Haskell 中匿名 record 的实现（1）

Felis sap...
函数式编程、编程语言、编程 话题的优秀回答者

Haskell 里，搞匿名 record 也算是个老生长谈的话题了——早的从 2004 年 HList 的文章刚发出来时就开始搞了，那个年代 ghc 还没有什么（伪）依赖类型的概念。很多库都有自己造匿名 record 的轮子，比如 Jon Sterling 的 vinyl 或者 Chris Done 的 labels —— 一个用 labels 做数据分析的例子见 Working with data in Haskell。

这个系列专栏会展示如何实现一个匿名 record 系统，讲解其中用到的一些 Haskell 高级特性（包含类型相关与底层相关），并演示在什么样的开发场景下匿名 record 比内建 record 使用更加方便。

因为忙期末季所以这一期就先贴实现吧，讲解下一期开始。这个匿名 record 系统实现的特性：

- 每个 record 的类型包含一个类型变量 ts，描述 record 的 schema。一个 schema 是一系列 field 的列表，每个 field 的定义包含 field name 以及 field type；为简单起见，这一版中所有 field type 都是 lifted 的类型，并且不考虑 strict field
- 不使用常见的 heterogeneous list，而使用 rts 提供的 SmallArray# 存储 record。get/set 的时间复杂度为 O(1)。代价是用到了一些底层相关的黑魔法
- 单个 field 的 getter/setter/lens，可以通过 field name 方便地获取 lens
- lens 自动支持重载，比如 getLens #yoo 可以用于所有包含 yoo 这个 field 的 record

实现代码：

```
{-# LANGUAGE DataKinds #-}
{-# LANGUAGE FlexibleContexts #-}
{-# LANGUAGE FlexibleInstances #-}
```

```
{-# LANGUAGE MagicHash #-}
{-# LANGUAGE MultiParamTypeClasses #-}
{-# LANGUAGE OverloadedLabels #-}
{-# LANGUAGE RankNTypes #-}
{-# LANGUAGE ScopedTypeVariables #-}
{-# LANGUAGE TypeFamilies #-}
{-# LANGUAGE TypeOperators #-}
{-# LANGUAGE UnboxedTuples #-}
{-# LANGUAGE UndecidableInstances #-}
{-# OPTIONS_GHC -Wno-incomplete-patterns #-}

import Control.Lens
import Data.Kind
import GHC.Exts
import GHC.Integer.GMP.Internals
import GHC.OverloadedLabels
import GHC.TypeLits

natInt# :: KnownNat n => Proxy# n -> Int#
natInt# n =
    case natVal' n of
        S# x -> x

type family SchemaSize (ts :: [(Symbol, Type)]) :: Nat where
    SchemaSize '[] = 0
    SchemaSize (_ ': ts) = 1 + SchemaSize ts

schemaSizeProxy# ::
       KnownNat (SchemaSize ts) => Proxy# ts -> Proxy# (SchemaSize ts)
schemaSizeProxy# _ = proxy#

schemaSize# :: KnownNat (SchemaSize ts) => Proxy# ts -> Int#
schemaSize# ts = natInt# (schemaSizeProxy# ts)

type family FieldIndex (ts :: [(Symbol, Type)]) (sym :: Symbol) (t :: Type) :: Nat whe
    FieldIndex ('( sym, t) ': _) sym t = 0
    FieldIndex (_ ': ts) sym t = 1 + FieldIndex ts sym t

fieldIndexProxy# ::
       KnownNat (FieldIndex ts sym t)
    => Proxy# ts
    -> Proxy# sym
    -> Proxy# t
```

```
fieldIndex# ::
       KnownNat (FieldIndex ts sym t)
    => Proxy# ts
    -> Proxy# sym
    -> Proxy# t
    -> Int#
fieldIndex# ts sym t = natInt# (fieldIndexProxy# ts sym t)

data Rec :: [(Symbol, Type)] -> Type where
    Rec :: SmallArray# Any -> Rec ts

newRec :: KnownNat (SchemaSize ts) => Proxy# ts -> Rec ts
newRec ts =
    case newSmallArray# (schemaSize# ts) undefined realWorld# of
        (# s1, buf #) -> case unsafeFreezeSmallArray# buf s1 of
            (# _, arr #) -> Rec arr

fieldGet :: KnownNat (FieldIndex ts sym t) => Proxy# ts -> Proxy# sym -> Proxy# t -> R
fieldGet ts sym t (Rec arr) =
    case indexSmallArray# arr (fieldIndex# ts sym t) of
        (# v #) -> unsafeCoerce# v

fieldSet :: KnownNat (FieldIndex ts sym t) => Proxy# ts -> Proxy# sym -> Proxy# t -> R
fieldSet ts sym t (Rec arr) v =
    case thawSmallArray# arr 0# (sizeofSmallArray# arr) realWorld# of
        (# s1, buf #) -> case unsafeFreezeSmallArray# buf (writeSmallArray# buf (field
            (# _, arr' #) -> Rec arr'

newtype LensWrapper ts t = LensWrapper { getLens :: Lens' (Rec ts) t }

instance KnownNat (FieldIndex ts sym t) => IsLabel sym (LensWrapper ts t) where
    fromLabel _ = LensWrapper (lens (fieldGet (proxy# :: Proxy# ts) (proxy# :: Proxy#
```

使用方法大致是：

```
-- Initialize a record
rec0 :: Rec '[ '("yoo", String)]
rec0 = newRec proxy#

-- Set a field
rec1 :: Rec '[ '("yoo", String)]
rec1 = set (getLens #yoo) "23333" rec0
```

```
s :: String
s = rec1 ^. getLens #yoo
```

下午要考试了所以不能再摸鱼了，先写到这吧。。。之后代码整理好，增加更多特性（比如 record 子类型的 cast、mutable record 等等）以后会发布到 Hackage 上。

EOF

编辑于 2017-06-14

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

幻想中的Haskell - Compiling Combinator

圆角骑士魔... 发表于雾雨魔法店

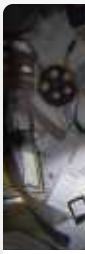
Notes on Haskell Debugging

Felis... 发表于不动点高校...

学 Haskell 果然是要趁早

几年前， length 的类型是介个样子滴length :: [a] -> Int 现在你打开 GHCi 捏，是介个样子滴， （搜 FTP ，并没有真相 length :: Foldable t => t a -> Int 当然了，这其实并不会...

Cosmia Fu



在H type

7 条评论 切换为时间排序

写下你的评论...

Felis sapiens (作者) 回复 祖与占 1 年前

没有。。他那个还是得基于有 Generic instance 的普通 record。而且代码肯定没我这个脏。。（

1 查看对话

以上为精选评论

parker liu 1 年前

哈哈，首讚

赞

MaskRay 1 年前

頷讚

赞

凉宫礼 1 年前

这代码高亮字体都瞎了

赞

禾木旁 1 年前

颈赞

赞

祖与占 1 年前

是不是参考了最近出的generic-lens?

赞

祖与占 1 年前

SuperRecord: Anonymous Records for Haskell 又一个库出来了...催更...

赞