Типы в языках программирования
Лекция 1. Язык арифметических выражений

Денис Николаевич Москвин

ИТМО, корпоративная магистратура JetBrains
Разработка ПО / Software Engineering

10.02.2021

План лекции

1. Что такое типы
2. Отношение вычисления
3. Бестиповая арифметика

План лекции

1. Что такое типы
2. Отношение вычисления
3. Бестиповая арифметика

Системы типов

Два подхода к типам:

- Теория типов как раздел логики. (Рассел - Рамсей/Черч - Карри/Говард - Мартин-Лёф - Берарди/Терлоу/Барендрегт - Воеводский)
- Типы в прикладной информатике. (Системы типов для языков программирования.)

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

Бенджамин Пирс

Для чего нужны типы?

- Выявление некоторых классов ошибок.

```
GHCi> length 42
GHCi> foldr (+) "ABCDE"
```

- Обеспечивают механизм *абстракции*, позволяя отделить протокол использования от деталей реализации.
- Обеспечивают *безопасность языка*, через механизм целостности абстракций.
- Документация.
- Эффективность.

Какие бывают системы типов?

Возможны классификации систем типам по разным аспектам:

- статические (static) vs динамические (dinamic);
- явные (explicit) vs неявные (implicit);
- сильные (strong) vs слабые (weak);
- структурные (structural) vs именные (nominal).
- консервативные (conservative) vs выразительные (expressive).

В слабой системе это можно типизировать

```
x = 5;
y = "37";
z = x + y;
```

План лекции

1. Что такое типы
2. Отношение вычисления
3. Бестиповая арифметика

Термы исчисления

В стандартной форме Бэкуса-Наура:

Термы (выражения) исчисления

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

Здесь `t` — метапеременная.

Можно описать и *индуктивно*, и в виде *схем правил вывода*, и в виде *конкретной иерархии*.

Термы исчисления, индуктивно

Термы, индуктивно

Множество термов T — это наименьшее множество, обладающее следующими свойствами:

$$\{\texttt{true}, \texttt{false}, 0\} \subset T$$

$$\texttt{t} \in T \Rightarrow \{\texttt{succ t}, \texttt{pred t}, \texttt{iszero t}\} \subset T$$

$$\texttt{t}_1, \texttt{t}_2, \texttt{t}_3 \in T \Rightarrow \texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \in T$$

Почему мы не говорим о скобках, хотя, вроде, надо бы?

Термы исчисления, индуктивно

Термы, индуктивно

Множество термов T — это наименьшее множество, обладающее следующими свойствами:

$$\{\texttt{true}, \texttt{false}, 0\} \subset T$$

$$\mathrm{t} \in T \Rightarrow \{\texttt{succ t}, \texttt{pred t}, \texttt{iszero t}\} \subset T$$

$$\mathrm{t}_1, \mathrm{t}_2, \mathrm{t}_3 \in T \Rightarrow \texttt{if } \mathrm{t}_1 \texttt{ then } \mathrm{t}_2 \texttt{ else } \mathrm{t}_3 \in T$$

Почему мы не говорим о скобках, хотя, вроде, надо бы?

Мы определяем термы как деревья в абстрактном синтаксисе. При записи в линейном, строковом виде используем группирующие скобки.

Термы исчисления, конкретно

Конкретная иерархия

$$
\begin{array}{lcll}
T_0 & = & \varnothing & \\
T_{i+1} & = & & \{\texttt{true}, \texttt{false}, 0\} \\
 & & \cup & \{\texttt{succ t}, \texttt{pred t}, \texttt{iszero t} \mid \texttt{t} \in T_i\} \\
 & & \cup & \{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \mid \texttt{t}_1, \texttt{t}_2, \texttt{t}_3 \in T_i\} \\
 & & & \\
T & = & \bigcup_i T_i &
\end{array}
$$

Сколько элементов содержит T_1? T_2?

Является ли эта иерархия кумулятивной?

Индукция на термах

Индуктивное определение множества констант терма:

Определение

$$
\begin{array}{lcl}
cnst(\texttt{true}) & = & \{\texttt{true}\} \\
cnst(\texttt{false}) & = & \{\texttt{false}\} \\
cnst(\texttt{0}) & = & \{\texttt{0}\} \\
cnst(\texttt{succ t}) & = & cnst(\texttt{t}) \\
cnst(\texttt{pred t}) & = & cnst(\texttt{t}) \\
cnst(\texttt{iszero t}) & = & cnst(\texttt{t}) \\
cnst(\texttt{if } \texttt{t}_1 \texttt{ then } \texttt{t}_2 \texttt{ else } \texttt{t}_3) & = & cnst(\texttt{t}_1) \cup cnst(\texttt{t}_2) \cup cnst(\texttt{t}_3)
\end{array}
$$

Определите $size$ (количество узлов всех типов) и $depth$ (глубина дерева).

