Типы в языках программирования
Лекция 2. Простые типы

Денис Николаевич Москвин

ИТМО, корпоративная магистратура JetBrains
Разработка ПО / Software Engineering

17.02.2021

План лекции

План лекции

Бестиповая арифметика

Термы и значения

```
t ::=
      true
      false
      if t then t else t
      0
      succ t
      pred t
      iszero t
v ::=
      true
      false
      nv
nv ::=
      0
      succ nv
```

Отношение вычисления

$$\texttt{if true then } t_2 \texttt{ else } t_3 \longrightarrow t_2 \qquad (E-\text{IfTrue})$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \longrightarrow t_3 \qquad (E-\text{IfFalse})$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \longrightarrow \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \qquad (E-\text{If})$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{succ } t_1 \longrightarrow \texttt{succ } t_1'} \qquad (E-\text{Succ})$$

$$\texttt{pred } 0 \longrightarrow 0 \qquad (E-\text{PredZero})$$

$$\texttt{pred}(\texttt{succ } nv_1) \longrightarrow nv_1 \qquad (E-\text{PredSucc})$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{pred } t_1 \longrightarrow \texttt{pred } t_1'} \qquad (E-\text{Pred})$$

$$\texttt{iszero } 0 \longrightarrow \texttt{true} \qquad (E-\text{IsZeroZero})$$

$$\texttt{iszero}(\texttt{succ } nv_1) \longrightarrow \texttt{false} \qquad (E-\text{IsZeroSucc})$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{iszero } t_1 \longrightarrow \texttt{iszero } t_1'} \qquad (E-\text{IsZero})$$

Типы

- У нас имелись тупиковые термы, вроде `pred(false)`.
- Хотелось бы иметь возможность *статически* проверять, зайдет ли вычисление в тупик.
- Типы позволят это сделать, но *консервативно*, то есть отбросив при этом и некоторые нетупиковые, например `if true then 0 else false`.

Введем новые синтаксические формы

Типы

```
T ::=
    Bool
    Nat
```

Отношение типизации

Правила типизации

$$\texttt{true} : \texttt{Bool} \qquad (\mathrm{T} - \mathrm{True})$$

$$\texttt{false} : \texttt{Bool} \qquad (\mathrm{T} - \mathrm{False})$$

$$\frac{t_1 : \texttt{Bool} \quad t_2 : T \quad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T} \qquad (\mathrm{T} - \mathrm{If})$$

$$0 : \texttt{Nat} \qquad (\mathrm{T} - \mathrm{Zero})$$

$$\frac{t_1 : \texttt{Nat}}{\texttt{succ } t_1 : \texttt{Nat}} \qquad (\mathrm{T} - \mathrm{Succ})$$

$$\frac{t_1 : \texttt{Nat}}{\texttt{pred } t_1 : \texttt{Nat}} \qquad (\mathrm{T} - \mathrm{Pred})$$

$$\frac{t_1 : \texttt{Nat}}{\texttt{iszero } t_1 : \texttt{Bool}} \qquad (\mathrm{T} - \mathrm{IsZero})$$

Отношение типизации

Определение

Отношение типизации (typing relation) для арифметических выражений — это наименьшее бинарное отношение между термами и типами, удовлетворяющее всем правилам с предыдущего слайда.

Определение

Терм $\mathtt{t}$ называется ***типизируемым*** (typable) (или ***корректно типизированным***, well-typed), если существует тип $\mathtt{T}$ такой, что $\mathtt{t} : \mathtt{T}$.

Лемма об инверсии

Лемма (генерации, порождения)

1. Если $\texttt{true} : \texttt{R}$, то $\texttt{R} = \texttt{Bool}$.
2. Если $\texttt{false} : \texttt{R}$, то $\texttt{R} = \texttt{Bool}$.
3. Если $\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : \texttt{R}$, то $t_1 : \texttt{Bool}$, $t_2 : \texttt{R}$ и $t_3 : \texttt{R}$.
4. Если $0 : \texttt{R}$, то $\texttt{R} = \texttt{Nat}$.
5. Если $\texttt{succ } t_1 : \texttt{R}$, то $\texttt{R} = \texttt{Nat}$ и $t_1 : \texttt{Nat}$.
6. Если $\texttt{pred } t_1 : \texttt{R}$, то $\texttt{R} = \texttt{Nat}$ и $t_1 : \texttt{Nat}$.
7. Если $\texttt{iszero } t_1 : \texttt{R}$, то $\texttt{R} = \texttt{Bool}$ и $t_1 : \texttt{Nat}$.

