

УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ



Андрија Д. Урошевић

ИНТУИЦИОНИСТИЧКА ТЕОРИЈА
ТИПОВА КАО УВОД У ХОМОТОПНУ
ТЕОРИЈУ ТИПОВА

мастер рад

Београд, 2024.

Ментор:

др Сана СТОЈАНОВИЋ-ЂУРЂЕВИЋ, доцент
Универзитет у Београду, Математички факултет

Чланови комисије:

проф. др Филип МАРИЋ, редовни професор
Универзитет у Београду, Математички факултет

др Иван ЧУКИЋ, доцент
Универзитет у Београду, Математички факултет

Датум одбране: 29. фебруар 2024.

Мами, пати и геги

Наслов мастер рада: Интуиционистичка теорија типова као увод у хомотопну теорију типова

Резиме: Homotopy Type Theory/Univalent Foundations (HoTT/UF) is a revolutionary approach to the foundation of mathematics. Although it's revolutionary, HoTT/UF is very slowly gaining popularity among a broader circle of mathematicians and computer scientists. One of the reasons is that during formalization one requires both theoretical knowledge and proof-assistance skills. Acquiring those prerequisites is partially based on one's background. Mathematicians lack functional programming skills, on the other hand, computer scientists lack theoretical knowledge. A few materials tackle both areas, but they are lacking interactability. This thesis proposes a material that formalizes one theoretical area of HoTT/UF in Agda and is doing so while interacting with the user input.

Кључне речи: хомотопна теорија типова, интерактивно доказивање, агда

Садржај

1	Увод	2
1.1	Филозофија и историја	3
1.2	Циљ рада	3
2	Интуиционистичка теорија типова	5
2.1	Правила закључивања	6
2.2	Зависни типови	7
2.3	Типови зависних функција	8
2.4	Индуктивни типови	9
2.5	Искази као типови	17
2.6	Хијерархија универзума и универзум типови	19
2.7	Типови идентитети	19
3	Агда	28
4	Закључак	29

Глава 1

Увод

Хомотопна теорија типова (ХоТТ) (енгл. *Homotopy Type Theory*) је нова област математике која повезује многе друге области. Ослања се на *хомотопну теорију* и *теорију типова*. Хомотопна теорија је област настала из алгебарске топологије и хомолошке алгебра, са идејама више теорије категорија, док теорија типова има корене у математичкој логици и теоријском рачунарству. Сматра се да ХоТТ представља алтернативно заснивање математике, поступком формализације уз помоћу интерактивних доказивача. Програм заснивања математике у ХоТТ се назива *унивалентно заснивање* (енгл. *Univalent Foundations*) [**hottbook**].

Хомотопна теорија типова представља надоградњу *Мартин-Луф теорије типова* (МЛТТ) (енгл. *Martin-Löf Type Theory*) са *вишим индуктивним типовама* и *аксиомом унивалентности*. Виши индуктивни типови омогућавају логичко описивање основних простора и конструкција у хомотопној теорији (сфере, цилиндри, итд.). Са друге стране, аксиома унивалентности тврди да је једнакост еквивалентна еквиваленцији, тј. $(A = B) \simeq (A \simeq B)$.

Постоји пуно разлога за изучавање ХоТТ и заснивање математике кроз интерактивне доказиваче теорема, али један од главних је објављен у раду *The Origins and Motivations of Univalent Foundations* [**vlad14**]:

A technical argument by a trusted author, which is hard to check and looks similar to arguments known to be correct, is hardly ever checked in detail. (Владимир Веоводски, 2014.)

1.1 Филозофија и историја

Теорију типова је оригинално представио Берtrand Расел [rus08], решавајући парадок у заснивању математике тог времена. Након њега Алонзо Черч, развија *једно типизирани ламбда рачун* (ПРЛР) (енгл. *Simply Typed Lambda Calculus*) [crc40, crc41]. Теорију типова даље развија Пер Мартин-Луф коју данас знамо као МЛТТ, или *зависна/интуиционистичка/конструктивистичка теорија типова* [pml98, pml75, pml82, pml84].

1.2 Циљ рада

Формализација основних објеката и конструкција у МЛТТ у типски зависном програмском језику *Agda*.

- Хомотопна теорија типова = интуиционистичка теорија типова + високи индуктивни типови + аксиома унивалантности.
- Пер Мартин-Луф теорија типова се заснива на интуиционистичком програму који је настао по Брауверу.
- Математичко резтоновање је људска активност и математика је језик у коме се математичке идеје преносе.
- Фундаментална људска активност.
- Конструктивна теорија је *доказно релевантна*, тј. доказ је математички објекат као и сваки други.
- Тврђења можемо интерпретирати као типове, те ће доказ представљати *јроверу тип*, тј. конструисање терма одређеног типа. (Јако битна уврнута идеја)
- Запажање: Хомотопна теорија и теорија типова представљају исту ствар.
- Хомотопна теорија се бави непрекидним пресликавањима која су *хомотопна* између себе, тј. могу се "непрекидно деформисати" једна у друге.
- Тројство израчуњивости: Програмерска интерпретација, хомотопна интерпретација и логичка интерпретација.

- Типско расуђивање $t : T$ читамо као t је терм типа T или терм t настањује T . У програмерској интерпретацији тип представља тип, док терм неког типа представља израз тог типа. У хомотопној интерпретацији тип представља простор, док терм неког типа представља тачку у том простору.
- Пример јединичног типа $\mathbb{1}$: јединични (**unit** у програмерском смислу), јединствени (*The* у логичком смислу), и контрактибилни (у хомотопном смислу) тип.
- Интенционални и екстенционални типови? (нешто чуно, проучити)
- Раселов парадокс као мотивација за теорију типова.

Глава 2

Интуиционистичка теорија типова

Интуиционистичка теорија типова или Пер Мартин-Луф теорија типова је математичка теорија конструкција. Тип представља врсту конструкције. Елемент, терм или тачка представља резултат конструкције неког типа. Прецизније, елемент a типа A записујемо као $a : A$, и кажемо да елемент a настањује тип A . Битно је напоменути да терм не може да „живи самостално“ тј. терм увек мора да настањује неки тип.

Конструкција типова се састоји из низа дедуктивних *правила закључивања*. Правило закључивања записујемо као

$$\frac{\mathcal{H}_1 \quad \mathcal{H}_2 \quad \dots \quad \mathcal{H}_n}{\mathcal{C}}$$

где расуђивања $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_n$ називамо *премисе* или *хипотезе*, а расуђивање \mathcal{C} називамо *закључак*.

Дефиниција 2.0.1. Свако *расуђивање* је облика $\Gamma \vdash \mathcal{J}$, где је Γ *контекст* и \mathcal{J} *теза* расуђивања.

