Рекурсия

Дмитрий Халанский

14 сентября 2020 г.

Каррирование

Введём такие лямбда-термы:

$$\begin{aligned} \text{curry} &:= \lambda f\ a\ b.\, f\ (\text{pair}\ a\ b) \\ \text{uncurry} &:= \lambda f\ p.\, f\ (\text{fst}\ p)\ (\text{snd}\ p) \end{aligned}$$

Они позволяют свободно переключаться между функциями, которые принимают один аргумент, и функциями, которые принимают несколько: просто засунем все нужные аргументы в пару, протащим её везде, где нужно, а потом подадим её компоненты в функцию, которая принимает их по отдельности.

$$\begin{aligned} \text{curry}\ f\ a\ b &= f\ (\text{pair}\ a\ b) \\ \text{uncurry}\ f\ (\text{pair}\ a\ b) &= f\ a\ b \end{aligned}$$

Цикл со счётчиком

Рассмотрим такую программу на Питоне:

```
def rec(f, a, n):
    acc = a
    for i in range(0, n):
        acc = f(i, acc)
    return acc
```

Просто вынесли в отдельную функцию цикл со счётчиком: осуществляем n шагов и на каждом шаге выполняем какие-то действия, меняя состояние программы.

Перепишем чуть иначе:

```
def rec(f, a, n):
    cur = 0 # almost the same as i
    acc = a
    for i in range(0, n):
        cur = cur + 1
        acc = f(cur, acc)
    return acc
```

Примитивная рекурсия

Перепишем в лямбда-исчисление:

- Исполняем n как число Чёрча с состоянием, заданным парой из `cur` и `acc`;
- Начальное состояние — `cur = 0` и `acc = a`;
- На каждом шаге обновляем состояние, кладя в `cur` значение $\texttt{cur} + 1$, а в аккумулятор — результат вызова $f(\texttt{cur}, \texttt{acc})$.

$$\text{step } f := \lambda p.\, \text{pair } (\text{suc } (\text{fst } p)) \; (\text{uncurry } f \; p)$$

Убедимся, что $\text{step } f \; (\text{pair } a \; b) = \text{pair } (\text{suc } a) \; (f \; a \; b)$. Тогда

$$\text{rec} := \lambda f \; a \; n.\, \text{snd } (n \; (\text{step } f) \; (\text{pair } 0 \; a))$$

Примитивная рекурсия: использование

$$\text{step } f \ (\text{pair } a \ b) = \text{pair } (\text{suc } a) \ (f \ a \ b)$$

$$\text{rec} := \lambda f \ a \ n. \text{snd } (n \ (\text{step } f) \ (\text{pair } 0 \ a))$$

Пример:

$$\text{fac} =$$

Примитивная рекурсия: использование

$$\text{step } f \text{ (pair } a \text{ } b) = \text{pair (suc } a) \text{ } (f \text{ } a \text{ } b)$$

$$\text{rec} := \lambda f \text{ } a \text{ } n. \text{snd } (n \text{ (step } f) \text{ (pair 0 } a))$$

Пример:

$$\text{fac} = \text{rec } (\lambda i \text{ } r. \text{mult (suc } i) \text{ } r) \text{ 1}$$

$$\begin{aligned}
&\text{fac 2} = \text{snd } (2 \text{ (step } (\lambda i \text{ } r. \text{mult (suc } i) \text{ } r)) \text{ (pair 0 1)}) = \\
&\text{snd } (\text{step } (\lambda i \text{ } r. \text{mult (suc } i) \text{ } r) \text{ ((step } (\lambda i \text{ } r. \text{mult (suc } i) \text{ } r)) \text{ (pair 0 1)))} = \\
&\text{snd } (\text{step } (\lambda i \text{ } r. \text{mult (suc } i) \text{ } r) \text{ (pair (suc 0) ((}\lambda i \text{ } r. \text{mult (suc } i) \text{ } r) \text{ 0 1)))} = \\
&\text{snd } (\text{step } (\lambda i \text{ } r. \text{mult (suc } i) \text{ } r) \text{ (pair 1 (mult (suc 0) 1)))} = \\
&\text{snd } (\text{step } (\lambda i \text{ } r. \text{mult (suc } i) \text{ } r) \text{ (pair 1 1)}) = \\
&\text{snd } (\text{pair (suc 1) ((}\lambda i \text{ } r. \text{mult (suc } i) \text{ } r) \text{ 1 1)}) = (\lambda i \text{ } r. \text{mult (suc } i) \text{ } r) \text{ 1 1} = \\
&\text{mult (suc 1) 1} = 2
\end{aligned}$$

