

Немає Нічого

Anders: базова бібліотека

Максим Сохацький¹

¹ Національний технічний університет України
«Київський Політехнічний Інститут» ім. Ігора Сікорського
26 листопада 2020

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: лаконічність, 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 постає 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 властивості 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 геометрії 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 результату, який є відкритою проблемою в теорії типів для однієї із математик, що є курсами на кафедрі чистої математики (KM-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 функція визначена на всьому просторі домена.

Простори залежних функцій використовуються в теорії типів для моделювання різних математичних конструкцій, об'єктів, типів, просторів, а також їхніх відображень: залежних функцій, неперервних відображень, еталних відображень, розшарувань, квантора узагальнення \forall , імплікації, тощо.

1.1 Формация

Визначення 1.1 (П-формация, 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 $B : A \rightarrow U_i$ над A .

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

```
def Pi (A : U) (B : A → U) : U
:= Pi (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 $x \mapsto b(x)$.

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

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

```
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 значення з домену функції $f : A \rightarrow B$ розглядаються в контексті System F_ω , залежний випадок розглядається в System P_ω або Calculus of Construction (CoC).

1.3 Елімінація

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

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

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

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

```
def apply (A : U) (B : A → U) (f : Π 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 oT (x y z : U) : U
  := (y → z) → (x → y) → (x → z)

def o (x y z : U) : oT 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 $\text{lam} \circ \text{app}$ може бути скорочена (fused).

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

```
def Π-β (A : U) (B : A → U) (a : A) (f : Π 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 композиція $\text{app} \circ \text{lam}$ може бути скорочена (fused).

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

```
def Π-η (A : U) (B : A → U) (a : A) (f : Π A B)
  : Path (Π A B) f (λ (x : A), f x)
  := idp (Π 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 непересічним об'єднанням.

Простори залежних пар використовуються в теорії типів для моделювання декартових добутків, непересічних сум, розшарувань, векторних просторів, телескопів, лінз, контекстів, об'єктів, алгебр, квантору існування \exists , 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 кодомону $B(x)$ присутнє в парі.

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

```
def Sigma (A: U) (B: A → U) : U
:=  $\Sigma$  (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.

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

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

```
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 $pr_1 : \Sigma(A, B) \rightarrow A$ і $pr_2 : \Pi_{x:\Sigma(A, B)} B(pr_1(x))$ є деконструкторами пари.

$$\begin{aligned} \mathbf{pr}_1 &: \prod_{A:U} \prod_{B:A \rightarrow U} \prod_{x:\Sigma(A, B)} A \\ &=_{def} .1 =_{def} (a, b) \mapsto a. \\ \mathbf{pr}_2 &: \prod_{A:U} \prod_{B:A \rightarrow U} \prod_{x:\Sigma(A, B)} B(x.1) \\ &=_{def} .2 =_{def} (a, b) \mapsto b. \end{aligned}$$

```
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
```

Якщо ви хочете досягти 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