Лекция L2
Типизированное λ–исчисление, I

Вадим Пузаренко

19 октября 2021 г.

Мотивация

Синтаксис этого исчисления отличается от синтаксиса бестипового λ–исчисления только тем, что каждому терму приписывается некоторый тип (класс объектов, к которому он относится). Это накладывает некоторые ограничения на применимость их друг к другу и, соответственно, на построение термов. Мы предполагаем, что имеется конечный или бесконечный набор некоторых так называемых **базовых** или **простейших** попарно различных типов σ_0, σ_1, ..., которые можно мыслить, как множества (впрочем, в наших рассмотрениях их можно понимать как просто некоторые символы). Из них мы будем строить составные типы следующим образом: если τ_0 и τ_1 — типы, то $(\tau_0 \to \tau_1)$ — тип, который удобно мыслить, как класс всех отображений из τ_0 в τ_1.

Построение типов

Определение

Определим понятие **типа** индукцией по построению.

Построение типов

Определение

Определим понятие **типа** индукцией по построению.

1. Каждый простейший тип является типом.

Построение типов

Определение

Определим понятие **типа** индукцией по построению.

1. Каждый простейший тип является типом.
2. Если π и τ — типы, то $(\pi \to \tau)$ также является типом.

Построение типов

Определение

Определим понятие **типа** индукцией по построению.

1. Каждый простейший тип является типом.
2. Если π и τ — типы, то $(\pi \to \tau)$ также является типом.
3. Других типов нет.

Построение типов

Определение

Определим понятие **типа** индукцией по построению.

1. Каждый простейший тип является типом.
2. Если π и τ — типы, то $(\pi \to \tau)$ также является типом.
3. Других типов нет.

Предполагается, что типы, имеющие различные записи, различны. Другими словами, нетривиальной пары синонимов нет.

Построение типов

Примеры

Пусть ω — единственный простейший тип, например, тип натуральных чисел. Тогда

Построение типов

Примеры

Пусть ω — единственный простейший тип, например, тип натуральных чисел. Тогда

1. $(\omega \to \omega)$ — тип всех функций из ω в ω;

Построение типов

Примеры

Пусть ω — единственный простейший тип, например, тип натуральных чисел. Тогда

1. $(\omega \to \omega)$ — тип всех функций из ω в ω;
2. $((\omega \to \omega) \to \omega)$ — тип отображений из множества функций из ω в ω во множество натуральных чисел ω, т.е. тип функционалов;

Построение типов

Примеры

Пусть ω — единственный простейший тип, например, тип натуральных чисел. Тогда

1. $(\omega \to \omega)$ — тип всех функций из ω в ω;
2. $((\omega \to \omega) \to \omega)$ — тип отображений из множества функций из ω в ω во множество натуральных чисел ω, т.е. тип функционалов;
3. $(\omega \to (\omega \to \omega))$ — множество отображений, действующих из ω во множество функций из ω в ω (фактически, имитация бинарной функции из ω^2 в ω);

Построение типов

Примеры

Пусть ω — единственный простейший тип, например, тип натуральных чисел. Тогда

1. $(\omega \to \omega)$ — тип всех функций из ω в ω;
2. $((\omega \to \omega) \to \omega)$ — тип отображений из множества функций из ω в ω во множество натуральных чисел ω, т.е. тип функционалов;
3. $(\omega \to (\omega \to \omega))$ — множество отображений, действующих из ω во множество функций из ω в ω (фактически, имитация бинарной функции из ω^2 в ω);
4. а каков смысл типов $((\omega \to \omega) \to (\omega \to \omega))$, $(((\omega \to \omega) \to \omega) \to \omega)$, $(\omega \to (\omega \to (\omega \to \omega))$?

Типизация λ–термов

Определение

Любое отображение γ, сопоставляющее каждой переменной некоторый тип, называется **типизацией** переменных. При этом будем говорить, что переменная x имеет тип $\gamma(x)$.

Упорядоченную пару, состоящую из переменной и её типа, будем называть **типизированной переменной** (будем записывать как $x : \tau$ или x^{τ}).

Типизация λ–термов

Определение

Любое отображение γ, сопоставляющее каждой переменной некоторый тип, называется **типизацией** переменных. При этом будем говорить, что переменная x имеет тип $\gamma(x)$.
Упорядоченную пару, состоящую из переменной и её типа, будем называть **типизированной переменной** (будем записывать как $x : \tau$ или x^{τ}).

Если каждой переменной приписан некоторый тип, то мы также можем приписать типы и некоторым λ–термам.

Типизация λ–термов

Определение

Любое отображение γ, сопоставляющее каждой переменной некоторый тип, называется **типизацией** переменных. При этом будем говорить, что переменная x имеет тип $\gamma(x)$.
Упорядоченную пару, состоящую из переменной и её типа, будем называть **типизированной переменной** (будем записывать как $x : \tau$ или x^τ).

Если каждой переменной приписан некоторый тип, то мы также можем приписать типы и некоторым λ–термам.

Замечание.

Всегда будем считать, что типизация γ переменных обязательно удовлетворяет следующему условию: каждый тип приписывается бесконечному количеству переменных.

