Вывод типов и некоторые задачи

Косарев Дмитрий a.k.a. Kakadu

матмех СПбГУ

27 сентября 2018 г.

Организация тестов к программе

Демо с просторов интернета
https://github.com/andybalaam/hunit-example.git
Можно собирать с помощью нового Stack или с помощью устаревшего ~~cabal~~

Запускать так

- stack init
- stack build
- stack test
- Тест падает, но это так и задумано.

Рандомная ссылка с просторов интернета

Как сделать змейку на Haskell

У нас был интерпретатор языка Λ ...

```
type Name = String
data Term = Var        Name
          | Con Int
          | Add Term Term
          | Lam Name Term
          | App Term Term
```

Обязуем интепретатор возвращать вот такое значение

```
data Value = Wrong
           | Num Int
           | Fun (Value -> Value)

showval :: Value -> String
```

... который умел вычислять

```
type Environment = [(Name, Value)]
eval :: Term -> Environment -> Value

test :: Term -> String
test t = showval (eval t [])

term0 = (App (Lam "x" (Add (Var "x") (Var "x")))
             (Add (Con 10) (Con 11)))
```

т.е. $((\lambda x.x + x)(10 + 11))$

И теперь `test term0` должен вычисляться в `Num 42`.

Добавим туда человеческие сообщения об ошибках (1/2)

```
data E a = Success a | Error String

showE (Success a) = "Success: " ++ showval a
showE (Error s) = "Error: " ++ s

lookup x [] = Error ("unbound variable: " ++ x)
add a b     =
  Error  ("should be numbers: "
             ++ showval a
             ++ " "
             ++ showval b)

apply f a   =
  Error ("should be function: " ++ showval f)
```

Добавим туда человеческие сообщения об ошибках (2/2)

```
term0 = (App (Lam "x" (Add (Var "x") (Var "x")))
             (Add (Con 10) (Con 11)))
```

И теперь `term0` (он же $((\lambda x.x + x)(10 + 11))$) должен вычисляться в `Success: 42`

А `test (App (Con 1) (Con 2))`, должен выдать, например, `"Error: should be function: 1"`

В “нечистых” языках это делается через исключения или callback’и.

Ошибки с позицией в файле

```
data Term = ...
          | At Position Term
```

Например, в `(At p (App t (At q u))` выражение `(App t u)` находится на позиции `p`, а подвыражение `u` на позиции `q`.

Добавьте в сообщениях об ошибке распечатку позиции.

В императивных языках, наверное, использовался бы стек позиций, куда/откуда надо было бы вовремя добавлять/снимать.

Число редукций при вычислении

Редукция – применений одного выражения языка Λ к другому.

Вместе с ответом надо вернуть сколько таких применений было произведено.

Например, для `(Add (Add (Con 1) (Con 2)) Count)` надо вернуть ответ 4 и количество редукций 2, потому что было произведено два сложения.

Язык, где возможен вывод куда-либо

```
data Term = ...
          | Out Term
```

Например, для `(Add (Out (Con 41)) (Out (Con 1)))` надо вернуть распечатанную по ходу дела строку `Output: 41; 1` и ответ `42`.

Язык, где есть неоднозначность

```
data Term = ...
          | Fail
          | Amb Term Term
```

Вычислитель выражений должен возвращать список (может быть пустой) возможных результатов исполнения.

В “нормальных” языках это не очень понятно как делать. Корутины?

P.S. Можно ещё две задачи про это же, но потом.

Разминка: какой тип у функции?

```
smth2 = helper
  where
    helper  []     = init
    helper  (x:xs) = acc + (helper xs)
```

Разминка: какой тип у функции?

```
smth2 = helper
  where
    helper  []     = init
    helper  (x:xs) = acc + (helper xs)
```

А у этой?

```
smth1 f = helper
  where
    helper  []     = init
    helper  (x:xs) = f acc (helper xs)
```

Разминка: какой тип у функции?

```
smth2 = helper
  where
    helper  []     = init
    helper  (x:xs) = acc + (helper xs)
```

А у этой?

```
smth1 f = helper
  where
    helper  []     = init
    helper  (x:xs) = f acc (helper xs)
```

Как вы это получили?

Ещё про типы

Можно ли применить `f :: (Int -> a) -> Int -> a)` к вот таким `g` ?

- `g :: Int -> String`
- `g :: b -> Int -> [b]`
- `g :: Int -> a -> a`
- `g :: String -> Int -> a`

Унификация. Неформально. (1/2)

Даны два дерева (или DAG), в которых некоторые листья “особенные”: в них стоят *метапеременные*, которые можно заменять на другие деревья. В дереве также есть кокретизированные (ground) значения, которые заменять запрещено.

Список пар из метапеременных и деревьев образует *подстановку*.

Унификация может завершиться успешно с (возможно пустой) подстановкой или с ошибкой.

Пример: $f(x, 1)$ и $g(2, y)$.

Унификация. Неформально. (1/2)

Даны два дерева (или DAG), в которых некоторые листья “особенные”: в них стоят *метапеременные*, которые можно заменять на другие деревья. В дереве также есть кокретизированные (ground) значения, которые заменять запрещено.

Список пар из метапеременных и деревьев образует *подстановку*.

Унификация может завершиться успешно с (возможно пустой) подстановкой или с ошибкой.