Лемма об инверсии дает нам рекурсивный алгоритм вывода типов.

Теорема

Все подтермы типизируемого терма типизируемы.

Дерево вывода типа

Правила типизации позволяют строить дерево вывода типа.
Докажем, например, что
`if iszero 0 then 0 else pred 0 : Nat`.

Единственность типа

Теорема о единственности типа

Всякий терм t имеет не более одного типа. Если терм типизируем, то его тип выводится с использованием правил типизации единственным образом.

Доказательство: Структурная индукция по t, с использованием леммы генерации. ■

Это свойство выполняется далеко не для всех систем.

Корректность типа (типобезопасность)

Безопасность = продвижение + сохранение (Харпер)

- Продвижение: Правильно типизированный терм не может быть тупиковым (либо это значение, либо может быть проделан следующий шаг в соответствии с правилами вычисления).
- Сохранение: Если над правильно типизированным термом выполнить шаг вычисления, то получившийся терм также правильно типизирован.

Второе свойство почти всегда можно сформулировать более сильным образом — тип сохраняется при вычислениях.

Канонические формы

Канонические формы (canonical forms) для некоторого типа — это корректно типизированные значения этого типа. Например, без правила (E-PredZero) у нас имелась бы (корректно типизированная) нормальная форма `pred 0`, не являющаяся значением.

Лемма о канонических формах

1. Если v — значение типа `Bool`, то v равно либо `true`, либо `false`.
2. Если v — значение типа `Nat`, то v является числовым значением.

Доказательство:
1. Значения — это `true`, `false`, `0` или `succ nv`. Первые два дают искомые КФ, а вторые два имеют тип `Nat`.
2. Самостоятельно. ■

Продвижение

Теорема о продвижении

Пусть $\mathtt{t} : \mathtt{T}$. Тогда либо $\mathtt{t}$ является значением, либо существует некоторый $\mathtt{t}'$, такой, что $\mathtt{t} \longrightarrow \mathtt{t}'$.

Доказательство: Индукция по дереву вывода $\mathtt{t} : \mathtt{T}$ с использованием леммы о канонических формах. 7 случаев.
Например, *случай* $\mathtt{t} = \mathtt{pred}\ \mathtt{t}_1$, $\mathtt{t}_1 : \mathtt{Nat}$. По IH либо (1) $\mathtt{t}_1$ — значение, либо (2) имеется $\mathtt{t}_1'$, такой что $\mathtt{t}_1 \longrightarrow \mathtt{t}_1'$.
(1) По лемме о канон.формах $\mathtt{t}_1$ это либо 0, либо $\mathtt{succ}\ \mathtt{nv}_1$. Тогда к $\mathtt{t}$ применимо (E-PredZero) или (E-PredSucc).
(2) По правилу (E-Pred) $\mathtt{pred}\ \mathtt{t}_1 \longrightarrow \mathtt{pred}\ \mathtt{t}_1'$.
и т.д. ■

Сохранение

Теорема о редукции субъекта (сохранении)

Пусть $\mathtt{t} : \mathtt{T}$ и $\mathtt{t} \longrightarrow \mathtt{t}'$, тогда $\mathtt{t}' : \mathtt{T}$.

Доказательство: Индукция по дереву вывода $\mathtt{t} : \mathtt{T}$ с анализом соответствующих вычислительных правил. ■

План лекции

Синтаксис

- В λ-исчислении вводятся переменные.

Термы

```
t ::=
    x
    λx.t
    t t
```

- Здесь `x` — переменная, а `t` — метапеременная.
- Значения для начала введем максимально простым образом

Значения

```
v ::=
    λx.t
```

Простейшая операционная семантика

Вычисление

$$\frac{t_1 \longrightarrow t_1'}{t_1\, t_2 \longrightarrow t_1'\, t_2} \quad (E - App1)$$

$$\frac{t_2 \to t_2'}{v_1\, t_2 \longrightarrow v_1\, t_2'} \quad (E - App2)$$

$$(\lambda x.\, t)v \longrightarrow [x \mapsto v]t \quad (E - AppAbs)$$

- Является ли эта семантика детерминированной?

Простейшая операционная семантика

Вычисление

$$\frac{t_1 \longrightarrow t_1'}{t_1\, t_2 \longrightarrow t_1'\, t_2} \quad (E - App1)$$

$$\frac{t_2 \to t_2'}{v_1\, t_2 \longrightarrow v_1\, t_2'} \quad (E - App2)$$

$$(\lambda x.\, t)v \longrightarrow [x \mapsto v]t \quad (E - AppAbs)$$

- Является ли эта семантика детерминированной?
- Есть ли тут тупиковые термы?

