

КЗ-свойства языков. Проверка и вывод простейших полиморфных типов



Теория формальных языков
2021 г.



Возникновение понятия типа

Изначально возник в трудах Б.Рассела, который заметил, что в наивной теории множеств существует парадокс:

Парадокс Рассела

$$\Omega = \{A \mid A \notin A\} \Rightarrow (\Omega \in \Omega \Leftrightarrow \Omega \notin \Omega)$$

Понятие типа ограничивает возможные операции над его сущностями \Rightarrow исключает парадоксы
(неожиданное/неприемлемое поведение программ).



Определение типа

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

Б.Пирс



Определение типа

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

Б.Пирс

Описание утверждения о типах — *логическая спецификация*.



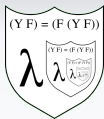
Определение типа

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

Б.Пирс

Описание утверждения о типах — *логическая спецификация*.

Записывается: $\Gamma \vdash M : \sigma$, где Γ — это перечисление $x_i : \tau_i$ — ака контекст.



Определение типа

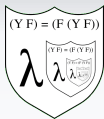
Система типов — гибко управляемый синтаксический метод доказательства отсутствия в программе определенных видов поведения при помощи классификации выражений языка по разновидностям вычисляемых ими значений.

Б.Пирс

Описание утверждения о типах — *логическая спецификация*.

Записывается: $\Gamma \vdash M : \sigma$, где Γ — это перечисление $x_i : \tau_i$ — ака контекст.

Читается: «в контексте Γ терм M имеет тип σ ».



Определение типа

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

Б.Пирс

Описание утверждения о типах — *логическая спецификация*.

Записывается: $\Gamma \vdash M : \sigma$, где Γ — это перечисление $x_i : \tau_i$ — ака контекст.

Читается: «в контексте Γ терм M имеет тип σ ».

Понимается: «если придать переменным x_i типы τ_i , тогда можно установить, что тип выражения M есть σ ».



Системы типов (неформально)

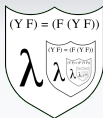
- 1 Простые типы (сорта);
- 2 Функциональные типы данных;
- 3 Алгебраические типы данных (сигнатуры + операции);
- 4 Упорядоченные сорта (решётки) типов данных — ООП.



Таблица связывания

КЗ-свойства имён вынуждают использовать таблицы связывания (имён и функций) с двумя базовыми операциями:

- $\text{bind} :: ([\text{таблица}], [\text{имя}], [\text{тип}]) \rightarrow [\text{таблица}];$
- $\text{lookup} :: ([\text{таблица}], [\text{имя}]) \rightarrow [\text{тип}].$



Пример правил типизации

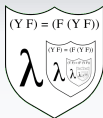
- Сорты (простые типы): Bool, Int.
- Операторы: =, +, условный, вызов функции.
- Синтаксис:

$$\begin{aligned} [\text{Prog}] &::= [\text{Fs}] & [\text{Fs}] &::= [\text{F}] \mid [\text{Fs}] \\ [\text{F}] &::= [\text{Typeld}] \text{ } ([\text{Tlds}]) = [\text{Exp}] \\ [\text{Exps}] &::= [\text{Exp}] \mid [\text{Exp}], [\text{Exps}] \\ [\text{Typeld}] &::= (\text{Bool} \mid \text{Int}) \text{ id} & [\text{Tlds}] &::= [\text{Typeld}], [\text{Tlds}] \mid [\text{Typeld}] \\ [\text{Exp}] &::= \text{num} \mid \text{id} \mid [\text{Exp}] + [\text{Exp}] \mid [\text{Exp}] = [\text{Exp}] \mid \text{id} ([\text{Exps}]) \\ & \mid \text{if } [\text{Exp}] \text{ then } [\text{Exp}] \text{ else } [\text{Exp}] \mid \text{let id} = [\text{Exp}] \text{ in } [\text{Exp}] \end{aligned}$$



Пример правил типизации

```
[Prog] ::= [Fs]           [Fs] ::= [F] | [Fs]
[F] ::= [Typeld] ([Tlds]) = [Exp]
[Exps] ::= [Exp] | [Exp], [Exps]
[Typeld] ::= (Bool | Int) id [Tlds] ::= [Typeld], [Tlds] | [Typeld]
[Exp] ::= num | id | [Exp]+[Exp] | [Exp] = [Exp] | id ([Exps])
          | if [Exp] then [Exp] else [Exp] | let id = [Exp] in [Exp]
```