Индуктивное доказательство

Принцип структурной индукции

Если из того, что свойство выполнено для всех непосредственных подтермов данного терма выводимо, что свойство выполнено для данного терма, то свойство верно для любого терма.

Лемма

$$\forall \mathrm{t}\ |cnst(\mathrm{t})| \leqslant size(\mathrm{t})$$

Доказываем структурной индукцией, перебирая всевозможные синтаксические формы.
Можно доказывать и индукцией по глубине и по размеру.

Семантические стили

- **Операционная семантика:** описываем *абстрактную машину*. Для простых языков *состояние* это терм, а поведение задается *функцией перехода*, которая либо описывает следующее состояние, либо говорит, что достигнуто конечное состояние. *Смысл* терма — его конечное состояние.
- **Денотационная семантика:** смысл терма — некоторый математический объект из *семантического домена*. Для каждого терма задается *функция интерпретации*.

Операционная семантика для булева подмножества

Для того, чтобы описать процесс вычисления, надо

- задать подмножество термов, называемых *значениями*;
- задать отношение *вычисления за один шаг*: $\mathrm{t} \longrightarrow \mathrm{t}'$.

На булевом подмножестве

Термы и значения

```
t ::=
      true
      false
      if t then t else t
v ::=
      true
      false
```

Здесь `t` и `v` описывают *синтаксические категории*.

Отношение вычисления на булевом подмножестве

Вычисление

$$\texttt{if true then } t_2 \texttt{ else } t_3 \;\rightarrow\; t_2 \qquad (\mathrm{E-IfT})$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \;\rightarrow\; t_3 \qquad (\mathrm{E-IfF})$$

$$\frac{t_1 \rightarrow t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \;\rightarrow\; \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \qquad (\mathrm{E-If})$$

Как будет вычисляться следующее выражение?

```
if true then (if false then false else false) else true
```

Можно построить дерево вывода для вычислений: листья — *рабочие правила (computation rules)*: $(\mathrm{E-If})$ и $(\mathrm{E-IfF})$, узлы — *правила соответствия (congruence rules)*: $(\mathrm{E-If})$.
Ветвлений нет!

Детерминированность одношагового вычисления

Теорема

Если $t \to t'$ и $t \to t''$, то $t' = t''$.

Доказательство: индукция по дереву вывода для вычисления $t \to t'$. Смотрим в корень, разбираем возможные варианты структуры t, исходя из последнего правила, затем сравниваем с деревом для $t \to t''$.

- $(E - IfT)$: $t = \texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3$ и $t_1 = \texttt{true}$. В дереве вывода для $t \to t''$ не может быть другого правила.
- $(E - IfF)$: аналогично.
- $(E - If)$: аналогично, с использованием IH.

■

Нормальная форма

Определение

Если к терму неприменимо ни одно вычислительное правило, то говорят, что он находится в *нормальной форме*.

Теорема

Любое значение является нормальной формой.

Теорема

Если t — нормальная форма, то t является значением.

Доказательство: пусть t не значение, докажем, что не NF, структурной индукцией. «Не значение» должно иметь вид $\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3$. $t_1 = \texttt{false}$ или $t_1 = \texttt{true}$ — не NF, в третьем случае пользуемся IH. ■

Второе утверждение верно только для нашего простого исчисления, первое должно выполнятся для любого разумного.

Многошаговое вычисление

Определение

Отношение многошагового вычисления $\mathrm{t} \twoheadrightarrow \mathrm{t}'$ — это рефлексивно-транзитивное замыкание отношения (одношагового) вычисления.

Теорема (единственность нормальной формы)

Если u и v — нормальные формы, и $\mathrm{t} \twoheadrightarrow u$ и $\mathrm{t} \twoheadrightarrow v$, то $u = v$.

Доказательство: следует из детерминированности одношагового вычисления. ■

Завершимость вычисления

Теорема

Для каждого терма t существует нормальная форма t' такая, что $t \twoheadrightarrow t'$.