Типизация λ–термов

Полагаем, что сигнатура σ также типизирована, т.е. каждой константе **c** заранее приписывается тип $t(\mathbf{c})$. Далее, пусть задана типизация γ переменных.

Типизация λ–термов

Полагаем, что сигнатура σ также типизирована, т.е. каждой константе **c** заранее приписывается тип $t(\mathbf{c})$. Далее, пусть задана типизация γ переменных.

Определение

Индукцией по построению термов зададим **приписывание типа** следующим образом:

Типизация λ–термов

Полагаем, что сигнатура σ также типизирована, т.е. каждой константе **c** заранее приписывается тип $t(\mathbf{c})$. Далее, пусть задана типизация γ переменных.

Определение

Индукцией по построению термов зададим **приписывание типа** следующим образом:

1. всякая переменная x получает тип $\tau = \gamma(x)$;

Типизация λ–термов

Полагаем, что сигнатура σ также типизирована, т.е. каждой константе **c** заранее приписывается тип $t(\mathbf{c})$. Далее, пусть задана типизация γ переменных.

Определение

Индукцией по построению термов зададим **приписывание типа** следующим образом:

1. всякая переменная x получает тип $\tau = \gamma(x)$;
2. всякая константа **c** получает некоторый тип $t(\mathbf{c})$, независимо от типизации γ;

Типизация λ–термов

Полагаем, что сигнатура σ также типизирована, т.е. каждой константе **c** заранее приписывается тип $t(\mathbf{c})$. Далее, пусть задана типизация γ переменных.

Определение

Индукцией по построению термов зададим **приписывание типа** следующим образом:

1. всякая переменная x получает тип $\tau = \gamma(x)$;
2. всякая константа **c** получает некоторый тип $t(\mathbf{c})$, независимо от типизации γ;
3. если M и N — λ–термы, уже получившие типы $(\pi \to \tau)$ и π соответственно, то λ–терм (MN) получает тип τ;

Типизация λ–термов

Полагаем, что сигнатура σ также типизирована, т.е. каждой константе **c** заранее приписывается тип $t(\mathbf{c})$. Далее, пусть задана типизация γ переменных.

Определение

Индукцией по построению термов зададим **приписывание типа** следующим образом:

1. всякая переменная x получает тип $\tau = \gamma(x)$;
2. всякая константа **c** получает некоторый тип $t(\mathbf{c})$, независимо от типизации γ;
3. если M и N — λ–термы, уже получившие типы $(\pi \to \tau)$ и π соответственно, то λ–терм (MN) получает тип τ;
4. если M — λ–терм, уже получивший тип π, а переменная x — тип τ, то λ–терм $\lambda x.M$ получает тип $(\tau \to \pi)$;

Типизация λ–термов

Полагаем, что сигнатура σ также типизирована, т.е. каждой константе **c** заранее приписывается тип $t(\mathbf{c})$. Далее, пусть задана типизация γ переменных.

Определение

Индукцией по построению термов зададим **приписывание типа** следующим образом:

1. всякая переменная x получает тип $\tau = \gamma(x)$;
2. всякая константа **c** получает некоторый тип $t(\mathbf{c})$, независимо от типизации γ;
3. если M и N — λ–термы, уже получившие типы $(\pi \to \tau)$ и π соответственно, то λ–терм (MN) получает тип τ;
4. если M — λ–терм, уже получивший тип π, а переменная x — тип τ, то λ–терм $\lambda x.M$ получает тип $(\tau \to \pi)$;
5. всякий λ–терм получает тип только согласно пп. 1–4.

Типизация λ–термов

Определение

Упорядоченную пару, состоящую из λ–терма и его типа, будем называть **типизированным λ–термом**. Для типизированных λ–термов используется та же запись, что и для переменных: $A : \tau$ обозначает λ–терм A типа τ.

Типизация λ–термов

Определение

Упорядоченную пару, состоящую из λ–терма и его типа, будем называть **типизированным λ–термом**. Для типизированных λ–термов используется та же запись, что и для переменных: $A : \tau$ обозначает λ–терм A типа τ.

Определение

Пусть задана типизация γ переменных. λ–Терм t назовем **типизируемым при типизации переменных** γ, если в результате этой типизации он получает некоторый тип. λ–Терм назовем **типизируемым**, если его переменным можно приписать типы так, что сам этот λ–терм получит некоторый тип.

Типизация λ–термов

Исчисление приписывания типов

Зафиксируем типизацию Γ переменных.

$\Gamma \vdash \mathbf{c} : t(\mathbf{c})$	— аксиома
$\Gamma \vdash x : \gamma(x)$	— аксиома
$\dfrac{\Gamma \vdash M : (\pi \to \tau);\ \ \Gamma \vdash N : \pi}{\Gamma \vdash (MN) : \tau}\ (\mathrm{I})$	$\dfrac{\Gamma \vdash M : \pi;\ \ \Gamma \vdash x : \tau}{\Gamma \vdash \lambda x.M : (\tau \to \pi)}\ (\mathrm{II})$

Типизация λ–термов

Предложение L1

Какова бы ни была типизация переменных, любой λ–терм может получить не более одного типа.

Типизация λ–термов

Предложение L1

Какова бы ни была типизация переменных, любой λ–терм может получить не более одного типа.

Доказательство.