Пример: $f(x, 1)$ и $g(2, y)$. Ответ: $[x \mapsto 2, y \mapsto 3]$

Унификация. Неформально. (1/2)

Даны два дерева (или DAG), в которых некоторые листья “особенные”: в них стоят *метапеременные*, которые можно заменять на другие деревья. В дереве также есть кокретизированные (ground) значения, которые заменять запрещено.

Список пар из метапеременных и деревьев образует *подстановку*.

Унификация может завершиться успешно с (возможно пустой) подстановкой или с ошибкой.

Пример: $f(x, 1)$ и $g(2, y)$. Ответ: $[x \mapsto 2, y \mapsto 3]$
Пример: $f(x, 1)$ и $g(y, y)$.

Унификация. Неформально. (1/2)

Даны два дерева (или DAG), в которых некоторые листья “особенные”: в них стоят *метапеременные*, которые можно заменять на другие деревья. В дереве также есть кокретизированные (ground) значения, которые заменять запрещено.

Список пар из метапеременных и деревьев образует *подстановку*.

Унификация может завершиться успешно с (возможно пустой) подстановкой или с ошибкой.

Пример: $f(x, 1)$ и $g(2, y)$. Ответ: $[x \mapsto 2, y \mapsto 3]$
Пример: $f(x, 1)$ и $g(y, y)$. Ответ: $[x \mapsto y, y \mapsto 1]$ или $[x \mapsto 1, y \mapsto 1]$

Унификация. Неформально. (1/2)

Даны два дерева (или DAG), в которых некоторые листья “особенные”: в них стоят *метапеременные*, которые можно заменять на другие деревья. В дереве также есть кокретизированные (ground) значения, которые заменять запрещено.

Список пар из метапеременных и деревьев образует *подстановку*.

Унификация может завершиться успешно с (возможно пустой) подстановкой или с ошибкой.

Пример: $f(x, 1)$ и $g(2, y)$. Ответ: $[x \mapsto 2, y \mapsto 3]$
Пример: $f(x, 1)$ и $g(y, y)$. Ответ: $[x \mapsto y, y \mapsto 1]$ или $[x \mapsto 1, y \mapsto 1]$
Пример: $f(x, 1)$ и $g(y, 2)$.

Унификация. Неформально. (1/2)

Даны два дерева (или DAG), в которых некоторые листья “особенные”: в них стоят *метапеременные*, которые можно заменять на другие деревья. В дереве также есть кокретизированные (ground) значения, которые заменять запрещено.

Список пар из метапеременных и деревьев образует *подстановку*.

Унификация может завершиться успешно с (возможно пустой) подстановкой или с ошибкой.

Пример: $f(x, 1)$ и $g(2, y)$. Ответ: $[x \mapsto 2, y \mapsto 3]$
Пример: $f(x, 1)$ и $g(y, y)$. Ответ: $[x \mapsto y, y \mapsto 1]$ или $[x \mapsto 1, y \mapsto 1]$
Пример: $f(x, 1)$ и $g(y, 2)$. Ответ: $\bot$

Унификация. Неформально. (1/2)

Даны два дерева (или DAG), в которых некоторые листья “особенные”: в них стоят *метапеременные*, которые можно заменять на другие деревья. В дереве также есть кокретизированные (ground) значения, которые заменять запрещено.

Список пар из метапеременных и деревьев образует *подстановку*.

Унификация может завершиться успешно с (возможно пустой) подстановкой или с ошибкой.

Пример: $f(x, 1)$ и $g(2, y)$. Ответ: $[x \mapsto 2, y \mapsto 3]$
Пример: $f(x, 1)$ и $g(y, y)$. Ответ: $[x \mapsto y, y \mapsto 1]$ или $[x \mapsto 1, y \mapsto 1]$
Пример: $f(x, 1)$ и $g(y, 2)$. Ответ: $\bot$

Унификация. Неформально. (2/2)

У подстановке есть область определения (domain):
`dom s = map fst s`.

Подстановку можно применять
(`app :: Subst -> Tree -> Tree`) к дереву, заменив в дереве переменные из домена на соответствующие правые части.

Из двух подстановок σ и ξ можно построить *композицию подстановок*, применяя подстановку ξ к правым частям подстановки σ.

Унификация. Скользкий момент.

? Что будет, если унифицировать x и $l(y, x)$?

? Ну или в терминах типов `r` и `ListG(a,r)` ?

Унификация. Скользкий момент.

? Что будет, если унифицировать x и $l(y, x)$?

? Ну или в терминах типов `r` и `ListG(a,r)` ?

Occurs check

Упражнение. Реализовать унификацию

Тут есть три варианта

- Для конкретного типа завести отдельный конструктор для метапеременных
  - Максимально прямолинейно
  - Неподставленные переменные никак не обозначены на уровне типов.

Упражнение. Реализовать унификацию

Тут есть три варианта

- Для конкретного типа завести отдельный конструктор для метапеременных
  - Максимально прямолинейно
  - Неподставленные переменные никак не обозначены на уровне типов.

Задачки про ✚

- ✚ Интерпретатор с человеческими сообщениями об ошибках
- ✚ Интерпретатор с позицией
- ✚ Интерпретатор с подсчетом числа редукций
- ✚ Интерпретатор с выводом
- ✚ Интерпретатор с неоднозначностью
- ✚ Унификация раз