Лямбда-исчисление

С. Л. Кузнецов

Математический институт им. В.А. Стеклова РАН
sk@mi-ras.ru https://homepage.mi-ras.ru/~sk/

Казанский университет · лекция 2 · 7 ноября 2024 г.

λ-термы

- Напомним определение **λ-терма**:

Определение

Множество λ-термов Λ определяется как наименьшее по включению множество со следующими свойствами:

1. $\mathrm{Var} \subseteq \Lambda$;
2. если $M, N \in \Lambda$, то $(MN) \in \Lambda$;
3. если $M \in \Lambda$ и $x \in \mathrm{Var}$, то $\lambda x.M \in \Lambda$.

λ-термы

- Напомним определение **λ-терма**:

Определение

Множество λ-термов Λ определяется как наименьшее по включению множество со следующими свойствами:

1. $\mathrm{Var} \subseteq \Lambda$;
2. если $M, N \in \Lambda$, то $(MN) \in \Lambda$;
3. если $M \in \Lambda$ и $x \in \mathrm{Var}$, то $\lambda x.M \in \Lambda$.

- Нет ограничений на применение: любой терм можно применить, как функцию, к любому другому.

λ-термы

- Напомним определение **λ-терма**:

Определение

Множество λ-термов Λ определяется как наименьшее по включению множество со следующими свойствами:

1. $\mathrm{Var} \subseteq \Lambda$;
2. если $M, N \in \Lambda$, то $(MN) \in \Lambda$;
3. если $M \in \Lambda$ и $x \in \mathrm{Var}$, то $\lambda x.M \in \Lambda$.

- Нет ограничений на применение: любой терм можно применить, как функцию, к любому другому.
- Это не соответствует стандартному математическому понятию функции: к объекту из множества A можно применять только функцию, чья область определения содержит A.

λ-термы

- Напомним определение **λ-терма**:

Определение

Множество λ-термов Λ определяется как наименьшее по включению множество со следующими свойствами:

1. $\mathrm{Var} \subseteq \Lambda$;
2. если $M, N \in \Lambda$, то $(MN) \in \Lambda$;
3. если $M \in \Lambda$ и $x \in \mathrm{Var}$, то $\lambda x.M \in \Lambda$.

- Нет ограничений на применение: любой терм можно применить, как функцию, к любому другому.
- Это не соответствует стандартному математическому понятию функции: к объекту из множества A можно применять только функцию, чья область определения содержит A.
- В частности, появляются странные термы вроде $\Omega = (\lambda x.(xx))(\lambda x.(xx))$.

Системы типов

- Таким образом, чтобы получить исчисление с лучшими свойствами, вводится **контроль типов** — ограничения на операцию применения.

Системы типов

- Таким образом, чтобы получить исчисление с лучшими свойствами, вводится **контроль типов** — ограничения на операцию применения.
- Каждому λ-терму присваивается **тип**, и при применении типы должны быть согласованы.

Системы типов

- Таким образом, чтобы получить исчисление с лучшими свойствами, вводится **контроль типов** — ограничения на операцию применения.
- Каждому λ-терму присваивается **тип**, и при применении типы должны быть согласованы.
- Соответствующие модификации λ-исчисления называются **типизованными.**

Системы типов

- Таким образом, чтобы получить исчисление с лучшими свойствами, вводится **контроль типов** — ограничения на операцию применения.
- Каждому λ-терму присваивается **тип**, и при применении типы должны быть согласованы.
- Соответствующие модификации λ-исчисления называются **типизованными.**
- Типизованные λ-исчисления обладают свойствами нормализуемости, а также естественной семантикой.

Системы типов

- Таким образом, чтобы получить исчисление с лучшими свойствами, вводится **контроль типов** — ограничения на операцию применения.
- Каждому λ-терму присваивается **тип**, и при применении типы должны быть согласованы.
- Соответствующие модификации λ-исчисления называются **типизованными.**
- Типизованные λ-исчисления обладают свойствами нормализуемости, а также естественной семантикой.
- Однако их вычислительные возможности слабее, чем у бестипового λ-исчисления.

Системы типов

- Таким образом, чтобы получить исчисление с лучшими свойствами, вводится **контроль типов** — ограничения на операцию применения.
- Каждому λ-терму присваивается **тип**, и при применении типы должны быть согласованы.
- Соответствующие модификации λ-исчисления называются **типизованными.**
- Типизованные λ-исчисления обладают свойствами нормализуемости, а также естественной семантикой.
- Однако их вычислительные возможности слабее, чем у бестипового λ-исчисления.
- Системы типов бывают разными, мы познакомимся с двумя из них.

Простая система типов

- Начнём с **простой системы типов**; соответствующая версия λ-исчисления обозначается $\lambda_{\rightarrow}$.

Простая система типов

- Начнём с **простой системы типов**; соответствующая версия λ-исчисления обозначается $\lambda_{\to}$.
- Типы строятся из базовых (переменные по типам) с помощью единственной операции $\to$.

Простая система типов

- Начнём с **простой системы типов**; соответствующая версия λ-исчисления обозначается $\lambda_{\to}$.
- Типы строятся из базовых (переменные по типам) с помощью единственной операции $\to$.
- Базовые типы обозначаем α, β, γ, а сложные — τ, σ.
  - Пример: $\tau = \alpha \to (\beta \to \alpha)$.

Простая система типов

- Начнём с **простой системы типов**; соответствующая версия λ-исчисления обозначается $\lambda_{\to}$.
- Типы строятся из базовых (переменные по типам) с помощью единственной операции $\to$.
- Базовые типы обозначаем α, β, γ, а сложные — τ, σ.
  - Пример: $\tau = \alpha \to (\beta \to \alpha)$.
- Тип $\tau \to \sigma$ — это тип функций из типа τ в тип σ.

Простая система типов

- Начнём с **простой системы типов**; соответствующая версия λ-исчисления обозначается $\lambda_{\to}$.
- Типы строятся из базовых (переменные по типам) с помощью единственной операции $\to$.
- Базовые типы обозначаем α, β, γ, а сложные — τ, σ.
  - Пример: $\tau = \alpha \to (\beta \to \alpha)$.
- Тип $\tau \to \sigma$ — это тип функций из типа τ в тип σ.
- Запись "$M : \tau$" читается как «терм M имеет тип τ».

Простая система типов

- Начнём с **простой системы типов**; соответствующая версия λ-исчисления обозначается $\lambda_{\to}$.
- Типы строятся из базовых (переменные по типам) с помощью единственной операции $\to$.
- Базовые типы обозначаем α, β, γ, а сложные — τ, σ.
  - Пример: $\tau = \alpha \to (\beta \to \alpha)$.
- Тип $\tau \to \sigma$ — это тип функций из типа τ в тип σ.
- Запись "$M : \tau$" читается как «терм M имеет тип τ».
- **Правила типизации:**
  1. если $M : \tau$ и $x : \sigma$, то $\lambda x.M : (\sigma \to \tau)$;
  2. если $M : (\sigma \to \tau)$ и $N : \sigma$, то $(MN) : \tau$.

Мягкая и жёсткая типизация

- Правила типизации оставляют определённую свободу для выбора типов, а именно, не зафиксировано, какие типы получают переменные.

Мягкая и жёсткая типизация

- Правила типизации оставляют определённую свободу для выбора типов, а именно, не зафиксировано, какие типы получают переменные.
- Для **свободных** переменных типы указываются явно. Присвоение типов свободным переменным: $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$ называется **контекстом.** Таким образом, терм со свободными переменными типизируется не сам по себе, а «в контексте».

Мягкая и жёсткая типизация

- Правила типизации оставляют определённую свободу для выбора типов, а именно, не зафиксировано, какие типы получают переменные.
- Для **свободных** переменных типы указываются явно. Присвоение типов свободным переменным: $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$ называется **контекстом.** Таким образом, терм со свободными переменными типизируется не сам по себе, а «в контексте».
- Для связанных переменных есть два варианта.