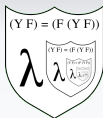


Пример правил типизации

$$\begin{aligned} [\text{Prog}] &::= [\text{Fs}] & [\text{Fs}] &::= [\text{F}] \mid [\text{Fs}] \\ [\text{F}] &::= [\text{Typeld}] \text{ } ([\text{Tlds}]) = [\text{Exp}] \\ [\text{Exps}] &::= [\text{Exp}] \mid [\text{Exp}], [\text{Exps}] \\ [\text{Typeld}] &::= (\text{Bool} \mid \text{Int}) \text{ id} & [\text{Tlds}] &::= [\text{Typeld}], [\text{Tlds}] \mid [\text{Typeld}] \\ [\text{Exp}] &::= \text{num} \mid \text{id} \mid [\text{Exp}] + [\text{Exp}] \mid [\text{Exp}] = [\text{Exp}] \mid \text{id} \text{ } ([\text{Exps}]) \\ &\quad \mid \text{if } [\text{Exp}] \text{ then } [\text{Exp}] \text{ else } [\text{Exp}] \mid \text{let id} = [\text{Exp}] \text{ in } [\text{Exp}] \end{aligned}$$

tchExp(Exp, vtable, ftable) = case Exp of

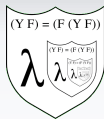
num	int
id	$\mid t == \text{undef} = \text{err}; \text{int}$ $\mid \text{otherwise} = t$ $\text{where } t = \text{lookup}(\text{vtable}, \text{id})$
$\text{Exp}_1 + \text{Exp}_2$	$\mid t_1 \neq \text{int} \parallel t_2 \neq \text{int} = \text{err}; \text{int}$ $\mid \text{otherwise} = \text{int}$ $\text{where } t_1 = \text{tchExp}(\text{Exp}_1, \text{vtable}, \text{ftable}),$ $t_2 = \text{tchExp}(\text{Exp}_2, \text{vtable}, \text{ftable})$



Пример правил типизации

tchExp(Exp, vtable, ftable) = case Exp of

num	int
id	t == undef = err; int otherwise = t where t = lookup(vtable, id)
Exp ₁ +Exp ₂	t ₁ ≠ int t ₂ ≠ int = err; int otherwise = int where t ₁ =tchExp(Exp ₁ ,vtable,ftable), t ₂ =tchExp(Exp ₂ ,vtable,ftable)
Exp ₁ =Exp ₂	t ₁ == t ₂ = bool otherwise = err; bool where t ₁ =tchExp(Exp ₁ ,vtable,ftable), t ₂ =tchExp(Exp ₂ ,vtable,ftable)



Правила типизации в форме вывода

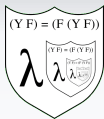
$$\frac{}{\Gamma \vdash \mathbf{num} : \text{int}} \quad \frac{\Gamma \vdash t_1 : \text{int}, \Gamma \vdash t_2 : \text{int}}{\Gamma \vdash t_1 + t_2 : \text{int}}$$

$$\frac{}{\Gamma, \mathbf{id} : \tau \vdash \mathbf{id} : \tau} \quad \frac{\Gamma \vdash t_1 : \sigma, \Gamma \vdash t_2 : \sigma}{\Gamma \vdash t_1 = t_2 : \text{bool}}$$

$$\frac{\Gamma \vdash t_1 : \text{bool}, \Gamma \vdash t_2 : \sigma, \Gamma \vdash t_3 : \sigma}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : \sigma}$$

$$\frac{\Gamma, \mathbf{f_id} : (\tau_1, \dots, \tau_n) \rightarrow \tau_0 \vdash t_i : \tau_i}{\Gamma, \mathbf{f_id} : (\tau_1, \dots, \tau_n) \rightarrow \tau_0 \vdash \mathbf{f_id}(t_1, \dots, t_n) : \tau_0}$$

$$\frac{\Gamma, \mathbf{id} : \tau \vdash s : \sigma, \Gamma \vdash t : \tau}{\Gamma \vdash M, \text{let } \mathbf{id} = t \text{ in } s : \sigma}$$



Параметрический полиморфизм

Параметрический полиморфизм — возможность определять функции равномерно для возможных типов аргументов. Альтернатива — ad hoc полиморфизм — функции перегружены, но определяются отдельно для каждого набора типов.



