Типы в языках программирования
Лекция 8. Экзистенциальные типы

Денис Николаевич Москвин

СПбАУ РАН

19.04.2018

План лекции

1. Определение экзистенциальных типов
2. Абстракция через экзистенцию
3. Кодирование экзистенциальных типов

План лекции

Мотивация

- *Универсальные типы с логической точки зрения:*
  элемент типа $\forall \mathtt{X}.\, \mathtt{T}$ является значением, которое имеет тип $[\mathtt{X} \mapsto \mathtt{S}]\mathtt{T}$ при любом выборе $\mathtt{S}$.
- *Универсальные типы операционно:*
  элементом типа $\forall \mathtt{X}.\, \mathtt{T}$ является функция $\lambda \mathtt{X} : *.\, \mathtt{t}$, которая переводит произвольный тип $\mathtt{S}$ в элемент типа $[\mathtt{X} \mapsto \mathtt{S}]\mathtt{T}$.
- *Экзистенциальные типы логически:*
  элемент типа $\{\exists \mathtt{X},\, \mathtt{T}\}$ является значением, которое имеет тип $[\mathtt{X} \mapsto \mathtt{S}]\mathtt{T}$ для некоторого $\mathtt{S}$.
- *Экзистенциальные типы операционно:*
  элементом типа $\{\exists \mathtt{X},\, \mathtt{T}\}$ является пара $\{^*\mathtt{S}, \mathtt{t}\}$, состоящая из некоторого типа $\mathtt{S}$ и элемента $\mathtt{t}$ типа $[\mathtt{X} \mapsto \mathtt{S}]\mathtt{T}$.

Экзистенциальный тип

- Значение типа $\{\exists \mathtt{X},\ \mathtt{T}\}$ рассматривается как *пакет* (или *модуль*) $\{^*\mathtt{S}, \mathtt{t}\}$ с одним (скрытым) компонентом-типом и одним компонентом-термом. (Альтернативный синтаксис: $\mathtt{pack\ X\ =\ S\ with\ t}\}$.)
- Тип $\mathtt{S}$ называют *скрытым типом представления* (hidden representation type), или *типом-свидетелем* (witness type) пакета.
- Например, пакет $\{^*\mathtt{Nat}, \{\mathtt{a} = 5, \lambda \mathtt{x} : \mathtt{Nat}.\, \mathtt{succ}(\mathtt{x})\}\}$ имеет экзистенциальный тип $\{\exists \mathtt{X}, \{\mathtt{a} : \mathtt{X}, \mathtt{f} : \mathtt{X} \to \mathtt{X}\}\}$.

Экзистенциальный тип

- Значение типа $\{\exists \mathtt{X},\ \mathtt{T}\}$ рассматривается как *пакет* (или *модуль*) $\{^{*}\mathtt{S}, \mathtt{t}\}$ с одним (скрытым) компонентом-типом и одним компонентом-термом. (Альтернативный синтаксис: $\mathtt{pack\ X\ =\ S\ with\ t}\}$.)
- Тип $\mathtt{S}$ называют *скрытым типом представления* (hidden representation type), или *типом-свидетелем* (witness type) пакета.
- Например, пакет $\{^{*}\mathtt{Nat}, \{\mathtt{a} = 5, \lambda \mathtt{x} : \mathtt{Nat}.\, \mathtt{succ}(\mathtt{x})\}\}$ имеет экзистенциальный тип $\{\exists \mathtt{X}, \{\mathtt{a} : \mathtt{X}, \mathtt{f} : \mathtt{X} \to \mathtt{X}\}\}$.
- Однако, с тем же успехом можно этому пакету приписать экзистенциальный тип $\{\exists \mathtt{X}, \{\mathtt{a} : \mathtt{X}, \mathtt{f} : \mathtt{X} \to \mathtt{Nat}\}\}$.
- Автоматическое введение экзистенциального типа невозможно, это дело рук программиста.

Введение экзистенциального типа

- Экзистенциальные типы вводим приписыванием

```
  p = {*Nat, {a=5, f=λx:Nat.succ(x)}} as {∃X, {a:X, f:X→X}};
▷ p : {∃X, {a:X, f:X→X}}
```

- Правило типизации для введения квантора существования

Типизация

$$\frac{\Gamma \vdash t : [X \mapsto S]T}{\Gamma \vdash \{*S, t\} \text{ as } \{\exists X,\ T\} : \{\exists X,\ T\}} \quad (T - \text{Pack})$$

- Разные типы-свидетели могут порождать один тип

```
  p2 = {*Nat, 0} as {∃X, X}
▷ p2 : {∃X, X}
  p3 = {*Bool, true} as {∃X, X}
▷ p3 : {∃X, X}
```

Элиминация экзистенциального типа

- Элиминация пакета подобна директиве `open` или `import`.
- Можно использовать содержимое модуля в каком-то другом месте программы, при этом абстрактная природа типа модуля сохраняется.