Мягкая и жёсткая типизация

- Правила типизации оставляют определённую свободу для выбора типов, а именно, не зафиксировано, какие типы получают переменные.
- Для **свободных** переменных типы указываются явно. Присвоение типов свободным переменным: $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$ называется **контекстом.** Таким образом, терм со свободными переменными типизируется не сам по себе, а «в контексте».
- Для связанных переменных есть два варианта.
- При **жёсткой** типизации (по Чёрчу), около каждой лямбды указывается тип переменной, что фиксирует тип всего терма.

Мягкая и жёсткая типизация

- Правила типизации оставляют определённую свободу для выбора типов, а именно, не зафиксировано, какие типы получают переменные.
- Для **свободных** переменных типы указываются явно. Присвоение типов свободным переменным: $\Gamma = x_1 : \tau_1, \ldots, x_n : \tau_n$ называется **контекстом.** Таким образом, терм со свободными переменными типизируется не сам по себе, а «в контексте».
- Для связанных переменных есть два варианта.
- При **жёсткой** типизации (по Чёрчу), около каждой лямбды указывается тип переменной, что фиксирует тип всего терма.
- При **мягкой** типизации (по Карри), типы связанных переменных не зафиксированы, и один терм может иметь много типов.

Мягкая и жёсткая типизация

- Жёсткая типизация вносит изменения в язык λ-термов. Например, вместо одного терма $\lambda x.x$ теперь будет семейство термов $\lambda x^{\tau}.x$ для каждого типа τ.

Мягкая и жёсткая типизация

- Жёсткая типизация вносит изменения в язык λ-термов. Например, вместо одного терма $\lambda x.x$ теперь будет семейство термов $\lambda x^{\tau}.x$ для каждого типа τ.
- При этом каждый из них имеет единственный тип: $\lambda x^{\alpha}.x : (\alpha \to \alpha)$, $\lambda x^{\alpha \to \beta}.x : ((\alpha \to \beta) \to (\alpha \to \beta))$ и т.п.

Мягкая и жёсткая типизация

- Жёсткая типизация вносит изменения в язык λ-термов. Например, вместо одного терма $\lambda x.x$ теперь будет семейство термов $\lambda x^{\tau}.x$ для каждого типа τ.
- При этом каждый из них имеет единственный тип: $\lambda x^{\alpha}.x : (\alpha \to \alpha)$, $\lambda x^{\alpha \to \beta}.x : ((\alpha \to \beta) \to (\alpha \to \beta))$ и т.п.
- Возможны ситуации, когда неправильная расстановка типов делает терм нетипизуемым, например $\lambda f^{\beta \to \gamma} g^{\alpha \to \beta} x^{\alpha}.f(gx) : (\beta \to \gamma) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$, а $\lambda f^{\alpha} g^{\beta} x^{\gamma}.f(gx)$ не типизируем.

Мягкая и жёсткая типизация

- Жёсткая типизация вносит изменения в язык λ-термов. Например, вместо одного терма $\lambda x.x$ теперь будет семейство термов $\lambda x^{\tau}.x$ для каждого типа τ.
- При этом каждый из них имеет единственный тип: $\lambda x^{\alpha}.x : (\alpha \to \alpha)$, $\lambda x^{\alpha \to \beta}.x : ((\alpha \to \beta) \to (\alpha \to \beta))$ и т.п.
- Возможны ситуации, когда неправильная расстановка типов делает терм нетипизуемым, например $\lambda f^{\beta \to \gamma} g^{\alpha \to \beta} x^{\alpha}.f(gx) : (\beta \to \gamma) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$, а $\lambda f^{\alpha} g^{\beta} x^{\gamma}.f(gx)$ не типизируем.
- Мягкая типизация **полиморфна.** Язык термов бестиповый, и каждый терм может иметь много типов (а может — ни одного).

Мягкая и жёсткая типизация

- Жёсткая типизация вносит изменения в язык λ-термов. Например, вместо одного терма $\lambda x.x$ теперь будет семейство термов $\lambda x^{\tau}.x$ для каждого типа τ.
- При этом каждый из них имеет единственный тип: $\lambda x^{\alpha}.x : (\alpha \to \alpha)$, $\lambda x^{\alpha\to\beta}.x : ((\alpha \to \beta) \to (\alpha \to \beta))$ и т.п.
- Возможны ситуации, когда неправильная расстановка типов делает терм нетипизуемым, например $\lambda f^{\beta\to\gamma} g^{\alpha\to\beta} x^{\alpha}.f(gx) : (\beta \to \gamma) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$, а $\lambda f^{\alpha} g^{\beta} x^{\gamma}.f(gx)$ не типизируем.
- Мягкая типизация **полиморфна.** Язык термов бестиповый, и каждый терм может иметь много типов (а может — ни одного).
- Таким образом, при мягкой типизации запись "$M : \tau$" правильно читать как «M *может* иметь тип τ».

Мягкая и жёсткая типизация

- Жёсткая типизация вносит изменения в язык λ-термов. Например, вместо одного терма $\lambda x.x$ теперь будет семейство термов $\lambda x^{\tau}.x$ для каждого типа τ.
- При этом каждый из них имеет единственный тип: $\lambda x^{\alpha}.x : (\alpha \to \alpha)$, $\lambda x^{\alpha\to\beta}.x : ((\alpha \to \beta) \to (\alpha \to \beta))$ и т.п.
- Возможны ситуации, когда неправильная расстановка типов делает терм нетипизуемым, например $\lambda f^{\beta\to\gamma} g^{\alpha\to\beta} x^{\alpha}.f(gx) : (\beta \to \gamma) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$, а $\lambda f^{\alpha} g^{\beta} x^{\gamma}.f(gx)$ не типизируем.
- Мягкая типизация **полиморфна.** Язык термов бестиповый, и каждый терм может иметь много типов (а может — ни одного).
- Таким образом, при мягкой типизации запись "$M : \tau$" правильно читать как «M *может* иметь тип τ».
- В присутствии контекста, "$\Gamma \vdash M : \tau$" читается как «M может иметь тип τ в контексте Γ».

Наиболее общий тип

- Если $\Gamma \vdash M : \tau$ и переменная по типам α не входит в Γ , то также $\Gamma \vdash M : \tau[\alpha := \sigma]$ для любого типа σ.

Наиболее общий тип

- Если $\Gamma \vdash M : \tau$ и переменная по типам α не входит в Γ , то также $\Gamma \vdash M : \tau[\alpha := \sigma]$ для любого типа σ.
- Например, $\vdash \lambda xy.x : \alpha \to (\beta \to \alpha)$, но также $\vdash \lambda xy.x : (\gamma \to \gamma) \to (\beta \to (\gamma \to \gamma))$.

Наиболее общий тип

- Если $\Gamma \vdash M : \tau$ и переменная по типам α не входит в Γ , то также $\Gamma \vdash M : \tau[\alpha := \sigma]$ для любого типа σ.
- Например, $\vdash \lambda xy.x : \alpha \to (\beta \to \alpha)$, но также $\vdash \lambda xy.x : (\gamma \to \gamma) \to (\beta \to (\gamma \to \gamma))$.
- Тип τ **более общий**, чем $\tau[\alpha := \sigma]$ (и наоборот, $\tau[\alpha := \sigma]$ — более специальный).

Наиболее общий тип

- Если $\Gamma \vdash M : \tau$ и переменная по типам α не входит в Γ, то также $\Gamma \vdash M : \tau[\alpha := \sigma]$ для любого типа σ.
- Например, $\vdash \lambda xy.x : \alpha \to (\beta \to \alpha)$, но также $\vdash \lambda xy.x : (\gamma \to \gamma) \to (\beta \to (\gamma \to \gamma))$.
- Тип τ **более общий**, чем $\tau[\alpha := \sigma]$ (и наоборот, $\tau[\alpha := \sigma]$ — более специальный).

Теорема

Если тип M типизуем в контексте Γ (т.е. существует τ, такой что $\Gamma \vdash M : \tau$), то существует наиболее общий тип τ_0 для M в контексте Γ. А именно, $\Gamma \vdash M : \tau_0$, и если $\Gamma \vdash M : \tau$, то τ получается из τ_0 подстановкой.