Дефиниција 2.0.2. *Контекст расуђивања* је коначна листа узајамно зависних променљивих декларисаних на следећи начин

$$x_1 : A_1, x_2 : A_2(x_1), \dots, x_n : A_n(x_1, \dots, x_{n-1}),$$

под условом да за свако $1 \leq k \leq n$ можемо да изведемо расуђивање

$$x_1 : A_1, x_2 : A_2(x_1), \dots, x_{k-1} : A_{k-1}(x_1, \dots, x_{k-2}) \vdash A_k(x_1, x_2, \dots, x_{k-1}).$$

Дефиниција 2.0.3. *Теза расуђивања* може имати четири врсте расуђивања и то су:

(i) A је (*добро-формиран*) *тип* у контексту Γ

$$\Gamma \vdash A \text{ type}$$

(ii) A и B су *расуђивачки једнаки типови* у контексту Γ

$$\Gamma \vdash A \equiv B \text{ type}$$

(iii) a је *елемент* типа A у контексту Γ

$$\Gamma \vdash a : A$$

(iv) a и b су *расуђивачки једнаки елементи* типа A у контексту Γ

$$\Gamma \vdash a \equiv_A b : A$$

2.1 Правила закључивања

Интуиционистичка теорија типова, као и други математички формализми, захтева скуп правила закључивања на којима ће се формализам заснивати. Та правила називамо *структурна правила*.

Пример структурних правила закључивања која описују да је расуђивачка једнакост релација еквиваленције:

$$\frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash A \equiv A \text{ type}} \quad \frac{\Gamma \vdash A \equiv A' \text{ type}}{\Gamma \vdash A' \equiv A \text{ type}} \quad \frac{\Gamma \vdash A \equiv A' \text{ type} \quad \Gamma \vdash A' \equiv A'' \text{ type}}{\Gamma \vdash A \equiv A'' \text{ type}}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash a \equiv_A a : A} \quad \frac{\Gamma \vdash a \equiv_A a' : A}{\Gamma \vdash a' \equiv_A a : A} \quad \frac{\Gamma \vdash a \equiv_A a' : A \quad \Gamma \vdash a' \equiv_A a'' : A}{\Gamma \vdash a \equiv_A a'' : A}$$

Исцрпна листа структурних правила закључивања у интуиционистичкој теорији типова се може наћи у [rijke2022intro]. **Da li sada ovo raspisivati?**

2.2 Зависни типови

Из дефиниције контекста можемо видети да неки типови могу зависити од неких термова. На пример, тип $A_2(x_1)$ зависи од терма $x_1 : A_1$, тј. за разне термове $x_1 : A_1$ имамо разне типове $A_2(x_1)$. Ову идеју можемо уопштити помоћу следећих дефиниција:

Дефиниција 2.2.1. Нека је тип A у контексту Γ . *Фамилија* типова над A у контексту Γ је тип $B(x)$ у контексту $\Gamma, x : A$, тј.

$$\Gamma, x : A \vdash B(x) \text{ type.}$$

Кажемо да је B фамилија типова над A у контексту Γ . Алтернативно, кажемо да је $B(x)$ тип индексиран са $x : A$ у контексту Γ .

Дефиниција 2.2.2. Нека је B фамилија типова над A у контексту Γ . *Секција* фамилије B над типом A у контексту Γ је елемент типа $B(x)$ у контексту $\Gamma, x : A$, тј.

$$\Gamma, x : A \vdash b(x) : B(x).$$

Кажемо да је b секција фамилије B над A у контексту Γ . Алтернативно, кажемо да је $b(x)$ елемент типа $B(x)$ индексиран са $x : A$ у контексту $\Gamma, x : A$.

Дефиниција 2.2.3. Нека је B фамилија типова над A у контексту Γ , и нека је $a : A$. Кажемо да је $B[a/x]$ *влакно* од B за параметар a , где $B[a/x]$ представља замену свих појављивања x у B са a . Нит од B за параметар a краће записујемо као $B(a)$.

Дефиниција 2.2.4. Нека је b секција фамилије типова B над A у контексту Γ . Кажемо да је $b[a/x]$ *вредност* од b за параметар a , где $b[a/x]$ представља замену свих појављивања x у b са a . Такође, вредност од b за параметар a краће записујемо као $b(a)$.

2.3 Типови зависних функција

У математици заснованој на теорији скупова функција $f : A \rightarrow B$ дефинисана је над одређеним доменом A и кодоменом B . У теорији типова то не мора да буде случај, тј. кодомен може зависити од елемента над којим се функција примењује. Прецизније, посматрајмо секцију b фамилије типова B над A у контексту Γ . Један начин је да b посматрамо као функцију $\text{mapstob}(x)$. Тада $b(x)$ наставује тип $B(x)$ који зависи од $x : A$. Због тога за разне елементе $x : A$ домена имамо разне кодомене, те има смисла говорити о типу *зависних функција* $\prod_{(x:A)} B(x)$.

Спецификација типа зависних функција $\prod_{(x:A)} B(x)$ је дата следећим правилима закључивања:

$$\begin{array}{c} \frac{[\prod\text{-form}]}{\Gamma, x : A \vdash B(x) \text{ type}} \quad \frac{[\prod\text{-intro}]}{\Gamma, x : A \vdash b(x) : B(x)} \quad \frac{[\prod\text{-elim}]}{\Gamma \vdash f : \prod_{(x:A)} B(x)} \\ \frac{[\prod\text{-comp}_1]}{\Gamma, x : A \vdash b(x) : B(x)} \quad \frac{[\prod\text{-comp}_2]}{\Gamma \vdash f : \prod_{(x:A)} B(x)} \\ \frac{}{\Gamma \vdash \prod_{(x:A)} B(x) \text{ type}} \quad \frac{}{\Gamma \vdash \lambda x. b(x) : \prod_{(x:A)} B(x)} \quad \frac{}{\Gamma, x : A \vdash f(x) : B(x)} \\ \frac{}{\Gamma \vdash (\lambda y. b(y))(x) \equiv b(x) : B(x)} \quad \frac{}{\Gamma \vdash \lambda x. f(x) \equiv f : \prod_{(x:A)} B(x)} \end{array}$$

Специјалан случај типа зависних функција је тип (уобичајених) *функција* $A \rightarrow B$. Уколико су типови A и B у контексту Γ , тј. тип B не зависи од елемената типа A , тада $\prod_{(x:A)} B$ представља тип (уобичајених) функција.

Дефиниција 2.3.1. Тип (уобичајених) *функција* $A \rightarrow B$ дефинишемо као:

$$A \rightarrow B := \prod_{(x:A)} B.$$

Ако је $f : A \rightarrow B$ функција, тада је A *домен*, а B *кодомен* функције f .

Дефиниција 2.3.2. За сваки тип A дефинишемо *функцију идентитета* $\text{id}_A : A \rightarrow A$ као $\text{id}_A := \lambda x. x$.

Дефиниција 2.3.3. За свака три типа A , B , и C дефинишемо *композицију* $\text{comp} : (B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C$ као $\text{comp} := \lambda g. \lambda f. \lambda g(f(x))$.

