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 S, $\mathbf{K}$, $\mathbf{I}$):

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

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

Упражнение

Известно, что композиция $\lambda xyz.x\,(y\,z)$ — 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
-- Это синоним комбинатора K.
  pure v = Rd (\_ -> v)
-- А это инфиксный синоним комбинатора S.
  Rd f <*> Rd g = Rd (\x -> f x (g x))
```

Упражнение

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

Посмотрим на структуру терма $\lambda x\,y\,z.x\;(y\;z)$. Он задействует все переменные, поэтому внешним комбинатором должен быть $\mathbf{S}$, а чтобы не появилось нежелательной композиции $(x\;y)$, у этого комбинатора должно быть два аргумента. Пробуем:
$\mathbf{S}\,M_1\,M_2\,x = M_1\,x\,(M_2\,x)$.
Здесь одно из вхождений x лишнее. Предположим, что первое, тогда $M_1 = \mathbf{K}\,M_3$, и результат: $M_3\,(M_2\,x)$.
Рассмотрим $M_3 = \mathbf{S}$, получаем $M_2\,x\,z\,(y\,z)$. Очевидно, $M_2 = \mathbf{K}$, и итоговый терм в комбинаторах $\mathbf{S}(\mathbf{KS})\mathbf{K}$. Ниже — его представление в языке аппликативных операторов.

Упражнение

Известно, что композиция $\lambda xyz.x\,(y\,z)$ — 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
-- Это синоним комбинатора K.
  pure v = Rd (\_ -> v)
-- А это инфиксный синоним комбинатора S.
  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 $(M\ N)$, 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». Потом мы опять вернёмся к минимальной логике, но сначала посмотрим, как всё сложно с извлечением термов из отрицания.

Теорема Гливенко

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

$*\neg(\neg\neg A \Rightarrow A)$

> $*A$
>
> > $*\neg\neg A$
> >
> > > A
> >
> > $\neg\neg A \Rightarrow A$
> >
> > $\bot$
>
> $\neg A$
>
> $*\neg\neg A$
>
> > A Из $\neg\neg A$ и $\neg A$
>
> $\neg\neg A \Rightarrow A$
>
> $\bot$

$\neg\neg(\neg\neg A \Rightarrow A)$

Теорема Гливенко

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

- $*\neg(\neg\neg A \Rightarrow A)$
  - $*A$
    - $*\neg\neg A$
      - A
    - $\neg\neg A \Rightarrow A$
    - $\bot$
  - $\neg A$
  - $*\neg\neg A$
    - A Из $\neg\neg A$ и $\neg A$
  - $\neg\neg A \Rightarrow A$
  - $\bot$
- $\neg\neg(\neg\neg A \Rightarrow A)$

- $*\neg(((A \Rightarrow B) \Rightarrow A) \Rightarrow A)$
  - $*A$
    - $*(A \Rightarrow B) \Rightarrow A$
      - A
    - $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
    - $\bot$
  - $\neg A$
  - $*(A \Rightarrow B) \Rightarrow A$
    - $*A$
      - B из A и $\neg A$
    - $A \Rightarrow B$
    - A
  - $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
  - $\bot$
- $\neg\neg(((A \Rightarrow B) \Rightarrow A) \Rightarrow A)$

Переход к относительному отрицанию

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

Рассмотрим предыдущие два вывода с точки зрения извлечения термов из их конструкции.