Индукцией по построению типизированного λ–терма. Всюду в индукционных предположениях следует понимать, что типизация однозначна.

$t \in \{v, \mathbf{c}\}$. Если $t = v$, то имеем $v : \gamma(v)$; если же $t = \mathbf{c}$, то имеем $\mathbf{c} : t(\mathbf{c})$.

$t = (MN)$. По предположению индукции, имеем $M : \tau_1$, $N : \tau_2$. Согласно правилу (I), должно выполняться $\tau_1 = (\tau_2 \to \tau_1')$ и $(MN) : \tau_1'$.

$t = \lambda x.M$. По предположению индукции, $x : \gamma(x)$ и $M : \tau$. По правилу (II), имеем $\lambda x.M : (\gamma(x) \to \tau)$.

$\square$

Типизация λ–термов

Примеры

1. Один и тот же λ–терм может иметь тип при одной типизации переменных и не иметь его при другой типизации. Так, если задать следующую типизацию переменных: $x : \sigma$, $y : \sigma$, то приписать тип λ–терму (xy) невозможно. При другой типизации переменных, например, $x : (\sigma \to \sigma)$, $y : \sigma$, λ–терм (xy) получит тип σ.

Типизация λ–термов

Примеры

1. Один и тот же λ–терм может иметь тип при одной типизации переменных и не иметь его при другой типизации. Так, если задать следующую типизацию переменных: $x : \sigma$, $y : \sigma$, то приписать тип λ–терму (xy) невозможно. При другой типизации переменных, например, $x : (\sigma \to \sigma)$, $y : \sigma$, λ–терм (xy) получит тип σ.
2. λ–Терм (xx) не типизируем, поскольку для приписывания ему типа необходимо, чтобы переменной x был приписан одновременно тип вида σ и $(\sigma \to \tau)$, что невозможно.

Типизация λ–термов

Примеры

1. Один и тот же λ–терм может иметь тип при одной типизации переменных и не иметь его при другой типизации. Так, если задать следующую типизацию переменных: $x : \sigma$, $y : \sigma$, то приписать тип λ–терму (xy) невозможно. При другой типизации переменных, например, $x : (\sigma \to \sigma)$, $y : \sigma$, λ–терм (xy) получит тип σ.
2. λ–Терм (xx) не типизируем, поскольку для приписывания ему типа необходимо, чтобы переменной x был приписан одновременно тип вида σ и $(\sigma \to \tau)$, что невозможно.
3. λ–Термы $((xy)x)$, $((xy)(xy))$ не типизируемы.

Типизация λ–термов

Примеры

1. Один и тот же λ–терм может иметь тип при одной типизации переменных и не иметь его при другой типизации. Так, если задать следующую типизацию переменных: $x : \sigma$, $y : \sigma$, то приписать тип λ–терму (xy) невозможно. При другой типизации переменных, например, $x : (\sigma \to \sigma)$, $y : \sigma$, λ–терм (xy) получит тип σ.
2. λ–Терм (xx) не типизируем, поскольку для приписывания ему типа необходимо, чтобы переменной x был приписан одновременно тип вида σ и $(\sigma \to \tau)$, что невозможно.
3. λ–Термы $((xy)x)$, $((xy)(xy))$ не типизируемы.
4. λ–Терм $(x(yx))$ типизируем.

Типизация λ–термов

Примеры

1. Один и тот же λ–терм может иметь тип при одной типизации переменных и не иметь его при другой типизации. Так, если задать следующую типизацию переменных: $x : \sigma$, $y : \sigma$, то приписать тип λ–терму (xy) невозможно. При другой типизации переменных, например, $x : (\sigma \to \sigma)$, $y : \sigma$, λ–терм (xy) получит тип σ.
2. λ–Терм (xx) не типизируем, поскольку для приписывания ему типа необходимо, чтобы переменной x был приписан одновременно тип вида σ и $(\sigma \to \tau)$, что невозможно.
3. λ–Термы $((xy)x)$, $((xy)(xy))$ не типизируемы.
4. λ–Терм $(x(yx))$ типизируем.
5. Если λ–терм типизируем, то и любой его подтерм также типизируем.

Исчисление λ–конверсий

Аксиомы

- $(\lambda x.TR) : \tau \Rightarrow [T]^x_R : \tau$, где R свободен для x в T (β–конверсия)
- $\lambda x^\sigma.T : \tau \Rightarrow \lambda y^\sigma.[T]^x_y : \tau$, где y свободна для x в T (α–конверсия)
- $T : \tau \Rightarrow T : \tau$ (рефлексивность)

Исчисление λ–конверсий

Аксиомы

- $(\lambda x.TR):\tau \Rightarrow [T]^x_R:\tau$, где R свободен для x в T (β–конверсия)
- $\lambda x^\sigma.T:\tau \Rightarrow \lambda y^\sigma.[T]^x_y:\tau$, где y свободна для x в T (α–конверсия)
- $T:\tau \Rightarrow T:\tau$ (рефлексивность)

Правила вывода

- $\dfrac{T:\tau \Rightarrow U:\tau;\ U:\tau \Rightarrow R:\tau}{T:\tau \Rightarrow R:\tau}$ (транзитивность)
- $\dfrac{T:(\tau\to\sigma) \Rightarrow U:(\tau\to\sigma);\ \ R:\tau}{(TR):\sigma \Rightarrow (UR):\sigma}$ (преобразование функции)
- $\dfrac{T:\tau \Rightarrow U:\tau;\ \ R:(\tau\to\sigma)}{(RT):\sigma \Rightarrow (RU):\sigma}$ (преобразование аргумента)
- $\dfrac{T:\sigma \Rightarrow U:\sigma;\ \ x:\tau}{\lambda x.T:(\tau\to\sigma) \Rightarrow \lambda x.U:(\tau\to\sigma)}$ (преобразование ξ)

Исчисление λ–конверсий

Предложение L2

В результате β–конверсии любого типизированного λ–терма получается снова типизированный λ–терм того же самого типа.

