Типы в языках программирования
Лекция 2. Простые типы

Денис Николаевич Москвин

СПбАУ РАН

22.02.2018

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
```

Типы

- У нас имелись тупиковые термы, вроде $\texttt{pred}(\texttt{false})$.
- 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

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

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

$$\frac{\mathtt{t}_1 : \mathtt{Bool} \quad \mathtt{t}_2 : \mathsf{T} \quad \mathtt{t}_3 : \mathsf{T}}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathsf{T}} \qquad (\mathrm{T-If})$$

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

$$\frac{\mathtt{t}_1 : \mathtt{Nat}}{\mathtt{succ}\ \mathtt{t}_1 : \mathtt{Nat}} \qquad (\mathrm{T-Succ})$$

$$\frac{\mathtt{t}_1 : \mathtt{Nat}}{\mathtt{pred}\ \mathtt{t}_1 : \mathtt{Nat}} \qquad (\mathrm{T-Pred})$$

$$\frac{\mathtt{t}_1 : \mathtt{Nat}}{\mathtt{iszero}\ \mathtt{t}_1 : \mathtt{Bool}} \qquad (\mathrm{T-IsZero})$$

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 (генерации)

1. Если $\mathtt{true} : \mathrm{R}$, то $\mathrm{R} = \mathtt{Bool}$.
2. Если $\mathtt{false} : \mathrm{R}$, то $\mathrm{R} = \mathtt{Bool}$.
3. Если $\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathrm{R}$, то $\mathtt{t}_1 : \mathtt{Bool}$, $\mathtt{t}_2 : \mathrm{R}$ и $\mathtt{t}_3 : \mathrm{R}$.
4. Если $0 : \mathrm{R}$, то $\mathrm{R} = \mathtt{Nat}$.
5. Если $\mathtt{succ}\ \mathtt{t}_1 : \mathrm{R}$, то $\mathrm{R} = \mathtt{Nat}$ и $\mathtt{t}_1 : \mathtt{Nat}$.
6. Если $\mathtt{pred}\ \mathtt{t}_1 : \mathrm{R}$, то $\mathrm{R} = \mathtt{Nat}$ и $\mathtt{t}_1 : \mathtt{Nat}$.
7. Если $\mathtt{iszero}\ \mathtt{t}_1 : \mathrm{R}$, то $\mathrm{R} = \mathtt{Bool}$ и $\mathtt{t}_1 : \mathtt{Nat}$.

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

Определение

Канонические формы (canonical forms) для некоторого типа — это правильно типизированные значения этого типа.

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. ■

Сохранение

Теорема о редукции субъекта (сохранении)

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

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

План лекции

1. Арифметика с типами
2. Бестиповое λ-исчисление
3. Просто типизированное λ-исчисление

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
```

Вычисление

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

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

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

- Является ли эта семантика детерминированной?

Простейшая операционная семантика

Вычисление

$$\frac{t_1 \longrightarrow t_1'}{t_1 t_2 \longrightarrow t_1' t_2} \qquad (\mathsf{E-App1})$$

$$\frac{t_2 \longrightarrow t_2'}{v_1 t_2 \longrightarrow v_1 t_2'} \qquad (\mathsf{E-App2})$$

$$(\lambda x.\, t)v \longrightarrow [x \mapsto v]t \qquad (\mathsf{E-AppAbs})$$

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

Полная β-редукция

Вычисление

$$\frac{t_1 \to t_1'}{t_1\,t_2 \to t_1'\,t_2} \qquad (EF-App1)$$

$$\frac{t_2 \to t_2'}{t_1\,t_2 \to t_1\,t_2'} \qquad (EF-App2)$$

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

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

- 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

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

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

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

$$(\lambda x.\, t_1)t_2 \longrightarrow [x \mapsto t_2]t_1 \quad (\mathsf{ENO-AppAbs})$$

Аппликативный порядок редукции до NF

Синтаксические категории

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

Вычисление

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

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

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

$$(\lambda x.\,t_1)\,\mathtt{nf} \longrightarrow [x \mapsto \mathtt{nf}]t_1 \quad (\mathrm{EAO-AppAbs})$$

$$\frac{t_1 \longrightarrow t_1'}{(\lambda x.\,t_2)t_1 \longrightarrow (\lambda x.\,t_2)t_1'} \quad (\mathrm{EAO-App3})$$

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

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

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

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

- При этом все α-эквивалентные термы кодируются одинаково, коллизий захвата переменной не возникает.

План лекции

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

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

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

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

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 `(λx:Bool.x) true : Bool`.

Лемма об инверсии

Лемма (генерации)

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

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 $\mathtt{S} \to \mathtt{R}$, то $v = \lambda x : \mathtt{T}.\, t$.

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

Продвижение

Теорема о продвижении

Пусть $\mathtt{t}$ — замкнутый, правильно типизированный терм, то есть $\vdash \mathtt{t} : \mathtt{T}$ для некоторого типа $\mathtt{T}$. 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.
Интересный случай: $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$, при этом $\mathtt{t}_1 : \mathtt{S} \rightarrow \mathtt{T}$, $\mathtt{t}_2 : \mathtt{S}$. По IH это либо значения, либо в них можно сделать шаг.
1. В $\mathtt{t}_1$ — шаг: к $\mathtt{t}$ применимо E-App1;
2. $\mathtt{t}_1$ — значение, в $\mathtt{t}_2$ — шаг: к $\mathtt{t}$ применимо E-App2;
3. $\mathtt{t}_1$ и $\mathtt{t}_2$ — значения: по лемме о КФ $\mathtt{t}_1$ имеет вид лямбда-абстракции, то есть к $\mathtt{t}$ 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 $t \to t'$, 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 можно отбросить.