Основания языков программирования

А. М. Пеленицын
ulysses4ever@gmail.com

Факультет математики, механики и компьютерных наук
Южный федеральный университет

Семинар «Введение в теоретическую информатику»
27 мая 2011 г.

Языки программирования

1. Синтаксис: правила составления программ.
2. Системы типов: правила отбрасывания неправильно составленных программ.
3. Семантика: способ получения результатов работы программы.

План

1. Бестиповое λ-исчисление
   - Язык
   - Вычисление
   - Программирование в λ-исчислении
2. Типизированное λ-исчисление
3. Семантики языков программирования

1. Бестиповое λ-исчисление
   - Язык
   - Вычисление
   - Программирование в λ-исчислении
2. Типизированное λ-исчисление
3. Семантики языков программирования

Функции как формулы

Определение функций из школы

- $f(x) = x^2$,
- $\mathrm{add}(x, y) = x + y$,
- $I(f) = \int_0^1 f\ dx$.

Определение функций как λ-термов

- $\lambda x\,.\,x^2$,
- $\lambda x\,.\,\lambda y\,.\,x + y$,
- $\lambda f\,.\ \int_0^1 f\ dx$.

Вычисление — подстановка

- $f(5) = 5^2 = 25$,
- $\mathrm{add}(3, 2) = 3 + 2 = 5$,
- $I(x^2) = \int_0^1 x^2\ dx = 1/3$.

- $(\lambda x\,.\,x^2)5 \to_\beta 5^2$,
- $((\lambda x\,.\,\lambda y\,.\,x + y)3)2 \to_\beta$
  $(\lambda y\,.\,3 + y)2 \to_\beta 3 + 2$,
- $(\lambda f\,.\ \int_0^1 f\ dx)x^2 \to_\beta \int_0^1 x^2\ dx$.

1. Бестиповое λ-исчисление
   - Язык
   - Вычисление
   - Программирование в λ-исчислении

2. Типизированное λ-исчисление

3. Семантики языков программирования

λ-термы

Определение

Грамматика G_λ задаётся правилами:

$$\Lambda ::= V \mid (\lambda V \,.\, \Lambda) \mid (\Lambda\Lambda),$$

где V обозначает имя переменной $(x, y, z \ldots)$. Язык $L(G_\lambda)$ называется множеством λ-термов.

- Правило 2 — λ-абстракция, правило 3 — аппликация.
- Примеры:
  1. $\underline{\Lambda} \to (\lambda x \,.\, \underline{\Lambda}) \to (\lambda x \,.\, x)$ — тождественная функция.
  2. $\underline{\Lambda} \to (\lambda x \,.\, \underline{\Lambda}) \to (\lambda x \,.\, y)$ — функция-константа.
  3. $\underline{\Lambda} \to (\underline{\Lambda}\Lambda) \to ((\lambda x \,.\, \underline{\Lambda})\Lambda) \to ((\lambda x \,.\, (\underline{\Lambda}\Lambda))\Lambda) \to ((\lambda x \,.\, (x\underline{\Lambda}))\Lambda) \to ((\lambda x \,.\, (xx))\underline{\Lambda}) \to ((\lambda x \,.\, (xx))(\lambda y \,.\, \underline{\Lambda})) \to ((\lambda x \,.\, (xx))(\lambda y \,.\, (\underline{\Lambda}\Lambda))) \to ((\lambda x \,.\, (xx))(\lambda y \,.\, (y\underline{\Lambda}))) \to ((\lambda x \,.\, (xx))(\lambda y \,.\, (yy)))$ — Ω-терм.
  4. $(\lambda f \,.\, (\lambda x \,.\, (f(fx))))$.

Соглашения о скобках

1. Внешние скобки опускаются.
2. Аппликация ассоциирует влево: $((MN)K)L \sim MNKL$.
3. Абстракция жадная вправо: $\lambda x \,.\, (MN) \sim \lambda x \,.\, MN$.

