Математические модели вычислений

Ещё немного про индуктивные типы

Типы идентичности (равенства)

Интенсиональное и экстенсиональное равенство

«Наблюдательная теория типов»

Гомотопическая теория типов и аксиома унивалентности

Высшие индуктивные типы

maxim.krivchikov@gmail.com

Материалы курса: https://maxxk.github.io/formal-models-2015/

«По следам наших публикаций»

1. $\#k|$ — тип из $k|$ элементов ($\#0, \#1, \#2|$ — типы ложных высказываний, истиных высказываний, булевских значений, соответственно)

2. Реализации λ-исчисления с зависимыми типами бывают и на C++: Lean (код, презентация, видео доклада)

Индексированные индуктивные типы

На индексированных индуктивных типах можно продемонстрировать отличия двух способов определения индуктивных типов (как набор конструкторов и как неподвижную точку функтора):

$$\mu_{\Pi\mathbb{N}.\,\mathbf{Type}} Even : \Pi\mathbb{N}.\,\mathbf{Type} \equiv \begin{array}{l} even_{zero} : Even(\mathbb{0}) \\ even_{plus2} : \Pi(k : \mathbb{N}).\, Even(k) \to Even(\mathbb{S}(\mathbb{S}(\mathbb{k}))) \end{array}\Bigg|$$

В первом случае структура типа $Even \cdot 2|$ от нас скрыта.

$$\begin{array}{r} Even' : \Pi\mathbb{N}.\,\mathbf{Type} \equiv \lambda(n : \mathbb{N}).\, ind_{\mathbb{N}}(\Pi\mathbb{N}.\,\mathbf{Type}, c_{\mathbb{0}} \equiv \lambda\mathbb{N}.\,\#1, \\ c_{\mathbb{S}} \equiv \lambda(k : \mathbb{N}).\,(step_1 : \Pi\mathbb{N}.\,\mathbf{Type}).\,\lambda(m : \mathbb{N}).\, ind_{\mathbb{N}}(\mathbf{Type}, c_{\mathbb{0}} \equiv \#0, \\ c_{\mathbb{S}} \equiv \lambda(k : \mathbb{N}).\,(step_2 : \mathbf{Type}).\, step_1 \cdot k, m), n) \cdot n \end{array}\Bigg|$$

Во втором случае для каждого конкретного числа $n|$ в канонической форме тип $Even'|$ *вычисляется* в #0 или #1.

Для некоторых индексированных типов такое преобразование из набора конструкторов в функтор затруднено (например, для типа на следующем слайде).

Индуктивные типы и сопоставление с образцом

В современных функциональных языках программирования часто используется конструкция сопоставления с образцом (pattern matching). Подробнее о конструкции в целом, наверное, в следующем семестре, пока что пример на Haskell:

```
interpret term context = case term of
    Var n   -> get context n
    Abs t   -> λn -> interpret t (add context n)
    App l r -> (interpret l context) (interpret r context)
```

Оператор простого удаления (итерации) для индуктивным типов, определённых как набор конструкторов, естественным образом представляется в виде сопоставления с образцом (образцы = конструкторы). Пример на Coq:

```
Fixpoint reverse lst := match lst with
| nil => nil
| cons head tail => (reverse tail) ++ head
end.
```

В простом случае это синтаксический сахар. Более сложные образцы не совсем повторяют структуру удаления индуктивного типа.

Индукция и сопоставление с образцом

Для зависимого удаления (индукции) простая схема сопоставления с образцом не работает. Пример — реализация функции «предыдущее натуральное число» с полной спецификацией в Coq:

```
pred : forall n : nat, n > 0 -> { m : nat | n = S m } :=
fun n : nat =>
match n as n0 return (n0 > 0 -> {m : nat | n0 = S m}) with
| 0 =>
    fun _H : 0 > 0 => False_rec ...
| S n' =>
    fun _H : S n' > 0 => exist (fun m : nat => S n' = S m) n' (... n _H)
end.
```