Доказательство: На каждом шаге вычисления размер терма сокращается. ■

(Эта теорема верна только для ограниченного круга исчислений.)

Упражнение

Добавим в отношение вычисления еще одно правило соответствия.

Вычисление

$$\ldots$$

$$\frac{t_2 \to t_2'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \;\to\; \texttt{if } t_1 \texttt{ then } t_2' \texttt{ else } t_3} \quad (E - Funny2)$$

Как это повлияет на приведенные выше теоремы?

Упражнение

Добавим в отношение вычисления еще одно правило соответствия.

Вычисление

$$\ldots$$

$$\frac{t_2 \to t_2'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \;\to\; \texttt{if } t_1 \texttt{ then } t_2' \texttt{ else } t_3} \quad (\mathrm{E-Funny2})$$

Как это повлияет на приведенные выше теоремы?

Все теоремы будут выполняться, кроме детерминированности одношагового вычисления.

Для получившегося исчисления будет верно свойство ромба с одношаговым ребром.

План лекции

1. Что такое типы
2. Отношение вычисления
3. Бестиповая арифметика

Расширение до арифметики

Термы и значения

```
t ::= ...
      0
      succ t
      pred t
      iszero t
v ::= ...
      nv
nv ::=
      0
      succ nv
```

Введена синтаксическая категория числовых значений `nv`.

Расширение вычислений

Вычисление (новые правила)

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

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

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

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

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

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

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

Детерминировано ли вычисление `pred(succ(pred 0))`?

Можно ли описать целые числа, добавив `pred nv` в `nv`?

Детерминированность одношагового вычисления

Для нашей расширенной системы по-прежнему верна теорема о детерминированности вычислений.

Теорема

Если $t \longrightarrow t'$ и $t \longrightarrow t''$, то $t' = t''$.

- Можно расширить вычисления для многошаговых и ввести понятие нормальной формы.
- Верна ли теорема о завершимости?

Детерминированность одношагового вычисления

Для нашей расширенной системы по-прежнему верна теорема о детерминированности вычислений.

Теорема

Если $t \longrightarrow t'$ и $t \longrightarrow t''$, то $t' = t''$.

- Можно расширить вычисления для многошаговых и ввести понятие нормальной формы.
- Верна ли теорема о завершимости? Да.

Детерминированность одношагового вычисления

Для нашей расширенной системы по-прежнему верна теорема о детерминированности вычислений.

Теорема

Если $t \longrightarrow t'$ и $t \longrightarrow t''$, то $t' = t''$.

- Можно расширить вычисления для многошаговых и ввести понятие нормальной формы.
- Верна ли теорема о завершимости? Да.
- Верна ли теорема о том, всякая нормальная форма является значением?

Детерминированность одношагового вычисления

Для нашей расширенной системы по-прежнему верна теорема о детерминированности вычислений.

Теорема

Если $t \longrightarrow t'$ и $t \longrightarrow t''$, то $t' = t''$.

- Можно расширить вычисления для многошаговых и ввести понятие нормальной формы.
- Верна ли теорема о завершимости? Да.
- Верна ли теорема о том, всякая нормальная форма является значением? Нет!

Тупиковое состояние

Определение

Терм называется *тупиковым* (stuck), если он находится в нормальной форме, но не является значением.

Пример тупикового терма

```
succ true
```

Для нашей абстрактной машины тупиковое состояние это ошибка времени исполнения.

Семантика с большим шагом

Семантика с большим шагом описывает вычислительные правила в посылках (и заключениях) через понятие «терм t имеет при вычислении значение v», нотация $t \Downarrow v$.

Вычисление

$$v \Downarrow v \qquad (B - \text{Value})$$

$$\frac{t_1 \Downarrow \texttt{true} \qquad t_2 \Downarrow v_2}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \Downarrow v_2} \qquad (B - \text{IfTrue})$$

$$\frac{t_1 \Downarrow \texttt{false} \qquad t_3 \Downarrow v_3}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \Downarrow v_3} \qquad (B - \text{IfFalse})$$

...

Упражнение: продолжите самостоятельно.

Упражнение: докажите, что $t \twoheadrightarrow v$ равносильно $t \Downarrow v$. Верно ли это для нормальных форм?

Упражнение

Предположим, что нам захотелось поменять стратегию вычисления для булева языка так, чтобы ветви `then` и `else` в условном выражении вычислялись (в указанном порядке) до того, как вычислится само условие.
Покажите, как нужно изменить правила вычисления, чтобы добиться такого поведения.