Параметрический полиморфизм

Параметрический полиморфизм — возможность определять функции равномерно для возможных типов аргументов. Альтернатива — ad hoc полиморфизм — функции перегружены, но определяются отдельно для каждого набора типов.

- Полиморфные типы — переменные, пробегающие все возможные сорта (простые типы)...



Параметрический полиморфизм

Параметрический полиморфизм — возможность определять функции равномерно для возможных типов аргументов. Альтернатива — ad hoc полиморфизм — функции перегружены, но определяются отдельно для каждого набора типов.

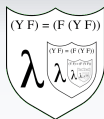
- Полиморфные типы — переменные, пробегающие все возможные сорта (простые типы)...
- ...если разрешены функции высших порядков (HOF) — тогда все возможные функциональные типы.

Как проверять типы полиморфных функций, если их алгебра бесконечнозначна?



Неформально о λ -исчислении

- Формальная модель вычислений, позволяет компактно описывать семантику ЯП с НОФ.
- Бестиповая версия — А. Черч, 1930-е (и много типизированных).
- Базисные операции — применение (функция \rightarrow данные) и абстракция (данные \rightarrow функция).



Неформально о λ -исчислении

Пусть F, X — термы. $F X$ — операция применения терма F (функции) к терму X (данному).

Пусть $M \equiv M[x]$ — терм, возможно содержащий x . Тогда абстракция $\lambda x.M$ обозначает анонимную (неименованную) функцию от x : $x \rightarrow M[x]$.



Неформально о λ -исчислении

Пусть F , X — термы. $F X$ — операция применения терма F (функции) к терму X (данному).

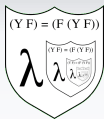
Scheme

; Первый элемент пары — функция, применяемая
; ко второму элементу.
(Fun1 Fun2)

Пусть $M \equiv M[x]$ — терм, возможно содержащий x .
Тогда абстракция $\lambda x.M$ обозначает анонимную
(неименованную) функцию от x : $x \rightarrow M[x]$.

Scheme

(lambda (x) M)

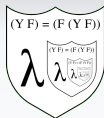


β -редукция и β -эквивалентность

Применение анонимной функции к аргументу:
 $(\lambda x. M[x]) N \rightarrow_{\beta} M[x := N]$ назовём β -редукцией.

Если $T_1 \rightarrow_{\beta}^+ T_2$ или $T_2 \rightarrow_{\beta}^+ T_1$, скажем, что термы T_1 и T_2 β -эквивалентны.

Считаем, что все связанные λ -абстракциями переменные имеют разные имена (как в исходном терме, так и в результате).



Полуформально о типизации λ OF

- 1 Если $M[x]$ имеет тип σ в контексте $x : \tau$, тогда естественно, что $\lambda x.M$ имеет тип $\tau \rightarrow \sigma$;
- 2 Если $(M N)$ имеет тип σ , а N имеет тип τ , тогда естественно, что M имеет тип $\sigma \rightarrow \tau$.



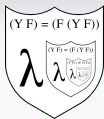
Полуформально о типизации λ OF

- 1 Если $M[x]$ имеет тип σ в контексте $x : \tau$, тогда естественно, что $\lambda x.M$ имеет тип $\tau \rightarrow \sigma$;
- 2 Если $(M N)$ имеет тип σ , а N имеет тип τ , тогда естественно, что M имеет тип $\sigma \rightarrow \tau$.

Логическая спецификация

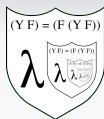
$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x.M : \tau \rightarrow \sigma}$$

$$\frac{\Gamma \vdash M : \tau \rightarrow \sigma, \Gamma \vdash N : \tau}{\Gamma \vdash (M N) : \sigma}$$



Проблема типизации HOF

Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?



Проблема типизации HOF

Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?

- Пусть тип аргумента применения (то есть x) — это τ , а тип результата применения (то есть $(x\ x)$) — это σ . Тогда $\tau = \tau \rightarrow \sigma$.



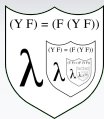
Проблема типизации HOF

Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?

- Пусть тип аргумента применения (то есть x) — это τ , а тип результата применения (то есть $(x\ x)$) — это σ .

Тогда $\tau = \tau \rightarrow \sigma$.

Уравнение $\tau = \tau \rightarrow \sigma$ не имеет неподвижной точки, отличной от \perp (унификация зацикливается).