И соответствующий ему терм, непосредственно использующий индукцию:

$$
\begin{aligned}
&\text{pred} \equiv \text{ind\_}\mathbb{N}(T = \lambda(n : \mathbb{N}).\Pi(\text{n-gt-zero} : n > 0).\Sigma(m : \mathbb{N}).(n = m{+}1),\\
&\quad c0 : T(0) \equiv\\
&\quad\ \ \lambda\ \text{n-gt-zero}\ .\ \text{exfalso n-gt-zero}\ T(0,\ \text{n-gt-zero}),\\
&\quad cS : \Pi(k : \mathbb{N}).T(k) \to T(k{+}1) \equiv\\
&\quad\ \ \lambda(k : \mathbb{N})\ .\ (tk : T(k))\ .\ (\text{sk\_gt\_zero} : k{+}1 > 0).\\
&\quad\quad \text{pair}(k,\ \text{refl}(k{+}1))\\
&\quad)
\end{aligned}
$$

Пояснения:

$T(0) \longrightarrow (\text{n\_gt\_zero} : 0 > 0).\ \Sigma(m : \mathbb{N}).(0 = m{+}1)$

$T(k{+}1) \longrightarrow \Pi(\text{n\_gt\_zero} : k{+}1 > 0).\Sigma(m : \mathbb{N}).(k{+}1 = k{+}1)$

Тип идентичности (равенства)

Для произвольного типа A — индексированный индуктивный тип $Id : \Pi A.\, A.\, \mathbf{Type}$

$$Id(a, b) \equiv a = b$$

Единственный конструктор — рефлексивность: $\mathrm{refl}_A : \Pi(a : A).\, Id(a, a)$.

Терм зависимого удаления — конвертация по равенству:

$$J(a, b : A, \varepsilon : a =_A b, C : \Pi(x, y : A).\, (\delta : x =_A y).\, \mathbf{Type}, x : C(a, a, \mathrm{refl}_A(a))) : C(a, b, \varepsilon)$$

Пример: пусть $P : \mathbb{N} \to \mathbf{Type}$ — предикат, $p_2 : P(2)$ — доказательство предиката для 2, $x : \mathbb{N}$ — некоторое натуральное число и $\varepsilon : 2 =_{\mathbb{N}} x$ — элемент типа идентичности.

Допустим, нужно вызвать функцию $fun : \Pi(y : \mathbb{N}).\, P(y) \to \mathbb{N}$. С помощью удаления ε это можно сделать так:

$$fun(x, J(2, x, \varepsilon, \lambda(x, y : \mathbb{N}).\, (\delta : x =_{\mathbb{N}} y).\, P(y), p_2))$$

Далее будем использовать запись:

fun(x, p_2 : ε ⇝ P(**2**))

В позициях, набранных полужирным, левая часть равенства заменяется на правую.

Использование типов идентичности

1. Покажем, что $\varepsilon : 0{=}1$ — ложное высказывание:
   $\text{helper} \equiv \lambda\ (n : \mathbb{N})\ .\ \text{iter\_}\mathbb{N}(\mathbf{Type}, \#1, \#0, n)$
   $0\#1 : \text{helper} \cdot 0 \longrightarrow \#1$
   $\varepsilon \rightsquigarrow 0\#1 : \text{helper} \cdot \mathbf{0} \longrightarrow \#0$
   $(\lambda\ (\varepsilon : 0{=}1).\ \varepsilon \rightsquigarrow 0\#1 : \text{helper} \cdot \mathbf{0}) : \Pi(0{=}1).\#0$

2. Оптимизация на уровне типов:

Равенство по определению и пропозициональное равенство

Ранее мы рассматривали понятие эквивалентности термов для проверки типизации.

Термы эквивалентны, если они за конечное число редукций могут быть приведены к одному виду (с точностью до индексов уровней универсумов, которые не должны образовывать циклов с ребром «$<$»).

$Id(a, b)$ — пропозициональное равенство (утверждение о равенстве)

Для любых эквивалентных термов мы можем получить доказательство пропозиционального равенства с использованием терма refl: ($Id(a, b) \longrightarrow^* Id(a, a)$, если $b \longrightarrow^* a$)

Если мы имеем $\varepsilon : 2 =_{\mathbb{N}} x$, то это не значит, что мы можем подставлять x вместо 2 везде (можем только с использованием J-терма)

Из пропозиционального равенства нельзя получить равенство по определению.

