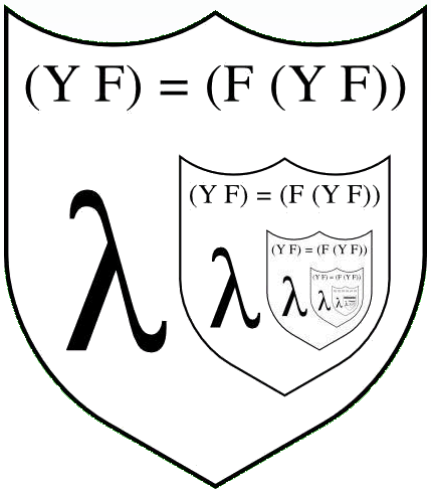
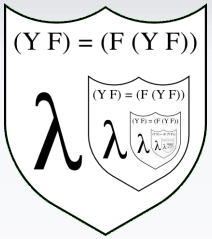


Соответствие Карри–Ховарда. Лекция вторая

А. Н. Непейвода
ИПС им. А.К. Айламазяна РАН,
МГТУ им. Н.Э. Баумана

MathClub,
25 февраля 2022, Иннополис



Аксиомы комбинаторной логики

Эквивалентность комбинаторов, имеющих нормальную форму, определяется в следующей системе переписывания термов (+ аксиомы для $\mathbf{S}$, $\mathbf{K}$, $\mathbf{I}$):

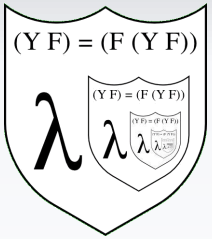
$$\mathbf{S}(\mathbf{KK}) = \mathbf{S}(\mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KK})(\mathbf{S}(\mathbf{KS})\mathbf{K})))(\mathbf{KK})$$

$$\mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KK})) = \mathbf{S}(\mathbf{KK})(\mathbf{S}(\mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KK})\mathbf{I}))(\mathbf{KI}))$$

$$\mathbf{S}(\mathbf{K}(\mathbf{S}(\mathbf{KS})))(\mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KS}))) = \mathbf{S}(\mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KK})(\mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{K}(\mathbf{S}(\mathbf{KS})))\mathbf{S}))))(\mathbf{KS})$$

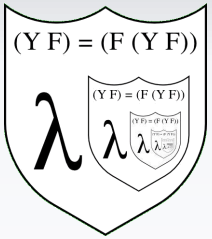
$$\mathbf{I} = \mathbf{S}(\mathbf{S}(\mathbf{KS})\mathbf{K})(\mathbf{KI})$$

Переписывание может применяться с любой стороны, и в процессе применения аксиом упрощаемый терм может удлиняться.



Упражнение

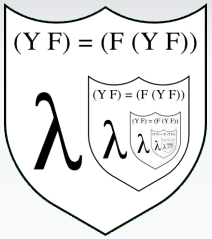
Известно, что композиция $\lambda x\, y\, z.x\ (y\ z)$ — это `fmap` для функтора `Reader`. Выразить её в бесточечном виде по алгоритму и по логике.



Упражнение

Известно, что композиция $\lambda x\, y\, z.x\ (y\ z)$ — это `fmap` для функтора `Reader`. Выразить её в бесточечном виде по алгоритму и по логике.

```
newtype MyRead r a = Rd run :: r -> a
instance Applicative (MyRead r) where
  pure v = Rd (\_ -> v)
  Rd f <*> Rd g = Rd (\x -> f x (g x))
```

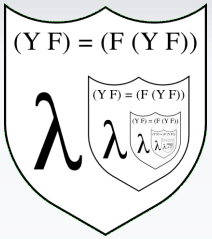


Упражнение

Известно, что композиция $\lambda x\, y\, z.x\ (y\ z)$ — это `fmap` для функтора `Reader`. Выразить её в бесточечном виде по алгоритму и по логике.