Полная β-редукция

Вычисление

$$\frac{t_1 \longrightarrow t_1'}{t_1 t_2 \longrightarrow t_1' t_2} \quad (EF - App1)$$

$$\frac{t_2 \longrightarrow t_2'}{t_1 t_2 \longrightarrow t_1 t_2'} \quad (EF - App2)$$

$$\frac{t \longrightarrow t'}{\lambda x.\, t \longrightarrow \lambda x.\, t'} \quad (EF - Abs)$$

$$(\lambda x.\, t_1) t_2 \longrightarrow [x \mapsto t_2] t_1 \quad (EF - AppAbs)$$

- Это недетерминированная семантика.
- Понятие значения не используется (фактически совпадает с NF).

Нормальный порядок редукции до NF

Синтаксические категории

```
nf  ::= λx.nf        nanf ::= x          na ::= x
        nanf                  nanf nf           t1 t2
```

Вычисление

$$\frac{\mathrm{na} \longrightarrow \mathrm{na}'}{\mathrm{na}\,\mathrm{t} \longrightarrow \mathrm{na}'\,\mathrm{t}} \quad (\mathrm{ENO}-\mathrm{App1})$$

$$\frac{\mathrm{t} \longrightarrow \mathrm{t}'}{\mathrm{nanf}\,\mathrm{t} \longrightarrow \mathrm{nanf}\,\mathrm{t}'} \quad (\mathrm{ENO}-\mathrm{App2})$$

$$\frac{\mathrm{t} \longrightarrow \mathrm{t}'}{\lambda x.\,\mathrm{t} \longrightarrow \lambda x.\,\mathrm{t}'} \quad (\mathrm{ENO}-\mathrm{Abs})$$

$$(\lambda x.\,\mathrm{t}_1)\mathrm{t}_2 \longrightarrow [x \mapsto \mathrm{t}_2]\mathrm{t}_1 \quad (\mathrm{ENO}-\mathrm{AppAbs})$$

Аппликативный порядок редукции до NF

Синтаксические категории

```
nf  ::= λx.nf        nanf ::= x          na ::= x
        nanf                  nanf nf           t1 t2
```

Вычисление

$$\frac{\mathrm{na} \longrightarrow \mathrm{na}'}{\mathrm{na}\,\mathrm{t} \longrightarrow \mathrm{na}'\,\mathrm{t}} \quad (\mathrm{EAO}-\mathrm{App1})$$

$$\frac{\mathrm{t} \to \mathrm{t}'}{\mathrm{nanf}\,\mathrm{t} \longrightarrow \mathrm{nanf}\,\mathrm{t}'} \quad (\mathrm{EAO}-\mathrm{App2})$$

$$\frac{\mathrm{t} \to \mathrm{t}'}{\lambda x.\,\mathrm{t} \longrightarrow \lambda x.\,\mathrm{t}'} \quad (\mathrm{EAO}-\mathrm{Abs})$$

$$(\lambda x.\,\mathrm{t}_1)\,\mathrm{nf} \longrightarrow [x \mapsto \mathrm{nf}]\mathrm{t}_1 \quad (\mathrm{EAO}-\mathrm{AppAbs})$$

$$\frac{\mathrm{t}_1 \longrightarrow \mathrm{t}_1'}{(\lambda x.\,\mathrm{t}_2)\mathrm{t}_1 \longrightarrow (\lambda x.\,\mathrm{t}_2)\mathrm{t}_1'} \quad (\mathrm{EAO}-\mathrm{App3})$$

План лекции

Индексы Де Брауна: знакомство

- ***Индексы Де Брауна (De Bruijn)*** представляют альтернативный способ представления термов.
- Переменные не именуются, а индексируются, индекс показывает, сколько лямбд «назад» переменная была связана:

$$\begin{aligned} \lambda x.\, \lambda y.\, x\, y &\leftrightarrow \lambda\, \lambda\, 1\, 0 \\ \lambda x.\, x\, (\lambda y.\, x\, y\, y) &\leftrightarrow \lambda\, 0\, (\lambda\, 1\, 0\, 0) \\ \lambda z.\, (\lambda y.\, y (\lambda x.\, x))(\lambda x.\, z\, x) &\leftrightarrow \lambda\, (\lambda\, 0\, (\lambda\, 0))\, (\lambda\, 1\, 0) \end{aligned}$$

- Свободные переменные при этом получают индексы, превышающие число лямбд слева:

$$\lambda x.\, z\, x\, y \leftrightarrow \lambda\, 2\, 0\, 1$$

- Порядок нумерации свободных переменных подразумевается фиксированным через набор биндеров в некотором внешнем окружении.