Интенсиональное и экстенсиональное равенство

Интенсиональная (intensional) теория типов — типы равенства могут содержать какие-то неизвестные нам элементы, не описываемые $\mathrm{refl}_A(c)$.

Экстенсиональная (extensional) теория типов — типы равенства содержат только тривиальные элементы (любой элемент $Id_A(a, b)$ — это $\mathrm{refl}_A(c)$ для какого-то c). Способы наложения такого ограничения:

1. Ввести правило редукции (с ним получить из пропозиционального равенства — равенство по определению можно):
$$\frac{\varepsilon : Id_A(x, y)}{x \leqslant y}$$
— но тогда проверка типов становится неразрешима, т.к. алгоритм проверки должен уметь проверять, выводимо ли равенство для любых пар термов.
2. Ввести одну из дополнительных аксиом:
   - единственность доказательств равенства (uniqueness of identity proofs) — все элементы типа идентичности равны
   $$UIP : \Pi(A : \mathbf{Type}).\,(x, y : A).\,(\varepsilon, \delta : x =_A y).\,\varepsilon =_{x =_A y} \delta$$
   - аксиома K (Streicher)
   $$\Pi(A : \mathbf{Type}).\,(x : A).\,(P : \Pi(x =_A x).\,\mathbf{Type}).\,P(\mathrm{refl}_A(x)) \to \Pi(h : x =_A x).\,P(h)$$

Расширенный Pattern Matching в Agda тоже делает теорию экстенсиональной.

Интензиональная и экстенсиональная теория типов

Интензиональность vs экстензиональность — внутренняя структура vs внешние проявления

Для экстензиональной теории типов проверка типов неразрешима (если у нас любое пропозициональное равенство это то же самое, что и равенство по определению, то алгоритм проверки должен уметь доказывать все возможные равенства, что, очевидно, является неразрешимой задачей).

Есть отдельное понятие *функциональной экстензиональности* — аксиома экстензиональности только для функций:

$$\Pi(A, B : \mathbf{Type}).\,(f, g : \Pi A.\,B).\,(\Pi(a : A).\,f(a) =_{B(a)} g(a)).\,f =_{\Pi A.B} g$$

— функции равны, если они дают равные ответы для любых входных данных.

Наблюдательная теория типов

Observational Type Theory; McBride, Altenkirch 2006, 2007.

Основная идея: рассматривать не равенство, а приведение типов.
$[S{=}T\rangle$ — метод получения объектов типа S из типа T.

Определяются отдельные правила для преобразования стандартных конструкторов:

$$[\Pi A_1.\,B_1 = \Pi A_2.\,B_2\rangle|$$

$$[\Sigma A_1.\,B_1 = \Sigma A_2.\,B_2\rangle|$$

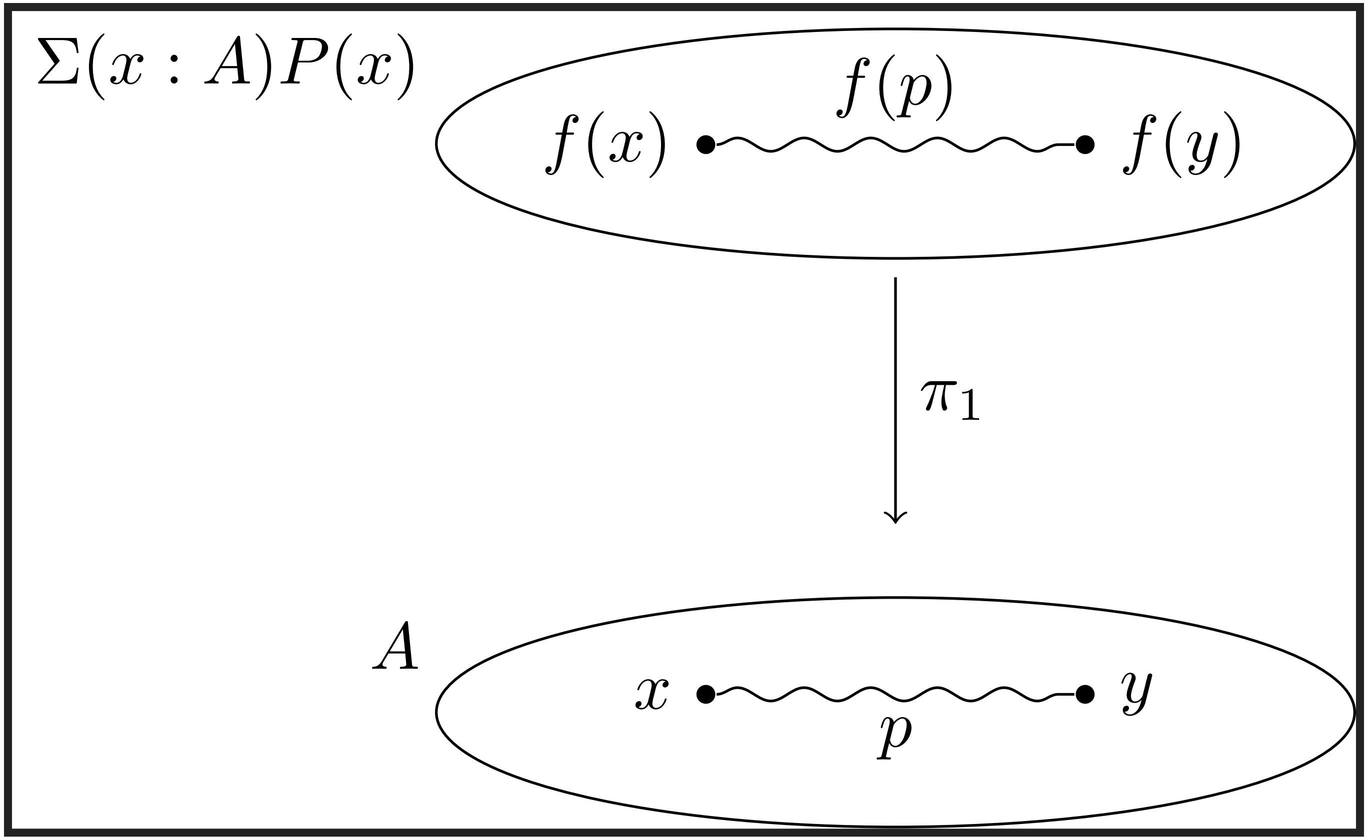
Недостаток — «удаление» такого типа идентичности менее выразительно, чем $J|$.

1.Altenkirch T., McBride C. Towards observational type theory. 2006.
2.Altenkirch T., McBride C., Swierstra W. Observational Equality, Now! // Proceedings of PLPV 2007. P. 57–68.

Гомотопическая теория типов

Awodey, Voevodsky; Homotopy Type Theory: The Univalent Foundations of Mathematics, 2013.

Теория типов интерпретируется в теории высших категорий. Если рассмотреть пример топологических пространств, типы идентичности — пути между двумя точками. Правило удаления J интерпретируется как перенос вдоль пути.



Гомотопическая теория типов

Типы идентичности имеют структуру группоида (определена частичная бинарная операция, которая на своей области определения удовлетворяет определению группы).
С помощью правила J можно показать, что типы идентичности удовлетворяют определению:

- операция — композиция (транзитивность) $\varepsilon : a = b, \delta : b = c \Longrightarrow \varepsilon \circ \delta : a = c$
- нейтральные элементы — рефлексивность $refl : a = a$
- обратный элемент $\varepsilon : a = b \longrightarrow \varepsilon^{-1} : b = a$
- ассоциативность, нейтральность, обратность.

Задача 9.1* Записать термы, доказывающие перечисленные выше свойства.