Примеры

1. $(\lambda x \,.\, x) \sim \lambda x \,.\, x$,
2. $((\lambda x \,.\, (xx))(\lambda y \,.\, (yy))) \sim (\lambda x \,.\, xx)(\lambda y \,.\, yy)$,
3. $(\lambda f \,.\, (\lambda x \,.\, (f(fx)))) \sim \lambda f \,.\, \lambda x \,.\, f(fx)$.

Переименование связанных переменных

Мотивация:

$$f(x) = x^2 \sim f(y) = y^2.$$

Определение (α-эквивалентность)

Термы M и N называются α-эквивалентными, если они отличаются только именами переменных, связанных λ-абстракцией.
Записывается: $M =_\alpha N$.

Примеры

1. $\lambda x \,.\, x =_\alpha \lambda y \,.\, y$;
2. $(\lambda x \,.\, xx)(\lambda y \,.\, yy) =_\alpha (\lambda x \,.\, xx)(\lambda x \,.\, xx) =_\alpha (\lambda z \,.\, zz)(\lambda x \,.\, xx)$;
3. $\lambda x \,.\, y \neq_\alpha \lambda x \,.\, z$.

Вычисление: отношение $\to_\beta$

Определение

λ-терм вида $(\lambda x\,.\,M)N$ называется редексом.

Определение

$\to_\beta$ — это бинарное отношение на множестве λ-термов:

$$(\lambda x\,.\,M)N \to_\beta [N/x]M,$$

где операция $[N/x]M$ означает подстановку терма N в терм M вместо всех свободных вхождений x.

Пример:

$$\begin{aligned}(\lambda x\,.\,y)(\underline{(\lambda z\,.\,zz)(\lambda w\,.\,w)}) &\to_\beta (\lambda x\,.\,y)(\underline{(\lambda w\,.\,w)(\lambda w\,.\,w)})\\ &\to_\beta \underline{(\lambda x\,.\,y)(\lambda w\,.\,w)}\\ &\to_\beta y.\end{aligned}$$

Или: $\underline{(\lambda x\,.\,y)((\lambda z\,.\,zz)(\lambda w\,.\,w))} \to_\beta y$.

Подстановка: проблема

1. Функция от двух аргументов, которая применяет первый аргумент ко второму: $\lambda f\,.\,\lambda x\,.\,fx$.
2. Константная функция, возвращающая x: $\lambda y\,.\,x$.
3. Переменная: z.

Применив константную функцию (2) к переменной (3) ожидаем получить x. Однако:

$$\underline{(\lambda f\,.\,\lambda x\,.\,fx)(\lambda y\,.\,x)}z \to_\beta \underline{(\lambda x\,.\,(\lambda y\,.\,x)x)z} \to_\beta \underline{(\lambda y\,.\,z)z} \to_\beta z.$$

Проблема: произошёл захват свободной переменной x на первом шаге редукции.

Решение: переименовывать связанные переменные.

$$\begin{aligned}(\lambda f\,.\,\lambda x\,.\,fx)(\lambda y\,.\,x)z &=_\alpha \underline{(\lambda f\,.\,\lambda w\,.\,fw)(\lambda y\,.\,x)}z \\ &\to_\beta \underline{(\lambda w\,.\,(\lambda y\,.\,x)w)z} \\ &\to_\beta \underline{(\lambda y\,.\,x)z} \\ &\to_\beta x.\end{aligned}$$

Свойства $\to_\beta$

Определение

Говорят что терм M находится в нормальной форме, если он не содержит редексов.

Теорема

У каждого терма M нормальная форма единственна, если она существует.

Пример терма без нормальной формы:

$$\underline{(\lambda x\,.\,xx)(\lambda x\,.\,xx)} \to_\beta \underline{(\lambda x\,.\,xx)(\lambda x\,.\,xx)} \to_\beta \ldots \tag{Ω}$$

Ура, бесконечный цикл!

Свойства $\to_\beta$

Определение

Стратегией редукции называется правило, по которому выбирается очередной редекс в редуцируемом терме.

Определение

Стратегия редукции называется нормализующей, если она приводит к нормальной форме любой терм, имеющий нормальную форму.

Теорема

Нормализующие стратегии существуют.