Наиболее общий тип

- Если $\Gamma \vdash M : \tau$ и переменная по типам α не входит в Γ, то также $\Gamma \vdash M : \tau[\alpha := \sigma]$ для любого типа σ.
- Например, $\vdash \lambda xy.x : \alpha \to (\beta \to \alpha)$, но также $\vdash \lambda xy.x : (\gamma \to \gamma) \to (\beta \to (\gamma \to \gamma))$.
- Тип τ **более общий**, чем $\tau[\alpha := \sigma]$ (и наоборот, $\tau[\alpha := \sigma]$ — более специальный).

Теорема

Если тип M типизуем в контексте Γ (т.е. существует τ, такой что $\Gamma \vdash M : \tau$), то существует наиболее общий тип τ_0 для M в контексте Γ. А именно, $\Gamma \vdash M : \tau_0$, и если $\Gamma \vdash M : \tau$, то τ получается из τ_0 подстановкой.

- Таким образом, для данных Γ и M множество типов $\{\tau \mid \Gamma \vdash M : \tau\}$ либо пусто, либо порождается одним типом τ_0.

Выведение типов

Теорема

Задача проверки (по данным M и Γ) типизуемости и, если ответ положителен, нахождения наиболее общего типа, алгоритмически разрешима.

Выведение типов

Теорема

Задача проверки (по данным M и Γ) типизуемости и, если ответ положителен, нахождения наиболее общего типа, алгоритмически разрешима.

- Процесс вычисления наиболее общего типа называется **выведением типов** (type inference).

Выведение типов

Теорема

Задача проверки (по данным M и Γ) типизуемости и, если ответ положителен, нахождения наиболее общего типа, алгоритмически разрешима.

- Процесс вычисления наиболее общего типа называется **выведением типов** (type inference).
- За счёт выведения типов в некоторых функциональных языках (например, Haskell) не обязательно указывать типы, алгоритм сам вычислит наиболее общий.

Выведение типов

Теорема

Задача проверки (по данным M и Γ) типизуемости и, если ответ положителен, нахождения наиболее общего типа, алгоритмически разрешима.

- Процесс вычисления наиболее общего типа называется **выведением типов** (type inference).
- За счёт выведения типов в некоторых функциональных языках (например, Haskell) не обязательно указывать типы, алгоритм сам вычислит наиболее общий.
- Код выглядит бестиповым, однако компилятор не разрешит использовать нетипизуемый терм (такой как $\lambda x.(xx)$).

Выведение типов

- При помещении терма в более сложный терм его наиболее общий тип может измениться (конкретизироваться).

Выведение типов

- При помещении терма в более сложный терм его наиболее общий тип может измениться (конкретизироваться).
- Например, наиболее общий тип $\lambda x.x$ — это $\alpha \to \alpha$, но внутри терма $(\lambda x.x)(\lambda y.y)$ у первого $(\lambda x.x)$ тип конкретизируется до $((\alpha \to \alpha) \to (\alpha \to \alpha))$.

Выведение типов

- При помещении терма в более сложный терм его наиболее общий тип может измениться (конкретизироваться).
- Например, наиболее общий тип $\lambda x.x$ — это $\alpha \to \alpha$, но внутри терма $(\lambda x.x)(\lambda y.y)$ у первого $(\lambda x.x)$ тип конкретизируется до $((\alpha \to \alpha) \to (\alpha \to \alpha))$.
- При β-редукции типизация не портится: если $\Gamma \vdash M_1 : \tau$ и $M_1 \to_\beta M_2$, то $\Gamma \vdash M_2 : \tau$.

Выведение типов

- При помещении терма в более сложный терм его наиболее общий тип может измениться (конкретизироваться).
- Например, наиболее общий тип $\lambda x.x$ — это $\alpha \to \alpha$, но внутри терма $(\lambda x.x)(\lambda y.y)$ у первого $(\lambda x.x)$ тип конкретизируется до $((\alpha \to \alpha) \to (\alpha \to \alpha))$.
- При β-редукции типизация не портится: если $\Gamma \vdash M_1 : \tau$ и $M_1 \to_\beta M_2$, то $\Gamma \vdash M_2 : \tau$.
- Обратное при этом неверно: множество допустимых типов после β-редукции может расшириться.

Выведение типов

- При помещении терма в более сложный терм его наиболее общий тип может измениться (конкретизироваться).
- Например, наиболее общий тип $\lambda x.x$ — это $\alpha \to \alpha$, но внутри терма $(\lambda x.x)(\lambda y.y)$ у первого $(\lambda x.x)$ тип конкретизируется до $((\alpha \to \alpha) \to (\alpha \to \alpha))$.
- При β-редукции типизация не портится: если $\Gamma \vdash M_1 : \tau$ и $M_1 \to_\beta M_2$, то $\Gamma \vdash M_2 : \tau$.
- Обратное при этом неверно: множество допустимых типов после β-редукции может расшириться.
- Например, $(\lambda xy.y)\Omega$ не типизуем, но $\vdash \lambda y.y : (\alpha \to \alpha)$.

Выведение типов

- При помещении терма в более сложный терм его наиболее общий тип может измениться (конкретизироваться).
- Например, наиболее общий тип $\lambda x.x$ — это $\alpha \to \alpha$, но внутри терма $(\lambda x.x)(\lambda y.y)$ у первого $(\lambda x.x)$ тип конкретизируется до $((\alpha \to \alpha) \to (\alpha \to \alpha))$.
- При β-редукции типизация не портится: если $\Gamma \vdash M_1 : \tau$ и $M_1 \to_\beta M_2$, то $\Gamma \vdash M_2 : \tau$.
- Обратное при этом неверно: множество допустимых типов после β-редукции может расшириться.
- Например, $(\lambda xy.y)\Omega$ не типизуем, но $\vdash \lambda y.y : (\alpha \to \alpha)$.
- Другой пример: $\lambda fz.(((\lambda xy.x)f)(fz))$ имеет наиболее общий тип $(\alpha \to \beta) \to (\alpha \to (\alpha \to \beta))$, но после редукции тип обобщается: $\lambda fz.f : \gamma \to (\alpha \to \gamma)$.

Нормализуемость

Теорема

Всякий типизуемый в $\lambda_{\to}$ терм сильно нормализуем.

Теоретико-множественная семантика

- В отличие от бестипового λ-исчисления, семантика которого весьма сложна, исчисление $\lambda_{\rightarrow}$ (с жёсткой типизацией) имеет простую и естественную **теоретико-множественную семантику.**

Теоретико-множественная семантика

- В отличие от бестипового λ-исчисления, семантика которого весьма сложна, исчисление $\lambda_{\to}$ (с жёсткой типизацией) имеет простую и естественную **теоретико-множественную семантику.**
- Типы интерпретируются множествами: каждому типу τ сопоставляется множество $D(\tau)$, так что

$$D(\sigma \to \tau) = D(\sigma) \to D(\tau) = D(\tau)^{D(\sigma)} = \\ = \{\text{все функции из } D(\sigma) \text{ в } D(\tau)\}.$$

Интерпретация $D(\alpha)$ базовых типов произвольна.

Теоретико-множественная семантика

- В отличие от бестипового λ-исчисления, семантика которого весьма сложна, исчисление $\lambda_{\to}$ (с жёсткой типизацией) имеет простую и естественную **теоретико-множественную семантику.**
- Типы интерпретируются множествами: каждому типу τ сопоставляется множество $D(\tau)$, так что

$$D(\sigma \to \tau) = D(\sigma) \to D(\tau) = D(\tau)^{D(\sigma)} = \\ = \{\text{все функции из } D(\sigma) \text{ в } D(\tau)\}.$$

  Интерпретация $D(\alpha)$ базовых типов произвольна.
- Каждый (типизованный по Чёрчу) терм M интерпретируется объектом $[\![M]\!] \in D(\tau)$, где τ — тип M.

Теоретико-множественная семантика

- Переменные (свободные) интерпретируются произвольно в соответствующих $D(\tau)$. Интерпретация переменных обозначается θ.