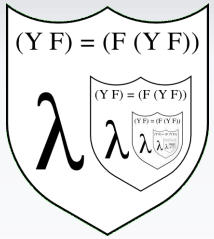
```
newtype MyRead r a = Rd run :: r -> a
instance Applicative (MyRead r) where
  pure v = Rd (\_ -> v)
  Rd f <*> Rd g = Rd (\x -> f x (g x))
```

```
fmap f (Rd x)
    = (Rd ((pure (<*>)) <*> pure) f x)
```



Вычисления «от противного»

Проблема термов в типизированном λ-исчислении (и чистого функционального стиля): «что упало, то пропало». Если вычислено $(\mathrm{M}\ \mathrm{N})$, то M и N по отдельности уже потеряны навсегда.



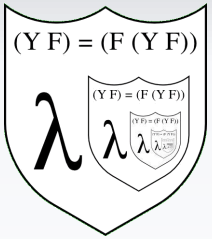
Вычисления «от противного»

Проблема термов в типизированном λ-исчислении (и чистого функционального стиля): «что упало, то пропало». Если вычислено $(M\ N)$, то M и N по отдельности уже потеряны навсегда.

Логичный выход — возвраты, если стало ясно, что вычисления зашли в тупик. Аналог в доказательствах — работа с отрицанием.

$$\frac{\Gamma \vdash \tau,\ \Gamma \vdash \neg\tau}{\Gamma \vdash \sigma}$$ (из лжи следует всё что угодно)

$$\frac{\Gamma \vdash \neg\neg\tau}{\Gamma \vdash \tau}$$ (снятие двойного отрицания)



Теорема Гливенко

Пусть Φ — пропозициональная формула. Тогда $\vdash \Phi$ в классической логике $\Leftrightarrow \vdash \neg\neg\Phi$ в конструктивной интуиционистской логике.

Здесь под конструктивной интуиционистской логикой понимаем минимальную логику + «из лжи следует всё что угодно».

Теорема Гливенко

Пусть Φ — пропозициональная формула. Тогда $\vdash \Phi$ в классической логике $\Leftrightarrow \vdash \neg\neg\Phi$ в конструктивной интуиционистской логике.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{A}
}{\neg\neg A \Rightarrow A}
\quad
\dfrac{}{\neg A}
}{\bot}
\quad
\dfrac{
\dfrac{}{A}
}{\neg\neg A \Rightarrow A}
}{\bot}
}{\neg\neg(\neg\neg A \Rightarrow A)}
$$

Из $\neg\neg A$ и $\neg A$

Теорема Гливенко

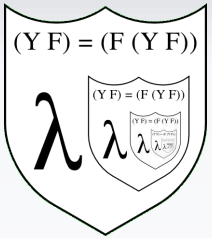
Пусть Φ — пропозициональная формула. Тогда $\vdash \Phi$ в классической логике $\Leftrightarrow \vdash \neg\neg\Phi$ в конструктивной интуиционистской логике.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{\neg\neg A \Rightarrow A}
}{\bot}
}{\neg A}
\quad
\dfrac{
\dfrac{}{A}
}{\neg\neg A \Rightarrow A}
}{\bot}
}{\neg\neg(\neg\neg A \Rightarrow A)}
}{\neg\neg(\neg\neg A \Rightarrow A)}
$$

Из $\neg\neg A$ и $\neg A$

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{(A \Rightarrow B) \Rightarrow A}
}{\bot}
}{\neg A}
\quad
\dfrac{
\dfrac{
\dfrac{}{A}
}{A \Rightarrow B}
}{(A \Rightarrow B) \Rightarrow A}
}{\bot}
}{\neg\neg((A \Rightarrow B) \Rightarrow A) \Rightarrow A}
}{\neg\neg((A \Rightarrow B) \Rightarrow A) \Rightarrow A}
$$

из A и $\neg A$



Переход к относительному отрицанию

«Вычисления зашли в тупик» $\Rightarrow$ приводят к нежелательному результату. Интерпретация $\neg_{\mathrm{R}}\Phi = (\Phi \Rightarrow \mathrm{R})$.
Рассмотрим предыдущие два вывода с точки зрения извлечения термов из их конструкции.

