Программирование с зависимыми типами

Эридан Доморацкий
(на основе курса Валерия Исаева)

7 сентября 2024 г.

Мотивация

- Типизация в языках программирования позволяет выражать свойства программ
- Чем мощнее система типов, тем больше свойств она позволяет выразить
- Зависимые типы позволяют полностью описывать спецификацию программы

Пример

- В простых системах типов мы можем приписать функции сортировки такой тип:

$$\texttt{sort} : \texttt{List}\ \alpha \to \texttt{List}\ \alpha$$

- Этот тип ничего не говорит о результате работы функции, кроме того, что это список элементов того же типа, что и исходный
- В языке с зависимыми типами мы можем уточнить её тип до:

$$\texttt{sort} : \texttt{List}\ \alpha \to \texttt{SortedList}\ \alpha$$

- И даже так мы всё ещё не полностью описали её...
  Что осталось за кадром?

Альтернативы

- Если мы хотим описать тип отсортированных списков, то нам нужно уметь выражать произвольные логические формулы
- Тогда мы можем определить этот тип следующим способом;

$$\{\mathtt{xs} : \mathtt{List}\ \alpha \mid \forall 2 \leq i \leq \mathtt{length}\ \mathtt{xs}, \mathtt{xs[i\ -\ 1]} \leq \mathtt{xs[i]}\}$$

- Если мы хотим реализовать функцию `sort`, то нам нужно не только уметь выражать утверждения, но и их доказательства
- Тогда мы могли бы разделить язык на две части: отдельно программы и отдельно доказательства

Соответствие Карри-Говарда

- Зависимые типы предоставляют более гибкий и удобный подход
- Логические формулы можно записывать на языке типов, тогда доказательство — это программа соответствующего типа:

Логическая связка	$\bot$	$\top$	$\to$	$\wedge$	$\vee$
Конструктор типа	`Void`	`()`	$\to$	$\times$	`Either`

- Благодаря этому нет необходимости в двух разных языках

Зависимые типы

- При помощи простых типов можно выражать только формулы пропозициональной логики:

$$((P \to Q) \to P) \to P \quad \simeq \quad ((\alpha \to \beta) \to \alpha) \to \alpha$$

- Для формулировки интересных утверждений нам нужны кванторы
- Аналогами кванторов являются зависимые типы

Лямбда-куб

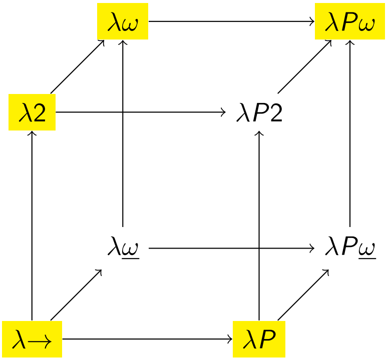

Рис.: Лямбда-куб

- $\lambda\to$ — просто-типизированное λ-исчисление, STLC
- $\lambda 2$ — System F, Haskell
- $\lambda\omega$ — GHC Haskell
- λP — STLC с зависимыми типами
- $\lambda P\omega$ — исчисление конструкций, Coq, Agda

Типы и виды

- В STLC мы отдельно обрабатываем типы и термы, это две разные синтаксические категории
- Чем дальше мы от STLC находимся на λ-кубе, тем больше сходства возникает между типами и термами
- В $\lambda P\omega$ мы уже не отличаем термы и типы
- Вместо этого, термам приписываются типы, типам приписываются виды...
- А дальше?

Уровни типов

- Вместо введения новых систем контроля для систем контроля (типы, кайнды, боксы, и т. п.) вводятся числовые уровни типов
- То есть типы, у которых под квантором расположены другие типы, получают уровень на единицу выше...

Σ-типы

- Как закодировать квантор существования в типах?
- Квантор существования можно представлять как дизъюнкцию:

$$\exists x \in A.\ P(x) \simeq P(x_1) \vee P(x_1) \vee \ldots \vee P(x_n)$$

- Дизъюнкцию мы кодируем как тип-сумму:

$$P(x_1) \vee P(x_1) \vee \ldots \vee P(x_n) \simeq P\ x_1 + P\ x_1 + \ldots + P\ x_n$$

- Обобщим тип-сумму до Σ-типа:

$$P\ x_1 + P\ x_1 + \ldots + P\ x_n \simeq \sum_{x:A} P\ x$$

- С другой стороны Σ-типы обобщают типы-произведения до типов зависимого произведения, то есть тип второго элемента зависимой пары зависит от значения первого элемента

Π-типы

- Как закодировать квантор всеобщности в типах?
- Квантор всеобщности можно представлять как конъюнкцию:

$$\forall x \in A.\ P(x) \simeq P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n)$$

- Конъюнкцию мы кодируем как тип-произведение:

$$P(x_1) \wedge P(x_2) \wedge \ldots \wedge P(x_n) \simeq P\ x_1 \times P\ x_2 \times \ldots \times P\ x_n$$

- Обобщим тип-произведение до Π-типа:

$$P\ x_1 \times P\ x_2 \times \ldots \times P\ x_n \simeq \prod_{x:A} P\ x$$

- С другой строны Π-типы обобщают стрелочные типы (типы-экспоненты) до типов зависимых функций, то есть тип возвращаемого значения зависимой функции зависит от значения переданного аргумента

Применения

Языки с зависимыми типами используют для двух разных целей:

- Во-первых, для верификации программ
- Во-вторых, так как язык является полноценной логикой, его можно использовать для формализации математики

Реализации

- Существует несколько языков с зависимыми типами: Agda, Idris, Coq, Arend и т. д.
- Мы будем использовать Arend
  - Использует гомотопическую теорию типов (HoTT), но нас это не будет обременять до поры до времени
  - Использует исключительно λ-синтаксис (в отличие от Coq и ему подобных)
  - Разрабатывается в JetBrains