Проблема типизации λ OF

Рассмотрим терм $\lambda x.(x x)$. Какой у него тип?

- Пусть тип аргумента применения (то есть x) — это τ , а тип результата применения (то есть $(x x)$) — это σ . Тогда $\tau = \tau \rightarrow \sigma$.

Уравнение $\tau = \tau \rightarrow \sigma$ не имеет неподвижной точки, отличной от \perp (унификация зацикливается).

Зацикливается не только унификация: см.

$(\lambda x.(x x)) (\lambda x.(x x))$. Однако в некоторых случаях всё успешно вычисляется: напр. $(\lambda x.(x x)) (\lambda x.(\lambda y.(y x)))$.



Проблема типизации NOF

Рассмотрим терм $\lambda x.(x\ x)$. Какой у него тип?

- Пусть тип аргумента применения (то есть x) — это τ , а тип результата применения (то есть $(x\ x)$) — это σ . Тогда $\tau = \tau \rightarrow \sigma$.

Уравнение $\tau = \tau \rightarrow \sigma$ не имеет неподвижной точки, отличной от \perp (унификация зацикливается).

Зацикливается не только унификация: см. $(\lambda x.(x\ x)) (\lambda x.(x\ x))$. Однако в некоторых случаях всё успешно вычисляется: напр. $(\lambda x.(x\ x)) (\lambda x.(\lambda y.(y\ x)))$.

$\lambda x.(x\ x)$ — частичная функция и не может быть конечным образом определена на всех полиморфных типах.



Нормальная форма HOF

Редекс — это подтерм вида $((\lambda x.M) N)$.
Нормальная форма λ -терма T — это λ -терм,
 β -эквивалентный исходному, не содержащий редексов.



Просто типизированное λ -исчисление

Ограничим множество λ -термов только такими, типы которых всегда выводимы по описанным выше правилам.

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x. M : \tau \rightarrow \sigma} \quad \frac{\Gamma \vdash M : \tau \rightarrow \sigma, \Gamma \vdash N : \tau}{\Gamma \vdash (M N) : \sigma}$$



Просто типизированное λ -исчисление

Ограничим множество λ -термов только такими, типы которых всегда выводимы по описанным выше правилам.

$$\frac{\Gamma, x : \tau \vdash M : \sigma}{\Gamma \vdash \lambda x. M : \tau \rightarrow \sigma} \quad \frac{\Gamma \vdash M : \tau \rightarrow \sigma, \Gamma \vdash N : \tau}{\Gamma \vdash (M N) : \sigma}$$

...а теперь забудем про термы и посмотрим только на типы. Что получилось?

$$\frac{\Gamma, \tau \vdash \sigma}{\Gamma \vdash \tau \rightarrow \sigma} \quad \text{(правило введения импликации)}$$

$$\frac{\Gamma \vdash \tau \rightarrow \sigma, \Gamma \vdash \tau}{\Gamma \vdash \sigma} \quad \begin{array}{l} \text{(правило удаления импликации} \\ \text{aka } \textit{modus ponens}) \end{array}$$



Связь логики и λ НФ: соответствие Карри–Ховарда

- Существует взаимно-однозначное соответствие между типами замкнутых термов в просто типизированном λ -исчислении и тавтологиями в минимальной импликативной логике.
- (теорема о нормализации) Все термы просто типизированного λ -исчисления имеют нормальную форму.
- Доказательствам в минимальной логике соответствуют всюду определенные полиморфные функции высшего порядка.



Древесная форма естественного вывода

Правила вывода для \Rightarrow в minLOG

Применению и абстракции соответствуют следующие правила вывода (modus ponens и правило дедукции):

$$\begin{array}{lcl} () : & \frac{A \quad A \Rightarrow B}{B} & \lambda. : \frac{\begin{array}{c} * A \\ | B \end{array}}{A \Rightarrow B} \end{array}$$

Каждое применение правила вывода в доказательстве minLOG соответствует использованию в терме из λ_{\rightarrow} конструкции с именем этого правила вывода.

В контексте соответствия Карри-Ховарда стрелочный тип далее обозначаем и как $\alpha \rightarrow \beta$ (в программировании), и как $\alpha \Rightarrow \beta$ (в логике).