Теоретико-множественная семантика

- Переменные (свободные) интерпретируются произвольно в соответствующих $D(\tau)$. Интерпретация переменных обозначается θ.
- Применение функции: $[\![FM]\!] = [\![F]\!]([\![M]\!])$. Корректность: если $F : \sigma \to \tau$ и $M \in \sigma$, то $[\![F]\!] \in (D(\sigma) \to D(\tau))$ и $[\![M]\!] \in D(\sigma)$.

Теоретико-множественная семантика

- Переменные (свободные) интерпретируются произвольно в соответствующих $D(\tau)$. Интерпретация переменных обозначается θ.
- Применение функции: $[\![FM]\!] = [\![F]\!]([\![M]\!])$. Корректность: если $F : \sigma \to \tau$ и $M \in \sigma$, то $[\![F]\!] \in (D(\sigma) \to D(\tau))$ и $[\![M]\!] \in D(\sigma)$.
- λ-абстракция: если $M : \tau$, то $[\![\lambda x^{\sigma}.M]\!]$ — это такая функция $f : D(\sigma) \to D(\tau)$, что для каждой $a \in D(\sigma)$ имеем

$$f(a) = [\![M]\!]_{\theta^x_a},$$

где $\theta^x_a(y) = \theta(y)$ для $y \neq x$ и $\theta^x_a(x) = a$.

Теоретико-множественная семантика

- Переменные (свободные) интерпретируются произвольно в соответствующих $D(\tau)$. Интерпретация переменных обозначается θ.
- Применение функции: $[\![FM]\!] = [\![F]\!]([\![M]\!])$. Корректность: если $F : \sigma \to \tau$ и $M \in \sigma$, то $[\![F]\!] \in (D(\sigma) \to D(\tau))$ и $[\![M]\!] \in D(\sigma)$.
- λ-абстракция: если $M : \tau$, то $[\![\lambda x^{\sigma}.M]\!]$ — это такая функция $f : D(\sigma) \to D(\tau)$, что для каждой $a \in D(\sigma)$ имеем

$$f(a) = [\![M]\!]_{\theta_a^x},$$

  где $\theta_a^x(y) = \theta(y)$ для $y \neq x$ и $\theta_a^x(x) = a$.
- Корректность: для любого M, типизованного по Чёрчу и имеющего тип τ, имеем $[\![M]\!] \in D(\tau)$.

Теоретико-множественная семантика

- Если $M_1 \to_\beta M_2$ (или, шире, $M_1 =_\beta M_2$, т.е. M_1 и M_2 редуцируются к одному и тому же терму), то имеем $[\![M_1]\!] = [\![M_2]\!]$.

Теоретико-множественная семантика

- Если $M_1 \to_\beta M_2$ (или, шире, $M_1 =_\beta M_2$, т.е. M_1 и M_2 редуцируются к одному и тому же терму), то имеем $[\![M_1]\!] = [\![M_2]\!]$.
- Существуют, однако, β-равные термы с одинаковой интерпретацией: пусть $f : (\alpha \to \beta)$. Тогда $[\![f]\!] = [\![\lambda x.(f\,x)]\!]$, однако $f \neq_\beta \lambda x.(f\,x)$, т.к. оба терма $\in$ NF.

Теоретико-множественная семантика

- Если $M_1 \to_\beta M_2$ (или, шире, $M_1 =_\beta M_2$, т.е. M_1 и M_2 редуцируются к одному и тому же терму), то имеем $[\![M_1]\!] = [\![M_2]\!]$.
- Существуют, однако, β-равные термы с одинаковой интерпретацией: пусть $f : (\alpha \to \beta)$. Тогда $[\![f]\!] = [\![\lambda x.(fx)]\!]$, однако $f \neq_\beta \lambda x.(fx)$, т.к. оба терма $\in$ NF.
- Ситуация исправляется добавлением **η-редукции**:

$$\lambda x.(Fx) \to_\eta F, \text{ если } x \notin \mathrm{FVar}(F).$$

Теоретико-множественная семантика

- Если $M_1 \to_\beta M_2$ (или, шире, $M_1 =_\beta M_2$, т.е. M_1 и M_2 редуцируются к одному и тому же терму), то имеем $[\![M_1]\!] = [\![M_2]\!]$.
- Существуют, однако, β-равные термы с одинаковой интерпретацией: пусть $f : (\alpha \to \beta)$. Тогда $[\![f]\!] = [\![\lambda x.(fx)]\!]$, однако $f \neq_\beta \lambda x.(fx)$, т.к. оба терма $\in$ NF.
- Ситуация исправляется добавлением **η-редукции**:

$$\lambda x.(Fx) \to_\eta F, \text{ если } x \notin \mathrm{FVar}(F).$$

- η-редукция равносильна принципу **экстенсиональности**: если $Fx = Gx$, то $F = G$.

Теоретико-множественная семантика

- Если $M_1 \to_\beta M_2$ (или, шире, $M_1 =_\beta M_2$, т.е. M_1 и M_2 редуцируются к одному и тому же терму), то имеем $[\![M_1]\!] = [\![M_2]\!]$.
- Существуют, однако, β-равные термы с одинаковой интерпретацией: пусть $f : (\alpha \to \beta)$. Тогда $[\![f]\!] = [\![\lambda x.(fx)]\!]$, однако $f \neq_\beta \lambda x.(fx)$, т.к. оба терма $\in$ NF.
- Ситуация исправляется добавлением **η-редукции**:

$$\lambda x.(Fx) \to_\eta F, \text{ если } x \notin \mathrm{FVar}(F).$$

- η-редукция равносильна принципу **экстенсиональности**: если $Fx = Gx$, то $F = G$.

Теорема

$M_1 =_{\beta\eta} M_2$ тогда и только тогда, когда $[\![M_1]\!] = [\![M_2]\!]$ при любой интерпретации.

Соответствие Карри – Говарда

- Запишем правила типизации (по Карри) в виде логического исчисления:

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \qquad \frac{\Gamma \vdash F : (\sigma \to \tau) \quad \Gamma \vdash M : \sigma}{\Gamma \vdash (FM) : \tau} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : (\sigma \to \tau)}$$

Соответствие Карри – Говарда

- Запишем правила типизации (по Карри) в виде логического исчисления:

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \qquad \frac{\Gamma \vdash F : (\sigma \to \tau) \quad \Gamma \vdash M : \sigma}{\Gamma \vdash (FM) : \tau} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : (\sigma \to \tau)}$$

- Если убрать термы и оставить только типы, получатся следующие правила:

$$\frac{}{\Gamma, \tau \vdash \tau} \qquad \frac{\Gamma \vdash \sigma \to \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau} \qquad \frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \to \tau}$$

Соответствие Карри – Говарда

- Запишем правила типизации (по Карри) в виде логического исчисления:

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \qquad \frac{\Gamma \vdash F : (\sigma \to \tau) \quad \Gamma \vdash M : \sigma}{\Gamma \vdash (FM) : \tau} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : (\sigma \to \tau)}$$

- Если убрать термы и оставить только типы, получатся следующие правила:

$$\frac{}{\Gamma, \tau \vdash \tau} \qquad \frac{\Gamma \vdash \sigma \to \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau} \qquad \frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \to \tau}$$

- Это в точности правило modus ponens и теорема о дедукции.

Соответствие Карри – Говарда

- Запишем правила типизации (по Карри) в виде логического исчисления:

$$\frac{}{\Gamma, x : \tau \vdash x : \tau} \qquad \frac{\Gamma \vdash F : (\sigma \to \tau) \quad \Gamma \vdash M : \sigma}{\Gamma \vdash (FM) : \tau} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : (\sigma \to \tau)}$$

- Если убрать термы и оставить только типы, получатся следующие правила:

$$\frac{}{\Gamma, \tau \vdash \tau} \qquad \frac{\Gamma \vdash \sigma \to \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau} \qquad \frac{\Gamma, \sigma \vdash \tau}{\Gamma \vdash \sigma \to \tau}$$

- Это в точности правило modus ponens и теорема о дедукции.
- Таким образом, если понимать типы как пропозициональные формулы, мы получили некоторую логику высказываний (в импликативном фрагменте).

