CFR-2
The Coral Standard Runtime Library

Version 0.1

Markéta Lisová

May 11, 2015

Contents

Preface

The *Coral Standard Runtime Library* (the CSRL[1], not to be confused with CLS, which is the Coral Language Specification, CFR-0) is a specification for a set of Coral modules, types and classes that come shipped together with every CVM (Coral Virtual Machine, CFR-1).

Some features of the CSL are not available on every platform/CVM combination, and these differences are documented in this document, along with advices on how to properly query for their presence or absence.

Many functions that are a part of the CSL are implemented natively in particular CVMs, mostly because they use the low-level APIs provided by a CVM, or simply for performance reasons, and therefore it is not possible to list their source in this specification, only function and method signatures.

Status of This CFR

This CFR is active and mandatory for every proper Coral implementation, without exceptions. The status of this CFR is not likely to change.

Some particular components defined in this CFR are only optional, as specified, and need not to be implemented in a proper Coral implementation.

[1]Or, as a mnemonic, CSRLib.

Chapter 1

The Lang Module

The `Lang~[coral]` module is implicitly imported into every Coral source, and there is no way to opt-out of this behaviour.

For brevity, we will omit the "`~[coral]`" in this and every following chapter, and unless specified otherwise, it will be implicitly added to every occurrence of any CSL module as vendor specifier.

Every Coral source can be viewed as starting with the following code:

```
use Lang~[coral].{_}
```

The `Lang` module is designed to be as much not intrusive as possible, e.g. the `Lang.Object` does not define any instance methods.[1] To allow users to use some typical methods querying `Object` instances, Coral offers attribute syntax along with those methods predefined in the `Lang.Predef` object (§1.10).

1.1 The Any Type

```
module Lang

type Any := ;; implementation-defined, blank slate
```

1.2 The Object Class

```
module Lang
```

[1]This is very much unlike e.g. Java, whose `java.lang.Object` is pre-flooded with methods.

```
class Object extends Any
end class

object Object
begin
  message new (params: Variadic_Arguments): self.type
  end message

  constructor ()
  end constructor

  clone ()
  end clone
end object
```

1.3 The Nothing & Undefined Classes

```
module Lang

immutable sealed class Nothing extends /* every type */
end class

immutable sealed class Undefined extends Nothing
end class
```

1.4 The Unit Class

```
module Lang

immutable sealed class Unit extends Object
end class
```

1.5 The Type Class

```
module Lang

abstract class Type [T] extends Object
end class

object Type
begin
end object
```

1.6 The ‘Class’ Class

```
module Lang

class Class [T] extends Type[T]
end class

object Class
begin
  message new (name: Symbol, &init: Class[_] -> Unit): Class[_]
  end message
end object
```

1.7 The Metaclass Class

```
module Lang

class Metaclass [T <: Class[T]] extends Type[T]
end class

object Metaclass
begin
end object
```

1.8 The Magnitude Class

```
module Lang

abstract class Magnitude extends Object
end class

object Magnitude
begin
end object
```

1.9 The Number Class

```
module Lang

trait Number_Like extends Magnitude
  constraint value as Number_Like
  end constraint

  message as_number (): Number
  end message
end trait

abstract sealed class Number
    extends Magnitude
       with Number_Like
end class

object Number
begin
  type Integer := ;; implementation-defined
  end type
  type Integer_Unsigned := ;; implementation-defined
    ;; could be: Integer with constraint { value >= 0 }
  end type
  type Real := ;; implementation-defined
  end type
  type Decimal := ;; implementation-defined
  end type
  type Decimal_Unsigned := ;; implementation-defined
    ;; could be: Decimal with constraint { value >= 0 }
  end type
  type Fixed_Point_Number [
      Delta  <: Literal_Singleton_Type[Decimal_Unsigned],
      Digits <: Literal_Singleton_Type[Integer_Unsigned],
      Range  <: (Literal_Singleton_Type[Decimal],
                 Literal_Singleton_Type[Decimal],
                 Literal_Singleton_Type[Boolean])]
      := ;; implementation-defined
  end type
  immutable class Rational (
      val numerator: Number_Like,
      val denominator: Number_Like with constraint { value /= 0 })
      extends Number
  end class
  immutable class Complex (
      val real: Number_Like,
      val imaginary: Number_Like) extends Number
  end class
end object
```

1.10 The Lang.Predef Object