Типизация

$$\frac{\Gamma \vdash \mathtt{t} : \{\exists \mathtt{X},\ \mathtt{T}\} \qquad \Gamma, \mathtt{X} : *, \mathtt{x} : \mathtt{T} \vdash \mathtt{s} : \mathtt{S}}{\Gamma \vdash \mathtt{let}\ \{\mathtt{X}, \mathtt{x}\} = \mathtt{t}\ \mathtt{in}\ \mathtt{s}\ :\ \mathtt{S}} \quad (\mathrm{T-UnPack})$$

- Здесь $\mathtt{X}$ и $\mathtt{x}$ связывают типовую и термовую компоненты пакета и используются при вычислении $\mathtt{s}$.
- Иногда используют синтаксис
  $\mathtt{open}\ \mathtt{t}\ \mathtt{as}\ \{\mathtt{X}, \mathtt{x}\}\ \mathtt{in}\ \mathtt{s}$.

Элиминация экзистенциального типа (примеры)

Типизация

$$\frac{\Gamma \vdash \mathtt{t} : \{\exists \mathtt{X}, \mathtt{T}\} \qquad \Gamma, \mathtt{X} : *, \mathtt{x} : \mathtt{T} \vdash \mathtt{s} : \mathtt{S}}{\Gamma \vdash \mathtt{let}\ \{\mathtt{X}, \mathtt{x}\} = \mathtt{t}\ \mathtt{in}\ \mathtt{s}\ :\ \mathtt{S}} \quad (\mathrm{T} - \mathrm{UnPack})$$

```
  p4={*Nat,{a=0,f=λx:Nat.succ(x)}} as {∃X,{a:X,f:X→Nat}};
▷ p4 : {∃X, {a:X, f:X→Nat}}

let {X,x} = p4 in x.f x.a;
▷ 1 : Nat

let {X,x} = p4 in (λy:X. x.f y) x.a;
▷ 1 : Nat
```

Последний пример показывает, что в теле формы устранения можно также упоминать типовую переменную X.

Элиминация экзистенциального типа (запреты)

Типизация

$$\frac{\Gamma \vdash \mathtt{t} : \{\exists \mathtt{X},\ \mathtt{T}\} \qquad \Gamma, \mathtt{X} : *, \mathtt{x} : \mathtt{T} \vdash \mathtt{s} : \mathtt{S}}{\Gamma \vdash \mathtt{let}\ \{\mathtt{X}, \mathtt{x}\} = \mathtt{t}\ \mathtt{in}\ \mathtt{s}\ :\ \mathtt{S}} \quad (\mathrm{T-UnPack})$$

Тип представления пакета при проверке в теле распаковки должен оставаться абстрактным:

```
  p4={*Nat,{a=0,f=λx:Nat.succ(x)}} as {∃X,{a:X,f:X→Nat}};
▷ p4 : {∃X, {a:X, f:X→Nat}}

let {X,x} = p4 in succ(x.a);
▷ Error!

  p5={*Bool,{a=true,f=λx:Bool.42}} as {∃X,{a:X,f:X→Nat}};
▷ p5 : {∃X, {a:X, f:X→Nat}}
```

Ошибка возвращается справедливо, p5 имеет тот же тип!

Элиминация экзистенциального типа (запреты 2)

Типизация

$$\frac{\Gamma \vdash \mathtt{t} : \{\exists \mathtt{X},\ \mathtt{T}\} \qquad \Gamma, \mathtt{X} : *, \mathtt{x} : \mathtt{T} \vdash \mathtt{s} : \mathtt{S}}{\Gamma \vdash \mathtt{let}\ \{\mathtt{X}, \mathtt{x}\} = \mathtt{t}\ \mathtt{in}\ \mathtt{s}\ :\ \mathtt{S}} \quad (\mathrm{T-UnPack})$$

В контексте заключения правила типовая переменная X отсутствует, поэтому следующий пример приведет к ошибке области видимости:

```
  p4={*Nat,{a=0,f=λx:Nat.succ(x)}} as {∃X,{a:X,f:X→Nat}};
▷ p4 : {∃X, {a:X, f:X→Nat}}

let {X,x} = p4 in x.a;
▷ Error!
```

Расширение System F

Новые синтаксические формы

```
t ::= ...
      {*T, t} as T
      let {X, x} = t in t

v ::= ...
      {*T, v} as T

T ::= ...
      {∃X, T}
```

Синтаксис термов расширяется упаковкой и распаковкой, вводится значение-пакет и экзистенциальный тип.