Примитивная рекурсия: свойство

В общем случае,
$\mathrm{rec}\ f\ a\ n = f\ (n-1)\ (f\ (n-2)\ (f\ (n-3)\ (\cdots\ (f\ 1\ (f\ 0\ a))\cdots)))$.
Теперь можно разобрать пример покрупнее:
$\mathrm{fac}\ 3 = (\lambda i.\,\mathrm{mult}\ (\mathrm{suc}\ i))\ 2\ ((\lambda i.\,\mathrm{mult}\ (\mathrm{suc}\ i))\ 1\ ((\lambda i.\,\mathrm{mult}\ (\mathrm{suc}\ i))\ 0\ 1 =$
$\mathrm{mult}\ (\mathrm{suc}\ 2)\ (\mathrm{mult}\ (\mathrm{suc}\ 1)\ (\mathrm{mult}\ (\mathrm{suc}\ 0)\ 1)) =$
$\mathrm{mult}\ 3\ (\mathrm{mult}\ 2\ (\mathrm{mult}\ 1\ 1)) = 6$.
Свойство также позволяет найти какие-нибудь полезные функции.
Например, pred =

Примитивная рекурсия: свойство

В общем случае,
$\mathsf{rec}\ f\ a\ n = f\ (n-1)\ (f\ (n-2)\ (f\ (n-3)\ (\cdots\ (f\ 1\ (f\ 0\ a))\cdots)))$.
Теперь можно разобрать пример покрупнее:
$\mathsf{fac}\ 3 = (\lambda i.\,\mathsf{mult}\ (\mathsf{suc}\ i))\ 2\ ((\lambda i.\,\mathsf{mult}\ (\mathsf{suc}\ i))\ 1\ ((\lambda i.\,\mathsf{mult}\ (\mathsf{suc}\ i))\ 0\ 1 =$
$\mathsf{mult}\ (\mathsf{suc}\ 2)\ (\mathsf{mult}\ (\mathsf{suc}\ 1)\ (\mathsf{mult}\ (\mathsf{suc}\ 0)\ 1)) =$
$\mathsf{mult}\ 3\ (\mathsf{mult}\ 2\ (\mathsf{mult}\ 1\ 1)) = 6$.
Свойство также позволяет найти какие-нибудь полезные функции.
Например, $\mathsf{pred} = \mathsf{rec}\ \mathsf{K}\ 0$.

Комбинатор неподвижной точки

Свойство: $\mathsf{Y}\ F' = F'\ (\mathsf{Y}\ F')$.

В практических целях значимо именно оно.

Комбинатор неподвижной точки: пример 1

Найти такой F, что для любых M и N выполняется $F\ M\ N = F$.

1. Перепишем это свойство как рекурсивное определение F:

Комбинатор неподвижной точки: пример 1

Найти такой F, что для любых M и N выполняется $F\ M\ N = F$.

1. Перепишем это свойство как рекурсивное определение F:

$$F\ M\ N = (\lambda m\ n.\, F)\ M\ N$$

$$F = \lambda m\ n.\, F$$

2. Переведём в вид $F = F'\ F$, где F не встречается в F':

Комбинатор неподвижной точки: пример 1

Найти такой F, что для любых M и N выполняется $F\ M\ N = F$.

1. Перепишем это свойство как рекурсивное определение F:

$$F\ M\ N = (\lambda m\ n.\, F)\ M\ N$$

$$F = \lambda m\ n.\, F$$

2. Переведём в вид $F = F'\ F$, где F не встречается в F':

$$F = (\lambda f\ m\ n.\, f)\ F$$

3. Применим Y-комбинатор:

Комбинатор неподвижной точки: пример 1

Найти такой F, что для любых M и N выполняется $F\ M\ N = F$.