Переход к относительному отрицанию

«Вычисления зашли в тупик» $\Rightarrow$ приводят к нежелательному результату. Интерпретация $\neg_R \Phi = (\Phi \Rightarrow R)$.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{A}\ (\text{терм } y)
}{\neg_R \neg_R A \Rightarrow A}\ (\text{терм } \lambda z.y)
}{R}\ (\text{терм } x\ (\lambda z.y))
}{A \Rightarrow R}\ (\text{терм } \lambda y.x\ (\lambda z.y))
\qquad
\dfrac{
\dfrac{}{A}\ (\text{??!!! из } A \Rightarrow R \text{ и } (A \Rightarrow R) \Rightarrow R)
}{\neg_R \neg_R A \Rightarrow A}
\qquad
R
}{((\neg_R \neg_R A \Rightarrow A) \Rightarrow R) \Rightarrow R}
$$

(тип x) (тип y) (тип z) (тип w)

Видно, что возникает проблема с правилом «из лжи следует всё что угодно», потому что относительное отрицание даёт вывести из противоречия лишь конкретную формулу R.

Переход к относительному отрицанию

«Вычисления зашли в тупик» $\Rightarrow$ приводят к нежелательному результату. Интерпретация $\neg_R \Phi = (\Phi \Rightarrow R)$.

$*(((A \Rightarrow B) \Rightarrow A) \Rightarrow A) \Rightarrow R$ (тип x)

- $*A$ (тип y)
  - $*(A \Rightarrow B) \Rightarrow A$ (тип z)
    - A (терм y)

    $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ (терм $\lambda z.y$)

    R (терм $x\ (\lambda z.y)$)

  $A \Rightarrow R$ (терм $\lambda y.x\ (\lambda z.y)$)
  - $*(A \Rightarrow B) \Rightarrow A$ (тип u)
    - $*A$ (тип w)
      - B ??!!!

      $A \Rightarrow B$

      A

    $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$

    R

$((((A \Rightarrow B) \Rightarrow A) \Rightarrow A) \Rightarrow R) \Rightarrow R$

Переход к относительному отрицанию

«Вычисления зашли в тупик» $\Rightarrow$ приводят к нежелательному результату. Интерпретация $\neg_R \Phi = (\Phi \Rightarrow R)$.

$*(((A \Rightarrow B) \Rightarrow A) \Rightarrow A) \Rightarrow B$ (тип x)

| $*A$ (тип y)

| | $*(A \Rightarrow B) \Rightarrow A$ (тип z)

| | | A (терм y)

| | $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ (терм $\lambda z.y$)

| | B (терм $x\ (\lambda z.y)$)

| $A \Rightarrow B$ (терм $\lambda y.x\ (\lambda z.y)$)

| $*(A \Rightarrow B) \Rightarrow A$ (тип u)

| | $*A$ (тип w)

| | | B (терм $(\lambda y.x\ (\lambda z.y))\ w$)

| | $A \Rightarrow B$ (терм $\lambda y.x\ (\lambda z.y)$)

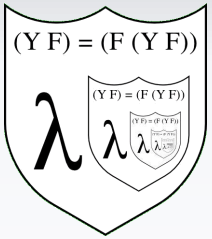
| | A (терм $u\ (\lambda y.x\ (\lambda z.y))$)

| $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ (терм $\lambda u.u\ (\lambda y.x\ (\lambda z.y))$)

| B (терм $x\ (\lambda u.u\ (\lambda y.x\ (\lambda z.y)))$)

$(((((A \Rightarrow B) \Rightarrow A) \Rightarrow A) \Rightarrow B) \Rightarrow B$

(терм $(\lambda x.x\ (\lambda u.u\ (\lambda y.x\ (\lambda z.y))))$)

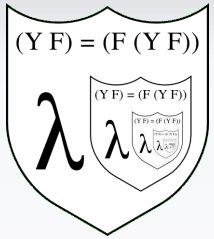


Навешивание двойного отрицания

Видно, что использование «возвратных» термов (переход от типа Φ к типу $\neg_R \neg_R \Phi$) расширяет возможности языка. Возникает вопрос, в каких подформулах лучше это делать?

