编程语言 编程 函数式编程 Haskell 惰性求值 修改

关注者 210 被浏览 19,386

Haskell中的惰性求值如何实现? 修改

他们也关注了该问题

有很多程序语言都实现了惰性求值，比如最典型的Haskell。通过惰性求值可以简化很多运算，但是我有一个步明白的地方，惰性求值在编译器中是如何实现的，难...显示全部

关注问题 写回答 邀请回答 添加评论 分享 邀请回答 举报 ...

查看全部 15 个回答

Felis sapiens
函数式编程、编程语言、编程 话题的优秀回答者

考古学家千里冰封、hhhhhhhhh、头顶青天红美铃、阅千人而惜知己、开源哥等 36 人赞同了该回答

有几个抽象机器专门用于实现惰性语言的graph reduction，以G-machine为代表。
读一下Implementing Functional Languages: A Tutorial就知道怎样实现了。

======

另外题主和若干个答主把两个概念搞混淆了：惰性求值（lazy evaluation）和记忆化（memoization）。

```
fib :: Int -> Integer
fib 0 = 0
fib 1 = 1
fib n = (fib (n-1))+(fib (n-2))
```

比如上面求fibonacci的代码，不管用什么策略求值，时间复杂度都是O(2^n)。但是如果在实现语言的编译器/解释器时，任何一次函数调用都存表/查表来缓存的话，可以在上层代码不变的情况下自动降时间复杂度为O(n)。但是这叫做记忆化，不是惰性求值，而且Haskell的编译器ghc没有实现这个机制。Haskell是需要手写记忆化的逻辑的，一个小例子：gist.github.com/TerrorJ...

惰性求值则是一种求值策略：

节选自《Programming in Haskell》。

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	文章	关注者
624	40	14,871

已关注 发私信

被收藏 8 次

l'Illumination de l'Ori Yutong Zhang 创建	4 人关注
编译原理与程序设计原理 张梓铖 创建	1 人关注
Haskell 姚奕正 创建	0 人关注
长知识 Lu Zheng 创建	0 人关注

为什么编译器不做记忆化优化，主要原因是用得到的场合太少，开销太大。

编辑于 2014-12-05

▲ 赞同 36 ▼ 8 条评论 分享 收藏 感谢 ··· 收起

更多回答

watashi
ゆっくりでいいさ

57 人赞同了该回答

GHC中的惰性求值是如何实现的

Haskell语言只要求惰性求值，并不考虑怎么实现。就比如Haskell语言对编译器们说你们可以提供个C++的FFI，但完全没规定怎么实现，结果也没有任何编译器实现了这个功能。所以假设问题是”GHC中的惰性求值是如何实现的“。

要一窥GHC是如何实现惰性求值的，这有个再合适不过的工具：ghc-heap-view: Extract the heap representation of Haskell values and thunks。其中的Demo.hs提供了几个了解惰性求值实现非常好的例子：

```
Here are a four different lists, where the first three are already evaluated.
The first one, l, was defined as a top level constant as
> l = [1,2,3]
and is now found at 0xaf305d90 (where the /2 is the pointer tag information) and fully
> ConsClosure {info = StgInfoTable ... ptrs = 0, tipe = CONSTR_2_0, srtlen = 1}
```

展开阅读全文

▲ 赞同 57 ▼ 1 条评论 分享 收藏 感谢 ···

亞首

30 人赞同了该回答

這週把惰性求值實現了一遍，終於覺得有自信回答這個問題了。我在這裡的解釋會比較簡略，不過推薦來讀我翻譯的文章《Haskell 怎麼實現惰性求值》（原文），和實現了惰性求值的 Loli 語言。

首先，Loli 類似 Haskell 惰性求值的實現主要在兩方面：第一，動態閉包的傳參方式實現 non-strict 的語意；第二，用自訂類型支持方便的 streaming，為了接近 Haskell，我在 Loli 中用的是 ADT 的支持 。

動態閉包（dynamic closure）的傳參方式就是說，求值器在求值一個 function application 的時候不會對其參數求值，而是會封到一個動態閉包裡。動態閉包其實就是一個帶 binding 的表達式，動態閉包有很多名字，有的叫之 thunk，也有些地方稱之為 redex，不過就我的理解動態閉包不能和 redex 畫上等號，因為在我的實現裡 redex 還包括其他一些表達式類型。所有的值只有在用到的時候才會被規約到模範型（normal form）。做到這點並不會很難，scheme 的 delay 和 force 函數就可以用來實現簡單的手動惰性求值功能。

展开阅读全文

▲ 赞同 30 ▼ 11 条评论 分享 收藏 感谢 ···

查看全部 15 个回答

研发 0 人关
大石匠 创建

相关问题

Haskell等语言中的模式匹配在C++中如何实现？ 8 个回答

Haskell 以后，就没有其他新语言默认采用惰性求值了，这是否说明（默认）惰性求值是个错误的决定？ 22 个回答

Haskell 的 Graph reduction 在现实机器中是怎样实现的？ 3 个回答

在Haskell里，每个类型都可以构造出来一个此类型的表达式吗？ 4 个回答

Haskell 最有代表性的一段程序是什么？ 6 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
289,063 人读过 阅读