Haskell 实现相关的 reading list

Felis sap...
函数式编程、编程语言、编程 话题的优秀回答者

记录一下我在开 Haskell 编译器坑时阅读过的材料。因为手头的坑主要是 backend 相关，所以像类型系统等 frontend 相关的材料，就不放上来了。
首先是 GHC 相关的材料：

- Commentary/Compiler - GHC：不了解 GHC 架构的话，自然先从 wiki 看起了，有一些基础 Haskell 知识就可以。不过看 wiki 时要留心该条目的编辑历史，比如 Last modified 10 years ago 这种就跟目前的 GHC codebase 不搭边。
- takenobu-hs/haskell-ghc-illustrated：综合讲述 GHC backend & runtime 的 slide，概括得不错。
- Making a fast curry: push/enter vs. eval/apply for higher-order languages：关于 STG（Spineless Tagless G-machine）最 up-to-date 的介绍材料，介绍了 STG 的语法、操作语义（基于一个抽象的栈和堆的重写规则）、实现相关的细节（内存布局等等）。那篇原始的 Implementing lazy functional languages on stock hardware: the Spineless Tagless G-machine 太旧了。
- C--: A Portable Assembly Language that Supports Garbage Collection：关于 C-- 的综合介绍。C-- 语法类似 C，设计目的是 portable assembler，支持多调用约定、显式尾调用、垃圾回收等特性。现在的 GHC pipeline 中的 Cmm 就是基于 C-- 的中间表示，所有的 codegen（C/ASM/LLVM）都是从 Cmm 开始翻译的。虽然这篇比较旧了，不过对 Cmm 不熟悉的话，还是需要读一下。
- Hoopl: a modular, reusable library for dataflow analysis and transformation：GHC 通过 Hoopl 框架对 Cmm 进行数据流优化，理解 Cmm 相关类型需要参考一下 hoopl 的 paper。
- Low level virtual machine for Glasgow Haskell Compiler：GHC 的 LLVM codegen 实现者的学士论文。将 Cmm 翻译到 LLVM IR，对于希望做其他 codegen 的项目而言是不错的参考

话也需要参考一下。另外对 GHC 的各个 IR 和运行时机制也有简单的介绍。
- Profiling optimised Haskell - causal analysis and implementation
- Understanding the Stack & Understanding the RealWorld：介绍 Cmm 运行机制的两个例子。

wiki 和 paper 的局限性也很明显，因为 GHC 的 codebase 进化实在是太快了，如果项目是基于 GHC HEAD 开发，那么不时会碰到一些过时的内容，这时最关键的参考资料，仍然是 ghc.git 。一些阅读 GHC 源代码的小技巧：

- build GHC 时，确保在 configure 之前，首先安装了 hscolour 工具。默认在 build 完成以后会调用 haddock 生成 ghc 等库的文档，如果 hscolour 可用的话，生成的文档中源代码部分会带上彩色的、scope-aware 的超链接，阅读起来非常非常方便，能省掉 n 多的 grep 工作。不过懒得 build 的话可以看我自己 build 的版本：ghc-8.3: The GHC API
- 碰到某个 IR 的类型很复杂，无从下手怎么办？可以尝试将其 abstract syntax 和 concrete syntax 对应起来 —— ghc 提供了各种 -ddump-xx 选项可以将不同 IR 给 dump 到文本文件，在 GHC 源代码里找到对应的类型，然后在该类型的定义模块里找到它的 Outputable instance。所有支持 -ddump-xx 选项的 GHC IR 都通过 Outputable class 实现了 pretty-print 的逻辑，所以从该 instance 的定义可以看到 abstract syntax 是如何变成 concrete syntax 的。一部分的 IR 同时提供了 parser，可以在 git repo 里搜索对应的 happy source file，观察 concrete syntax 如何被解析。
- GHC 的源代码里有很多有用的注释，而且会标明 Notes on xxx，这些注释在 haddock 生成的网页上不会显示，但在源代码页面里有。这些注释比任何 wiki 和 paper 都更加 up-to-date。

接下来是 GHC 以外的一些参考材料：

- Towards a Declarative Web：haste-compiler 作者的硕士论文，详细介绍了 haste-compiler 的架构。haste-compiler 是一个将 STG 翻译到 JavaScript 的 codegen，支持除 Template Haskell 以外的所有 GHC 语言特性，值得参考。
- Composable scheduler activations for Haskell：介绍了如何将 rts 中涉及并发调度的部分以 Haskell API 的形式提供给用户自行定制。这项工作最初是在 Haskell 上做的，但是并未在 GHC 上实现，后来一作转去实现 Multicore OCaml 去了。
- PAEAN: Portable and scalable runtime support for parallel Haskell dialects：提出了面向各种支持并行编程的 Haskell dialect 的运行时框架。

说起来，Haskell 作为一门有 20 余年历史的语言，献祭了这么多 PhD，搞出了这么多 bleeding-edge 而又不优雅、不正交的特性，编译器架构也十分庞杂。那我为什么不像 @Belleve 一样另起炉灶，开一个优雅的坑，而是想做一个兼容 GHC 的 Haskell 编译器呢？

编辑于 2017-04-21

「真诚赞赏，手留余香」

赞赏

还没有人赞赏，快来当第一个赞赏的人吧！

Haskell GHC（编程套件） 函数式编程

文章被以下专栏收录

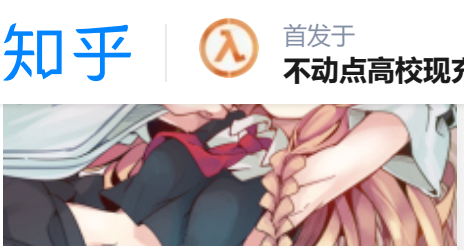
不动点高校现充部
一切与编程语言理论、函数式编程相关的杂谈。

已关注

推荐阅读

初窥Haskell：解析一个数学表

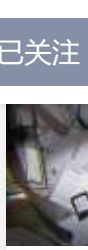
学 Haskell 果然是要趁早



真的是一门令我着迷的语言，lazy和纯函数式等特性都非常吸引我，不过短时间内还无法掌握得很好，最重要是思维的转变非常苦难。 曾几何时我觉得Python是一门...

FTP ，并没有真相 length :: Foldable t => t a -> Int 当

Cosmia Fu



8 条评论

切换为时间排序

写下你的评论...

dram 1 年前

be5 已被大家手动忽略

4

以上为精选评论

Belleve 1 年前

「因为喜欢啊」（楼下保持队形）

赞

草莓大福 1 年前

高产似母× （逃

1

Neuromancer 1 年前

哇都是现在最需要的信息😁

赞

Hypnoes Liu 1 年前

想起来lisp梗：为了解决lisp方言太多的问题，我们决定开发一种新的lisp方言。

1

vczh 1 年前

图是什么

赞

Felis sapiens (作者) 回复 vczh 1 年前

路灯下的喵帕斯

1 年前

看见大图书馆的牧羊人我就进来了..

赞