Исчисление λ–конверсий

Предложение L2

В результате β–конверсии любого типизированного λ–терма получается снова типизированный λ–терм того же самого типа.

Сначала докажем лемму.

Исчисление λ–конверсий

Предложение L2

В результате β–конверсии любого типизированного λ–терма получается снова типизированный λ–терм того же самого типа.

Сначала докажем лемму.

Лемма

Пусть задана типизация переменных, при которой λ–терм M типизируем, а переменная x и λ–терм N получают один и тот же тип. Тогда при этой типизации λ–терм $[M]_N^x$ также типизируем, причём имеет тот же самый тип, что и M.

Исчисление λ–конверсий

Предложение L2

В результате β–конверсии любого типизированного λ–терма получается снова типизированный λ–терм того же самого типа.

Сначала докажем лемму.

Лемма

Пусть задана типизация переменных, при которой λ–терм M типизируем, а переменная x и λ–терм N получают один и тот же тип. Тогда при этой типизации λ–терм $[M]_N^x$ также типизируем, причём имеет тот же самый тип, что и M.

Доказательство леммы.

Индукцией по сложности построения λ–терма M. Пусть в дальнейшем $M : \sigma$, $x : \tau$ и $N : \tau$.

- $M \in \{y, \mathbf{c}\}$, где $y \neq x$ — переменная. Тогда $[M]_N^x = M$ и утверждение выполняется.

Исчисление λ–конверсий

Доказательство леммы (продолжение)

$M = x$. Тогда $[M]^x_N = N$ и утверждение также выполняется.

$M = \lambda x.M_1$. Тогда $[M]^x_N = M$ и утверждение также выполняется.

$M = \lambda y.M_1$, где $y \neq x$. Тогда $\sigma = (\sigma_1 \to \sigma_2)$, $M_1 : \sigma_2$, $y : \sigma_1$ и, по предположению индукции, $[M_1]^x_N : \sigma_2$. Следовательно, $[M]^x_N = \lambda y.[M_1]^x_N$ получает тип $(\sigma_1 \to \sigma_2) = \sigma$.

$M = (M_1 M_2)$. Тогда $M_1 : (\rho \to \sigma)$, $M_2 : \rho$ и, по предположению индукции, $[M_1]^x_N : (\rho \to \sigma)$, $[M_2]^x_N : \rho$. Далее, $[M]^x_N = ([M_1]^x_N [M_2]^x_N)$ и $[M]^x_N$ получает тип σ. $\square$

Исчисление λ–конверсий

Доказательство предложения L2.

Пусть задана β–конверсия $(\lambda x.MN) \Rightarrow [M]^x_N$, и λ–терм $(\lambda x.MN)$ получает тип τ; покажем, что $[M]^x_N$ также получает тип τ.
По определению типизации, $N : \sigma$ и $\lambda x.M : (\sigma \to \tau)$.
Следовательно, $x : \sigma$ и $M : \tau$. Так как x и N получают один и тот же тип, по лемме получаем, что $[M]^x_N : \tau$. $\square$

Алгоритм типизации

Пусть дана типизация γ переменных и заранее фиксируется типизация констант. Опишем алгоритм проверки типизации λ–термов.

Пусть последовательность T_0, T_1, $\ldots$, $T_n = T$ описывает построение λ–терма T в следующем смысле:

1. T_i есть переменная или константа;
2. $T_i = \lambda x.T_j$, где $j < i$;
3. $T_i = (T_j T_k)$, где $j, k < i$.

Предполагается, что все λ–термы последовательности являются подтермами T.

Алгоритм типизации

Далее, индукцией по $i \leqslant n$ определим приписывание λ–термам $T_0, T_1, \ldots, T_i$ типов. Если T_j не получает тип при некотором $j < i$, то процесс останавливается. Пусть теперь все λ–термы $T_0, T_1, \ldots, T_{i-1}$ получили типы. Разберём несколько случаев для T_i:

1. если T_i — переменная или константа, то он получает тип (согласно γ или заранее фиксированной типизации констант);
2. если $T_i = \lambda x.T_j$ и $x : \sigma$, $T_j : \tau$, то $T_i : (\sigma \to \tau)$;
3. пусть теперь $T_i = (T_j T_k)$ и $T_j : \sigma$, $T_k : \tau$; если $\sigma = (\tau \to \rho)$ для некоторого типа ρ, то $T_i : \rho$; иначе λ–терм T_i не получает тип.