- Преобразование Колмогорова:
  $\tau(A) = \neg_R \neg_R A \quad \tau(\Phi \Rightarrow \Psi) = \neg_R \neg_R (\tau(\Phi) \Rightarrow \tau(\Psi))$
- Вариант более слабого преобразования (в стиле Куроды) — это $\neg_R \neg_R \tau'(\Phi)$, где:
  $\tau'(A) = A \quad \tau'(\Phi \Rightarrow \Psi) = \tau'(\Phi) \Rightarrow \neg_R \neg_R \tau'(\Psi))$

В примерах для краткости $(\Phi \Rightarrow R) \Rightarrow R$ переобозначим как Φ'.



Переход по Колмогорову

$*(A' \Rightarrow ((A' \Rightarrow B')' \Rightarrow B')') \Rightarrow R$ (тип k_0)

$*(A \Rightarrow R) \Rightarrow R$ (тип x)

$*((A' \Rightarrow B')' \Rightarrow B') \Rightarrow R$ (тип k_1)

$*((A' \Rightarrow B') \Rightarrow R) \Rightarrow R$ (тип y)

$*B \Rightarrow R$ (тип k_2)

$*A' \Rightarrow B'$ (тип k_3)

$*A \Rightarrow R$ (тип k_4)

R (терм $x\ k_4$)

$(A \Rightarrow R) \Rightarrow R$ (терм $\lambda k_4.x\ k_4$)

$(B \Rightarrow R) \Rightarrow R$ (терм $k_3\ (\lambda k_4.x\ k_4)$)

R (терм $(k_3\ (\lambda k_4.x\ k_4))\ k_2$)

$(A' \Rightarrow B') \Rightarrow R$ (терм $\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2$)

R (терм $y\ (\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2)$)

$(B \Rightarrow R) \Rightarrow R$ (терм $\lambda k_2.y\ (\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2)$)

$(A' \Rightarrow B')' \Rightarrow B'$ (терм $\lambda y k_2.y\ (\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2)$)

R (терм $k_1\ (\lambda y k_2.y\ (\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2))$)

$(((A' \Rightarrow B')' \Rightarrow B') \Rightarrow R) \Rightarrow R$ (терм $\lambda k_1.k_1\ (\lambda y k_2.y\ (\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2))$)

$A' \Rightarrow ((A' \Rightarrow B')' \Rightarrow B')'$ (терм $\lambda x k_1.k_1\ (\lambda y k_2.y\ (\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2))$)

R (терм $k_0\ (\lambda x k_1.k_1\ (\lambda y k_2.y\ (\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2)))$)

$((A' \Rightarrow ((A' \Rightarrow B')' \Rightarrow B')') \Rightarrow R) \Rightarrow R$

Извлечённый терм: $\lambda k_0.k_0\ (\lambda x k_1.k_1\ (\lambda y k_2.y\ (\lambda k_3.k_3\ (\lambda k_4.x\ k_4)\ k_2)))$

Слабый переход по Куроде

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
*(A \Rightarrow ((A \Rightarrow B') \Rightarrow B') \Rightarrow R \qquad (\text{тип } k_0)
}{
*A \qquad (\text{тип } x)
}
}{
*((A \Rightarrow B') \Rightarrow B') \Rightarrow R \qquad (\text{тип } k_1)
}
}{
*A \Rightarrow B' \qquad (\text{тип } y)
}
}{
*B \Rightarrow R \qquad (\text{тип } k_2)
}
}{
(B \Rightarrow R) \Rightarrow R \qquad (\text{тип } y\ x)
}
}{
R \qquad (\text{тип } y\ x\ k_2)
}
}{
(B \Rightarrow R) \Rightarrow R \qquad (\text{тип } \lambda k_2.y\ x\ k_2)
}
}{
(A \Rightarrow B') \Rightarrow B' \qquad (\text{тип } \lambda y k_2.y\ x\ k_2)
}
}{
R \qquad (\text{тип } k_1\ (\lambda y k_2.y\ x\ k_2))
}
}{
(((A \Rightarrow B') \Rightarrow B') \Rightarrow R) \Rightarrow R \qquad (\text{тип } \lambda k_1.k_1\ (\lambda k_2.y\ x\ k_2))
}
}{
A \Rightarrow ((A \Rightarrow B') \Rightarrow B')' \qquad (\text{тип } \lambda x k_1.k_1\ (\lambda k_2.y\ x\ k_2))
}
}{
R \qquad (\text{тип } k_0\ (\lambda x k_1.k_1\ (\lambda k_2.y\ x\ k_2)))
}
}{
((A \Rightarrow ((A \Rightarrow B') \Rightarrow B')') \Rightarrow R) \Rightarrow R
}
$$