Може се показати да је композиција асоцијативна, као и да је функција идентитета неутрал за композицију функција. Због сагласности типова имамо леви неутрал id_B и десни неутрал id_A .

2.4 Индуктивни типови

Поред типова зависних функција постоји и класа *индуктивних типова*. Сваки индуктивни тип се дефинише помоћу следеће спецификације:

- (i) *Формирање* типа описује начин на који се дати тип формира.
- (ii) *Конструисање* описује на који начин се уводе нови канонични термови датог типа.
- (iii) *Индуктивни принцип* описује податке који су потребни да би се конструисала секција произвољне фамилије типова над датим типом.
- (iv) *Правила израчунавања* захтевају да се индуктивно дефинисана секција произвољне фамилије типова над датим типом слаже по конструкторима који уводе нове каноничне термове.

Обично се, поред ових спецификација, уводи и *правило рекурзије* које је специјални случај правила индукције. Код правила рекурзије не конструисемо секцију произвољне фамилије типова над датим типом, већ само константну фамилију над датим типом.

У наставку су наведене спецификације за уобичајене индуктивне типове: тип природних бројева \mathbb{N} , празни тип \emptyset , јединични тип $\mathbb{1}$, типови копроизвода $A + B$, тип зависних парова $\sum_{(x:A)} B(x)$, као и специјални случајеви ових типова. Поред њих, у засебном поглављу ће бити представљени типови идентитети $x =_A y$.

Тип природних бројева

Тип природних бројева \mathbb{N} представља тип кога настањују природни бројеви $0_{\mathbb{N}}, 1_{\mathbb{N}}, 2_{\mathbb{N}}, \dots$. Прецизније, тип природних бројева \mathbb{N} дефинишемо следећом спецификацијом:

$$\begin{array}{c}
 \frac{[\mathbb{N}\text{-form}]}{\vdash \mathbb{N} \text{ type}} \qquad \frac{[\mathbb{N}\text{-intro}_{0_{\mathbb{N}}}] }{\vdash 0_{\mathbb{N}} : \mathbb{N}} \qquad \frac{[\mathbb{N}\text{-intro}_{\text{succ}_{\mathbb{N}}}] }{\vdash \text{succ}_{\mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N}} \\
 \\
 \frac{
 \begin{array}{c}
 [\mathbb{N}\text{-ind}] \\
 \Gamma, n : \mathbb{N} \vdash P(n) \text{ type} \\
 \Gamma \vdash p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}}) \\
 \Gamma \vdash p_{\text{succ}_{\mathbb{N}}} : \prod_{(n:\mathbb{N})} P(n) \rightarrow P(\text{succ}_{\mathbb{N}}(n))
 \end{array}
 }{\Gamma \vdash \text{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\text{succ}_{\mathbb{N}}}) : \prod_{(n:\mathbb{N})} P(n)} \qquad
 \frac{
 \begin{array}{c}
 [\mathbb{N}\text{-comp}_{0_{\mathbb{N}}}^{\text{ind}_{\mathbb{N}}}] \\
 \Gamma, n : \mathbb{N} \vdash P(n) \text{ type} \\
 \Gamma \vdash p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}}) \\
 \Gamma \vdash p_{\text{succ}_{\mathbb{N}}} : \prod_{(n:\mathbb{N})} P(n) \rightarrow P(\text{succ}_{\mathbb{N}}(n))
 \end{array}
 }{\Gamma \vdash \text{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\text{succ}_{\mathbb{N}}}, 0_{\mathbb{N}}) \equiv p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}})} \\
 \\
 \frac{
 \begin{array}{c}
 [\mathbb{N}\text{-comp}_{\text{succ}_{\mathbb{N}}}^{\text{ind}_{\mathbb{N}}}] \\
 \Gamma, n : \mathbb{N} \vdash P(n) \text{ type} \\
 \Gamma \vdash p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}}) \\
 \Gamma \vdash p_{\text{succ}_{\mathbb{N}}} : \prod_{(n:\mathbb{N})} P(n) \rightarrow P(\text{succ}_{\mathbb{N}}(n))
 \end{array}
 }{\Gamma, n : \mathbb{N} \vdash \text{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\text{succ}_{\mathbb{N}}}, \text{succ}_{\mathbb{N}}(n)) \equiv p_{\text{succ}_{\mathbb{N}}}(n, \text{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\text{succ}_{\mathbb{N}}}, n)) : P(\text{succ}_{\mathbb{N}}(n))} \\
 \\
 \frac{
 \begin{array}{c}
 [\mathbb{N}\text{-rec}_{\mathbb{N}}] \\
 \Gamma \vdash A \text{ type} \\
 \Gamma \vdash a_{0_{\mathbb{N}}} : A \\
 \Gamma \vdash a_{\text{succ}_{\mathbb{N}}} : \mathbb{N} \rightarrow A \rightarrow A
 \end{array}
 }{\Gamma \vdash \text{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\text{succ}_{\mathbb{N}}}) : \mathbb{N} \rightarrow A} \qquad
 \frac{
 \begin{array}{c}
 [\mathbb{N}\text{-comp}_{0_{\mathbb{N}}}^{\text{rec}_{\mathbb{N}}}] \\
 \Gamma \vdash A \text{ type} \\
 \Gamma \vdash a_{0_{\mathbb{N}}} : A \\
 \Gamma \vdash a_{\text{succ}_{\mathbb{N}}} : \mathbb{N} \rightarrow A \rightarrow A
 \end{array}
 }{\Gamma \vdash \text{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\text{succ}_{\mathbb{N}}}, 0_{\mathbb{N}}) \equiv a_{0_{\mathbb{N}}} : A} \\
 \\
 \frac{
 \begin{array}{c}
 [\mathbb{N}\text{-comp}_{\text{succ}_{\mathbb{N}}}^{\text{rec}_{\mathbb{N}}}] \\
 \Gamma \vdash A \text{ type} \\
 \Gamma \vdash a_{0_{\mathbb{N}}} : A \\
 \Gamma \vdash a_{\text{succ}_{\mathbb{N}}} : \mathbb{N} \rightarrow A \rightarrow A
 \end{array}
 }{\Gamma, n : \mathbb{N} \vdash \text{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\text{succ}_{\mathbb{N}}}, \text{succ}_{\mathbb{N}}(n)) \equiv a_{\text{succ}_{\mathbb{N}}}(n, \text{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\text{succ}_{\mathbb{N}}}, n)) : A}
 \end{array}$$

По правилу $\mathbb{N}\text{-form}$, тип природних бројева \mathbb{N} може да се формира из празног контекста. Другим речима, постојање типа природних бројева \mathbb{N} не зависи од постојања других типова. Даље, имамо два конструктора помоћу којих конструишемо све каноничке термове типа \mathbb{N} . Први конструктор је константа $0_{\mathbb{N}} : \mathbb{N}$ и он говори да је $0_{\mathbb{N}}$ канонични терм типа \mathbb{N} . Други конструктор је функција $\text{succ}_{\mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N}$ и она говори да ће $\text{succ}_{\mathbb{N}}(n)$ бити канонични терм

типа \mathbb{N} ако је $n : \mathbb{N}$ канонични терм. Због тога су $0_{\mathbb{N}}, \text{succ}_{\mathbb{N}}(0_{\mathbb{N}}), \text{succ}_{\mathbb{N}}(\text{succ}_{\mathbb{N}}(0_{\mathbb{N}})), \dots$ канонични термови који настајују тип \mathbb{N} .