Переход к относительному отрицанию

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

$\quad *A$ (тип y)

$\quad\quad * \neg_R \neg_R A$ (тип z)

$\quad\quad\quad A$ (терм y)

$\quad\quad \neg_R \neg_R A \Rightarrow A$ (терм $\lambda z.y$)

$\quad\quad R$ (терм $x\ (\lambda z.y)$)

$\quad A \Rightarrow R$ (терм $\lambda y.x\ (\lambda z.y)$)

$\quad *(A \Rightarrow R) \Rightarrow R$ (тип w)

$\quad\quad A$ (??!!! из $A \Rightarrow R$ и $(A \Rightarrow R) \Rightarrow R$)

$\quad \neg_R \neg_R A \Rightarrow A$

$\quad R$

$((\neg_R \neg_R A \Rightarrow A) \Rightarrow R) \Rightarrow R$

Проблема с правилом «из лжи следует всё что угодно»: относительное отрицание даёт вывести из противоречия лишь одну формулу R. Вывод требует, чтобы $R = A$, но это превращает исходную формулу в тривиальность:

$((((A \Rightarrow A) \Rightarrow A) \Rightarrow A) \Rightarrow A) \Rightarrow A.$

Переход к относительному отрицанию

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

$$
\begin{array}{ll}
*(((A \Rightarrow B) \Rightarrow A) \Rightarrow A) \Rightarrow R & \text{(тип } x\text{)} \\
\quad \mid\ *A & \text{(тип } y\text{)} \\
\quad \mid\ \quad \mid\ *(A \Rightarrow B) \Rightarrow A & \text{(тип } z\text{)} \\
\quad \mid\ \quad \mid\ \quad \mid\ A & \text{(терм } y\text{)} \\
\quad \mid\ \quad \mid\ ((A \Rightarrow B) \Rightarrow A) \Rightarrow A & \text{(терм } \lambda z.y\text{)} \\
\quad \mid\ \quad \mid\ R & \text{(терм } x\ (\lambda z.y)\text{)} \\
\quad \mid\ A \Rightarrow R & \text{(терм } \lambda y.x\ (\lambda z.y)\text{)} \\
\quad \mid\ *(A \Rightarrow B) \Rightarrow A & \text{(тип } u\text{)} \\
\quad \mid\ \quad \mid\ *A & \text{(тип } w\text{)} \\
\quad \mid\ \quad \mid\ \quad \mid\ B & \text{??!!!} \\
\quad \mid\ \quad \mid\ A \Rightarrow B & \\
\quad \mid\ \quad \mid\ A & \\
\quad \mid\ ((A \Rightarrow B) \Rightarrow A) \Rightarrow A & \\
\quad \mid\ R & \\
((((A \Rightarrow B) \Rightarrow A) \Rightarrow A) \Rightarrow R) \Rightarrow R &
\end{array}
$$

Переход к относительному отрицанию

$$
\begin{array}{ll}
*(((A \Rightarrow B) \Rightarrow A) \Rightarrow A) \Rightarrow B & \text{(тип } x\text{)} \\
\quad *A & \text{(тип } y\text{)} \\
\quad\quad *(A \Rightarrow B) \Rightarrow A & \text{(тип } z\text{)} \\
\quad\quad\quad A & \text{(терм } y\text{)} \\
\quad\quad ((A \Rightarrow B) \Rightarrow A) \Rightarrow A & \text{(терм } \lambda z.y\text{)} \\
\quad\quad B & \text{(терм } x\ (\lambda z.y)\text{)} \\
\quad A \Rightarrow B & \text{(терм } \lambda y.x\ (\lambda z.y)\text{)} \\
\quad *(A \Rightarrow B) \Rightarrow A & \text{(тип } u\text{)} \\
\quad\quad *A & \text{(тип } w\text{)} \\
\quad\quad\quad B & \text{(терм } (\lambda y.x\ (\lambda z.y))\ w\text{)} \\
\quad\quad A \Rightarrow B & \text{(терм } \lambda y.x\ (\lambda z.y)\text{)} \\
\quad\quad A & \text{(терм } u\ (\lambda y.x\ (\lambda z.y))\text{)} \\
\quad ((A \Rightarrow B) \Rightarrow A) \Rightarrow A & \text{(терм } \lambda u.u\ (\lambda y.x\ (\lambda z.y))\text{)} \\
\quad B & \text{(терм } x\ (\lambda u.u\ (\lambda y.x\ (\lambda z.y)))\text{)} \\
((((A \Rightarrow B) \Rightarrow A) \Rightarrow A) \Rightarrow B) \Rightarrow B & \\
 & \text{(терм } (\lambda x.x\ (\lambda u.u\ (\lambda y.x\ (\lambda z.y))))\text{)}
\end{array}
$$

Здесь тоже сложно с правилом «из лжи следует всё что угодно», но вопрос решается, если положить $R = B$. Тогда мы получаем минимальный вывод нетривиальной формулы Крипке.