1. Перепишем это свойство как рекурсивное определение F:

$$F\ M\ N = (\lambda m\ n.\, F)\ M\ N$$

$$F = \lambda m\ n.\, F$$

2. Переведём в вид $F = F'\ F$, где F не встречается в F':

$$F = (\lambda f\ m\ n.\, f)\ F$$

3. Применим Y-комбинатор:

$$F = \mathrm{Y}\ (\lambda f\ m\ n.\, f)$$

4. Проверим, выполняется ли свойство:

$$\begin{aligned} F\ M\ N &= \mathrm{Y}\ (\lambda f\ m\ n.\, f)\ M\ N = (\lambda f\ m\ n.\, f)\ (\mathrm{Y}\ (\lambda f\ m\ n.\, f))\ M\ N \\ &= (\lambda f\ m\ n.\, f)\ F\ M\ N = F \end{aligned}$$

Комбинатор неподвижной точки: пример 2 (1)

(Считаем, что умеем сравнивать числа на равенство)

$$\begin{aligned}\text{fib } n = \text{if} \quad & (\text{equal } n\ 0) \\ & 0 \\ & (\text{if } (\text{equal } n\ 1)\ 1\ (\text{plus } (\text{fib } (\text{minus } n\ 2))\ (\text{fib } (\text{minus } n\ 1))))\end{aligned}$$

1. Перепишем это свойство как рекурсивное определение fib:

Комбинатор неподвижной точки: пример 2 (1)

(Считаем, что умеем сравнивать числа на равенство)

$$\begin{aligned}\text{fib } n = \text{if} \quad & (\text{equal } n\ 0) \\ & 0 \\ & (\text{if } (\text{equal } n\ 1)\ 1\ (\text{plus } (\text{fib } (\text{minus } n\ 2))\ (\text{fib } (\text{minus } n\ 1))))\end{aligned}$$

1. Перепишем это свойство как рекурсивное определение fib:

$$\begin{aligned}\text{fib} = \lambda n.\, \text{if} \quad & (\text{equal } n\ 0) \\ & 0 \\ & (\text{if } (\text{equal } n\ 1)\ 1\ (\text{plus } (\text{fib } (\text{minus } n\ 2))\ (\text{fib } (\text{minus } n\ 1))))\end{aligned}$$

2. Переведём в вид $\text{fib} = \text{fib}'\ \text{fib}$, где fib не встречается в fib':

Комбинатор неподвижной точки: пример 2 (1)

(Считаем, что умеем сравнивать числа на равенство)

$$\begin{aligned}\text{fib } n = \text{if} \quad & (\text{equal } n\ 0) \\ & 0 \\ & (\text{if } (\text{equal } n\ 1)\ 1\ (\text{plus } (\text{fib } (\text{minus } n\ 2))\ (\text{fib } (\text{minus } n\ 1))))\end{aligned}$$

1. Перепишем это свойство как рекурсивное определение fib:

$$\begin{aligned}\text{fib} = \lambda n.\,\text{if} \quad & (\text{equal } n\ 0) \\ & 0 \\ & (\text{if } (\text{equal } n\ 1)\ 1\ (\text{plus } (\text{fib } (\text{minus } n\ 2))\ (\text{fib } (\text{minus } n\ 1))))\end{aligned}$$

2. Переведём в вид $\text{fib} = \text{fib}'\ \text{fib}$, где fib не встречается в fib':

$$\begin{aligned}\text{fib} = (\lambda f \quad & n.\,\text{if } (\text{equal } n\ 0) \\ & 0 \\ & (\text{if } (\text{equal } n\ 1)\ 1\ (\text{plus } (f\ (\text{minus } n\ 2))\ (f\ (\text{minus } n\ 1))))) \\ & \text{fib}\end{aligned}$$

Комбинатор неподвижной точки: пример 2 (2)

3. Применим Y-комбинатор:

Комбинатор неподвижной точки: пример 2 (2)

3. Применим Y-комбинатор:

$$\begin{aligned}\text{fib} = \text{Y}\quad & (\lambda f\ n.\,\text{if}\ (\text{equal}\ n\ 0)\\ & 0\\ & (\text{if}\ (\text{equal}\ n\ 1)\ 1\ (\text{plus}\ (f\ (\text{minus}\ n\ 2))\ (f\ (\text{minus}\ n\ 1)))))\end{aligned}$$

4. Проверим, выполняется ли свойство (сами).