Правила формирања и конструкције нам говоре о томе под којим условима се може формирати тип, и како конструисати каноничне термове тог типа. Потребно је још дефинисати и начин на који се тип и елементи тог типа користе. Због тога се уводи индуктивно правило и правила израчунавања. Да би конструисали елемент $\text{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\text{succ}_{\mathbb{N}}}) : \prod_{(n:\mathbb{N})} P(n)$ потребно је конструисати елемент $p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}})$ (*база индукције*) и $p_{\text{succ}_{\mathbb{N}}} : \prod_{n:\mathbb{N}} P(n) \rightarrow P(\text{succ}_{\mathbb{N}}(n))$ (*индуктивни корак*). Даље, за сваки од конструктора треба увести правило израчунавања у складу са зависном функцијом $\text{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\text{succ}_{\mathbb{N}}}) : \prod_{(n:\mathbb{N})} P(n)$. Због тога имамо два правила израчунавања $\mathbb{N}\text{-comp}_{0_{\mathbb{N}}}$ и $\mathbb{N}\text{-comp}_{\text{succ}_{\mathbb{N}}}$.

Специјални случај индукције типа природних бројева је рекурзија типа природних бројева, у којој тип P не зависи од \mathbb{N} . Тада добијамо функцију $\text{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\text{succ}_{\mathbb{N}}}) : \mathbb{N} \rightarrow A$, под условом да имамо елементе $a_{0_{\mathbb{N}}} : A$ и $a_{\text{succ}_{\mathbb{N}}} : \mathbb{N} \rightarrow A \rightarrow A$.

Правило индукције, заједно са правилом рекурзије, омогућава дефинисање разних функција над природним бројевима. Да би дефинисали операцију сабирања природних бројева $+_{\mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ можемо искористити правило рекурзије, тј. функцију $\text{rec}_{\mathbb{N}} : A \rightarrow (\mathbb{N} \rightarrow A \rightarrow A) \rightarrow \mathbb{N} \rightarrow A$. За тип A узећемо \mathbb{N} . Због тога, сабирање природних бројева дефинишемо као:

$$m +_{\mathbb{N}} n \equiv \text{rec}_{\mathbb{N}}(m, \lambda n. \lambda r. \text{succ}_{\mathbb{N}}(r), n).$$

Заиста, за овако дефинисану операцију сабирања важи:

$$\begin{aligned} m +_{\mathbb{N}} 0_{\mathbb{N}} &\equiv m; \\ m +_{\mathbb{N}} \text{succ}_{\mathbb{N}}(n) &\equiv \text{succ}_{\mathbb{N}}(m +_{\mathbb{N}} n). \end{aligned}$$

Слично, множење природних бројева $\times_{\mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ можемо дефинисати као

$$m \times_{\mathbb{N}} n \equiv \text{rec}_{\mathbb{N}}(0_{\mathbb{N}}, \lambda n. \lambda r. m +_{\mathbb{N}} r, n).$$

Такође, за овако дефинисану операцију множења важи:

$$\begin{aligned} m \times_{\mathbb{N}} 0_{\mathbb{N}} &\equiv 0_{\mathbb{N}}; \\ m \times_{\mathbb{N}} \text{succ}_{\mathbb{N}}(n) &\equiv (m +_{\mathbb{N}} (m \times_{\mathbb{N}} n)). \end{aligned}$$

Можемо приметити шаблон између дефинисања операција преко рекурзивног правила и правила која захтевамо да важе по конструкторима. Наиме,

уколико желимо да дефинишемо функцију $f : \mathbb{N} \rightarrow A$ за коју важи:

$$\begin{aligned} f(0_{\mathbb{N}}) &\equiv \Phi_{0_{\mathbb{N}}}; \\ f(\text{succ}_{\mathbb{N}}(n)) &\equiv \Phi_{\text{succ}_{\mathbb{N}}}, \end{aligned}$$

где је $\Phi_{0_{\mathbb{N}}}$ израз типа A , и $\Phi_{\text{succ}_{\mathbb{N}}}$ израз типа A који може садржати n и $f(n)$. Тада функцију $f : \mathbb{N} \rightarrow A$ дефинишемо као:

$$f \equiv \text{rec}_{\mathbb{N}}(\Phi_{0_{\mathbb{N}}}, \lambda n. \lambda r. \Phi'_{\text{succ}_{\mathbb{N}}}),$$

где $\Phi'_{\text{succ}_{\mathbb{N}}}$ добијемо из $\Phi_{\text{succ}_{\mathbb{N}}}$ тако што сва појављивања $f(n)$ заменимо са r . Овај поступак дефинисања можемо уопштити и на индуктивно правило, и тада се он назива *ујаривање шаблона* (енгл. *pattern matching*).

Празни тип

Празни тип \emptyset је дегенерисани пример индуктивног типа кога не настањује ни један елемент. Прецизније, празни тип \emptyset дефинишемо следећом спецификацијом.

$$\begin{array}{lll} [\emptyset\text{-form}] & \frac{}{\vdash \emptyset \text{ type}} & [\emptyset\text{-ind}] \quad \frac{\Gamma, 0 \vdash P(x) \text{ type}}{\Gamma \vdash \text{ind}_{\emptyset} : \prod_{(x:\emptyset)} P(x)} \quad [\emptyset\text{-rec}] \quad \frac{\Gamma \vdash A \text{ type}}{\Gamma \vdash \text{rec}_{\emptyset} : \emptyset \rightarrow A} \end{array}$$

Како празан тип \emptyset не настањује ни један елемент, за њега не постоји ни један конструктор, и самим тим нема ни једно правило израчунавања. Може да се формира из празног контекста, а његово правило индукције тврди да за било коју фамилију типова P над \emptyset постоји елемент $\text{ind}_{\emptyset} : \prod_{(x:\emptyset)} P(x)$. Чешће се користи правило рекурзије које тврди да уколико конструишемо елемент $x : \emptyset$, онда можемо да конструишемо елемент $\text{rec}_{\emptyset}(x) : A$ било ког типа A . Правило рекурзије за празни тип \emptyset се обично назива и *уравило контрадикције* или *уравило противречности*.

Дефиниција 2.4.1. За сваки тип A дефинишемо тип *негације* од A као $\neg A := A \rightarrow \emptyset$. Поред тога, кажемо да је тип A *уразан* ако његову негацију настањује неки елемент, тј. $\text{empty}(A) := A \rightarrow \emptyset$.

