Anders: гомотопічна бібліотека

Максим Сохацький[1]

[1] Інститут математики «Групоїд Інфініті»
26 листопада 2023

Анотація

Тут представлена базова гомотопічна 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.

Ключові слова: Теорія типів, формалізація математики

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: 1) сигнатури типу або формації; 2) функції encode, способи конструювання елементів типу або конструкція; 3) функції decode, залежні елімінатори принципу індукції типу або елімінація; 4) рівняння бета правила або обчислювальності; 5) рівняння ета правила або унікальності. Таке визначення було дано Мартіном-Льофом, від чого теорія типів носить його ім'я MLTT.

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.

Основи

Перша частина базової бібліотеки — модальні унівалентні MLTT основи, що розділені на три групи. Перша група містить класичні типи MLTT системи описані Мартіном-Льофом, які присутні у мовах **Per** та **Anders**. Друга група містить унівалентні ідентифікаційні системи мови **Anders**. Третя група містить модальності мови **Anders**, 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 **Anders** 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 однеї із математик, що є курсами на кафедрі чистої математики (КМ-111):

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

Програми

Третя частини базової бібліотеки, присутня у мовах **Per** та **Anders**, присвячена прикладам з промислового програмування в області автоматизації підприємств та інформаційних технологій, а саме для специфікації програмних інтерфейсів.

- Формалізація двонаправленого тракту
- Формалізація графічного веб інтерфейсу
- Формалізація бази даних з єдиним простором ключів
- Формалізація реляційної бази даних
- Формалізація системи управління процесами

Філософії

З сучасників формальною філософією в HoTT загалом займається Девід Корфілд, а формалізацією свідомості як окремий предмет вивчають Хенк Барендрехт та Горо Като. Формальна теорія природніх мов теж формалізується за допомогою MLTT, а основні теореми доводять в HoTT. В четвертій частині базової бібліотеки **Anders** наводяться приклади програм, які маніфестують висловлювання і теореми з формальної філософії про пустотність всіх феноменів та синтаксис, морфологію і семантику природньої української мови.

- Формалізація Мадг'яміки
- Формалізація української мови в кванторах

Структура верифікатора

На відміну від одноаксіоматичного верифікатора **Henk**, який містить тільки один індексований всесвіт U_i, рівність за визначенням для примітивів єдиного П-типу, та функцію верифікації τ = **type**, верифікатори **Per** і **Anders** містять додатково Σ-тип для контекстів та телескопів, більш деталізовану функцію типізації τ, та багато інших досніпових модулів, крім П-типу, але які теж підпорядковуються системі типів Мартіна-Льофа.

Космос $\mathbb{N}$-індексованих всесвітів ω

В теорії типів всі сигнатури всіх типів живуть в ієрархіях всесвітів індексованих натуральними числами. Множина таких ієрархій називається космосом. В імплементаціях $\mathbb{N}$ завжди реалізовано як Big Integer. Верифікатор **Anders** має космос, що складається з двох ієрархій всесвітів $\omega = \{\mathbf{V}_i, \mathbf{U}_i\}$.

Рівність $=_{def}$ з точністю до α-β конверсій

Рівність за визначенням (інтенсіональна) двох термів означає, що за допомогою серії альфа та бета перетворень можна довести що терми дорівнюють посимвольно (бінарно). Саме ця функція повинна бути імплементована для всіх типів у верифікаторі. Програми, які доводять рівність двох термів в теорії самого верифікатора за допомогою конструкторів рефлексивності називаються екстенсіональними рівностями. Якшо це відбувається для =-типу у всесвітах V_i — це називаються пропрозиційними рівностями, а якшо в інших ідентифікаційних системах у всесвітах U_i – називаються типовими рівностями. Інтерналізація інтенсіональної рівності за визначенням всередині теорії за допомогою П-типів називається рівністю Лейбніца і є виводимою у всіх фібраційних верифікаторах.

Функція верифікації τ

Головна функція верифікації розпадається на систему взаємозалежних функцій $\tau = \{\mathbf{infer}, \mathbf{app}, \mathbf{check}, \mathbf{act}, \mathbf{conv}, \mathbf{eval}\}$, які повинні бути імплементовані для кожного типу, вбудованого в верифікатор.

Контексти та телескопи Σ

В теорії типів контексти, як алгебраїчні послідовності які містять сигнатури, які теж у свою чергу складаються з послідовністей пар, що складаються з імені змінної та її типу, визначаються Σ-типами.

Досніпові модулі $\int$ вбудованих типів

Кожен досніповий модуль повинен бути представлений у вигляді п'яти синтаксичних примітивів: 1) формації; 2) конструкції; 3) елімінації; 4) обчислювальності; 5) унікальності. Ці примітиви повинні бути узгоджені в сенсі Мартіна-Льофа та представлені у цій статті, як документація на бібліотеку верифікатора, як у тому числі дає формальне визначення примітивам в конкретній теорії $\int = \{\Pi, \Sigma, =, \mathbf{W}, \mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{Path}, \mathbf{Glue}\}$.

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 $x \mapsto b(x)$.