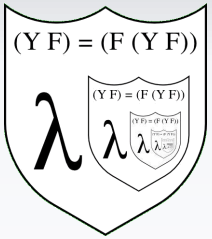
Извлечённый терм: $\lambda k_0.k_0\ (\lambda x k_1.k_1\ (\lambda y k_2.y\ x\ k_2))$.

Детали преобразования

- Переменные x и y получают имена так: смотрим на исходный терм $\lambda xy.y\ x$. В нём тип x — это просто A, тип y — это $A \Rightarrow B$. Теперь преобразуем их типы по Колмогорову или Куроде как подформулы. Образы этих типов и породят термы x и y.
- В выводе терма по Куроде есть странный подвывод, в котором выводится $(B \Rightarrow R) \Rightarrow R$, и так выводимая без него:

$$
\dfrac{
\begin{array}{l|ll}
*A \Rightarrow B' & & (\text{тип } y) \\
\quad *B \Rightarrow R & & (\text{тип } k_2) \\
\quad\quad (B \Rightarrow R) \Rightarrow R & (\text{тип } y\ x) & \\
\quad\quad R & (\text{тип } y\ x\ k_2) & \\
\hline
\quad (B \Rightarrow R) \Rightarrow R & (\text{тип } \lambda k_2.y\ x\ k_2) &
\end{array}
}{(A \Rightarrow B') \Rightarrow B' \qquad (\text{тип } \lambda y k_2.y\ x\ k_2)}
$$

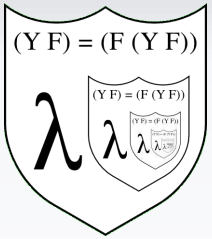
Однако если выводить формулу без подвывода с допущением $B \Rightarrow R$ напрямую, то вместо $A \Rightarrow B'$ можно просто подставить формулу $A \Rightarrow B$, и мы получим вывод не требуемой формулы, а более общей: $(A \Rightarrow ((A \Rightarrow B) \Rightarrow B)')'$, а мы хотим доказать именно ту формулу, которую получили преобразованием.



Система передачи продолжений

Система передачи продолжений в CBN-стиле:

- $\tau_N(const) = \lambda k.k\ const$
- $\tau_N(x) = \lambda k.x\ k$
- $\tau_N(\lambda x.M) = \lambda k.k\ (\lambda x.\tau_N(M))$
- $\tau_N(M\ N) = \lambda k.\tau_N(M)\ (\lambda f.f\ \tau_N(N)\ k)$



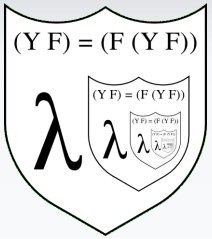
Система передачи продолжений

Система передачи продолжений в CBN-стиле:

- $\tau_N(const) = \lambda k.k\ const$
- $\tau_N(x) = \lambda k.x\ k$
- $\tau_N(\lambda x.M) = \lambda k.k\ (\lambda x.\tau_N(M))$
- $\tau_N(M\ N) = \lambda k.\tau_N(M)\ (\lambda f.f\ \tau_N(N)\ k)$

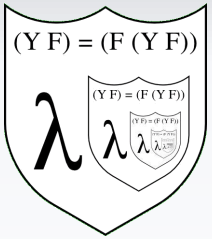
Система передачи продолжений в CBV-стиле:

- $\tau_V(const) = \lambda k.k\ const$
- $\tau_V(x) = \lambda k.k\ x$
- $\tau_V(\lambda x.M) = \lambda k.k\ (\lambda x.\tau_V(M))$
- $\tau_V(M\ N) = \lambda k.\tau_V(M)\ (\lambda f.\tau_V(N)\ (\lambda a.f\ a\ k))$



