Лабораторная # 3 Кодогенерация

Авторы:

- Иван Якимов: ivan.yakimov.research@yandex.ru
- Александр Кузнецов: askuznetsov@sfu-kras.ru

Исходные коды примеров доступны на github: https://github.com/forjadores/Haskable

Экспорт кода осуществляется командой **export_code**:
Для примера с *app* из предыдущей методички:
export_code Nil Cons app rev
in Haskell module_name App file "haskable"
На выходе получим файл на языке Хаскел:
В него добавлено определение нашего кастомного списка (через конструкторы Nil и Cons, которые, как мы помним, на самом деле являются функциями), а также двух функций над ним — app и rev.

```
{-# LANGUAGE EmptyDataDecls, RankNTypes, ScopedTypeVariables #-}

module App(List(..), app, rev) where {

import Prelude ((==), (/=), (<), (<=), (>=), (>), (+), (-), (*), (/), (**),
  (>>=), (>>), (=<<), (&&), (||), (^), (^^), (.), ($), ($!), (++), (!!), Eq,
  error, id, return, not, fst, snd, map, filter, concat, concatMap, reverse,
  zip, null, takeWhile, dropWhile, all, any, Integer, negate, abs, divMod,
  String, Bool(True, False), Maybe(Nothing, Just));
import qualified Prelude;

data List a = Nil | Cons a (List a);

app :: forall a. List a -> List a -> List a;
app Nil ys = ys;
app (Cons x xs) ys = Cons x (app xs ys);

rev :: forall a. List a -> List a;
rev Nil = Nil;
rev (Cons x xs) = app (rev xs) (Cons x Nil);

}
```

Для работы с этим файлом нужно написать небольшой драйвер, в котором будут инстанциированы функции вывода на экран и сравнения наших кастомных списков.

```
-- author: Ivan Yakimov
-- e-mail: ivan.yakimov.research@yandex.ru
-- NOTE: this software is licensed under the z-lib license

import App (List(..), app, rev)

instance (Show a) => Show (List a) where
    show Nil = "[]"
    show (Cons x xs) = show x ++ "#" ++ (show xs)

instance (Eq a) => Eq (List a) where
    Nil == Nil = True
    (Cons x xs) == Nil = False
    Nil == (Cons y ys) = False
    (Cons x xs) == (Cons y ys) = (x == y) && (xs == ys)

l0 = Nil
l1 = Cons 1 Nil
l2 = Cons 2 l1
l3 = Cons 3 l2
l4 = Cons 4 l3
```

Для демонстрации работы напишем небольшой bash-скрипт, который просто подает на вход интерпретатору ghci текст скрипта так, как если бы мы вбивали его с клавиатуры:

```
#!/bin/bash
# author: Ivan Yakimov
# e-mail: ivan.yakimov.research@yandex.ru
# this software is licensed under the z-lib license
echo "
:load Driver.hs
\"################ START ##################\"
\"the original lists are:\"
l0
l1
l2
l3
l4
\"reversed:\"
rev l0
rev l1
rev l2
rev l3
rev l4
\"=============================================================================\"
\" Mathematical PROOF BY INDUCTION for the following theorem: \"
\"rev (rev xs) = xs\"
\" Isabelle automatically generates 2 subgoals which are \"
\"-----------------------------------------------------------------\"
\" # BASE ...  \"
\" First goal is a BASE of the induction:\"
\" 1. rev (rev []) = [] \"
\" we can check whether is it a case with haskell by: \"
\" (rev (rev Nil)) == Nil \"
\" which is:  \"
(rev (rev Nil)) == Nil
\"-----------------------------------------------------------------\"
\" # ... and INDUCTIVE STEP \"
\" Second goal is an INDUCTIVE STEP:\"
\"  2. !! x1 x2. rev (rev x2) = x2 ==> rev (rev (x1 # x2)) = x1 # x2 \"
\" first part of the second goal is an INDUCTIVE HYPOTHESIS: \"
\" !! x2. rev (rev x2) = x2:\"
\" we expand it with fixed-length list l2\"
\"(rev (rev l2) == l2)\"
\" the result is: \"
(rev (rev l2)) == l2
\" futher, we demonstrate that INDUCTUVE STEP ITSELF is true: \"
\" rev (rev (x1 # x2)) = x1 # x2 \"
\" with list produced from l2 by adding single element\"
\" (rev (rev (Cons 28 l2))) == (Cons 28 l2) \"
\" we can see that it is indeed true:  \"
(rev (rev (Cons 28 l2))) == (Cons 28 l2)
\"####\"
\"################ END ##################\"
:quit
" | ghci
```

Задание:
Возьмите ваш вариант с предыдущей работы. Экспортируйте код в Хаскел и напишите соответствующий драйвер. Напишите сценарий, демонстрирующий примеры работы — базовый случай, индуктивный переход, как в примере.