1. Бестиповое λ-исчисление
    - Язык
    - Вычисление
    - Программирование в λ-исчислении

2. Типизированное λ-исчисление

3. Семантики языков программирования

Программирование в λ-исчислении

- $\texttt{true} \equiv \lambda x\,.\,\lambda y\,.\,x$; $\texttt{false} \equiv \lambda x\,.\,\lambda y\,.\,y$.
- $\texttt{if\_then\_else}\ C\ M\ N \equiv CMN$.

Пример:

$$\begin{aligned}
&\texttt{if_then_else}\,\texttt{true}\ \ M\ N \equiv \\
&\qquad \texttt{true}\ \ M\ N \equiv (\lambda x\,.\,\lambda y\,.\,x)MN \to_\beta \\
&\qquad\qquad (\lambda y\,.\,M)N \to_\beta M.
\end{aligned}$$

Программирование в λ-исчислении

- $\texttt{and} \ = \lambda x\,.\,\lambda y\,.\,\texttt{if\_then\_else}\; x\; y\; \texttt{false}$

Пример:

$$\texttt{and true false} \;\equiv$$
$$(\lambda x\,.\,\lambda y\,.\,\texttt{if\_then\_else}\; x\; y\; \texttt{false}\;)\texttt{true false} \;\to_\beta^*$$
$$\texttt{if\_then\_else}\,\texttt{true}\;\; \texttt{false}\;\; \texttt{false} \;\equiv$$
$$\texttt{true false false} \;\equiv (\lambda x\,.\,\lambda y\,.\,x)\texttt{false false} \;\to_\beta^*$$
$$\texttt{false}\;.$$

- $\texttt{or} \ = ?$
- $\texttt{not} \ = ?$

Свойства $\to_\beta$

Теорема

λ-исчисление с отношением $\to_\beta$ является полным по Тьюрингу вычислительным формализмом.

λ-исчисление vs языки программирования

Питер Ландин

- "Correspondence between ALGOL 60 and Church's Lambda-notation" // Com. ACM, 1965;
- "The next 700 programming languages" // Com. ACM, 1966;

  "A possible first step in the research program is 1700 doctoral theses called "A Correspondence between x and Church's λ-notation."

Введение типов: мотивация

Предположим, в λ-исчисление некоторым способом введены целые числа. Как понимать терм:

$$\texttt{if\_then\_else}\;(42)\;(1)\;(2)?$$

Худшая ситуация:

$$\texttt{if\_then\_else}\;(\langle\text{описание длинного и сложного вычисления}\rangle)\;(1)\;(2)?$$

Можно добавить к языку λ-исчисления типы — это поможет, не выполняя вычислений, исключать некоторые «неправильные» термы.

Язык типизированного λ-исчисления

Определение

Грамматика для типов задаётся следующими правилами:

$$T \ ::= \ B \mid T \to T,$$

где B обозначает «базовый тип» из некоторого фиксированного набора.

Пример: (Nat $\to$ Bool) $\to$ Nat $\to$ Bool (считая Nat и Bool базовыми типами).

Определение

Грамматика для термов задаётся следующими правилами:

$$\Lambda \ ::= \ V \mid \lambda V : T \,.\, \Lambda \mid \Lambda\Lambda.$$

Правила типизации

Обозначения

- $M\colon T$ — «терм M имеет тип T»;
- $x_1\colon T_1, \dots, x_n\colon T_n \vdash M\colon T$ — при условии, что x_i, свободные переменные в M, имеют типы T_i соответственно, терм M имеет тип T — утверждение о типизации (type judgement).

Правила

$$\textrm{VAR}\ \frac{}{\Gamma, x\colon A \vdash x\colon A}$$

$$\textrm{APP}\ \frac{\Gamma \vdash M\colon A \to B \qquad \Gamma \vdash N\colon A}{\Gamma \vdash MN\colon B}$$

$$\textrm{ABS}\ \frac{\Gamma, x\colon A \vdash M\colon B}{\Gamma \vdash \lambda x : A\,.\,M\colon A \to B}$$

Семантики языков программирования

- операционная,
- денотационная,
- аксиоматическая.