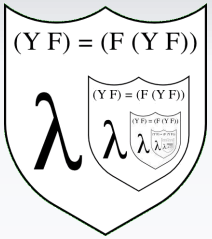
Применение CPS-преобразования

- $\mathrm{M} \to^* \mathrm{const}$ при CBV-стратегии $\Leftrightarrow$
  $\tau_{\mathrm{V}}(\mathrm{M})\ \mathrm{id} \to^* \mathrm{const}$ при какой угодно стратегии.
- $\mathrm{M} \to^* \mathrm{const}$ при CBN-стратегии $\Leftrightarrow$
  $\tau_{\mathrm{N}}(\mathrm{M})\ \mathrm{id} \to^* \mathrm{const}$ при какой угодно стратегии.



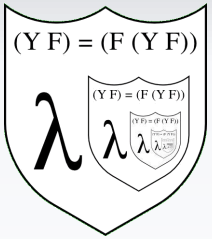
Монада Cont

```
newtype MyCont r a = Mc { runCont :: (a -> r) -> r
}
instance Functor (MyCont r) where
  fmap f (Mc cps_a) = ...
instance Applicative (MyCont r) where
  pure v = ...
  Mc cps_f <*> Mc cps_a = ...
```



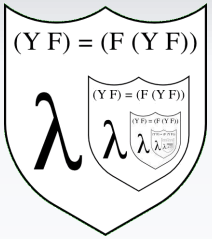
Монада Cont

```
newtype MyCont r a = Mc { runCont :: (a -> r) -> r
}
instance Functor (MyCont r) where
  fmap f (Mc cps_a) =  Mc (\cps -> cps_a (cps . f))
instance Applicative (MyCont r) where
  pure v = ...
  Mc cps_f <*> Mc cps_a = ...
```



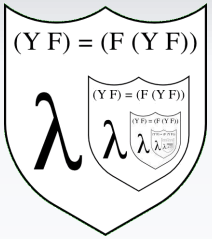
Монада Cont

```
newtype MyCont r a = Mc { runCont :: (a -> r) -> r
}
instance Functor (MyCont r) where
  fmap f (Mc cps_a) =  Mc (\cps -> cps_a (cps . f))
instance Applicative (MyCont r) where
  pure v = Mc (\r -> r v)
  Mc cps_f <*> Mc cps_a = ...
```



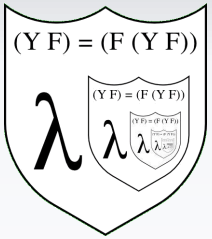
Монада Cont

```
newtype MyCont r a = Mc { runCont :: (a -> r) -> r
}
instance Functor (MyCont r) where
  fmap f (Mc cps_a) =  Mc (\cps -> cps_a (cps . f))
instance Applicative (MyCont r) where
  pure v = Mc (\r -> r v)
  Mc cps_f <*> Mc cps_a =
      Mc (\cps_b -> cps_f (\f -> cps_a (cps_b . f)))
```



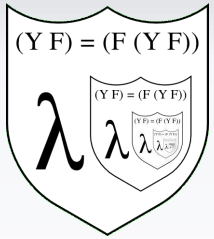
Упражнение

Пользуясь соответствием Карри–Ховарда, построить явным образом терм, реализующий bind-оператор >>= для монады Cont.
Тип >>= :: MyCont a -> (a -> MyCont b) -> MyCont b, и это всё, что нужно знать для решения задачи.



Прерывания вычислений

- Жёсткое прерывание — полный выход из контекста ($\neg_R \neg_R A \Rightarrow A$);
- Мягкое прерывание — возвращает текущий контекст вычислений ($(\neg_R A \Rightarrow A) \Rightarrow A$) — закон Пирса, а также тип оператора call-with-current-continuation.