Индексы Де Брауна: формализм

- Термы задаются индуктивно:

$$\begin{array}{lcl} n \in \mathbb{N} & \Rightarrow & n \in \Lambda, \\ M, N \in \Lambda & \Rightarrow & (M\,N) \in \Lambda, \\ M \in \Lambda & \Rightarrow & (\lambda\,M) \in \Lambda. \end{array}$$

- Стандартные соглашения:
  - внешние скобки опускаются;
  - операция применения термов ассоциативна влево, то есть $0\,1\,2$ обозначает $((0\,1)\,2)$;
  - тело терма простирается вправо насколько это возможно, то есть $\lambda\,0\,0$ обозначает $\lambda\,(0\,0)$, а не $(\lambda\,0)\,0$.

Индексы Де Брауна: сдвиг

- Вводится операция *сдвига* или *подъема* (shift или lifting), нотация $\uparrow_m^k$:

$$\uparrow_m^k n = \begin{cases} n, & \text{если } n < m, \\ n + k, & \text{если } n \geqslant m, \end{cases}$$

$$\uparrow_m^k (P\,Q) = (\uparrow_m^k P)\,(\uparrow_m^k Q),$$

$$\uparrow_m^k (\lambda\,P) = \lambda\,(\uparrow_{m+1}^k P).$$

- Величина m называется *отсечением*, меньшие индексы не сдвигаются.
- В сдвиге участвуют лишь свободные переменные: см. увеличение отсечения в последнем равенстве.
- На практике обычно используется версия с нулевым отсечением, для нее вводят нотацию $\uparrow^k = \uparrow_0^k$.

Индексы Де Брауна: подстановка

- Вводится операция *подстановки* $[j \mapsto N]\, M$ терма N вместо свободной переменной j в терм M :

$$
\begin{aligned}
[j \mapsto N]\, n &= \begin{cases} N, & \text{если } n = j, \\ n, & \text{если } n \neq j, \end{cases} \\
[j \mapsto N]\, (P\, Q) &= ([j \mapsto N]\, P)\, ([j \mapsto N]\, Q), \\
[j \mapsto N]\, (\lambda\, P) &= \lambda\, ([j + 1 \mapsto \uparrow^1 N]\, P).
\end{aligned}
$$

- При «переходе через лямбду» подставляемый индекс увеличивается для сохранения ссылки на тот же внешний биндер.
- При «переходе через лямбду» ссылки на свободные переменные в N тоже нужно увеличить, отсюда $\uparrow^1 N$.

Индексы Де Брауна: редукция

- В терминах сдвига и подстановки одношаговая β-редукция определяется так

$$(\lambda\, M)\, N \longrightarrow_\beta \uparrow^{-1} [0 \mapsto \uparrow^1 N]\, M.$$

- Сдвиг $\uparrow^1$ в $[0 \mapsto \uparrow^1 N]$ необходим для согласованности ссылок на свободные переменные, поскольку M жило под лямбдой.
- Сдвиг $\uparrow^{-1}$ необходим, потому что при редукции M «вылупляется» из лямбды.
- Почему при отрицательном сдвиге здесь не могут возникнуть отрицательные индексы?

Индексы Де Брауна: альтернативная реализация

- Можно все операции со сдвигами упаковать во вспомогательную функцию псевдоподстановки:

$$(\lambda\, M)\, N \longrightarrow_\beta [0 \rightsquigarrow N]\, M.$$

- Здесь псевдоподстановка $[j \rightsquigarrow N]\ M$ определена так

$$[j \rightsquigarrow N]\, n = \begin{cases} n, & \text{если } n < j, \\ \uparrow^n N, & \text{если } n = j, \\ n - 1, & \text{если } n > j, \end{cases}$$

$$[j \rightsquigarrow N]\, (P\, Q) = ([j \rightsquigarrow N]\, P)\, ([j \rightsquigarrow N]\, Q),$$

$$[j \rightsquigarrow N]\, (\lambda\, P) = \lambda\, ([j + 1 \rightsquigarrow N]\, P).$$

- Эта реализация слегка эффективнее, поскольку накопленный сдвиг в N осуществляется только один раз.

План лекции

Просто типизированное λ-исчисление

Синтаксис

```
t ::= ...
      x
      λx:T.t
      t t
v ::= ...
     λx:T.t
T ::= ...
      T→T
Γ ::=
      <>
      Γ, x:T
```

Здесь многоточие означает, что у нас есть все, связанное с `Bool` (и, возможно, с `Nat`).