1. Примитивная рекурсия
2. Комбинатор неподвижной точки
3. Кобирование в лямбда-исчислении
   - Списки
   - Типы-суммы

Списки

Мы уже знаем натуральные числа:

$$\mathsf{k} = \lambda s\ z.\ \underbrace{s\ (s\ (\cdots (s}_{k}\ (z)\ \underbrace{\cdots))}_{k}$$

Обобщим их таким образом:

$$\lambda s\ z.\ \underbrace{s\ a_0\ (s\ a_1\ (\cdots (s\ a_{k-1}}_{k}\ (z)\ \underbrace{\cdots))}_{k}$$

Каждая такая конструкция кодирует список $[a_0, a_1, \cdots, a_{k-1}]$.

Конструирование списков

$$\lambda s\ z.\, s\ a_0\ (s\ a_1\ (\cdots (s\ a_{k-1}\ (z)\cdots))$$

Как мы строили натуральные числа:

$$\begin{aligned} \mathbf{0} &:= \lambda s\ z.\, z \\ \mathbf{suc} &:= \lambda\ n\ s\ z.\, s\ (n\ s\ z) \end{aligned}$$

Списки можно строить по аналогии.

$$\mathbf{nil} := $$

Конструирование списков

$$\lambda s\ z.\, s\ a_0\ (s\ a_1\ (\cdots (s\ a_{k-1}\ (z)\cdots))$$

Как мы строили натуральные числа:

$$\begin{aligned} \mathbf{0} &:= \lambda s\ z.\, z \\ \mathbf{suc} &:= \lambda\ n\ s\ z.\, s\ (n\ s\ z) \end{aligned}$$

Списки можно строить по аналогии.

$$\begin{aligned} \mathbf{nil} &:= \mathbf{0} \\ \mathbf{cons} &:= \lambda h\ t\ s\ z. \end{aligned}$$

Конструирование списков

$$\lambda s\ z.\, s\ a_0\ (s\ a_1\ (\cdots (s\ a_{k-1}\ (z)\cdots))$$

Как мы строили натуральные числа:

$$\begin{aligned} \mathbf{0} &:= \lambda s\ z.\, z \\ \mathbf{suc} &:= \lambda\ n\ s\ z.\, s\ (n\ s\ z) \end{aligned}$$

Списки можно строить по аналогии.

$$\begin{aligned} \mathbf{nil} &:= \mathbf{0} \\ \mathbf{cons} &:= \lambda h\ t\ s\ z.\, s\ h\ (t\ s\ z) \end{aligned}$$

Пример списка:

$$[1, 3, 4] := \mathbf{cons}\ 1\ (\mathbf{cons}\ 3\ (\mathbf{cons}\ 4\ \mathbf{nil}))$$

Проверка на пустоту списка

$$\lambda s\ z.\, s\ a_0\ (s\ a_1\ (\cdots (s\ a_{k-1}\ (z)\cdots))$$

На натуральных числах задана операция

$$\textbf{isZero} = \lambda n.\, n\ (\lambda z'.\, \mathsf{false})\ \mathsf{true}$$

Аналогично, на списках можно определить

$$\textbf{isEmpty} = \lambda l.$$

Проверка на пустоту списка

$$\lambda s\ z.\, s\ a_0\ (s\ a_1\ (\cdots (s\ a_{k-1}\ (z)\cdots))$$

На натуральных числах задана операция

$$\textbf{isZero} = \lambda n.\, n\ (\lambda z'.\, \text{false})\ \text{true}$$

Аналогично, на списках можно определить

$$\textbf{isEmpty} = \lambda l.\, l\ (\lambda e\ z'.\, \text{false})\ \text{true}$$

Сложение списков?

Возьмём операцию

$$\textbf{plus} := \lambda n\ m\ s\ z.\ n\ s\ (m\ s\ z)$$

Что будет, если подать ей два списка?

Сложение списков?