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

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

```
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

Σ-тип — це простір, що містить залежні пари, де тип другого елемента залежить від значення першого елемента. Оскільки в кожній визначеній парі присутня лише одна точка домену волокна, — тип також є залежною сумою, де основа волокна є неперетинним об'єднанням.

Простори залежних пар використовуються в теорії типів для моделювання декартових добутків, неперетинних сум, розшарувань, векторних просторів, телескопів, лінз, контекстів, об'єктів, алгебр, квантору існування ∃, тощо.

2.1 Формація

Визначення 2.1 (Σ-формація, залежна сума). Тип залежної суми індексований типом A в сенсу кодобутку або диз'юнктивної суми, де тільки одне волокно кодомену $B(x)$ присутнє в парі.

$$\Sigma : U =_{def} \sum_{x:A} B(x).$$

```
def Sigma (A: U) (B: A → U) : U
 := Σ (x: A), B(x)
```

2.2 Конструкція

Визначення 2.2 (Залежна пара). Конструктор залежної пари — це спосіб визначення індексованої пари над типом A елементу кодобутку або диз'юнктивного об'єднання.

$$\mathbf{pair} : \Sigma(A, B) =_{def}$$

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

```
def pair (A: U) (B: A → U) (a: A) (b: B a)
  : Sigma A B := (a, b)
```

2.3 Елімінація

Визначення 2.3 (Проекції). Залежні проекції $pr_1 : \Sigma(A,B) \to A$ і $pr_2 : \Pi_{x:\Sigma(A,B)} B(pr_1(x))$ є деконструкторами пари.

$$\mathbf{pr}_1 : \prod_{A:U} \prod_{B:A\to U} \prod_{x:\Sigma(A,B)} A$$

$$=_{def} .1 =_{def} (a,b) \mapsto a.$$

$$\mathbf{pr}_2 : \prod_{A:U} \prod_{B:A\to U} \prod_{x:\Sigma(A,B)} B(x.1)$$

$$=_{def} .2 =_{def} (a,b) \mapsto b.$$

```
def pr₁ (A: U) (B: A → U) (x: Sigma A B) : A := x.1
def pr₂ (A: U) (B: A → U) (x: Sigma A B) : B (pr₁ A B x) := x.2
```

Якшо ви хочете доступитися до глибокого (>1) поля в сігма-типі — ви повинні використати серію елімінаторів .2, яка закінчується елімінатором .1.

Визначення 2.3.1 (Принцип індукції Σ). Каже, що предикат, який виконується для двох проекцій, він виконується також і для всього простору пар.

```
def Σ–ind (A : U) (B : A → U)
    (C : Π (s: Σ (x: A), B x), U)
    (g: Π (x: A) (y: B x), C (x,y))
    (p: Σ (x: A), B x) : C p := g p.1 p.2
```

2.4 Обчислювальність

Визначення 2.4 (Σ-обчислювальність).

```
def Σ–β₁ (A : U) (B : A → U) (a : A) (b : B a)
  : Path A a (pr₁ A B (a, b)) := idp A a

def Σ–β₂ (A : U) (B : A → U) (a : A) (b : B a)
  : Path (B a) b (pr₂ A B (a, b)) := idp (B a) b
```

2.5 Унікальність

Визначення 2.5 (Σ-унікальність).

```
def Σ–η (A : U) (B : A → U) (p : Sigma A B)
  : Path (Sigma A B) p (pr₁ A B p, pr₂ A B p)
 := idp (Sigma A B) p
```

3 Ідентифікаційні простори

=-тип — це індуктивна родина функій індексована елементами $x, y : A$, які містять доведення того факту, що ці елементи рівні між собою $x = y$.

3.1 Формація

Визначення 3.1 (=-формація, родина залежних функцій). Індуктивна родина $Id_V : A \to A \to V$ з доменом і кодоменом у всесвіті V представляє елементи, що містять доведення факту, що індексовані $x, y : A$ елменти рівні між собою.

$$= : V =_{def} \prod_{A:V} \prod_{x,y:A} \mathbf{Id}_V(A, x, y).$$

```
def IdV (A: V) (x y: A)
  : V := Id A x y
```

3.2 Конструкція

Визначення 3.2 (=-конструкція, рефлексивність). Елементи індуктивної родини інстанціюються єдиним конструктором який містить доведення, що елемент типу $x : A$ дорівнює сам собі.

$$\mathbf{ref}_V : x =_A x =_{def} \prod_{A:V} \prod_{x:A} \mathbf{ref}_A(x)$$

```
def Id-ref (A : V) (a: A)
  : Id A a a := ref a
```

3.3 Елімінація

Визначення 3.3 (=-елімінатор, J). Для індуктивної родини C і функції c існує функція f така, що $f(x, x, \mathbf{ref}_V(A, x)) = c(x)$.

$$\mathbf{ind}_{=_A} : (p : x =_A y) \to C(x, y, p) =_{def}$$

$$\prod_{A:V} \prod_{C:\Pi_{x,y:A} x=y \to V} \prod_{c:C(x,x,\mathbf{ref}(x))}$$

$$\lambda p.\mathbf{J}(A, C, x, c, y, p).$$

```
def =-ind (A : V)
    (C : Π (x y : A), Id A x y → V)
    (x y : A) (c : C x x (ref x))
    (p : Id A x y)
  : C x y p
 := ind_J A C x c y p
```

3.4 Обчислювальність

Визначення 3.4 (=-обчислювальність).

```
def J^S–β (A : V)
    (C : Π (a b : A), Id A a b → V)
    (a : A) (c : C a a (ref a))
  : Id (C a a (ref a)) (J^S A C a a c (ref a)) c
 := ref c
```

3.5 Унікальність

Визначення 3.5 (=-унікальність).

```
def UIP (A : V) (a b : A) (p q : Id A a b)
  : Id (Id A a b) p q
 := ref p
```

4 Індуктивні простори