Мотивация

Метод унификации типов позволяет выяснить, в частности, для λ–терма, типизируем он или нет. Ввиду того, что имеется алгоритм, отвечающий на вопрос, имеется ли унификатор (даже наиболее общий унификатор) или нет, проблема типизируемости λ–термов также оказывается алгоритмически разрешимой.

Подстановка

В дальнейшем, будем называть простейшие типы как **типовые переменные**.

Подстановка

В дальнейшем, будем называть простейшие типы как **типовые переменные**.

Определение

Пусть α_1, α_2, …, α_n — попарно различные типовые переменные, а ρ_1, ρ_2, …, ρ_n — произвольные типы. Обозначим через $\sigma[\alpha_1/\rho_1, \alpha_2/\rho_2, \ldots, \alpha_n/\rho_n]$ **результат подстановки** в типе σ вместо всех вхождений α_1, α_2, …, α_n типов ρ_1, ρ_2, …, ρ_n соответственно. Рекурсивно процесс построения типа может быть записан следующим образом (здесь α — типовая переменная, а σ и τ — произвольные типы; $\mathbb{S} = [\alpha_1/\rho_1, \alpha_2/\rho_2, \ldots, \alpha_n/\rho_n]$):

$$\alpha[\alpha_1/\rho_1, \alpha_2/\rho_2, \ldots, \alpha_n/\rho_n] = \begin{cases} \rho_i, & \text{если } \alpha \equiv \alpha_i,\ 1 \leqslant i \leqslant n; \\ \alpha & \text{в противном случае;} \end{cases} \qquad (1)$$

$$(\sigma \to \tau)\mathbb{S} = (\sigma\mathbb{S} \to \tau\mathbb{S}). \qquad (2)$$

“Равенство”

Определение

Будем говорить, что “равенство” $\sigma = \tau$ имеет **унификатор**, если существует подстановка $\mathbb{S}$, такая что $\sigma\mathbb{S} \equiv \tau\mathbb{S}$. В этом случае, $\mathbb{S}$ будем называть **унификатором** σ и τ.

“Равенство”

Определение

Будем говорить, что “равенство” $\sigma = \tau$ имеет **унификатор**, если существует подстановка $\mathbb{S}$, такая что $\sigma\mathbb{S} \equiv \tau\mathbb{S}$. В этом случае, $\mathbb{S}$ будем называть **унификатором** σ и τ.

Определение

Подстановку $\mathbb{S}$ будем называть **наиболее общим унификатором (НОУ)** для σ и τ, если $\mathbb{S}$ действительно является их унификатором и, к тому же, для любого другого унификатора $\mathbb{S}'$ для σ и τ найдется подстановка $\mathbb{T}$, для которой выполняются следующие соотношения:

$$\sigma\mathbb{S}' \equiv \sigma\mathbb{S}\mathbb{T}, \ \tau\mathbb{S}' \equiv \tau\mathbb{S}\mathbb{T}.$$

“Равенство”

Примеры.

1. Пусть $\sigma \equiv ((\alpha \to \alpha) \to \beta)$, $\tau \equiv (\beta \to (\gamma \to \gamma))$. Тогда $\mathbb{S} = [\beta/(\alpha \to \alpha), \gamma/\alpha]$ является унификатором для σ и τ. Действительно,
$$\sigma\mathbb{S} \equiv ((\alpha \to \alpha) \to (\alpha \to \alpha)) \equiv \tau\mathbb{S}.$$
Пусть ϱ — какой-либо тип; тогда подстановка $\mathbb{S}' = [\beta/(\varrho \to \varrho), \gamma/\varrho]$ также будет унификатором для σ и τ. Более того, $\mathbb{S}' = \mathbb{S}\mathbb{T}$.
2. Равенство $\alpha = ((\alpha \to \alpha) \to \beta)$ не имеет унификаторов, поскольку, независимо от того, какой тип будет подставляться вместо α, длина типа слева будет оставаться меньше, чем длина типа справа.
3. Пусть α — типовая переменная и тип τ, $\mathrm{lh}(\tau) > 1$, в который входит α. Тогда $\sigma = \tau$ не имеет унификаторов. В этом случае, для каждой подстановки $\mathbb{S}$ будем иметь $\mathrm{lh}(\alpha\mathbb{S}) < \mathrm{lh}(\tau\mathbb{S})$ и, следовательно, $\alpha = \tau$ не имеет унификаторов.

Алгоритм унификации (Дж.А. Робинсон)

Тип $\Rightarrow$ Дерево

Каждому типу σ сопоставим бинарное помеченное дерево $\mathrm{T}(\sigma)$ (помечаются только листья типовыми переменными) согласно следующей процедуре.

- Если $\sigma \equiv \alpha$ — типовая переменная, то $\mathrm{T}(\sigma)$ состоит только из одной вершины, помеченной α.
- Если $\sigma \equiv (\sigma_1 \to \sigma_2)$, то, предполагая, что $\mathrm{T}(\sigma_1)$ и $\mathrm{T}(\sigma_2)$ уже определены, определим дерево $\mathrm{T}(\sigma)$ как дерево с корнем, левым потомком которого является $\mathrm{T}(\sigma_1)$, а правым — $\mathrm{T}(\sigma_2)$.

