

LispSICP✎修改

怎么理解邱奇计数？✎修改

SICP里CONS CAR CDR的实现已经把我吓尿过一次了。它居然在习题2.6里说还有一个更让人醍醐灌顶的东西：church计数。(define z...显示全部

关注问题

✎写回答

+邀请回答

💬1条评论

🚩分享

★邀请回答

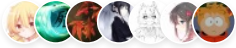
🚩举报

...

关注者159

被浏览8,479

他们也关注了该问题



查看全部 8 个回答

 Felis sapiens ⭐
函数式编程、编程语言、编程 话题的优秀回答者

霜月琉璃、考古学家千里冰封、Belleve、开源哥、圆角骑士魔理沙等 34 人赞同了该回答

在静态类型函数式语言的背景下，理解了 ADT (algebraic data type) 的 Church encoding，回头看 Church numerals 就豁然开朗了。

```
{-# LANGUAGE RankNTypes #-}

data Nat = Zero | Succ Nat

newtype NatChurch = NatChurch (forall r . r -> (r -> r) -> r)

zeroChurch :: NatChurch
zeroChurch = NatChurch const

succChurch :: NatChurch -> NatChurch
succChurch (NatChurch n) = NatChurch (\z s -> s (n z s))

addChurch :: NatChurch -> NatChurch -> NatChurch
addChurch (NatChurch n) (NatChurch m) = NatChurch (\z s -> n (m z s) s)

evalChurch :: NatChurch -> Int
evalChurch (NatChurch n) = n 0 succ

newtype NatScott = NatScott (forall r . r -> (NatScott -> r) -> r)

zeroScott :: NatScott
zeroScott = NatScott const

succScott :: NatScott -> NatScott
succScott n = NatScott (\_ s -> s n)

addScott :: NatScott -> NatScott -> NatScott
addScott n (NatScott m) = m n (succScott . addScott n)

evalScott :: NatScott -> Int
evalScott (NatScott n) = n 0 (succ . evalScott)
```

以上定义了一个 inductive Nat。这个 data type 可以用不同的方法做 encoding，也就是用普通的函数来进行编码。代码展示了 Church 和 Scott 两种不同的编码方式，包括相应的 zero/succ



关于作者

 Felis sapiens

⭐ 函数式编程、编程语言、编程 话题的优秀回答者

👤 电影旅行敲代码、Antokha Yuuki、暮无井见铃也关注了她

回答181

文章40

关注者14,871

已关注

💬发私信

被收藏 16 次

- l'Illumination de l'Ori
Yutong Zhang 创建

4 人关注
- 好东西
Ramen 创建

0 人关注
- 程序
一听silence 创建

0 人关注
- 计算机
Arcueid Brunestud 创建

0 人关注



constructor, 以及一个加法操作的样例, 最后可以用 eval 转回普通的 Int, 用于检验结果是否正确 (不妨在 ghci 下试几个例子)

Church encoding 的实质, 就是把 datatype 上的 fold 操作作用 CPS 方式表示。Scott encoding 的实质, 就是把 datatype 的每种 case 上的模式匹配操作作用 CPS 方式表示。

最后, 去掉各种 newtype 相关的转换, 用 untyped lambda calculus 把 zero/succ/add 等定义写出来, 对照 wikipedia 上的定义, 一切尽在不言中。

quiz:

1. Church/Scott encoding 孰优孰劣? 适用范围分别是?
2. Church encoding 与 Finally tagless style 之间的隐藏关联?

编辑于 2016-12-25

赞同 34

添加评论

分享

收藏

感谢

...

收起

更多回答



Pingan Cheng

To be a cognitive scientist.

26 人赞同了该回答

我也是刚看到这里, 我的理解如下:

首先看对于数的定义:

```
(define zero (lambda (f) (lambda (x) x)))
(define one (lambda (f) (lambda (x) (f x))))
(define two (lambda (f) (lambda (x) (f (f x)))))
```

这里我们可以看到所谓的数只是将过程f应用于x的次数。

对于add-1:

```
(define (add-1 n)
  (lambda (f) (lambda (x) (f ((n f) x)))))
```

结果就等于将f再应用于n一次。

上面这些估计不好理解, 点击右侧的

展开阅读全文

赞同 26

1 条评论

分享

收藏

感谢

...



6hu2t32

今天我们都是凡学家

13 人赞同了该回答

就是带括号的扳手指数数

扳手指的时候

1 表示 1

11 表示 2

111 表示 3

现在Church说了, 数数也要用括号

(1 ()) 表示 1

大概是这样

sophomore2 创建

0 人关



相关问题

SICP换零钱迭代方法实现, 是如何写的?

18 个回答

SICP第二章里的“图形语言”在 DrRacket 或者MIT Scheme上有没有办法实现啊? 5 个回答

如何看待Berkeley开设的CS61A:SICP in Python课程? 8 个回答

如何评价学军中学和SICP? 22 个回答

SICP 是不是被高估了? 36 个回答

相关推荐



淼懂物理学：理解世界的极简指南

共 31 节课

试听



数学妙啊！妙！

张英锋 等

289,069 人读过

阅读



刘看山 · 知乎指南 · 知乎协议 · 隐私政策

应用 · 工作 · 申请开通知乎机构号

侵权举报 · 网上有害信息举报专区

违法和不良信息举报: 010-82716601

儿童色情信息举报专区

电信与服务业务经营许可证

网络文化经营许可证

联系我们 © 2018 知乎



(1 (1 0)) 表示 2



(1 (1 (1 0))) 表示 3

[展开阅读全文](#)

赞同 13

3 条评论

分享

收藏

感谢

[查看全部 8 个回答](#)