Аксиома унивалентности

Основной смысл, за исключением деталей, требуемых для корректности: равенство между типами можно получить с помощью биективного отображения между элементами типов. Т.е. равенство есть изоморфизм.

$$\mathit{univalence} : A\ B \to A =_{\mathbf{Type}} B$$

Введем следующие сокращения (по книге п. 4.2 – 4.3):

$$\mathrm{linv}(f : \Pi A.\, B) \equiv \Sigma(\Pi A.\, B).\, \Pi A.1 \cdot (f \cdot 0) =_A 0$$

$$\mathrm{rinv}(f : \Pi A.\, B) \equiv \Sigma(\Pi A.\, B).\, \Pi B.f \cdot (1 \cdot 0) =_B 0$$

$$\mathrm{biinv}(f : \Pi A.\, B) \equiv \Sigma \mathrm{linv}(f).\, \mathrm{rinv}(f)$$

Положим также $id_A : \Pi A.\, A \equiv \lambda A.0$.

Для типов **Type** эквивалентности высших размерностей рассматриваются следующим образом:

$$A =_{\mathbf{Type}} B \Leftrightarrow \Sigma(\Pi A.\, B).\, \mathrm{biinv}(0)$$

Вычислительное содержание аксиомы унивалентности

$$plus0 \equiv \lambda(x : \mathbb{N}).\, 0 + x =_{\Pi\mathbb{N}.\mathbb{N}} \lambda(x : \mathbb{N}).\, x + 0$$

$$J(\Pi\mathbb{N}.\,\mathbb{N}, \lambda x.0 + x, \lambda x.\, x + 0, \mathbb{N}, 0, plus0) \not\longrightarrow x$$

(переписывание формулы $0 + x$ по коммутативности при данном $x + 0 \to x$ не приводится к x).

Варианты решения:

1. Интерпретация в кубических множествах (cubical sets).
   Bezem M., Coquand T., Huber S. A model of type theory in cubical sets. TYPES 2013.

2. Расширение наблюдательной теории типов.
   Моя диссертация — раздел 2.2 :) http://istina.msu.ru/dissertations/10283583/

Высшие индуктивные типы

Можно определить типы индуктивно с конструкторами не только непосредственно элементов типа, но и элементов типа идентичности на нём.

Пример из топологии:
$Circle = base : Circle, loop : base =_{Circle} base$

Пример:

R : Type
doc : R
_↔_@_ : (s1 s2 : String) (i : Fin n) → (doc = doc)
indep : (i ≠ j) → (s ↔ t @ i) ∘ (u ↔ v @ j)
= (u ↔ v @ i) ∘ (s ↔ t @ j)
noop : s ↔ s @ i = refl

Задачи со звёздочкой

Решение задач должно быть представлено в виде термов разновидности исчисления конструкций (с выбранными вами обозначениями для стандартных конструкторов) с индуктивными типами, в том числе — со стандартным образом определёнными натуральными числами. Можно использовать Coq.

Задача 9.2.** Определить операцию умножения на натуральных числах и показать свойства 0 и 1 через равенство.

Пример на сложении:

---

plus : $\Pi(a, b : \mathbb{N}).\mathbb{N} \equiv \lambda(a, b : \mathbb{N}).\mathrm{iter}\mathbb{N}(\mathbb{N}, c0 \equiv b, cS \equiv \lambda(c : \mathbb{N}).(res : \mathbb{N}).res+1, a)$

plus_zero : $\Pi(a : \mathbb{N}).(plus(0, a) = a) \equiv \lambda\ a.\ \mathrm{refl}\ a$, т.к. $plus(0, a) \equiv \mathrm{iter}\mathbb{N}(\mathbb{N}, c0\equiv a, cS\equiv\ldots, 0) \longrightarrow c0 \longrightarrow a$

zero_plus : $\Pi(a : \mathbb{N}).(plus(a, 0) = a)$ — можно показать через симметричность равенства или по индукции по a (последнее можно сделать в качестве отдельной задачи на *)

(бонусная *) Показать коммутативность и ассоциативность сложения.

Задача 9.3*** Получить представление индуктивного типа — неподвижной точки функтора через натуральные числа и высшие индуктивные типы.