Алгоритм унификации (Дж.А. Робинсон)

Обход дерева

Пусть σ и τ — типы, которым приписываются деревья $\mathrm{T}(\sigma)$ и $\mathrm{T}(\tau)$ соответственно. Одновременно проводим обход деревьев, пока не найдем первого различия, в котором хотя бы одна из вершин помечена; при этом листья, помеченные различными переменными, считаются различными. При обходе дерева придерживаемся левой стороны: движения проводим по левым потомкам. Если достигнуты листья и различия не найдены, то возвращаемся на их предков (по всем правым путям и ровно одному левому пути), от которых производим один такт к правым потомкам, после чего повторяем описанную выше процедуру.

Алгоритм унификации (Дж.А. Робинсон)

Конструкция

Опишем шаг $n \in \omega$ конструкции.

1. Если $n = 0$, то полагаем $\mathbb{S}_0 = id_{\beta_1,\beta_2,\ldots,\beta_k}$, где β_1, β_2, $\ldots$, β_k — список всех типовых перменных, входящих в σ и τ.
2. Согласно обходу дерева, находим первый шаг, на котором вершины различны. Тогда эти вершины являются корнями деревьев $\mathrm{T}(\alpha_n)$ и $\mathrm{T}(\sigma_n)$. Отметим, что хотя бы одно из деревьев соответствует типовой переменной (скажем, α_n). Если такой шаг не существует, то унификатор уже построен.
3. Если тип α_n входит в тип σ_n (и, в этом случае, $\mathrm{lh}(\sigma_n) > 1$, поскольку $\alpha_n \not\equiv \sigma_n$), то равенство $\sigma = \tau$ не имеет унификатора, и алгоритм останавливается.
4. В противном случае, полагаем $\sigma := \sigma[\alpha_n/\sigma_n]$, $\tau := \tau[\alpha_n/\sigma_n]$, $\mathbb{S}_n := \mathbb{S}_{n-1}[\alpha_n/\sigma_n]$ и переходим к следующему шагу.

Алгоритм унификации (Дж.А. Робинсон)

Пример.

Пусть даны типы $\sigma \equiv ((\alpha \to \alpha) \to \beta)$ и $\tau \equiv (\beta \to (\gamma \to \gamma))$.

Алгоритм унификации (Дж.А. Робинсон)

Пример.

Пусть даны типы $\sigma \equiv ((\alpha \to \alpha) \to \beta)$ и $\tau \equiv (\beta \to (\gamma \to \gamma))$.

- На первом шаге строим $\mathbb{S}_1 = [\beta/(\alpha \to \alpha)]$ и получаем $\sigma\mathbb{S}_1 \equiv ((\alpha \to \alpha) \to (\alpha \to \alpha))$, $\tau\mathbb{S}_1 \equiv ((\alpha \to \alpha) \to (\gamma \to \gamma))$;

Алгоритм унификации (Дж.А. Робинсон)

Пример.

Пусть даны типы $\sigma \equiv ((\alpha \to \alpha) \to \beta)$ и $\tau \equiv (\beta \to (\gamma \to \gamma))$.

- На первом шаге строим $\mathbb{S}_1 = [\beta/(\alpha \to \alpha)]$ и получаем $\sigma\mathbb{S}_1 \equiv ((\alpha \to \alpha) \to (\alpha \to \alpha))$, $\tau\mathbb{S}_1 \equiv ((\alpha \to \alpha) \to (\gamma \to \gamma))$;
- на втором шаге строим $\mathbb{S}_2 = \mathbb{S}_1[\gamma/\alpha]$ и получаем $\sigma\mathbb{S}_2 \equiv (\sigma\mathbb{S}_1)[\gamma/\alpha] \equiv ((\alpha \to \alpha) \to (\alpha \to \alpha)) \equiv (\tau\mathbb{S}_1)[\gamma/\alpha] \equiv \tau\mathbb{S}_2$;

Алгоритм унификации (Дж.А. Робинсон)

Пример.

Пусть даны типы $\sigma \equiv ((\alpha \to \alpha) \to \beta)$ и $\tau \equiv (\beta \to (\gamma \to \gamma))$.

- На первом шаге строим $\mathbb{S}_1 = [\beta/(\alpha \to \alpha)]$ и получаем $\sigma\mathbb{S}_1 \equiv ((\alpha \to \alpha) \to (\alpha \to \alpha))$, $\tau\mathbb{S}_1 \equiv ((\alpha \to \alpha) \to (\gamma \to \gamma))$;
- на втором шаге строим $\mathbb{S}_2 = \mathbb{S}_1[\gamma/\alpha]$ и получаем $\sigma\mathbb{S}_2 \equiv (\sigma\mathbb{S}_1)[\gamma/\alpha] \equiv ((\alpha \to \alpha) \to (\alpha \to \alpha)) \equiv (\tau\mathbb{S}_1)[\gamma/\alpha] \equiv \tau\mathbb{S}_2$;
- таким образом, $\mathbb{S}_2$ является унификатором для σ и τ (более того, этот унификатор будет НОУ).

Алгоритм унификации (Дж.А. Робинсон)

Теорема L4

Алгоритм Дж. Робинсона позволяет для каждой пары типов σ и τ определить, имеет ли равенство $\sigma = \tau$ унификатор или нет. В первом случае, он выдаёт наиболее общий унификатор.