Соответствие Карри – Говарда

- В частности, если для некоторого M имеем $\vdash M : \tau$, т.е. тип τ населён термом без свободных переменных, что τ (как пропозициональная формула) — тавтология.

Соответствие Карри – Говарда

- В частности, если для некоторого M имеем $\vdash M : \tau$, т.е. тип τ населён термом без свободных переменных, что τ (как пропозициональная формула) — тавтология.
- Типы контекста играют роль гипотез: если $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$, то τ логически следует из σ_1, ..., σ_n.

Соответствие Карри – Говарда

- В частности, если для некоторого M имеем $\vdash M : \tau$, т.е. тип τ населён термом без свободных переменных, что τ (как пропозициональная формула) — тавтология.
- Типы контекста играют роль гипотез: если $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$, то τ логически следует из σ_1, ..., σ_n.
- Это следует из корректности теоретико-множественной интерпретации. А именно, будем следить только за (не)пустотой $D(\tau)$. «Пусто» — 0, «непусто» — 1.

Соответствие Карри – Говарда

- В частности, если для некоторого M имеем $\vdash M : \tau$, т.е. тип τ населён термом без свободных переменных, что τ (как пропозициональная формула) — тавтология.
- Типы контекста играют роль гипотез: если $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$, то τ логически следует из σ_1, ..., σ_n.
- Это следует из корректности теоретико-множественной интерпретации. А именно, будем следить только за (не)пустотой $D(\tau)$. «Пусто» — 0, «непусто» — 1.
- $D(\sigma) \to D(\tau)$ пусто тогда и только тогда, когда $D(\sigma) \neq \varnothing$ и $D(\tau) = \varnothing$ (т.е. интерпретация импликации совпадает с булевой).

Соответствие Карри – Говарда

- В частности, если для некоторого M имеем $\vdash M : \tau$, т.е. тип τ населён термом без свободных переменных, что τ (как пропозициональная формула) — тавтология.
- Типы контекста играют роль гипотез: если $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$, то τ логически следует из σ_1, ..., σ_n.
- Это следует из корректности теоретико-множественной интерпретации. А именно, будем следить только за (не)пустотой $D(\tau)$. «Пусто» — 0, «непусто» — 1.
- $D(\sigma) \to D(\tau)$ пусто тогда и только тогда, когда $D(\sigma) \neq \varnothing$ и $D(\tau) = \varnothing$ (т.е. интерпретация импликации совпадает с булевой).
- Для населённого типа $D(\tau) \neq \varnothing$.

Соответствие Карри – Говарда

- Полнота, однако, не имеет места.

Соответствие Карри – Говарда

- Полнота, однако, не имеет места.
- А именно, **закон Пирса** $((\alpha \to \beta) \to \alpha) \to \alpha$ является классической тавтологией, однако соответствующий тип не населён замкнутым λ-термом.

Соответствие Карри – Говарда

- Полнота, однако, не имеет места.
- А именно, **закон Пирса** $((\alpha \to \beta) \to \alpha) \to \alpha$ является классической тавтологией, однако соответствующий тип не населён замкнутым λ-термом.
- Последнее устанавливается с помощью нормализации: если существовал бы какой-то терм в этом типе, то существовал бы и терм в нормальной форме.

Соответствие Карри – Говарда

- Полнота, однако, не имеет места.
- А именно, **закон Пирса** $((\alpha \to \beta) \to \alpha) \to \alpha$ является классической тавтологией, однако соответствующий тип не населён замкнутым λ-термом.
- Последнее устанавливается с помощью нормализации: если существовал бы какой-то терм в этом типе, то существовал бы и терм в нормальной форме.
- Построенное исчисление соответствует более слабой логике, а именно импликативному фрагменту **интуиционистской** логики высказываний.

Соответствие Карри – Говарда

- Полнота, однако, не имеет места.
- А именно, **закон Пирса** $((\alpha \to \beta) \to \alpha) \to \alpha$ является классической тавтологией, однако соответствующий тип не населён замкнутым λ-термом.
- Последнее устанавливается с помощью нормализации: если существовал бы какой-то терм в этом типе, то существовал бы и терм в нормальной форме.
- Построенное исчисление соответствует более слабой логике, а именно импликативному фрагменту **интуиционистской** логики высказываний.
- Соответствие Карри – Говарда:
  формулы = типы, доказательства = λ-термы.

Соответствие Карри – Говарда

- Полнота, однако, не имеет места.
- А именно, **закон Пирса** $((\alpha \to \beta) \to \alpha) \to \alpha$ является классической тавтологией, однако соответствующий тип не населён замкнутым λ-термом.
- Последнее устанавливается с помощью нормализации: если существовал бы какой-то терм в этом типе, то существовал бы и терм в нормальной форме.
- Построенное исчисление соответствует более слабой логике, а именно импликативному фрагменту **интуиционистской** логики высказываний.
- Соответствие Карри – Говарда:
  формулы = типы, доказательства = λ-термы.
- Нормализация термов = нормализация доказательства.

Соответствие Карри – Говарда

- Базовое исчисление можно расширять, добавляя в него новые конструкции типов и соответствующие им новые конструкции λ-термов.

Соответствие Карри – Говарда

- Базовое исчисление можно расширять, добавляя в него новые конструкции типов и соответствующие им новые конструкции λ-термов.
- Таким образом добавляются остальные логические операции, кванторы (не только первого порядка, а по произвольным объектам) и т.д.

Соответствие Карри – Говарда

- Базовое исчисление можно расширять, добавляя в него новые конструкции типов и соответствующие им новые конструкции λ-термов.
- Таким образом добавляются остальные логические операции, кванторы (не только первого порядка, а по произвольным объектам) и т.д.
- Парадигма при этом не меняется: доказательство формулы записывается как λ-терм (в расширенном языке), а проверка правильности доказательства = проверка корректности типизации.

Соответствие Карри – Говарда

- Базовое исчисление можно расширять, добавляя в него новые конструкции типов и соответствующие им новые конструкции λ-термов.
- Таким образом добавляются остальные логические операции, кванторы (не только первого порядка, а по произвольным объектам) и т.д.
- Парадигма при этом не меняется: доказательство формулы записывается как λ-терм (в расширенном языке), а проверка правильности доказательства = проверка корректности типизации.
- Таким образом, на базе функциональных языков программирования строятся системы полуавтоматического построения и проверки доказательства, такие как Coq (поверх OCaml) или Agda (поверх Haskell).

Представимость вычислимых функций

- Вернёмся к вопросу о представимости функций $f : \mathbb{N} \to \mathbb{N}$ в λ-исчислении.

Представимость вычислимых функций

- Вернёмся к вопросу о представимости функций $f : \mathbb{N} \to \mathbb{N}$ в λ-исчислении.
- Напомним, что натуральное число n представляется **нумералом Чёрча** $\underline{n} = \lambda f x. \underbrace{f(f(\ldots(f}_{n \text{ раз}} x)\ldots))$.

Представимость вычислимых функций

- Вернёмся к вопросу о представимости функций $f : \mathbb{N} \to \mathbb{N}$ в λ-исчислении.
- Напомним, что натуральное число n представляется **нумералом Чёрча** $\underline{n} = \lambda f x. \underbrace{f(f(\ldots(f}_{n \text{ раз}} x)\ldots))$.
- Нумерал Чёрча типизируем: $\underline{n} : (\alpha \to \alpha) \to (\alpha \to \alpha)$.

Представимость вычислимых функций

- Вернёмся к вопросу о представимости функций $f : \mathbb{N} \to \mathbb{N}$ в λ-исчислении.
- Напомним, что натуральное число n представляется **нумералом Чёрча** $\underline{n} = \lambda f x. \underbrace{f(f(\ldots(f}_{n \text{ раз}} x)\ldots))$.
- Нумерал Чёрча типизируем: $\underline{n} : (\alpha \to \alpha) \to (\alpha \to \alpha)$.
- Введём обозначение $\mathrm{nat} = (\alpha \to \alpha) \to (\alpha \to \alpha)$.

