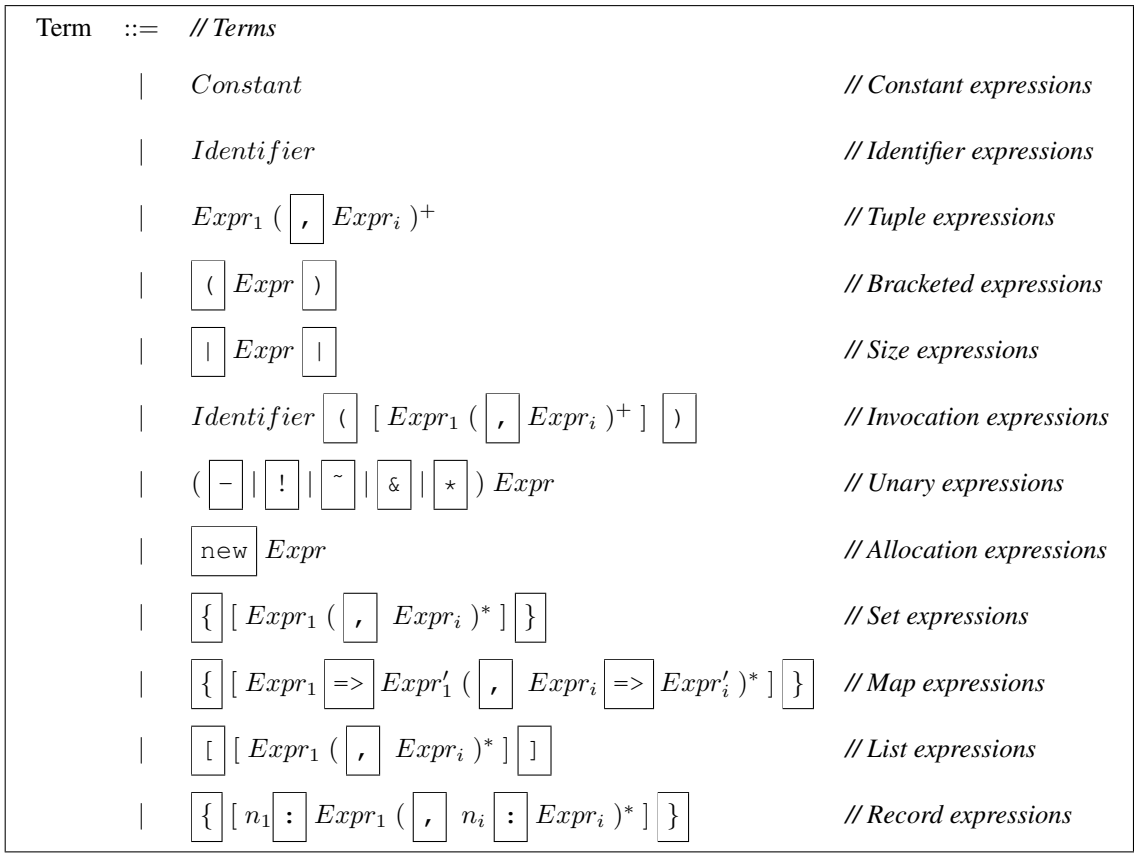


Figure 5.2: Syntax for Term Expressions

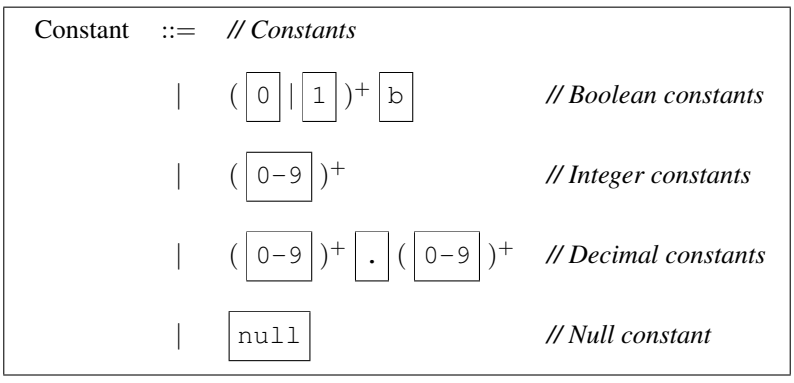


Figure 5.3: Syntax for Constant Expressions

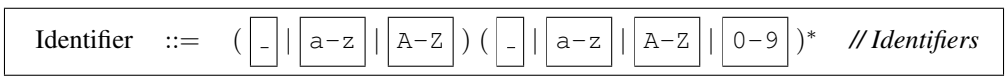


Figure 5.4: Syntax for Identifiers

Chapter 6

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