Алгоритм унификации (Дж.А. Робинсон)

Теорема L4

Алгоритм Дж. Робинсона позволяет для каждой пары типов σ и τ определить, имеет ли равенство $\sigma = \tau$ унификатор или нет. В первом случае, он выдаёт наиболее общий унификатор.

Доказательство.

Заметим сначала, что процесс конечен, поскольку если происходит переход к следующему шагу, то количество переменных уменьшается на единицу. Поэтому если алгоритм завершается корректно, то равенство $\sigma = \tau$ унифицируемо.
В обратную сторону, пусть равенство $\sigma = \tau$ унифицируемо и пусть m — последний шаг конструкции. Докажем индукцией по $i < m$, что **для любого унификатора $\mathbb{S}'$ существует подстановка $\mathbb{T}_i$ такая, что $\mathbb{S}' = \mathbb{S}_i\mathbb{T}_i$.**

Алгоритм унификации (Дж.А. Робинсон)

Доказательство (продолжение).

Пусть $\mathbb{S}'$ — унификатор для σ и τ, т.е. $\sigma\mathbb{S}' \equiv \tau\mathbb{S}'$.
$i = 0$. Полагаем $\mathbb{T}_0 = \mathbb{S}'$; действительно,
$\mathbb{S}' = id_{\beta_1,\beta_2,\ldots,\beta_k}\mathbb{S}' = \mathbb{S}_0\mathbb{T}_0$.
$i \mapsto i+1$. Предположим, что $\mathbb{S}' = \mathbb{S}_i\mathbb{T}_i$ для подходящей подстановки $\mathbb{T}_i$. Из соотношения
$(\sigma\mathbb{S}_i)\mathbb{T}_i \equiv \sigma(\mathbb{S}_i\mathbb{T}_i) \equiv \sigma\mathbb{S}' \equiv \tau\mathbb{S}' \equiv \tau(\mathbb{S}_i\mathbb{T}_i) \equiv (\tau\mathbb{S}_i)\mathbb{T}_i$ вытекает, что $\alpha_{i+1}\mathbb{T}_i \equiv \sigma_{i+1}\mathbb{T}_i$. Кроме того, типовая переменная α_{i+1} не встречается в σ_{i+1}. Напомним также, что $\mathbb{S}_{i+1} = \mathbb{S}_i[\alpha_{i+1}/\sigma_{i+1}]$.
Положим $\mathbb{T}_{i+1} = [\alpha_{i+1}/\sigma_{i+1}]\mathbb{T}_i$. Покажем, что
$\mathbb{T}_i = [\alpha_{i+1}/\sigma_{i+1}]\mathbb{T}_{i+1}$. В самом деле, для $\alpha \not\equiv \alpha_{i+1}$ имеет место
$\alpha([\alpha_{i+1}/\sigma_{i+1}]\mathbb{T}_{i+1}) \equiv (\alpha[\alpha_{i+1}/\sigma_{i+1}])\mathbb{T}_{i+1} \equiv \alpha\mathbb{T}_{i+1} \equiv$
$\alpha([\alpha_{i+1}/\sigma_{i+1}]\mathbb{T}_i) \equiv (\alpha[\alpha_{i+1}/\sigma_{i+1}])\mathbb{T}_i \equiv \alpha\mathbb{T}_i$. Далее,
$\alpha_{i+1}([\alpha_{i+1}/\sigma_{i+1}]\mathbb{T}_{i+1}) \equiv (\alpha_{i+1}[\alpha_{i+1}/\sigma_{i+1}])\mathbb{T}_{i+1} \equiv \sigma_{i+1}\mathbb{T}_{i+1} \equiv$
$\sigma_{i+1}([\alpha_{i+1}/\sigma_{i+1}]\mathbb{T}_i) \equiv (\sigma_{i+1}[\alpha_{i+1}/\sigma_{i+1}])\mathbb{T}_i \equiv \sigma_{i+1}\mathbb{T}_i \equiv \alpha_{i+1}\mathbb{T}_i$.

Алгоритм унификации (Дж.А. Робинсон)

Доказательство (окончание).

Остаётся проверить равенство $\mathbb{S}' = \mathbb{S}_i\mathbb{T}_i = \mathbb{S}_{i+1}\mathbb{T}_{i+1}$:
$\mathbb{S}_{i+1}\mathbb{T}_{i+1} = (\mathbb{S}_i[\alpha_{i+1}/\sigma_{i+1}])\mathbb{T}_{i+1} = \mathbb{S}_i([\alpha_{i+1}/\sigma_{i+1}]\mathbb{T}_{i+1}) = \mathbb{S}_i\mathbb{T}_i$.
И, наконец, заключаем, что $\mathbb{S}_{m-1}$ является унификатором для σ и τ, а по доказанному, $\mathbb{S}_{m-1}$ является НОУ. $\square$

Алгоритм унификации (Дж.А. Робинсон)

Доказательство (окончание).