Операционная семантика и типизация

Вычисление

$$\frac{t_1 \longrightarrow t_1'}{t_1\, t_2 \longrightarrow t_1'\, t_2} \quad (E - App1)$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\, t_2 \longrightarrow v_1\, t_2'} \quad (E - App2)$$

$$(\lambda x : T.\, t)v \longrightarrow [x \mapsto v]t \quad (E - AppAbs)$$

Типизация

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \quad (T - Var)$$

$$\frac{\Gamma, x : T \vdash t : S}{\Gamma \vdash \lambda x : T.\, t \ : \ T \to S} \quad (T - Abs)$$

$$\frac{\Gamma \vdash t_1 : T \to S \qquad \Gamma \vdash t_2 : T}{\Gamma \vdash t_1\, t_2 : S} \quad (T - App)$$

Дерево вывода типа

Покажем, что $(\lambda x{:}\mathtt{Bool}.x)\ \mathtt{true} : \mathtt{Bool}$.

Лемма об инверсии

Лемма (генерации)

1. Если $\Gamma \vdash \mathtt{true} : \mathtt{R}$, то $\mathtt{R} = \mathtt{Bool}$.
2. Если $\Gamma \vdash \mathtt{false} : \mathtt{R}$, то $\mathtt{R} = \mathtt{Bool}$.
3. Если $\Gamma \vdash \mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 : \mathtt{R}$, то $\Gamma \vdash t_1 : \mathtt{Bool}$, $\Gamma \vdash t_2 : \mathtt{R}$ и $\Gamma \vdash t_3 : \mathtt{R}$.
4. Если $\Gamma \vdash x : \mathtt{R}$, то $x : \mathtt{R} \in \Gamma$.
5. Если $\Gamma \vdash \lambda x : \mathtt{T}.\, t\ :\ \mathtt{R}$, то $\mathtt{R} = \mathtt{T} \to \mathtt{S}$ для некоторого $\mathtt{S}$, и $\Gamma, x : \mathtt{T} \vdash t : \mathtt{S}$.
6. Если $\Gamma \vdash t_1\, t_2\ :\ \mathtt{R}$, то существует тип $\mathtt{S}$, такой что $\Gamma \vdash t_1 : \mathtt{S} \to \mathtt{R}$ и $\Gamma \vdash t_2 : \mathtt{S}$.

Теорема

Все подтермы типизируемого терма типизируемы.

Единственность типа

Теорема о единственности типа

В любом заданном контексте Γ терм t имеет не более одного типа.

Доказательство: Структурная индукция по t, с использованием леммы генерации. ■

Что должно входить в контекст, чтобы мы могли получить дерево вывода типа?

Канонические формы

Лемма о канонических формах

1. Если v — значение типа `Bool`, то v равно либо `true`, либо `false`.
2. Если v — значение типа $S \to R$, то $v = \lambda x : S.\, t$.

- Следующий шаг — лемма о продвижении, но здесь она будет верна только для замкнутых термов.
  Например, терм $x\,$`true` — NF, но не значение.
- Типизируем ли этот терм?

Продвижение

Теорема о продвижении

Пусть t — замкнутый, правильно типизированный терм, то есть $\vdash t : T$ для некоторого типа T. Тогда либо t является значением, либо существует некоторый t', такой, что $t \longrightarrow t'$.

Доказательство: Индукция по дереву вывода $t : T$ с использованием леммы о канонических формах.
Интересный случай: $t = t_1\ t_2$, при этом $t_1 : S \to T$, $t_2 : S$. По IH это либо значения, либо в них можно сделать шаг.
1. В t_1 — шаг: к t применимо E-App1;
2. t_1 — значение, в t_2 — шаг: к t применимо E-App2;
3. t_1 и t_2 — значения: по лемме о КФ t_1 имеет вид лямбда-абстракции, то есть к t применимо E-AppAbs. ■

Сохранение

Лемма (сохранение типа при подстановке)

Если $\Gamma, x : S \vdash t : T$ и $\Gamma \vdash s : S$, то $\Gamma \vdash [x \mapsto s]t : T$

Доказательство: Индукция по глубине вывода $t : T$. ■

Теорема о редукции субъекта (сохранении)

Пусть $\Gamma \vdash t : T$ и $t \longrightarrow t'$, тогда $\Gamma \vdash t' : T$.

Доказательство: Индукция по дереву вывода $t : T$ с анализом соответствующих вычислительных правил. ■

Замечание: поскольку мы работаем только с замкнутыми термами, контексты при вычислениях можно отбросить.