Унификация

Алгоритм построения $\text{mgu}(E_1, \dots, E_k)$

$$\frac{t = x \quad t = t}{x = t} \quad \frac{f t_1 \dots t_n = f s_1 \dots s_n}{t_1 = s_1 \dots t_n = s_n}$$

$$\frac{x = t \quad x = s}{x = t \quad t = s} \quad \frac{x = t \quad r = s}{x = t \quad r[x := t] = s[x := t]}$$

Условия завершения унификации

1. Существует уравнение $f t_1 \dots t_n = g s_1 \dots s_m$, где $f \neq g$.
 2. Существует уравнение $x = f t_1 \dots t_n$, где x входит в некоторое t_i .
 3. Все уравнения имеют вид $x_i = t_i$, причем x_i не имеет вхождений в t_i — успех.
- } неудача



Алгоритм Хиндли для λ_{\rightarrow}

Пусть дан терм T . В изначально пустом контексте Γ параллельно строятся приближение Φ типа терма T и система уравнений E на переменные типа Φ обратным применением следующих правил.

Правила вывода (Y — свежий тип)

$$\frac{\Gamma, x : X \vdash P : \Psi, E}{\Gamma \vdash \lambda x. P : Y, E \cup \{Y = X \rightarrow \Psi\}} \quad \frac{\Gamma, x : X \vdash x : X, E}{\Gamma, x : X \vdash x : Y, E \cup \{X = Y\}}$$

$$\frac{\Gamma \vdash P : \Phi, E_1 \quad \Gamma \vdash Q : \Psi, E_2}{\Gamma \vdash (P Q) : Y, E_1 \cup E_2 \cup \{\Phi = \Psi \rightarrow Y\}}$$

Пусть построено $\vdash T : \Phi, E$. Из этого приближения строится $\vdash T : \Phi[mgu(E)]$ — окончательный тип терма T .



Пример на типизацию в λ_{\rightarrow}

Вывести тип терма $(\lambda k.k (\lambda x y.x))$

Соглашения об обозначениях:

- \square — символ конца ветки подвывода.
- $\Phi = \Psi_1 \rightarrow \Psi_2$ — уравнение абстракции.
- $\Psi_1 = \Psi_2 \rightarrow \Phi$ — уравнение применения.
- $X_i = X_j$ — уравнение извлечения из контекста.
- Шаг извлечения из контекста в дереве вывода приближения типа не выписываем, только выписываем уравнение.



Пример вывода типа

$$(\lambda k.k (\lambda x y.(y x))) : T_0$$

$$k : T_1 \vdash k (\lambda x y.(y x)) : T_2$$

$$k : T_1 \vdash \lambda x y.(y x) : T_4$$

$$k : T_1, x : T_5 \vdash \lambda y.(y x) : T_6$$

$$k : T_1, x : T_5, y : T_7 \vdash y x : T_8$$

$$k : T_1, x : T_5, y : T_7 \vdash y : T_9$$

□

$$k : T_1 \vdash k : T_3 \quad \square$$

$$k : T_1, x : T_5, y : T_7 \vdash x : T_{10}$$

□

$$T_0 = T_1 \rightarrow T_2$$

$$T_3 = T_4 \rightarrow T_2$$

$$T_1 = T_3$$

$$T_4 = T_5 \rightarrow T_6$$

$$T_6 = T_7 \rightarrow T_8$$

$$T_9 = T_{10} \rightarrow T_8$$

$$T_9 = T_7 \quad T_5 = T_{10}$$

Уравнения **применения**: $T_3 = T_4 \rightarrow T_2$, $T_9 = T_{10} \rightarrow T_8$.

Уравнения **абстракции**: $T_0 = T_1 \rightarrow T_2$, $T_4 = T_5 \rightarrow T_6$, $T_6 = T_7 \rightarrow T_8$.

Подставляем уравнения извлечения из контекста в систему и строим mgu системы $\{T_1 = T_4 \rightarrow T_2, T_7 = T_5 \rightarrow T_8, T_0 = T_1 \rightarrow T_2, T_4 = T_5 \rightarrow T_6, T_6 = T_7 \rightarrow T_8\}$.

Ответ: искомый тип терма есть

$$T_0 = ((T_5 \rightarrow ((T_5 \rightarrow T_8) \rightarrow T_8)) \rightarrow T_2) \rightarrow T_2.$$



Пример обратного соответствия

Покажем, что тип $((A \Rightarrow ((A \Rightarrow B) \Rightarrow B)) \Rightarrow C) \Rightarrow C$ корректен. Поскольку это — функциональный тип, то внешним конструктором его терма будет абстракция (соответствует правилу дедукции).

