Функциональное программирование с зависимыми типами на языке Idris

Лекция 2. Теоретические основы верификации ПО средствами зависимых типов.

В. Н. Брагилевский

21 ноября 2017 г.

Факультет компьютерных наук, НИУ «Высшая школа экономики»

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

Типы в языках программирования [Pierce, 2002]

Формальные методы в разработке ПО

Спецификация и поведение программы

- системы типов
- формальные подходы к тестированию

- проверка моделей
- SMT-солверы
- абстрактная интерпретация
- мониторинг времени выполнения

- логики Хоара
- модальные логики
- языки алгебраических спецификаций
- денотационные семантики

Системы типов

Определение [Pierce, 2002]

Система типов — это гибко управляемый синтаксический метод доказательства отсутствия в программе определенных видов поведения при помощи классификации выражений языка по разновидностям вычисляемых ими значений.

- Статическая типизация
  - проверка типов (type checking)
  - вывод типов (type inference)
  - консервативность
- Динамическая проверка типов
  - ошибки типов времени выполнения

Предназначение типов

- Выявление ошибок
- Абстракция
- Документация
- Безопасность
- Эффективность
- Интерактивная разработка

Зависимые типы

Зависимые типы

для программистов

Классический пример: векторы фиксированного размера

Тип вектора

```
Vect : Nat -> Type -> Type
```

Пример вектора

```
v : Vect 3 Integer
v = [10, 5, 1]
```

Зависимый тип = тип, зависящий от значения

```
v : Vect 3 Integer
v = [10, 5]
```

```
When checking right hand side of v with expected type
         Vect 3 Int

When checking argument xs to constructor Data.Vect.:::
        Type mismatch between
                Vect 0 elem (Type of [])
        and
                Vect 1 Int (Expected type)

        Specifically:
                Type mismatch between
                        0
                and
                        1
```

Функции над векторами

```
cons : a -> Vect n a -> Vect (n + 1) a

head : Vect (n + 1) a -> a

vjoin : Vect n a -> Vect m a
             -> Vect (n + m) a
```

Type-Driven Development (интерактивная разработка)

- 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

Пример: длины строк из вектора

allLengths-vect.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: Сумма векторов

allLengths-vect.idr

```
vadd : Num a => Vect n a ->
        Vect n a -> Vect n a
vadd [] [] = []
vadd (x :: xs) (y :: ys) = x + y ::
                             vadd xs ys
```

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

Зависимые типы

для теоретиков

$$
\begin{tikzcd}
 & \lambda\omega \arrow[rr] & & \lambda\Pi\omega \\
\lambda 2 \arrow[ur] \arrow[rr] & & \lambda\Pi 2 \arrow[ur] & \\
 & \lambda\underline{\omega} \arrow[uu] \arrow[rr] & & \lambda\Pi\underline{\omega} \arrow[uu] \\
\lambda{\to} \arrow[uu] \arrow[ur] \arrow[rr] & & \lambda\Pi \arrow[uu] \arrow[ur] &
\end{tikzcd}
$$

Из чего состоит *теория*?

- термы и вычисление (операционная семантика), контексты
- система типизации (правила вывода)
- алгоритмы проверки и/или вывода типов
- отношение к равенству типов и импредикативности
- отношение к данным (например, индуктивные типы данных)
- вопросы разрешимости
- мета-теория: свойство Чёрча–Россера, унификация, нормализация
- теоретико-множественная и теоретико-категорная реализации

Соответствие Карри–Ховарда

Высказывание A	**Тип** A
истинность	населённость (наличие термов)
True	любой населённый тип, $() : ()$
False	$\perp$ (тип без значений)
доказательство	терм требуемого типа, $M : A$
$A \& B$ (конъюнкция)	$A \times B$ (произведение)
$\dfrac{A \quad B}{A \& B}$	$\dfrac{M : A \qquad N : B}{(M, N) : A \times B}$
$\dfrac{A \& B}{A} \qquad \dfrac{A \& B}{B}$	$\dfrac{M : A \times B}{\pi_1 M : A} \qquad \dfrac{M : A \times B}{\pi_2 M : B}$

Высказывание A	**Тип** A
истинность	населённость (наличие термов)
$A \vee B$ (дизъюнкция)	$A + B$ (сумма, disjoint sum)
$\dfrac{A}{A \vee B} \qquad \dfrac{B}{A \vee B}$	$\dfrac{M : A}{M_L : A + B} \qquad \dfrac{N : B}{N_R : A + B}$
$\dfrac{A \vee B \qquad \begin{array}{c}[A]^x \\ \vdots \\ C\end{array} \qquad \begin{array}{c}[B]^y \\ \vdots \\ C\end{array}}{C}$	сопоставление с образцом (case-выражение)

Высказывание A	**Тип** A
истинность	населённость (наличие термов)
$A \supset B$ (импликация)	$A \to B$ (функция)
$$\dfrac{\begin{matrix}[A]^x \\ \vdots \\ B\end{matrix}}{A \supset B}$$	$$\dfrac{\begin{matrix}[x : A]^x \\ \vdots \\ N : B\end{matrix}}{\lambda x.N : A \to B}$$
$$\dfrac{A \qquad A \supset B}{B}$$	$$\dfrac{L : A \qquad M : A \to B}{ML : B}$$

Пример программы $(B \times A) \to (A \times B)$

$$\dfrac{\dfrac{\dfrac{[z : B \times A]^z}{\pi_2 z : A} \quad \dfrac{[z : B \times A]^z}{\pi_1 z : B}}{(\pi_2 z, \pi_1 z) : A \times B}}{\lambda z.(\pi_2 z, \pi_1 z) : (B \times A) \to (A \times B)}$$

$$\dfrac{\dfrac{\dfrac{[B \& A]^z}{A}\,\&\text{-E}_2 \quad \dfrac{[B \& A]^z}{B}\,\&\text{-E}_1}{A \& B}\,\&\text{-I}}{(B \& A) \supset (A \& B)}\,\supset\text{-I}^z$$

Высказывание A	**Тип** A
истинность	населённость (наличие термов)
Предикат $P(x)$, $x \in X$	зависимый тип, $T : X \to \mathtt{Type}$
$\forall x \in X,\ P(x)$	$\Pi_{x:X} T(x)$, зависимая функция
$\exists \in X,\ P(x)$	$\Sigma_{x:X} T(x)$, зависимая пара

Верификация ПО

Доказательство утверждения в логике предикатов соответствует наличию терма в λ-исчислении с подходящей системой типов!

- Coq
- Agda
- **Idris**
- NuRPL
- F*
- ...

Где можно узнать подробнее?

1. Пирс. Типы в языках программирования.
2. Löh, McBride, Swierstra. A tutorial implementation of a dependently typed lambda calculus (2010).
3. Расширение языка Haskell зависимыми типами, лекции 1–3 (видео): GHC Core и SystemD, `https://youtube.com/bravit111`.
4. Соответствие Карри–Ховарда: от математической логики к программированию, `http://www.mccme.ru/dubna/2017/notes/bragilevsky-notes.pdf` (записки лекций), `http://www.mathnet.ru/conf982` (видео)

Список литературы

- *Curry–Howard correspondence*. URL: https://en.wikipedia.org/wiki/Curry-Howard_correspondence.
- Pierce, Benjamin C. (2002). *Types and Programming Languages*. Имеется русский перевод: Типы в языках программирования, Москва, 2012.
- Wadler, Philip (2015). "Propositions As Types". В: *Commun. ACM* 58.12, с. 75—84. ISSN: 0001-0782. DOI: 10.1145/2699407.