Приметимо да је *двукратна негација* од A дефинисана као $\neg\neg A := (A \rightarrow \emptyset) \rightarrow \emptyset$. Због тога, не мора да важи $\neg\neg A \rightarrow A$, те није могуће изводити доказе контрадикцијом.

Јединични тип

Јединични тип $\mathbb{1}$ је индуктивни тип кога настањује само елемент \star . Прецизније, јединични тип $\mathbb{1}$ дефинишемо следећом спецификацијом.

$$\begin{array}{c}
 [\mathbb{1}\text{-form}] \quad \overline{\vdash \mathbb{1} \text{ type}} \qquad [\mathbb{1}\text{-intro}_\star] \quad \overline{\vdash \star : \mathbb{1}} \\
 \\
 [\mathbb{1}\text{-ind}] \quad \frac{\Gamma, x : \mathbb{1} \vdash P(x) \text{ type} \quad \Gamma \vdash p_\star : P(\star)}{\Gamma \vdash \text{ind}_{\mathbb{1}}(p_\star) : \prod_{(x:\mathbb{1})} P(x)} \qquad [\mathbb{1}\text{-comp}] \quad \frac{\Gamma, 1 \vdash P(x) \text{ type} \quad \Gamma \vdash p_\star : P(\star)}{\Gamma \vdash \text{ind}_{\mathbb{1}}(p_\star, \star) \equiv p_\star : P(\star)} \\
 \\
 [\mathbb{1}\text{-rec}] \quad \frac{\Gamma \vdash A \text{ type} \quad \Gamma \vdash a : A}{\Gamma \vdash \text{rec}_{\mathbb{1}}(a) : \mathbb{1} \rightarrow A}
 \end{array}$$

Јединични тип $\mathbb{1}$ може да се формира из празног контекста, а његово правило индукције тврди да за било коју фамилију типова P над $\mathbb{1}$ постоји елемент $\text{ind}_{\mathbb{1}}(p_\star) : \prod_{(x:\mathbb{1})} P(x)$ уколико постоји елемент $p_\star : P(\star)$. Како постоји само један конструктор $\star : \mathbb{1}$, имамо једно правило израчунавања које треба да се сложи са индуктивним правилом. Због тога, $\text{ind}_{\mathbb{1}}(p_\star, \star) \equiv p_\star : P(\star)$.

Специјални случај правила индукције типа $\mathbb{1}$ је правило рекурзије типа $\mathbb{1}$, које добијамо када фамилија типова P над $\mathbb{1}$ не зависи од $x : \mathbb{1}$. Тада за сваки елемент $a : A$ имамо функцију $\text{rec}_{\mathbb{1}}(a) : \mathbb{1} \rightarrow A$.

Дефиниција 2.4.2. За сваки тип A дефинишемо тип *јединствене функције* од A као $!A := A \rightarrow \mathbb{1}$. Специјално, јединствена функција од $\mathbb{0}$, тј. $\mathbb{0} \rightarrow \mathbb{1}$, се назива *вакумска функција*.

У хомотопној теорији типова за вакумску функцију важи да је јединствена.

Типови копроизвода

За типове A и B из контекста Γ можемо дефинисати тип копроизвода $A + B$ кога ће настањивати елементи или из типа A (ако $a : A$, онда $\text{inl}(a) : A + B$) или из типа B (ако $b : B$, онда $\text{inr}(b) : A + B$).

$$\begin{array}{c}
 \frac{\Gamma \vdash A \text{ type}, B \text{ type}}{\Gamma \vdash A + B \text{ type}} \quad \frac{[+ \text{-intro}_{\text{inl}}]}{\Gamma \vdash \text{inl} : A \rightarrow A + B} \quad \frac{[+ \text{-intro}_{\text{inr}}]}{\Gamma \vdash \text{inr} : B \rightarrow A + B} \\
 \\
 \begin{array}{c}
 \Gamma, z : A + B \vdash P(z) \text{ type} \\
 \Gamma \vdash p_{\text{inl}} : \prod_{(a:A)} P(\text{inl}(a)) \\
 \Gamma \vdash p_{\text{inr}} : \prod_{(b:B)} P(\text{inr}(b)) \\
 \hline
 \Gamma \vdash \text{ind}_+(p_{\text{inl}}, p_{\text{inr}}) : \prod_{(z:A+B)} P(z)
 \end{array} \\
 \\
 \begin{array}{c}
 \Gamma, z : A + B \vdash P(z) \text{ type} \\
 \Gamma \vdash p_{\text{inl}} : \prod_{(a:A)} P(\text{inl}(a)) \\
 \Gamma \vdash p_{\text{inr}} : \prod_{(b:B)} P(\text{inr}(b)) \\
 \hline
 \Gamma, a : A \vdash \text{ind}_+(p_{\text{inl}}, p_{\text{inr}}, \text{inl}(a)) \equiv p_{\text{inl}}(a) : P(\text{inl}(a)) \\
 \Gamma, b : B \vdash \text{ind}_+(p_{\text{inl}}, p_{\text{inr}}, \text{inr}(b)) \equiv p_{\text{inr}}(b) : P(\text{inr}(b))
 \end{array} \\
 \\
 \begin{array}{c}
 \Gamma \vdash \text{type} \\
 \Gamma \vdash f : A \rightarrow X \\
 \Gamma \vdash g : B \rightarrow X \\
 \hline
 \Gamma \vdash \text{rec}_+(f, g) : A + B \rightarrow X
 \end{array}
 \end{array}$$

Тип копроизвода $A + B$ због своје природе има два конструктора $\text{inl} : A \rightarrow A + B$ и $\text{inr} : B \rightarrow A + B$. Правило индукције тврди да за било коју фамилију типова P над $A + B$ постоји елемент $\text{ind}_+(p_{\text{inl}}, p_{\text{inr}}) : \prod_{(z:A+B)} P(z)$ уколико постоје елементи $p_{\text{inl}} : \prod_{(a:A)} P(\text{inl}(a))$ и $p_{\text{inr}} : \prod_{(b:B)} P(\text{inr}(b))$. Како постоје два конструктора, имамо два правила израчунавања која треба да се сложе са правилом индукције. Због тога $\text{ind}_+(p_{\text{inl}}, p_{\text{inr}}, \text{inl}(a)) \equiv p_{\text{inl}}(a) : P(\text{inl}(a))$ и $\text{ind}_+(p_{\text{inl}}, p_{\text{inr}}, \text{inr}(b)) \equiv p_{\text{inr}}(b) : P(\text{inr}(b))$.

Специјални случај правила индукције типа $A + B$ је правило рекурзије типа $A + B$, које добијамо када фамилија типова P над $A + B$ не зависи од $z : A + B$. Тада за сваку функцију $f : A \rightarrow X$ и за сваку функцију $g : B \rightarrow X$ имамо функцију $\text{rec}_+(f, g) : A + B \rightarrow X$. Из правила индукције, за свако $f : A \rightarrow X$ и за свако $g : B \rightarrow Y$, имамо функцију $f + g : A + B \rightarrow X + Y$.