Представимость вычислимых функций

- Вернёмся к вопросу о представимости функций $f : \mathbb{N} \to \mathbb{N}$ в λ-исчислении.
- Напомним, что натуральное число n представляется **нумералом Чёрча** $\underline{n} = \lambda f x. \underbrace{f(f(\ldots(f}_{n \text{ раз}} x)\ldots))$.
- Нумерал Чёрча типизируем: $\underline{n} : (\alpha \to \alpha) \to (\alpha \to \alpha)$.
- Введём обозначение $\mathrm{nat} = (\alpha \to \alpha) \to (\alpha \to \alpha)$.
  - Для всех нумералов, кроме $\underline{0} = \lambda f x.x$ и $\underline{1} = \lambda f x.(f x)$, nat является наиболее общим типом.

Представимость вычислимых функций

- Вернёмся к вопросу о представимости функций $f : \mathbb{N} \to \mathbb{N}$ в λ-исчислении.
- Напомним, что натуральное число n представляется **нумералом Чёрча** $\underline{n} = \lambda f x. \underbrace{f(f(\ldots(f}_{n \text{ раз}} x)\ldots))$.
- Нумерал Чёрча типизируем: $\underline{n} : (\alpha \to \alpha) \to (\alpha \to \alpha)$.
- Введём обозначение $\mathrm{nat} = (\alpha \to \alpha) \to (\alpha \to \alpha)$.
  - Для всех нумералов, кроме $\underline{0} = \lambda f x.x$ и $\underline{1} = \lambda f x.(f x)$, nat является наиболее общим типом.
- Соответственно, имеет смысл говорить о представимости вычислимой функции f термом F типа $\mathrm{nat} \to \mathrm{nat}$.

Представимость вычислимых функций

- Насколько велики вычислительные возможности типизованного λ-исчисления?

Представимость вычислимых функций

- Насколько велики вычислительные возможности типизованного λ-исчисления?
- Во-первых, за счёт сильной нормализуемости все представимые функции всюду определены (тотальны). Представить не тотальную функцию невозможно.

Представимость вычислимых функций

- Насколько велики вычислительные возможности типизованного λ-исчисления?
- Во-первых, за счёт сильной нормализуемости все представимые функции всюду определены (тотальны). Представить не тотальную функцию невозможно.
- Во-вторых, с помощью диагональной процедуры можно установить, что существует тотальная, но при этом не представимая вычислимая функция.

Представимость вычислимых функций

- Насколько велики вычислительные возможности типизованного λ-исчисления?
- Во-первых, за счёт сильной нормализуемости все представимые функции всюду определены (тотальны). Представить не тотальную функцию невозможно.
- Во-вторых, с помощью диагональной процедуры можно установить, что существует тотальная, но при этом не представимая вычислимая функция.
- К сожалению, возможности $\lambda_{\to}$ весьма скромны: а именно, в нём представимы только «обобщённые многочлены» — функции, которые строятся при помощи арифметических действий и сравнения с нулём (Schwichtenberg 1976).

Представимость вычислимых функций

- Почему не получается представить хотя бы все примитивно-рекурсивные функции?

Представимость вычислимых функций

- Почему не получается представить хотя бы все примитивно-рекурсивные функции?
- При кодировании ПРФ мы использовали функцию Step, задающую переход от $\langle i, f(i)\rangle$ к $\langle i+1, f(i+1)\rangle$.

Представимость вычислимых функций

- Почему не получается представить хотя бы все примитивно-рекурсивные функции?
- При кодировании ПРФ мы использовали функцию Step, задающую переход от $\langle i, f(i)\rangle$ к $\langle i+1, f(i+1)\rangle$.
- Напомним, что пара $\langle M, N\rangle$ кодируется термом $\lambda z.((zM)N)$, и если $M, N : \mathrm{nat}$, то $\langle M, N\rangle : (\mathrm{nat} \to (\mathrm{nat} \to \gamma)) \to \gamma$. Обозначим этот тип через nat^2.

Представимость вычислимых функций

- Почему не получается представить хотя бы все примитивно-рекурсивные функции?
- При кодировании ПРФ мы использовали функцию Step, задающую переход от $\langle i, f(i)\rangle$ к $\langle i+1, f(i+1)\rangle$.
- Напомним, что пара $\langle M, N\rangle$ кодируется термом $\lambda z.((zM)N)$, и если $M, N : \mathrm{nat}$, то $\langle M, N\rangle : (\mathrm{nat} \to (\mathrm{nat} \to \gamma)) \to \gamma$. Обозначим этот тип через nat^2.
- Далее, к функции Step исходный аргумент-нумерал $\underline{n}$ применялся как итератор: $\underline{n}\ \mathrm{Step}\ G_0$. Значит, $\underline{n}$ должен иметь тип $(\mathrm{nat}^2 \to \mathrm{nat}^2) \to (\mathrm{nat}^2 \to \mathrm{nat}^2)$.

Представимость вычислимых функций

- Почему не получается представить хотя бы все примитивно-рекурсивные функции?
- При кодировании ПРФ мы использовали функцию Step, задающую переход от $\langle i, f(i)\rangle$ к $\langle i+1, f(i+1)\rangle$.
- Напомним, что пара $\langle M, N\rangle$ кодируется термом $\lambda z.((zM)N)$, и если $M, N : \mathrm{nat}$, то $\langle M, N\rangle : (\mathrm{nat} \to (\mathrm{nat} \to \gamma)) \to \gamma$. Обозначим этот тип через nat^2.
- Далее, к функции Step исходный аргумент-нумерал $\underline{n}$ применялся как итератор: $\underline{n}\ \mathrm{Step}\ G_0$. Значит, $\underline{n}$ должен иметь тип $(\mathrm{nat}^2 \to \mathrm{nat}^2) \to (\mathrm{nat}^2 \to \mathrm{nat}^2)$.
- Результирующий же тип (тип нумерала $\underline{f(n)}$) должен быть просто $\mathrm{nat} = (\alpha \to \alpha) \to (\alpha \to \alpha)$.

Нехватка полиморфизма

- При построении более сложных функций (с помощью композиции и многократного применения примитивной рекурсии) эта проблема будет только нарастать.

Нехватка полиморфизма

- При построении более сложных функций (с помощью композиции и многократного применения примитивной рекурсии) эта проблема будет только нарастать.
- Таким образом, нам «не хватает полиморфизма» типа nat.

Нехватка полиморфизма

- При построении более сложных функций (с помощью композиции и многократного применения примитивной рекурсии) эта проблема будет только нарастать.
- Таким образом, нам «не хватает полиморфизма» типа nat.
- Хочется, чтобы нумерал $\underline{n}$ реализовывался в любом из типов вида $(\tau \to \tau) \to (\tau \to \tau)$, в зависимости от наших потребностей.

Нехватка полиморфизма

- При построении более сложных функций (с помощью композиции и многократного применения примитивной рекурсии) эта проблема будет только нарастать.
- Таким образом, нам «не хватает полиморфизма» типа nat.
- Хочется, чтобы нумерал $\underline{n}$ реализовывался в любом из типов вида $(\tau \to \tau) \to (\tau \to \tau)$, в зависимости от наших потребностей.
- При этом, в частности, внутри τ может использоваться сам исходный тип nat.

Нехватка полиморфизма

- При построении более сложных функций (с помощью композиции и многократного применения примитивной рекурсии) эта проблема будет только нарастать.
- Таким образом, нам «не хватает полиморфизма» типа nat.
- Хочется, чтобы нумерал $\underline{n}$ реализовывался в любом из типов вида $(\tau \to \tau) \to (\tau \to \tau)$, в зависимости от наших потребностей.
- При этом, в частности, внутри τ может использоваться сам исходный тип nat.
- Такой уровень полиморфизма доступен в **системе F**.

