Программирование с зависимыми типами на языке Idris

Лекция 2. Интерактивная разработка через типы

В. Н. Брагилевский

11 февраля 2017 г.

Computer Science клуб (Санкт-Петербург)

Институт математики, механики и компьютерных наук
имени И. И. Воровича, Южный федеральный университет (Ростов-на-Дону)

Процесс TDD — Type Driven Development

- Type
- Define
- Refine

Пример: длины строк из списка

allLengths.idr

```
allLengths : List String -> List Nat
allLengths [] = []
allLengths (x :: xs) = length x :: allLengths xs
```

Команды редактора

- Добавление определения по типу
- Генерация образцов
- Генерация тела функции
- Тип и документация для элемента под курсором

Использованные типы (по документации)

```
Data type Prelude.List.List : (elem : Type) -> Type
    Generic lists

Constructors:
    Nil : List elem
        Empty list

    (::) : (x : elem) -> (xs : List elem) -> List elem
        A non-empty list, consisting of a head element and the rest of the list.
        infixr 7


Data type Prelude.Nat.Nat : Type
    Natural numbers: unbounded, unsigned integers which can be pattern matched.

Constructors:
    Z : Nat
        Zero

    S : Nat -> Nat
        Successor


Primitive type String : Type
    Strings in some unspecified encoding
```

Пример: длины строк из вектора

allLengthsVect.idr

```
import Data.Vect

allLengths : Vect n String -> Vect n Nat
allLengths [] = []
allLengths (x :: xs) = length x :: allLengths xs
```

- Тип может давать достаточно много информации для автоматической генерации реализации.

Пример: сортировка вектора натуральных чисел

insSort.idr

```
import Data.Vect

insert : (x : Nat) -> (xsSorted : Vect len Nat)
                   -> Vect (S len) Nat
insert x [] = [x]
insert x ys@(y :: xs) = case x < y of
                          False => y :: insert x xs
                          True => x :: ys

insSort : Vect n Nat -> Vect n Nat
insSort [] = []
insSort (x :: xs) = let xsSorted = insSort xs in
                    insert x xsSorted
```

Пример: сортировка вектора (с ограничением на тип)

```
import Data.Vect

insert : Ord elem => (x : elem)
                  -> (xsSorted : Vect len elem)
                  -> Vect (S len) elem
insert x [] = [x]
insert x ys@(y :: xs) = case x < y of
                          False => y :: insert x xs
                          True => x :: ys

insSort : Ord elem => Vect n elem -> Vect n elem
insSort [] = []
insSort (x :: xs) = let xsSorted = insSort xs in
                    insert x xsSorted
```

Команды редактора (Emacs)

- C-c C-l: Загрузка файла в интерпретатор
- C-c C-s: Создание заготовки для функции по её типу
- C-c C-a: Автоматическое решение
- C-c C-e: Извлечение функции (леммы)
- C-c C-c: Генерация образцов для параметра или case-выражения
- C-c C-t: Тип элемента
- C-c C-z: Переход в буфер с интерпретатором
- C-c C-d d: Отображение документации по элементу

Явные и неявные параметры функций

```
vhead : Vect (1 + n) elem -> elem
vhead (x :: _) = x

vhead' : (n : Nat) -> (elem : Type)
         -> Vect (1 + n) elem -> elem
vhead' n elem (x :: _) = x
```

```
idris> vhead [1,2,3]
1 : Nat
idris> vhead' 2 Nat [1,2,3]
1 : Nat
```

Использование неявных параметров

zeroes.idr

```
zeroes : Vect n Nat
zeroes {n = Z} = []
zeroes {n = (S k)} = 0 :: zeroes

idris> zeroes
(input):Can't infer argument n to zeroes
idris> zeroes {n=5}
[0, 0, 0, 0, 0] : Vect 5 Nat

v : Vect 10 Nat
v = zeroes
```

Список литературы

 Brady, Edwin (Март, 2017). *Type-Driven Development with Idris*. Manning.