Специјални случај типа копроизвода је *буловски тип* $2 := \mathbb{1} + \mathbb{1}$, чије једине елементе дефинишемо као $\text{true} := \text{inl}(\star)$ и $\text{false} := \text{inr}(\star)$. Из спецификације типа копроизвода можемо извући правило индукције и правило израчунавања, за буловски тип 2. Правило индукције 2-ind се назива и *if-then-else*.

$$\begin{array}{c} \Gamma, x : 2 \vdash P(x) \text{ type} \\ \text{[2-ind]} \quad \frac{\Gamma \vdash p_{\text{true}} : P(\text{true}) \quad \Gamma \vdash p_{\text{false}} : P(\text{false})}{\Gamma \vdash \text{ind}_2(p_{\text{true}}, p_{\text{false}}) : \prod_{(x:2)} P(x)} \end{array}$$

$$\begin{array}{c} \Gamma, x : 2 \vdash P(x) \text{ type} \\ \Gamma \vdash p_{\text{true}} : P(\text{true}) \\ \text{[2-comp]} \quad \Gamma \vdash p_{\text{false}} : P(\text{false}) \\ \hline \Gamma \vdash \text{ind}_2(p_{\text{true}}, p_{\text{false}}, \text{true}) \equiv p_{\text{true}} : P(\text{true}) \\ \Gamma \vdash \text{ind}_2(p_{\text{true}}, p_{\text{false}}, \text{false}) \equiv p_{\text{false}} : P(\text{true}) \end{array}$$

Типови зависних парова

Ако је B фамилија типова над A из контекста Γ , онда можемо формирати тип зависних парова $\sum_{(x:A)} B(x)$ кога ће настајивати *парови* $(x, y(x))$, где је $x : A$ и $y(x) : B(x)$. Прецизније, тип зависних парова $\sum_{(x:A)} B(x)$ дефинишемо следећом спецификацијом.

$$\begin{array}{c} \text{[}\sum\text{-form]} \\ \frac{\Gamma, x : A \vdash B(x) \text{ type}}{\Gamma \vdash \sum_{(x:A)} B(x) \text{ type}} \end{array} \quad \begin{array}{c} \text{[}\sum\text{-intro]} \\ \frac{\Gamma, x : A \vdash y(x) : B(x)}{\Gamma \vdash (x, y(x)) : \sum_{(x:A)} B(x)} \end{array}$$

$$\begin{array}{c} \Gamma, (x, y) : \sum_{(x:A)} B(x) \vdash P((x, y)) \text{ type} \\ \text{[}\sum\text{-ind]} \quad \frac{\Gamma \vdash f : \prod_{(x:A)} \prod_{(y:B(x))} P((x, y))}{\Gamma \vdash \text{ind}_{\sum}(f) : \prod_{(p:\sum_{(x:A)} B(x))} P(p)} \end{array}$$

$$\begin{array}{c} \Gamma, (x, y) : \sum_{(x:A)} B(x) \vdash P((x, y)) \text{ type} \\ \text{[}\sum\text{-comp]} \quad \frac{\Gamma \vdash f : \prod_{(x:A)} \prod_{(y:B(x))} P((x, y))}{\Gamma, (x, y) : \sum_{(x:A)} B(x) \vdash \text{ind}_{\sum}(f, (x, y)) \equiv f(x, y) : P((x, y))} \end{array}$$

Тип зависних парова $\sum_{(x:A)} B(x)$ има један конструктор помоћу кога се могу формирати елементи који га настајују, и то једноставним упаривањем елемената $x : A$ и $y(x) : B(x)$. Правило индукције тврди да за било коју фамилију типова P над $\sum_{(x:A)} B(x)$ постоји елемент $\text{ind}_{\sum}(f) : \prod_{p:\sum_{(x:A)} B(x)} P(p)$

уколико постоји елемент $f : \prod_{(x:A)} \prod_{(y:B(x))} P((x, y))$. Како постоји само један конструктор, имамо само једно правило израчунавања које треба да се сложи са правилом индукције. Због тога важи $\text{ind}_{\Sigma}(f, (x, y)) \equiv f(x, y) : P((x, y))$.

Правило индукције нам омогућава да дефинишемо функције у нставку.

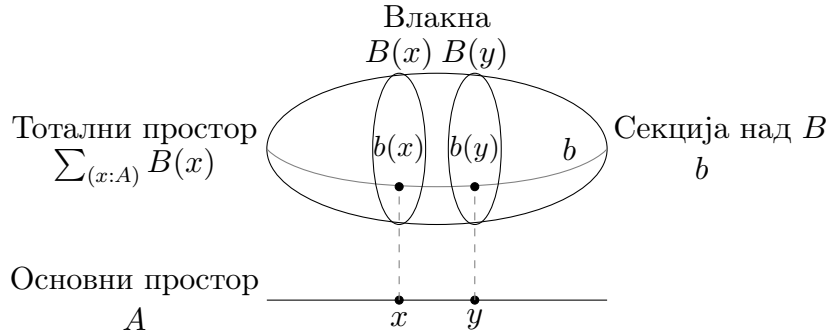
Дефиниција 2.4.3. Нека је B фамилија типова над A . Тада елемент $\text{pr}_1 : \sum_{(x:A)} B(x) \rightarrow A$ *пројекције на први елементи* дефинишемо као:

$$\text{pr}_1((a, b)) \equiv a, \quad (2.1)$$

а елемент $\text{pr}_2 : \prod_{p:\sum_{(x:A)} B(x)} B(\text{pr}_1(p))$ *пројекције на други елементи* дефинишемо као:

$$\text{pr}_2((a, b)) \equiv b. \quad (2.2)$$

Ако претпоставимо да имамо елемент $f : \prod_{((x,y):\sum_{(x:A)} B(x))} P((x, y))$ тада конструишемо елемент типа $\prod_{(x:A)} \prod_{(y:B(x))} P((x, y))$ као $\lambda x. \lambda y. f((x, y))$. Ова конструкција се назива *каријевање*, и како је супротна правилу Σ -ind, правило Σ -ind често наивамо *одкаријевање* (енгл. *uncurry*).



Слика 2.1: Геометријска репрезентација типа зависних парова.

Специјални случај типа зависних парова је тип (независних) *парова* или (Декартов) *производ* $A \times B$. Уколико су типови A и B у контексту Γ , тј. тип B не зависи од елемената типа A , тада $\sum_{(x:A)} B$ представља тип (независних) парова.