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:
  $\sigma_K(A) = \neg_R \neg_R A$
  $\sigma_K(\Phi \Rightarrow \Psi) = \neg_R \neg_R(\sigma_K(\Phi) \Rightarrow \sigma_K(\Psi))$

- Вариант более слабого преобразования (в стиле Куроды) — это $\sigma_W(\Phi) = \neg_R \neg_R \sigma'_W(\Phi)$, где:
  $\sigma'_W(A) = A$
  $\sigma'_W(\Phi \Rightarrow \Psi) = \sigma'_W(\Phi) \Rightarrow \neg_R \neg_R \sigma'_W(\Psi))$

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

$$
\begin{array}{ll}
*(A \Rightarrow ((A \Rightarrow B') \Rightarrow B')') \Rightarrow R & \text{(тип } k_0\text{)} \\
\quad *A & \text{(тип } x\text{)} \\
\quad\quad *((A \Rightarrow B') \Rightarrow B') \Rightarrow R & \text{(тип } k_1\text{)} \\
\quad\quad\quad *A \Rightarrow B' & \text{(тип } y\text{)} \\
\quad\quad\quad\quad *B \Rightarrow R & \text{(тип } k_2\text{)} \\
\quad\quad\quad\quad\quad (B \Rightarrow R) \Rightarrow R & \text{(тип } y\ x\text{)} \\
\quad\quad\quad\quad\quad R & \text{(тип } y\ x\ k_2\text{)} \\
\quad\quad\quad\quad (B \Rightarrow R) \Rightarrow R & \text{(тип } \lambda k_2.y\ x\ k_2\text{)} \\
\quad\quad\quad (A \Rightarrow B') \Rightarrow B' & \text{(тип } \lambda y k_2.y\ x\ k_2\text{)} \\
\quad\quad\quad R & \text{(тип } k_1\ (\lambda y k_2.y\ x\ k_2)\text{)} \\
\quad\quad (((A \Rightarrow B') \Rightarrow B') \Rightarrow R) \Rightarrow R & \text{(тип } \lambda k_1.k_1\ (\lambda k_2.y\ x\ k_2)\text{)} \\
\quad A \Rightarrow ((A \Rightarrow B') \Rightarrow B')' & \text{(тип } \lambda x k_1.k_1\ (\lambda k_2.y\ x\ k_2)\text{)} \\
\quad R & \text{(тип } k_0\ (\lambda x k_1.k_1\ (\lambda k_2.y\ x\ k_2))\text{)} \\
((A \Rightarrow ((A \Rightarrow B') \Rightarrow B')') \Rightarrow R) \Rightarrow R &
\end{array}
$$

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:

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

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:

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

Система передачи продолжений

Система передачи продолжений в CBN-стиле:

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

Система передачи продолжений в CBV-стиле:

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

Применение CPS-преобразования

Утверждение

Если тип исходного терма M — это Φ, то тип терма $\tau_N(M)$ — это $\sigma_K(\Phi)$, тип терма $\tau_V(M)$ — это $\sigma_W(\Phi)$.

Пример формулы Пирса $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$ показывает, что некоторые термы типизируются только после CPS-преобразования. Это и есть практический смысл теоремы Гливенко.

- $M \to^* \mathrm{const}$ при CBV-стратегии $\Leftrightarrow$ $\tau_V(M)\ \mathrm{id} \to^* \mathrm{const}$ при какой угодно стратегии.
- $M \to^* \mathrm{const}$ при CBN-стратегии $\Leftrightarrow$ $\tau_N(M)\ \mathrm{id} \to^* \mathrm{const}$ 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 ($\neg_{\mathrm{R}} \neg_{\mathrm{R}} A \Rightarrow A$);
- Мягкое прерывание — возвращает текущий контекст вычислений ($(\neg_{\mathrm{R}} A \Rightarrow A) \Rightarrow A$) — закон Пирса, а также тип оператора call-with-current-continuation.

Вывод = конструкция

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

> $*B \Rightarrow A$
>
> > $*(A \Rightarrow B) \Rightarrow B$
> >
> > > $*A \Rightarrow B$
> > >
> > > > B
> > > >
> > > > A
> > >
> > > $(A \Rightarrow B) \Rightarrow A$
> > >
> > > A Закон Пирса!
> >
> > $((A \Rightarrow B) \Rightarrow B) \Rightarrow A$
> >
> > B
>
> $(B \Rightarrow A) \Rightarrow B$
>
> B 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)

$\quad *B \Rightarrow A$ (тип терма y)

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

$\quad\quad\quad *A \Rightarrow B$ (тип терма w)

$\quad\quad\quad\quad B$ (терм $z\,w$)

$\quad\quad\quad\quad A$ (терм $y\,(z\,w)$)

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

$\quad\quad\quad A$ ($T_1 = \mathrm{callcc}\ \lambda w.y\,(z\,w)$)

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

$\quad\quad B$ (терм $x\,(\lambda z.T_1)$)

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

$\quad B$ ($T_2 = \mathrm{callcc}\ \lambda y.x\,(\lambda z.T_1)$)

$((((A \Rightarrow B) \Rightarrow B) \Rightarrow A) \Rightarrow B) \Rightarrow B$ 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 минимальной 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!