Idris 初学手札

大概介绍所谓的 Idris 编程语言的特点，还有所学到的东西，做个总结。

什么是 Idris ¶

Idris 是一个由 Edwin Brady 设计的编程语言。是个纯函数式 (Purely Functional Programming) 和依赖类型 (Dependent Type) 的混合体，默认严格求值，有着可选的惰性求值类型，自带程序验证功能 (Totality Checker) 让这个编程语言更适合在工程上体验依赖类型。

目录

为什么选择 Idris ? ¶

- Idris 有着各种后端 codegen，有 JVM, CIL, Ocaml 和部分支持 LLVM
- Idris 能够编译成 Javascript 和带垃圾回收机制 (Garbage Collection) 的 C 程序
- Idris 提供很好的 FFI 接口 (Foreign Function Interface)，还有 Effect system 来处理状态，这个特点允许在嵌入式平台上探索函数式编程的实用性
- Idris 的语法和 Haskell 很接近，甚至少了很多语法坑
- Idris 还很新，年轻力盛的少年可以考虑献祭下

Idris 的特点

Dependent Type，自带证明器 就是它的杀手锏。依赖类型把语言实现简约了，比起 HM 类型系统，强调区分 Kinds, Functions, Variables, Type 和 Typeclasses。它实际上把类型变成了第一公民，而值能够出现在类型上。

从官方文档拿出个例子，我们能够写个函数去计算一个类型

```
isSingleton : Bool → Type
isSingleton True  = Nat
isSingleton False = List Nat
```

或者可以用这个函数来计算一个类型，反正这个函数返回的是一个类型

```
mkSingle : (x : Bool) → isSingleton x
mkSingle True  = 0
mkSingle False = []
```

然后还能够检查参数类型，通过传递这个函数来计算，返回类型

```
sum : (single : Bool) → isSingleton single → Nat
sum True x = x
sum False [] = 0
sum False (x :: xs) = x + sum False xs
```

语法介绍

下载试一下是 Idris 的 Hello World 程序

个简述，以下是 Idris 的 Hello World 程序。

```
module Main
main : IO ()
main = putStrLn "Hello World!"
```

然后我们编译上面的代码，hello.idr

```
$ idris hello.idr -o hello
$ ./hello
Hello World!
```

声明和定义 ¶

Idris 用的是和 Haskell 的一套。靠模式匹配做数据之间的映射。类型的声明靠：来表示，后面跟着的是类型。而 = 用来定义返回的值。就像这样:

```
branchingFactor : Int
branchingFactor = 32
```

函数 ¶

函数也是同样的方式来定义。 -- 后面接着的是注释，而 Idris 是没有块注释的。

```
isOdd : Int -> Bool -- 定义 isOdd 函数接受 Int 返回 Bool 类型
isOdd x = (mod x 2) == 1
```

模式匹配 ¶

而模式匹配能够匹配不同条件，比如说下面的代码。

isSingleton 根据接受的 Bool 类型参数来决定返回的类型 (Type)

```
isSingleton : Bool → Type
isSingleton True = Nat
isSingleton False = List Nat
```

模式匹配和数组 ¶

模式匹配是个很强的语言特性，事实上，它能够让我们更方便地书写条件。比如说模式匹配列表就是一个很好的例子。

举个例子：

列表 (List) 就是一个优化连接的数据结构。写成

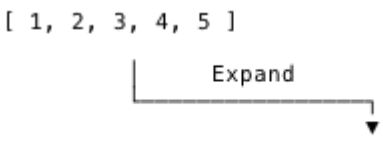
```
[1, 2, 3, 4] -- 1 :: 2 :: 3 :: 4 :: Nil
```

同时 Nil 也可以表示 []

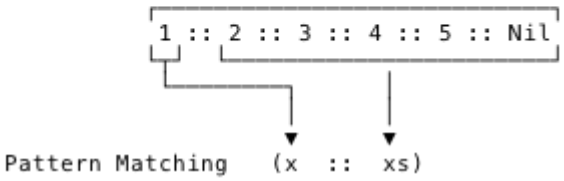
所以当我们模式匹配的时候。

```
isNil : List a → Bool
isNil []      = True
isNil (x::xs) = False
```

模式匹配的 xs 就是多个 x



的意思。也就是说除了从第二个元素开始的链表



最后一行的 (x::xs) 其实就是取列表的首元素和接下来的元素列表。

待续

© 2017 - 2018 zypeh, powered by Hexo and hexo-theme-brutalist.