Система F

- Система типов второго порядка (обозначение: $\lambda 2$), или система F, допускает квантор всеобщности по типовым переменным.

Система F

- Система типов второго порядка (обозначение: $\lambda 2$), или система F, допускает квантор всеобщности по типовым переменным.
- Например, $\lambda x.x : \forall\alpha.(\alpha \to \alpha)$,
  $\underline{2} = \lambda f x.f(fx) : \forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))$.

Система F

- Система типов второго порядка (обозначение: $\lambda 2$), или система F, допускает квантор всеобщности по типовым переменным.
- Например, $\lambda x.x : \forall\alpha.(\alpha \to \alpha)$,
  $\underline{2} = \lambda f x.f(fx) : \forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))$.
- Тип $\forall\alpha.\tau$ — это тип **полиморфных** термов, которые имеют одновременно все типы вида $\tau[\alpha := \sigma]$.

Система F

- Система типов второго порядка (обозначение: $\lambda 2$), или система F, допускает квантор всеобщности по типовым переменным.
- Например, $\lambda x.x : \forall\alpha.(\alpha \to \alpha)$,
  $\underline{2} = \lambda f x.f(fx) : \forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))$.
- Тип $\forall\alpha.\tau$ — это тип **полиморфных** термов, которые имеют одновременно все типы вида $\tau[\alpha := \sigma]$.
- Возможности $\lambda 2$ больше, чем $\lambda_{\to}$. Например, $\lambda x.(xx)$ типизуем в $\lambda 2$:

$$\lambda x.(xx) : (\forall\alpha.(\alpha \to \alpha)) \to (\forall\alpha.(\alpha \to \alpha)).$$

Здесь тип $\sigma = \forall\alpha.(\alpha \to \alpha)$ может специфицироваться в любой тип вида $\tau \to \tau$, в т.ч. в тип $\sigma \to \sigma$.

Система F

- Система типов второго порядка (обозначение: $\lambda 2$), или система F, допускает квантор всеобщности по типовым переменным.
- Например, $\lambda x.x : \forall\alpha.(\alpha \to \alpha)$,
  $\underline{2} = \lambda f x.f(fx) : \forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))$.
- Тип $\forall\alpha.\tau$ — это тип **полиморфных** термов, которые имеют одновременно все типы вида $\tau[\alpha := \sigma]$.
- Возможности $\lambda 2$ больше, чем $\lambda_\to$. Например, $\lambda x.(xx)$ типизуем в $\lambda 2$:

  $$\lambda x.(xx) : (\forall\alpha.(\alpha \to \alpha)) \to (\forall\alpha.(\alpha \to \alpha)).$$

  Здесь тип $\sigma = \forall\alpha.(\alpha \to \alpha)$ может специфицироваться в любой тип вида $\tau \to \tau$, в т.ч. в тип $\sigma \to \sigma$.
- Терм $\Omega = (\lambda x.(xx))(\lambda x.(xx))$ нетипизируем даже в $\lambda 2$.

Система F

Зададим систему F с мягкой типизацией (по Карри):

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}$$

$$\frac{\Gamma \vdash F : (\sigma \to \tau) \quad \Gamma \vdash M : \sigma}{\Gamma \vdash (FM) : \tau} \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x.M) : (\sigma \to \tau)}$$

$$\frac{\Gamma \vdash M : \tau}{\Gamma \vdash M : (\forall \alpha.\tau)} \qquad \frac{\Gamma \vdash M : (\forall \alpha.\tau)}{\Gamma \vdash M : \tau[\alpha := \sigma]}$$

условие: $\alpha \notin \mathrm{FVar}(\Gamma)$ условие: подстановка корректна

Система F

- Терм $\lambda x.(xx)$ имеет в системе F две **независимые** типизации:

$$(\forall\alpha.(\alpha \to \alpha)) \to (\forall\alpha.(\alpha \to \alpha));$$
$$(\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))) \to (\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))).$$

Система F

- Терм $\lambda x.(xx)$ имеет в системе F две **независимые** типизации:

  $$(\forall\alpha.(\alpha \to \alpha)) \to (\forall\alpha.(\alpha \to \alpha));$$
  $$(\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))) \to (\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))).$$

- Во втором типе указано, что x имеет более специальный функциональный тип, за счёт чего можно получить более специальный результат.

Система F

- Терм $\lambda x.(xx)$ имеет в системе F две **независимые** типизации:

  $$(\forall\alpha.(\alpha \to \alpha)) \to (\forall\alpha.(\alpha \to \alpha));$$
  $$(\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))) \to (\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))).$$

- Во втором типе указано, что x имеет более специальный функциональный тип, за счёт чего можно получить более специальный результат.
- Из этих двух типов ни один не получается подстановкой из другого (вместо связанной переменной подставлять нельзя).

Система F

- Терм $\lambda x.(xx)$ имеет в системе F две **независимые** типизации:

$$(\forall\alpha.(\alpha \to \alpha)) \to (\forall\alpha.(\alpha \to \alpha));$$
$$(\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))) \to (\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))).$$

- Во втором типе указано, что x имеет более специальный функциональный тип, за счёт чего можно получить более специальный результат.
- Из этих двух типов ни один не получается подстановкой из другого (вместо связанной переменной подставлять нельзя).
- Следовательно, в системе F не всегда существует наиболее общий тип, что не позволяет выведение типов.

Система F

- Терм $\lambda x.(xx)$ имеет в системе F две **независимые** типизации:

$$(\forall\alpha.(\alpha \to \alpha)) \to (\forall\alpha.(\alpha \to \alpha));$$
$$(\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))) \to (\forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha))).$$

- Во втором типе указано, что x имеет более специальный функциональный тип, за счёт чего можно получить более специальный результат.
- Из этих двух типов ни один не получается подстановкой из другого (вместо связанной переменной подставлять нельзя).
- Следовательно, в системе F не всегда существует наиболее общий тип, что не позволяет выведение типов.
- Более того, задача проверки типизуемости в системе F неразрешима.

Натуральные числа и рекурсия в системе F

- Тип натуральных чисел в системе F — полиморфный:

$$\mathrm{Nat} = \forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha)).$$

Натуральные числа и рекурсия в системе F

- Тип натуральных чисел в системе F — полиморфный:

$$\mathrm{Nat} = \forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha)).$$

- При кодировании ПРФ функция перехода Step получает тип $\mathrm{Nat}^2 \to \mathrm{Nat}^2$, где $\mathrm{Nat}^2 = (\mathrm{Nat} \to (\mathrm{Nat} \to \gamma)) \to \gamma$.

Натуральные числа и рекурсия в системе F

- Тип натуральных чисел в системе F — полиморфный:

$$\mathrm{Nat} = \forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha)).$$

- При кодировании ПРФ функция перехода Step получает тип $\mathrm{Nat}^2 \to \mathrm{Nat}^2$, где $\mathrm{Nat}^2 = (\mathrm{Nat} \to (\mathrm{Nat} \to \gamma)) \to \gamma$.
- Тип Nat можно конкретизировать в $(\mathrm{Nat}^2 \to \mathrm{Nat}^2) \to (\mathrm{Nat}^2 \to \mathrm{Nat}^2)$, поэтому нумерал $\underline{n}$ можно применить как итератор к Step и $G_0 : \mathrm{Nat}^2$.

Натуральные числа и рекурсия в системе F

- Тип натуральных чисел в системе F — полиморфный:

$$\mathrm{Nat} = \forall\alpha.((\alpha \to \alpha) \to (\alpha \to \alpha)).$$

- При кодировании ПРФ функция перехода Step получает тип $\mathrm{Nat}^2 \to \mathrm{Nat}^2$, где $\mathrm{Nat}^2 = (\mathrm{Nat} \to (\mathrm{Nat} \to \gamma)) \to \gamma$.
- Тип Nat можно конкретизировать в $(\mathrm{Nat}^2 \to \mathrm{Nat}^2) \to (\mathrm{Nat}^2 \to \mathrm{Nat}^2)$, поэтому нумерал $\underline{n}$ можно применить как итератор к Step и $G_0 : \mathrm{Nat}^2$.
- Таким образом, в системе F представимы все примитивно-рекурсивные функции, а также функции, задаваемые рекурсией высших порядков (такие как функция Аккермана).