Возьмём операцию

$$\textbf{plus} := \lambda n\ m\ s\ z.\ n\ s\ (m\ s\ z)$$

Что будет, если подать ей два списка?Они сконкатенируются.

$$\begin{aligned}
\textbf{plus}\ [1,2]\ [3,4] &:= (\lambda n\ m\ s\ z.\ n\ s\ (m\ s\ z))\ [1,2]\ [3,4] \\
&= \lambda s\ z.\ [1,2]\ s\ ([3,4]\ s\ z) = \lambda s\ z.\ [1,2]\ s\ (s\ 3\ (s\ 4\ z)) \\
&= \lambda s\ z.\ s\ 1\ (s\ 2\ (s\ 3\ (s\ 4\ z))) \\
&= [1,2,3,4]
\end{aligned}$$

Взятие головы списка

Хотим операцию **headOrDefault** с таким поведением:

$$\begin{aligned} \textbf{headOrDefault}\ d\ (\textbf{cons}\ a\ t) &= a \\ \textbf{headOrDefault}\ d\ \textbf{nil} &= d \end{aligned}$$

Взятие головы списка

Хотим операцию **headOrDefault** с таким поведением:

$$\begin{aligned} \textbf{headOrDefault}\ d\ (\textbf{cons}\ a\ t) &= a \\ \textbf{headOrDefault}\ d\ \textbf{nil} &= d \end{aligned}$$

$$\textbf{headOrDefault} = \lambda l.\, l\ \mathsf{K} = \textbf{fst}$$

1. Примитивная рекурсия
2. Комбинатор неподвижной точки
3. Кобирование в лямбда-исчислении
   - Списки
   - Типы-суммы

Полиморфизм

Хотим написать компьютерную игру, в которой есть два вида сущностей: мирные жители и злые монстры. Мирные жители выглядят иначе, чем монстры, но хотим, чтобы отрисовщик игры был реализован без знания о том, что рисует:

```
def drawAll(allObjects):
  for obj in allObjects:
    draw(obj)
```

Таким образом, функция должна вызываться одна и та же, но выполнять совсем разный код.

Типы-суммы, первая попытка

Введём термы

$$\begin{aligned}
\textbf{inl} &:= \lambda a.\,\text{pair true } a \\
\textbf{inr} &:= \lambda a.\,\text{pair false } a \\
\textbf{either} &:= \lambda f\ g\ e.\,\text{if (fst } e\text{) (}f\text{ (snd } e\text{)) (}g\text{ (snd } e\text{))}
\end{aligned}$$

Они обладают свойствами

$$\begin{aligned}
\textbf{either}\ f\ g\ (\textbf{inl}\ a) &= f\ a \\
\textbf{either}\ f\ g\ (\textbf{inr}\ a) &= g\ a
\end{aligned}$$

Типы-суммы, первая попытка (2)

$$\begin{aligned}
\textbf{inl} &:= \lambda a.\, \text{pair true } a \\
\textbf{inr} &:= \lambda a.\, \text{pair false } a \\
\textbf{either} &:= \lambda f\ g\ e.\, \text{if (fst } e) \text{ (} f \text{ (snd } e)) \text{ (} g \text{ (snd } e))
\end{aligned}$$

Упростим **either**:
$\lambda f\ g\ e.\, \text{if (fst } e) \text{ (} f \text{ (snd } e)) \text{ (} g \text{ (snd } e)) = \lambda f\ g\ e.\, \text{if (fst } e)\ f\ g\ \text{(snd } e) =$
$\lambda f\ g\ e.\, \text{(fst } e)\ f\ g\ \text{(snd } e) = \lambda f\ g\ e.\, e\ (\lambda b\ a.\, b\ f\ g\ a) = \lambda f\ g\ e.\, e\ (\lambda b.\, b\ f\ g)$
Заметим, что тут лишний код: можно было бы вместо $(\lambda b.\, b\ f\ g)$ подавать в e сами f и g, потому что это единственная информация, которая e недоступна.

Типы-суммы, вторая попытка

$$\begin{aligned} \textbf{inl} &:= \lambda a\ f\ g.\, f\ a \\ \textbf{inr} &:= \lambda a\ f\ g.\, g\ a \\ \textbf{either} &:= \lambda f\ g\ e.\, e\ f\ g \end{aligned}$$

Свойства по-прежнему выполняются.