Дефиниција 2.4.4. Тип (независних) *парова* $A \times B$ дефинишемо као:

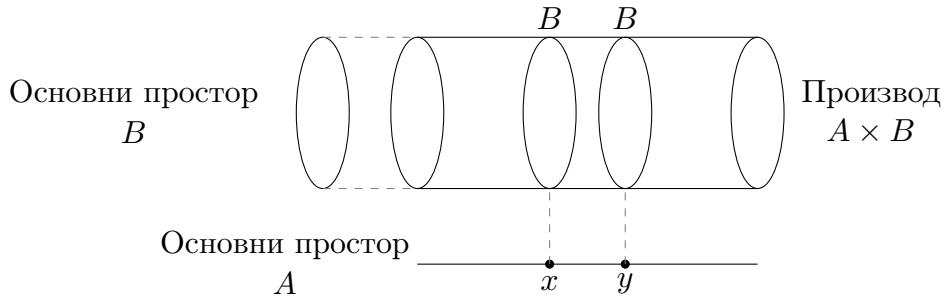
$$A \times B := \sum_{(x:A)} B.$$

Такође, *пројекцију на први елемент* $\text{fst} : A \times B \rightarrow A$ и *пројекцију на други елемент* $\text{snd} : A \times B \rightarrow B$ дефинишемо као:

$$\text{fst}((a, b)) \equiv a, \quad \text{snd}((a, b)) \equiv b.$$

Правило индукције и израчунавања за тип (независних) парова $A \times B$ директно добијамо из правила индукције и израчунавања за тип зависних парова $\sum_{(x:A)} B(x)$.

$$\begin{array}{c} \Gamma, (x, y) : A \times B \vdash P((x, y)) \text{ type} \\ [\times\text{-ind}] \quad \frac{\Gamma \vdash f : \prod_{(x:A)} \prod_{(y:B)} P((x, y))}{\Gamma \vdash \text{ind}_{\times}(f) : \prod_{(p:A \times B)} P(p)} \\ \\ \Gamma, (x, y) : A \times B \vdash P((x, y)) \text{ type} \\ [\times\text{-comp}] \quad \frac{\Gamma \vdash f : \prod_{(x:A)} \prod_{(y:B)} P((x, y))}{\Gamma, (x, y) : A \times B \vdash \text{ind}_{\times}(f, (x, y)) \equiv f(x, y) : P((x, y))} \end{array}$$



Слика 2.2: Геометријска репрезентација типа независних парова.

Тип независних парова можемо уопштити на тип *k-шорки* $A_1 \times A_2 \times \dots \times A_k$.

2.5 Искази као типови

Кари-Хавардова интерпретација неформално посматра исказе као типове, доказе као елементе типова, и предикате као фамилије типова. Да би показали да је исказ тачан у теорији типова треба конструисати елемент који настањује одговарајући тип. Прецизније, за дати исказ A (добро-формирани тип) уколико конструишемо елемент $x : A$ (кога често називамо и *сведок* за A) тада сматрамо да је исказ A тачан. Приметимо да исказ није тачан или

Искази	Типови
\perp	0
\top	1
$\neg A$	$A \rightarrow 0$
$A \implies B$	$A \rightarrow B$
$A \wedge B$	$A \times B$
$A \vee B$	$\ A + B\ $
$\forall x.P(x)$	$\prod_{(x:A)} P(x)$
$\exists x.P(x)$	$\ \sum_{(x:A)} P(x)\ $

Табела 2.1: Кари-Хавардова интерпретација

нетачан, већ да представља колекцију својих сведока који могу да потврде његову истинитост. Због тога су и сами докази математички објекти. У табели 2.1 приказани су искази заједно са њиховом одговарајућом интерпретацијом у теорији типова.

Прокоментаришимо неке интерпретације из табеле 2.1. Да би показали да важи $A \implies B$ треба претпоставити да важи A и доказати да важи B . У теорији типова треба конструисати елемент типа $A \rightarrow B$, тј. треба конструисати елемент типа B који користи претпоставку дату постојањем елемент типа A . Остали типови имају сличне интерпретације сем типа копроизвода $A + B$ и типа зависних парова $\sum_{(x:A)} B(x)$.

Да би показали $A \vee B$ треба показати да важи бар један од A и B . У теорији типова треба конструисати елемент типа $A + B$, помоћу једног од конструктора `inl` или `inr`. Због тога тип $A + B$, у односу на $A \vee B$, носи информацију о исказу који је тачан (тачно је или A или тачно је B). Слично, да би показали $\exists x.P(x)$ у теорији типова треба конструисати елемент типа $\sum_{(x:A)} P(x)$. У овом случају теорија типова нам даје и више од тога. Наиме, P је фамилија типова, што значи да $P(x)$ не мора да буде типа 2, тј. P не мора да буде предикат. Поред тога, тип $\sum_{(x:A)} P(x)$ можемо схватити као тип свих елемената $x : A$ за које $P(x)$.

Како ова два типа дају више информација у односу на традиционално значење исказа $A \vee B$ и $\exists x.P(x)$, користи се *окрњени искази* (енгл. *propositional truncation*) $\|A + B\|$ и $\|\sum_{(x:A)} B(x)\|$ који заборављају све информације о својим сведоцима сем да они постоје. Окрњени искази су ван опсега овог рада, тако да се неће детаљно описивати.

2.6 Хијерархија универзума и универзум типови

Универзум *типови* се могу посматрати као типови које настањују други типови. Универзум тип \mathcal{U} омогућава да се исказ „ A type” запише формално као $A : \mathcal{U}$. Поред тога, омогућава да се фамилија типова B над типом A дефинише као функција $B : A \rightarrow \mathcal{U}$.

Желимо да типови који могу да се формирају из празног контекста настањују универзум \mathcal{U} (то су, на пример, 0 , 1 , и \mathbb{N}). Штавише, како универзум \mathcal{U} настањују и други типови, желимо да универзум \mathcal{U} буде затворен по свим конструкторима који користе типове универзума \mathcal{U} . На пример, ако $A : \mathcal{U}$ и $B : A \rightarrow \mathcal{U}$, онда $\prod_{(x:A)} B(x) : \mathcal{U}$. Међутим, не сме доћи то тога да универзум настањује сам себе, тј. не сме да важи $\mathcal{U} : \mathcal{U}$. Другим речима, не смемо обезбедити услове настанка раселовог парадокса.

У многим случајевима довољно је постојање једног универзума \mathcal{U} , међутим, некада желимо да универзум настањује неки други универзум. Како би избегли Раселов парадокс захтевамо постојање *хијерархије универзума*

$$\mathcal{U}_0, \mathcal{U}_1, \mathcal{U}_2, \dots \quad (2.3)$$

за коју важе следећа правила:

$$[\mathcal{U}\text{-intro}] \quad \frac{}{\Gamma \vdash \mathcal{U}_i : \mathcal{U}_{i+1}} \quad [\mathcal{U}\text{-cumul}] \quad \frac{\Gamma \vdash A : \mathcal{U}_i}{\Gamma \vdash A : \mathcal{U}_{i+1}}$$

Универзум \mathcal{U}_0 називамо *базни универзум*. Базни универзум настањују типови који могу да се формирају из празног контекста, као и сви типови чији конструктори користе типове који се већ налазе у базном универзуму. За универзум \mathcal{U}_i има смисла посматрати и \mathcal{U}_{i+1} кога називамо и *универзум следбеник*. Често није битно знати редни број универзума у хијерархији, те се следбеник универзума \mathcal{U} обележава са \mathcal{U}^+ . За два универзума \mathcal{U} и \mathcal{V} можемо дефинисати њихову најмању горњу границу $\mathcal{U} \sqcup \mathcal{V}$. На пример, за \mathcal{U}_0 и \mathcal{U}_1 , најмања горња граница $\mathcal{U}_0 \sqcup \mathcal{U}_1$ је \mathcal{U}_1 .