Правила вычисления

Вычисление

$$\texttt{let}\ \{\mathtt{X}, \mathtt{x}\} = (\{^*\mathtt{S}, \mathtt{v}\}\ \texttt{as}\ \mathtt{T})\ \texttt{in}\ \mathtt{s} \longrightarrow [\mathtt{X} \mapsto \mathtt{S}][\mathtt{x} \mapsto \mathtt{v}]\mathtt{s} \qquad (\mathrm{E-UnPackPack})$$

$$\frac{\mathtt{t} \longrightarrow \mathtt{t}'}{\{^*\mathtt{S}, \mathtt{t}\}\ \texttt{as}\ \mathtt{T} \longrightarrow \{^*\mathtt{S}, \mathtt{t}'\}\ \texttt{as}\ \mathtt{T}} \qquad (\mathrm{E-Pack})$$

$$\frac{\mathtt{t} \longrightarrow \mathtt{t}'}{\texttt{let}\ \{\mathtt{X}, \mathtt{x}\} = \mathtt{t}\ \texttt{in}\ \mathtt{s} \longrightarrow \texttt{let}\ \{\mathtt{X}, \mathtt{x}\} = \mathtt{t}'\ \texttt{in}\ \mathtt{s}} \qquad (\mathrm{E-UnPack})$$

Первое правило аналогично компоновке модулей: символические имена ($\mathtt{X}$ и $\mathtt{x}$), заменяются реальным содержимым подключаемого модуля.

План лекции

Абстрактные типы данных

- Абстрактный тип данных (ADT) состоит из
  1. имени типа A;
  2. типа конкретного представления T;
  3. реализации некоторых операций над типом T;
  4. барьера абстракции.

```
ADT counter =
  type Counter
  representation Nat
  signature
    new : Counter,
    get : Counter → Nat,
    inc : Counter → Counter;
  operations
    new = 1,
    get = λi:Nat. i,
    inc = λi:Nat. succ(i);
counter.get (counter.inc counter.new);
```

АТД на экзистенциальных типах

```
counterADT =
  {*Nat,
    {new = 1,
    get = λi:Nat. i,
    inc = λi:Nat. succ(i)}}
 as {∃Counter,
      {new: Counter,
      get: Counter → Nat,
      inc: Counter → Counter}};
```

```
  let {Counter, counter} = counterADT in
    counter.get (counter.inc counter.new);
▷ 2 : Nat
```

Замена внутреннего представления на, скажем, запись с полем типа Nat не повлияет на использование.

План лекции

Кодирование пар в System F

Для пары конкретных типов A и B пара значений кодируется так

```
PairAB = ∀Y.(A→B→Y)→Y

pairAB = λa:A. λb:B. λY:*. λf:A→B→Y. f a b

fstAB = λp:PairAB. p [A] (λa:A. λb:B. a)
sndAB = λp:PairAB. p [B] (λa:A. λb:B. b)
```

Идея: закодировать пару из типа и значения аналогично:

$$\{\exists X, T\} \doteq \forall Y.(\forall X.T \to Y) \to Y$$

Кодирование упаковки и распаковки пакета

Идея: закодировать пару из типа и значения как пару:

$$\{\exists X,T\} \doteq \forall Y.(\forall X.T \to Y) \to Y$$

Кодирование упаковки пакета

$$\{^{*}S,t\} \text{ as } \{\exists X,T\} \doteq \lambda Y{:}*.\ \lambda f{:}(\forall X.T \to Y).\ f\ [S]\ t$$

Кодирование распаковки пакета

$$\text{let } \{X,x\} = t \text{ in } s \doteq t\ [S]\ (\lambda X{:}*.\ \lambda x{:}T.\ s)$$

Экзистенциальные типы в Haskell

```
{-# LANGUAGE ExistentialQuantification #-}
module Existentials where

data Obj = forall a. (Show a) ⇒ Obj a

xs :: [Obj]
xs = [Obj 42, Obj True, Obj 'c']
```

```
*Existentials> foldr (λ(Obj x) s → show x ++ s) "" xs
"42True'c'"
```

Экзистенциальные типы в Haskell (2)

Включив `RankNTypes`, можно задать кодирование и декодирование в универсальные функциональные типы

```
fromObj :: Obj→forall r.(forall a.Show a⇒a→r)→r
fromObj (Obj x) k = k x

toObj :: (forall r.(forall a.Show a⇒a→r)→r)→Obj
toObj f = f Obj
```

```
*Existentials> :t fromObj (Obj 5)
fromObj (Obj 5) :: (forall a. Show a ⇒ a → r) → r
*Existentials> fromObj (Obj 5) (λa → show a ++ show a)
"55"
```