Anders: гомотопічна базова бібліотека

Максим Сохацький [1]

[1] Національний технічний університет України
«Київський Політехнічний Інститут» ім. Ігора Сікорського
29 жовтня 2018

Анотація

Тут представлена базова бібліотека мови Anders для курсу «Теорія типів», яка сумісна з позначеннями, що використовуються в підручнику HoTT. Серед принципів, які покладені в основу бібліотеки, головними є: лаконічность, академічність, педагогічність. Кожна сторінка має на меті повністю висвітлити компоненти типу, використовуючи тільки ті типи, що були викладені попередньо, кожне визначення повинно містити як математичну нотацію так і код верифікатора та бути вичерпним посібником користувача мови програмування Anders та її базової бібліотеки. Загалом передбачається, що бібліотека повинна відповідати підручнику HoTT, та бути його практичним дослідницьким артефактом.

Теорія типів

Теорія типів — це універсальна мова програмування чистої математики (для доведення теорем), яка може містити довільну кількість консистентних аксіом, впорядкованих у вигляді псевдо-ізоморфізмів: функцій encode (способи конструювання елементів типу) і decode (залежні елімінатори принципу індукції типу) та їх рівнянь — бета і ета правил обчислювальності та унікальності. Зазвичай теорія типів, як мова програмування, вже постачається з наступними типами (примітивами-аксіомами) та коментарями у вигляді окремих лекцій (конспекти, документація).

Головна мотивація гомотопічної теорії — надати обчислювальну семантику гомотопічним типам та CW-комплексам. Головна ідея гомотопічної теорії [1] полягає в поєднанні просторів функцій, просторів контекстів і просторів шляхів таким чином, що вони утворюють фібраційну рівність яка збігається (доводиться в самій теорії) з простором шляхів.

Завдяки відсутності ета-правила у рівності, не кожні два доведення одного простору шляхів дорівнюють між собою, отже простір шляхів утворює багатовимірну структуру інфініті-групоїда.

Групоїдна інтерпретація теорії типів ставить питання про існування мови, в якій можна довести механічно всі всластивості категорного визначення групоїда.

Основи

Модальні унівалентні MLTT основи розділені на три частини. Перша частина містить класичні типи MLTT системи описані Мартіном-Льофом. Друга частина містить унівалентні ідентифікаційні системи. Третя частина містить модальності, які використовуються в диференціальній геометріїї та в теорії гомотопій. Основи пропонують фундаментальний базис який використовується для формалізації сучасної математики в таких системах доведення теорем як: Coq, Agda, Lean.

- Фібраційні
- Унівалентні
- Модальні

Математики

Друга частина базової бібліотеки містить формалізації математичних теорій з різних галузей математики: аналіз, алгебра, геометрія, теорія гомотопій, теорія категорій.

Слухачам курсу (10) пропонується застосувати теорію типів для доведення початкового але нетривіального результу, який є відкритою проблемою в теорії типів для однєї із математик, що є курсами на кафедрі чистої математики (КМ-111):

- Функціональний аналіз
- Гомологічна алгебра
- Диференціальна геометрія
- Теорія гомотопій
- Теорія категорій

1 Простори функцій

П-тип — це простір, що містить залежні функції, кодомен яких залежить від значення з домену. Так як всі розшарування домену присутні повністю в кожній функції з простору, П-тип також називається залежним добутком, так як фунція визначена на всьому просторі домена.

Простори залежних функції використовуються в теорії типів для моделювання різних математичних конструкцій, об'єктів, типів, просторів, а також їхніх відображень: залежних функцій, неперервниї відображень, етальних відображень, розшарувань, квантора узанальнення $\forall$, імплікації, тощо.

1.1 Формація

Визначення 1.1 (П-формація, залежний добуток). П-типи репрезентують спосіб створення просторів залежних функцій $f : \Pi(x : A), B(x)$ в певному всесвіті U_i, з доменом в A і кодоменом в сім'ї функцій $B : A \to U_i$ над A.

$$\Pi : U =_{def} \prod_{x:A} B(x).$$

```
def Pi (A : U) (B : A → U) : U
 := Π (x : A), B x
```

1.2 Конструкція

Визначення 1.2 (λ-функція). Лямбда конструктор визначає нову лямбда функцію в просторі залежних функцій, вона ще називається лямбда абстракцією і позначається як $\lambda x.b(x)$ або $x \mapsto b(x)$.

$$\backslash(x : A) \to b : \Pi(A, B) =_{def}$$

$$\prod_{A:U} \prod_{B:A\to U} \prod_{a:A} \prod_{b:B(a)} \lambda x.b.$$

```
def lambda (A: U) (B: A → U) (b: Pi A B)
   : Pi A B := λ (x : A), b x
```

```
def lam (A B: U) (f: A → B)
   : A → B := λ (x : A), f x
```

Коли кодомен не залежить від значеення з домену функції $f : A \to B$ розглядаються в контексті System F_ω, залежний випадок розглядається в Systen P_ω або Calculus of Construction (CoC).

1.3 Елімінація

Визначення 1.3 (Принцип індукції). Якшо предикат виконується для лямбда функції тоді існує функція з простору функцій в простіп предикатів.

```
def Π—ind (A : U) (B : A → U) (C : Pi A B → U)
      (g: Π (x: Pi A B), C x)
   : Π (p: Pi A B), C p := λ (p: Pi A B), g p
```

Визначення 1.3.1 (λ-аплікація). Застосування функції до аргументів редукує терм використовуючи рекурсивну підстановку аргументів в тіло функції.

$$f\ a : B(a) =_{def} \prod_{A:U} \prod_{B:A\to U} \prod_{a:A} \prod_{f:\prod_{x:A} B(a)} f(a).$$

```
def apply (A: U) (B: A → U)
      (f: Pi A B) (a: A) : B a := f a

def app (A B: U) (f: A → B)
      (x: A): B := f x
```

Визначення 1.3.2 (Композиція функцій).

```
def ∘^T (x y z: U) : U
 := (y → z) → (x → y) → (x → z)

def ∘ (x y z : U) : ∘^T x y z
 := λ (g: x → z) (f: x → y) (a: x), g (f a)
```

1.4 Обчислювальність

Теорема 1.4 (Обчислювальність Π_β). β-правило показує, що композиція lam $\circ$ app може бути скорочена (fused).

$$f(a) =_{B(a)} (\lambda(x : A) \to f(a))(a).$$

```
def Π—β (A : U) (B : A → U) (a : A) (f : Pi A B)
   : Path (B a) (apply A B (lambda A B f) a) (f a)
   := idp (B a) (f a)
```

1.5 Унікальність

Теорема 1.5 (Унікальність Π_η). η-правило показує, що композиація app $\circ$ lam можу бути скоронеча (fused).

$$f =_{(x:A)\to B(a)} (\lambda(y : A) \to f(y)).$$

```
def Π—η (A : U) (B : A → U) (a : A) (f : Pi A B)
   : Path (Pi A B) f (λ (x : A), f x)
:= idp (Pi A B) f
```

2 Простори контекстів