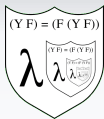
$*(A \Rightarrow ((A \Rightarrow B) \Rightarrow B)) \Rightarrow C$ (тип терма x)

(Здесь нужно придумать, как построить терм типа C , имея только x)

C

$((A \Rightarrow ((A \Rightarrow B) \Rightarrow B)) \Rightarrow C) \Rightarrow C$ (тип терма $\lambda x. \dots$)

Поскольку x — это функциональный терм, принимающий аргументом функцию типа $\tau = A \Rightarrow ((A \Rightarrow B) \Rightarrow B)$, нужно попробовать построить терм, имеющий тип τ . Для этого опять понадобится абстракция (даже две).



Пример обратного соответствия

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

$*A$ (тип терма y)

Тут опять не хватает шагов: нужно построить терм типа $(A \Rightarrow B) \Rightarrow B$, имея y .

Так как это функция, добавляем ещё абстракцию.

$(A \Rightarrow B) \Rightarrow B$

$A \Rightarrow ((A \Rightarrow B) \Rightarrow B)$ (тип терма $\lambda y. \dots$)

C (тип терма $x (\lambda y. \dots)$)

$((A \Rightarrow ((A \Rightarrow B) \Rightarrow B)) \Rightarrow C) \Rightarrow C$ (тип терма $\lambda x. x (\lambda y. \dots)$)



Пример обратного соответствия

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

$*A$ (тип терма y)

$*A \Rightarrow B$ (тип терма z)

Как получить терм типа B из y и z ?

B

$(A \Rightarrow B) \Rightarrow B$ (тип терма $\lambda z. \dots$)

$A \Rightarrow ((A \Rightarrow B) \Rightarrow B)$ (тип терма $\lambda y. (\lambda z. \dots)$)

C (тип терма x ($\lambda y. (\lambda z. \dots)$))

$((A \Rightarrow ((A \Rightarrow B) \Rightarrow B)) \Rightarrow C) \Rightarrow C$ (тип терма $\lambda x. x$ ($\lambda y. \lambda z. \dots$)))



Пример обратного соответствия

$$\begin{array}{l}
 *(A \Rightarrow ((A \Rightarrow B) \Rightarrow B)) \Rightarrow C \text{ (тип терма } x) \\
 | *A \text{ (тип терма } y) \\
 | | *A \Rightarrow B \text{ (тип терма } z) \\
 | | | B \text{ (тип терма } z \ y) \\
 | | \hline
 | | (A \Rightarrow B) \Rightarrow B \text{ (тип терма } \lambda z.(z \ y)) \\
 | \hline
 | A \Rightarrow ((A \Rightarrow B) \Rightarrow B) \text{ (тип терма } \lambda y.(\lambda z.(z \ y))) \\
 | C \text{ (тип терма } x \ (\lambda y.(\lambda z.(z \ y))) \\
 \hline
 ((A \Rightarrow ((A \Rightarrow B) \Rightarrow B)) \Rightarrow C) \Rightarrow C \text{ (тип терма } \lambda x.x \ (\lambda y.\lambda z.(z \ y)))
 \end{array}$$

Мы не только построили доказательство тавтологии $((A \Rightarrow ((A \Rightarrow B) \Rightarrow B)) \Rightarrow C) \Rightarrow C$ в minLOG, но ещё и предъявили всюду определённую функцию высшего порядка, которая реализует это доказательство.



Теория полиморфных типов

- Добавление других конструкций (пары, объединения по ключу) — расширение логики типов дополнительными операциями ($\&$, \vee).
- Добавление контейнерных типов (списки, обработка исключений) — расширение логики типов модальностями.
- Результат алгоритмизации проверки типов — *theorems for free*: семантические свойства полиморфных функций выполняются вследствие корректности их типизации.



Теория полиморфных типов

- Добавление других конструкций (пары, объединения по ключу) — расширение логики типов дополнительными операциями ($\&$, \vee).
- Добавление контейнерных типов (списки, обработка исключений) — расширение логики типов модальностями.
- Результат алгоритмизации проверки типов — *theorems for free*: семантические свойства полиморфных функций выполняются вследствие корректности их типизации.

Логические системы, описывающие полиморфные типы, отличаются от классической! У их алгебр нет конечного носителя. Некоторые классические тавтологии (например, $A \vee (A \Rightarrow B)$) не являются конструктивными.