Остаётся проверить равенство $\mathbb{S}' = \mathbb{S}_i\mathbb{T}_i = \mathbb{S}_{i+1}\mathbb{T}_{i+1}$:
$\mathbb{S}_{i+1}\mathbb{T}_{i+1} = (\mathbb{S}_i[\alpha_{i+1}/\sigma_{i+1}])\mathbb{T}_{i+1} = \mathbb{S}_i([\alpha_{i+1}/\sigma_{i+1}]\mathbb{T}_{i+1}) = \mathbb{S}_i\mathbb{T}_i$.
И, наконец, заключаем, что $\mathbb{S}_{m-1}$ является унификатором для σ и τ, а по доказанному, $\mathbb{S}_{m-1}$ является НОУ. □

Определение

Пусть $\sigma_1, \sigma_2, \ldots, \sigma_n, \tau_1, \tau_2, \ldots, \tau_n$ — типы. Будем говорить, что система равенств

$$\sigma_1 = \tau_1, \ \sigma_2 = \tau_2, \ \ldots, \ \sigma_n = \tau_n \tag{3}$$

унифицируема, если существует подстановка $\mathbb{S}$ (называемая унификатором системы), такая что имеет место $\sigma_1\mathbb{S} \equiv \tau_1\mathbb{S}$, $\sigma_2\mathbb{S} \equiv \tau_2\mathbb{S}$, $\ldots$, $\sigma_n\mathbb{S} \equiv \tau_n\mathbb{S}$.

Алгоритм типизации (Хиндли)

Проблема существования унификатора системы равенств (3) сводится к проблеме существования унификатора для σ и τ, где $\sigma = (\ldots(\sigma_1 \to \sigma_2) \to \ldots \to \sigma_n)$, $\tau = (\ldots(\tau_1 \to \tau_2) \to \ldots \to \tau_n)$.

Алгоритм типизации (Хиндли)

Проблема существования унификатора системы равенств (3) сводится к проблеме существования унификатора для σ и τ, где $\sigma = (\ldots(\sigma_1 \to \sigma_2) \to \ldots \to \sigma_n)$, $\tau = (\ldots(\tau_1 \to \tau_2) \to \ldots \to \tau_n)$.

Определение

Будем говорить, что σ — **наиболее общий тип (НОТ)** λ–терма M, если $\Gamma \vdash M : \sigma$ (здесь Γ — типизация переменных) и для любого другого типа σ', для которого выполняется $\Gamma' \vdash M : \sigma'$, найдётся подстановка $\mathbb{S}$ такая, что $\sigma' = \sigma\mathbb{S}$.

Алгоритм типизации (Хиндли)

Ниже Γ — типизация переменных (более точно, конечная её часть). Определим **наиболее общую типизацию (НОТ)** λ–термов рекурсивно по их построению:

1. $M \equiv x$. Полагаем $x : \alpha \vdash x : \alpha$.
2. $M \equiv \lambda x.P$, где $\Gamma \vdash P : \tau$ — НОТ λ–терма P. Тогда $\Gamma \setminus \{x : \sigma\} \vdash \lambda x.P : (\sigma \to \tau)$, если $x : \sigma \in \Gamma$; $\Gamma \vdash \lambda x.P : (\alpha \to \tau)$, если $x : \sigma \notin \Gamma$ (α — новая переменная).
3. $M \equiv (PQ)$, где $\Gamma_1 \vdash P : \sigma_1$ и $\Gamma_2 \vdash Q : \sigma_2$ — НОТ λ–термов P и Q. Сначала переименуем типовые переменные так, чтобы в Γ_1 и Γ_2 отсутствовали общие типовые переменные. Далее, рассмотрим систему равенств
$$\sigma_1 = (\sigma_2 \to \eta),$$
$$\varrho_i = \tau_i,$$
причём η — новая типовая переменная, а равенство $\varrho_i = \tau_i$ помещается в систему в точности в тех случаях, когда имеет место $x_i : \varrho_i \in \Gamma_1$ и $x_i : \tau_i \in \Gamma_2$.

Алгоритм типизации (Хиндли)

Если эта система равенств не унифицируема, то λ–терм не типизируем. В противном случае, возьмём НОУ $\mathbb{S}$. Тогда НОТ λ–терма M будет иметь вид $\Gamma_1\mathbb{S} \cup \Gamma_2\mathbb{S} \vdash M : \eta\mathbb{S}$.

Алгоритм типизации (Хиндли)

Если эта система равенств не унифицируема, то λ–терм не типизируем. В противном случае, возьмём НОУ $\mathbb{S}$. Тогда НОТ λ–терма M будет иметь вид $\Gamma_1\mathbb{S} \cup \Gamma_2\mathbb{S} \vdash M : \eta\mathbb{S}$.

Теорема L5

Алгоритм Хиндли позволяет для каждого λ–терма определить, типизируем он или нет. В первом случае, он выдаёт наиболее общий терм.

Алгоритм типизации (Хиндли)

Если эта система равенств не унифицируема, то λ–терм не типизируем. В противном случае, возьмём НОУ $\mathbb{S}$. Тогда НОТ λ–терма M будет иметь вид $\Gamma_1\mathbb{S} \cup \Gamma_2\mathbb{S} \vdash M : \eta\mathbb{S}$.

Теорема L5

Алгоритм Хиндли позволяет для каждого λ–терма определить, типизируем он или нет. В первом случае, он выдаёт наиболее общий терм.

Доказательство.

Повторяет фактически рассуждения теоремы L4. $\square$

Спасибо за внимание.