Сильная нормализуемость и доказуемая тотальность

Теорема

Все термы, типизуемые в системе F, сильно нормализуемы.

Сильная нормализуемость и доказуемая тотальность

Теорема

Все термы, типизуемые в системе F, сильно нормализуемы.

- Следовательно, все функции, представимые в системе F, тотальны.

Сильная нормализуемость и доказуемая тотальность

Теорема

Все термы, типизуемые в системе F, сильно нормализуемы.

- Следовательно, все функции, представимые в системе F, тотальны.
- Кроме того, из соображений диагонализации существуют тотальные вычислимые функции, не представимые в системе F.

Сильная нормализуемость и доказуемая тотальность

Теорема

Все термы, типизуемые в системе F, сильно нормализуемы.

- Следовательно, все функции, представимые в системе F, тотальны.
- Кроме того, из соображений диагонализации существуют тотальные вычислимые функции, не представимые в системе F.

Теорема

В системе F представимы в точности рекурсивные функции, доказуемо тотальные в арифметике второго порядка PA_2.

Сильная нормализуемость и доказуемая тотальность

Теорема

Все термы, типизуемые в системе F, сильно нормализуемы.

- Следовательно, все функции, представимые в системе F, тотальны.
- Кроме того, из соображений диагонализации существуют тотальные вычислимые функции, не представимые в системе F.

Теорема

В системе F представимы в точности рекурсивные функции, доказуемо тотальные в арифметике второго порядка PA_2.

- Утверждение о тотальности: $\forall n\, \exists m\, (F\, \underline{n} \twoheadrightarrow_\beta \underline{m})$ должно быть не только истинно, но и доказуемо в некоторой (достаточно богатой) теории.

Сильная нормализуемость и доказуемая тотальность

- Для конкретного терма F сильная нормализуемость термов $F\,\underline{n}$ оказывается доказуемой в PA_2, отсюда следует доказуемая тотальность представимой функции.

Сильная нормализуемость и доказуемая тотальность

- Для конкретного терма F сильная нормализуемость термов $F\,\underline{n}$ оказывается доказуемой в PA_2, отсюда следует доказуемая тотальность представимой функции.
- Общее утверждение о сильной нормализуемости для системы F, однако, в PA_2 недоказуемо.

Сильная нормализуемость и доказуемая тотальность

- Для конкретного терма F сильная нормализуемость термов $F\,\underline{n}$ оказывается доказуемой в PA_2, отсюда следует доказуемая тотальность представимой функции.
- Общее утверждение о сильной нормализуемости для системы F, однако, в PA_2 недоказуемо.
- В обратную сторону: доказуемая тотальность функции f конструктивизируется в доказуемую тотальность в гейтинговой (интуиционистской) арифметике второго порядка HA_2, откуда извлекается λ-терм.

Сильная нормализуемость и доказуемая тотальность

- Для конкретного терма F сильная нормализуемость термов $F\,\underline{n}$ оказывается доказуемой в PA_2, отсюда следует доказуемая тотальность представимой функции.
- Общее утверждение о сильной нормализуемости для системы F, однако, в PA_2 недоказуемо.
- В обратную сторону: доказуемая тотальность функции f конструктивизируется в доказуемую тотальность в гейтинговой (интуиционистской) арифметике второго порядка HA_2, откуда извлекается λ-терм.
- Таким образом, мы получили неполную по Тьюрингу, но достаточно мощную модель вычислений, в которой гарантирован останов вычисления.

Сильная нормализуемость и доказуемая тотальность

- Для конкретного терма F сильная нормализуемость термов $F\,\underline{n}$ оказывается доказуемой в PA_2, отсюда следует доказуемая тотальность представимой функции.
- Общее утверждение о сильной нормализуемости для системы F, однако, в PA_2 недоказуемо.
- В обратную сторону: доказуемая тотальность функции f конструктивизируется в доказуемую тотальность в гейтинговой (интуиционистской) арифметике второго порядка HA_2, откуда извлекается λ-терм.
- Таким образом, мы получили неполную по Тьюрингу, но достаточно мощную модель вычислений, в которой гарантирован останов вычисления.
- Доказуемой тотальности в арифметике Пеано первого порядка PA соответствует гёделева система T.

Система F в Haskell

- Типы системы F доступны, например, в языке Haskell, однако при этом нужно явно указывать типы (выведение типов не работает).

Система F в Haskell

- Типы системы F доступны, например, в языке Haskell, однако при этом нужно явно указывать типы (выведение типов не работает).
- Выведение типов доступно для небольшого фрагмента системы F, расширяющей $\lambda_{\to}$ — системы типов Хиндли–Милнера.

Система F в Haskell

- Типы системы F доступны, например, в языке Haskell, однако при этом нужно явно указывать типы (выведение типов не работает).
- Выведение типов доступно для небольшого фрагмента системы F, расширяющей $\lambda_{\to}$ — системы типов Хиндли–Милнера.
- Поскольку конструкции неподвижной точки не нормализуемы, Y-комбинатор $\mathbb{Y} = \lambda f.\big((\lambda x.f(xx)).(\lambda x.f(xx))\big)$ не типизуем в системе F.

Система F в Haskell

- Типы системы F доступны, например, в языке Haskell, однако при этом нужно явно указывать типы (выведение типов не работает).
- Выведение типов доступно для небольшого фрагмента системы F, расширяющей $\lambda_{\to}$ — системы типов Хиндли–Милнера.
- Поскольку конструкции неподвижной точки не нормализуемы, Y-комбинатор $\mathbb{Y} = \lambda f.\big((\lambda x.f(xx)).(\lambda x.f(xx))\big)$ не типизуем в системе F.
- Тем не менее, $\mathbb{Y}$ можно типизовать как «чёрный ящик» — добавить константу $\mathbb{Y} : \forall\alpha.((\alpha \to \alpha) \to \alpha)$ и редукцию $\mathbb{Y}F \to F(\mathbb{Y}F)$.

Система F в Haskell

- Типы системы F доступны, например, в языке Haskell, однако при этом нужно явно указывать типы (выведение типов не работает).
- Выведение типов доступно для небольшого фрагмента системы F, расширяющей $\lambda_{\to}$ — системы типов Хиндли–Милнера.
- Поскольку конструкции неподвижной точки не нормализуемы, Y-комбинатор $\mathbb{Y} = \lambda f.\big((\lambda x.f(xx)).(\lambda x.f(xx))\big)$ не типизуем в системе F.
- Тем не менее, $\mathbb{Y}$ можно типизовать как «чёрный ящик» — добавить константу $\mathbb{Y} : \forall\alpha.((\alpha \to \alpha) \to \alpha)$ и редукцию $\mathbb{Y}F \to F(\mathbb{Y}F)$.
- Добавление такой константы возвращает полноту по Тьюрингу, но ценой потери нормализуемости.

Система F в Haskell

- Типы системы F доступны, например, в языке Haskell, однако при этом нужно явно указывать типы (выведение типов не работает).
- Выведение типов доступно для небольшого фрагмента системы F, расширяющей $\lambda_{\to}$ — системы типов Хиндли–Милнера.
- Поскольку конструкции неподвижной точки не нормализуемы, Y-комбинатор $\mathbb{Y} = \lambda f.\big((\lambda x.f(xx)).(\lambda x.f(xx))\big)$ не типизуем в системе F.
- Тем не менее, $\mathbb{Y}$ можно типизовать как «чёрный ящик» — добавить константу $\mathbb{Y} : \forall\alpha.((\alpha \to \alpha) \to \alpha)$ и редукцию $\mathbb{Y}F \to F(\mathbb{Y}F)$.
- Добавление такой константы возвращает полноту по Тьюрингу, но ценой потери нормализуемости.
- Сильная нормализуемость неверна для любого подтерма вида $\mathbb{Y}F$, следовательно, нужно следить за ленивостью стратегии редукций.