Вывод = конструкция

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{*((((A \Rightarrow B) \Rightarrow B) \Rightarrow A) \Rightarrow B)}}{*B \Rightarrow A}}{*(A \Rightarrow B) \Rightarrow B}}{(B \Rightarrow A) \Rightarrow B} \qquad \dfrac{\dfrac{\dfrac{}{*A \Rightarrow B}}{(A \Rightarrow B) \Rightarrow A}}{((A \Rightarrow B) \Rightarrow B) \Rightarrow A}}{B}
$$

Закон Пирса!

$$
\dfrac{B \qquad \text{Закон Пирса!}}{((((A \Rightarrow B) \Rightarrow B) \Rightarrow A) \Rightarrow B) \Rightarrow B}
$$

Вывод = конструкция

$*((((A \Rightarrow B) \Rightarrow B) \Rightarrow A) \Rightarrow B)$ (тип терма x)

$*B \Rightarrow A$ (тип терма y)

$*(A \Rightarrow B) \Rightarrow B$ (тип терма z)

$*A \Rightarrow B$ (тип терма w)

B (терм $z\ w$)

A (терм $y\ (z\ w)$)

$(A \Rightarrow B) \Rightarrow A$ (терм $\lambda w.y\ (z\ w)$)

A ($T_1 = callcc\ \lambda w.y\ (z\ w)$)

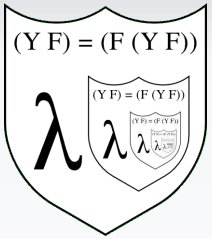
$((A \Rightarrow B) \Rightarrow B) \Rightarrow A$ (терм $\lambda z.T_1$)

B (терм $x\ (\lambda z.T_1)$)

$(B \Rightarrow A) \Rightarrow B$ (терм $\lambda y.x\ (\lambda z.T_1)$)

B ($T_2 = callcc\ \lambda y.x\ (\lambda z.T_1)$)

$((((A \Rightarrow B) \Rightarrow B) \Rightarrow A) \Rightarrow B) \Rightarrow B$ терм $\lambda x.T_2$

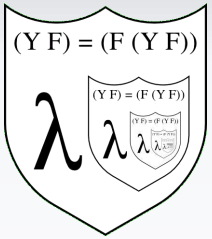


Определение монады

```
class Applicative M => Monad M where
  return :: a -> M a
  (>>=) :: M a -> (a -> M b) -> M b
```

Дополнительные монадические операторы:

```
join :: M M a -> M a
fmap :: (a -> b) -> M a -> M b
```



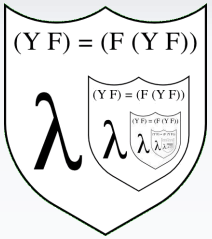
Соответствие Карри–Ховарда

Propositional Lax Logic — монадическая логика с единственным модальным оператором $\bigcirc$.

Правила вывода (+ стандартные правила естественного вывода в интуиционистской логике):

- $\dfrac{A}{\bigcirc A}$ (Return)
- $\dfrac{A \Rightarrow \bigcirc B}{\bigcirc A \Rightarrow \bigcirc B}$ (Bind)

Сохраняет возможность извлечь монадический оператор из вывода типов! Обратим внимание на инвертированный порядок аргументов.



Пример разбора задачи на монадическое соответствие

Построим оператор `join` из операторов `>>=` и `return`.

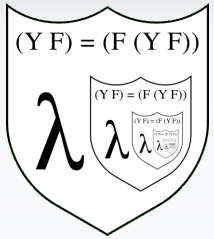
$*\bigcirc\bigcirc A$ (тип терма x)

(Тут нужно придумать, как объединить контейнеры, оборачивающие x)

$\bigcirc A$

$\bigcirc\bigcirc A \Rightarrow \bigcirc A$ (тип терма $\lambda x. \ldots$)

Единственный оператор, умеющий заглядывать внутрь контейнера — это `bind`. Он требует два аргумента: контейнерный типа `M a` и монадическую функцию типа `a -> M b`. Поскольку нужно заглянуть внутрь только одной из двух контейнерных оболочек, в роли типа `a` выступает также монада `M a'`. Значит, нам нужна стрелка типа `M a' -> M a'`. Это функция `id`, вывод которой мы уже умеем строить.



Пример разбора задачи на монадическое соответствие

$*\bigcirc\bigcirc A$ (тип терма x)

$*\bigcirc A$ (тип терма y)

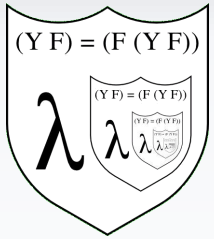
$\bigcirc A$ (просто возвращаем сам y)

$\bigcirc A \Rightarrow \bigcirc A$ (тип терма $\lambda y.y$)

(Аргументы `bind`-оператора построены, осталось применить их в правильном порядке)

$\bigcirc A$

$\bigcirc\bigcirc A \Rightarrow \bigcirc A$ (тип терма $\lambda x. \ldots$)



Пример разбора задачи на монадическое соответствие

$*\bigcirc\bigcirc A$ (тип терма x)

$*\bigcirc A$ (тип терма y)

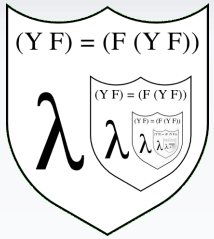
$\bigcirc A$ (просто возвращаем сам y)

$\bigcirc A \Rightarrow \bigcirc A$ (тип терма $\lambda y.y$)

(Аргументы `bind`-оператора построены, осталось применить их в правильном порядке)

$\bigcirc A$ (тип терма $x>>=(\lambda y.y)$)

$\bigcirc\bigcirc A \Rightarrow \bigcirc A$ (тип терма $\lambda x.(x>>=(\lambda y.y))$)



Пример разбора задачи на монадическое соответствие

$*\ \bigcirc\ \bigcirc A$ (тип терма x)

$*\ \bigcirc\ A$ (тип терма y)

$\bigcirc A$ (просто возвращаем сам y)

$\bigcirc A \Rightarrow \bigcirc A$ (тип терма $\lambda y.y$)

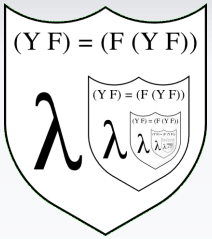
(Аргументы `bind`-оператора построены, осталось применить их в правильном порядке)

$\bigcirc A$ (тип терма $x{>>=}(\lambda y.y)$)

$\bigcirc\ \bigcirc\ A \Rightarrow \bigcirc A$ (тип терма $\lambda x.(x{>>=}(\lambda y.y))$)

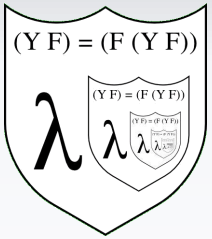
С помощью построения вывода в модальной логике мы просто решили уравнение в ФВП, получив ответ:

```
join x = x >>= id.
```



Задачи

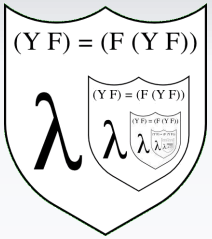
- Построить вывод $(A \Rightarrow B) \Rightarrow \bigcirc A \Rightarrow \bigcirc B$.
- Построить вывод $\bigcirc(A \Rightarrow B) \Rightarrow \bigcirc A \Rightarrow \bigcirc B$.



Задачи

- Построить вывод $(A \Rightarrow B) \Rightarrow \bigcirc A \Rightarrow \bigcirc B$.
- Построить вывод $\bigcirc(A \Rightarrow B) \Rightarrow \bigcirc A \Rightarrow \bigcirc B$.

liftM — аналог fmap для монад, стандартно определяется через bind и return. Как?
А как определить через них <*>?



Задачи

- Построить вывод $(A \Rightarrow B) \Rightarrow \bigcirc A \Rightarrow \bigcirc B$.
- Построить вывод $\bigcirc(A \Rightarrow B) \Rightarrow \bigcirc A \Rightarrow \bigcirc B$.

liftM — аналог fmap для монад, стандартно определяется через bind и return. Как?

```
liftM f m = m >>= \i -> return (f i)
```

А как определить через них <*>?

```
f <*> a
    = f >>= (\xf -> a >>= (\xa -> return (xf xa)))
```

Всё!

Спасибо за внимание!