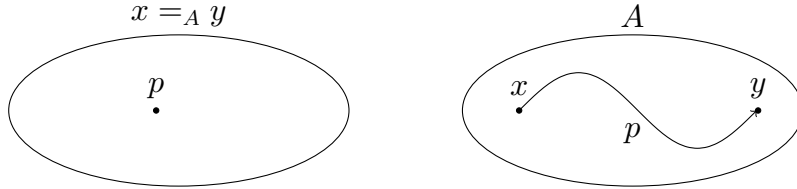
2.7 Типови идентитети

Подсетимо се да из дефиниције операције $+$ на \mathbb{N} важи $m +_{\mathbb{N}} 0_{\mathbb{N}} \equiv m$. Природно се намеће питање: Да ли важи $0_{\mathbb{N}} +_{\mathbb{N}} m \equiv m$? Јасно је да одговор на ово

питање треба да буде позитиван, али то није случај у интуиционистичкој теорији типова. Тиме долазимо до фундаменталног проблема интуиционистичке теорије типова: *Шта значи да су елементи неког типа једнаки?*

Како расуђивачка једнакост не може описати све врсте једнакости, потребно је дефинисати *исказну једнакост* (енгл. *propositional equality*) која тврди да ће два елемента $x, y : A$ бити исказно једнака. Исказна једнакост је исказ, и по Кари–Хавардовој интерпретацији представља неки тип, а како зависи од два елемента типа A мора бити фамилија типова. Исказне једнакости другачије називамо *tipovi identiteti* (енгл. *identity types*), и обележавамо као $\text{Id}_A : A \rightarrow A \rightarrow \mathcal{U}$. За два конкретна елемента $x, y : A$, $\text{Id}_A(x, y)$ обележавамо и као $x =_A y$ и кажемо да су x и y *једнаки* или *исказно једнаки*.

У хомотопној теорији типова, уколико интерпретирамо тип као простор, и елементе типа као тачке тог простора, онда елементе типа $x =_A y$ можемо интерпретирати као *путовање* или *еквиваленције* између тачака x и y у простору A . Као што је могуће да између две тачке у простору постоји више различитих путања, тако је могуће да постоји више од једног сведока једнакости $x =_A y$. Другим речима, $x =_A y$ се може сматрати као тип *идентификација* елемената x и y , и може постојати више начина на који x и y могу да се *идентификују*.



Слика 2.3: Геометријска репрезентација типова идентитета.

Ако је A тип и ако су дати елементи $x, y : A$ у контекста Γ , онда можемо формирати тип идентитета $x =_A y$ кога ће настањивати путање, еквиваленције или идентификације. Основна идентификација коју можемо да конструисемо је *рефлексива*

$$\text{refl}_x : x =_A x$$

која тврди да је било који елемент $x : A$ једнак самом себи. Рефлексиву refl_x , у хомотопном смислу, можемо посматрати као константном путањом у тачки

$x : A$. Формално, начин формирања и конструисања типова идентитета дат је следећом спецификацијом.

$$\frac{\Gamma \vdash A \text{ type} \quad \frac{[-\text{form}]}{\Gamma \vdash x : A} \quad \Gamma \vdash y : A}{\Gamma \vdash x =_A y \text{ type}} \quad \frac{\Gamma \vdash A \text{ type} \quad \frac{[-\text{intro}]}{\Gamma \vdash x : A}}{\Gamma \vdash \text{refl}_x : x =_A x}$$

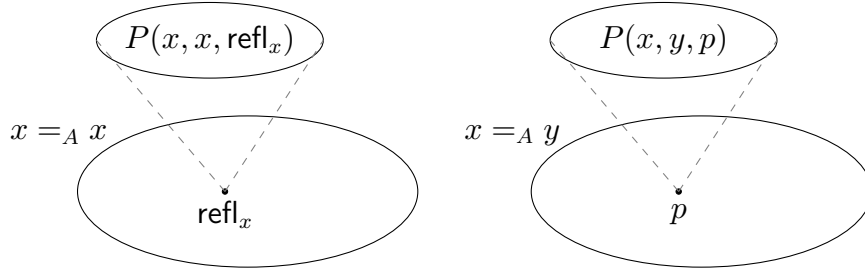
Уколико су два елемента $x, y : A$ расуђивачки једнака, тј. важи $x \equiv_A y$, онда су и исказно једнака и важи $\text{refl}_x : x =_A y$. Ово је добро засновано како је $\text{refl}_x : (x =_A x) \equiv (x =_A y)$ јер важи $x \equiv_A y$.

Индукција путање

Правило индукције за типове идентитета називамо *индукција путање*. Индукција путање тврди да за било коју фамилију типова P над типовима A и $x =_A y$ постоји функција $\text{ind}_= : \prod_{(x,y:A)} \prod_{(p:x=_A y)} P(x, y, p)$ уколико постоји функција $f : \prod_{(x:A)} P(x, x, \text{refl}_x)$. Како постоји само један конструктор $\text{refl}_x : x =_A x$, постоји само једно правило израчунавања које треба да се сложи са правилом индукције. Због тога правило израчунавања тврди $\text{ind}_=(x, x, \text{refl}_x) \equiv f(x) : P(x, x, \text{refl}_x)$. Формално, правило индукције и правило израчунавања је дато следећом спецификацијом.

$$\frac{\Gamma, x : A, y : A, p : x =_A y \vdash P(x, y, p) \text{ type} \quad \frac{[-\text{ind}]}{\Gamma \vdash f : \prod_{(x:A)} P(x, x, \text{refl}_x)}}{\Gamma \vdash \text{ind}_= : \prod_{(x,y:A)} \prod_{(p:x=_A y)} P(x, y, p)} \\ \frac{[-\text{comp}]}{\Gamma, x : A \vdash \text{ind}_=(x, x, \text{refl}_x) \equiv f(x) : P(x, x, \text{refl}_x)}$$

Једно од кључних питања је шта оправдава индукцију путањом? Другим речима, зашто ће $P(x, y, p)$ важити за било које тачке $x, y : A$ и било коју путању $p : x =_A y$ уколико важи $P(x, x, \text{refl}_x)$ за било коју тачку $x : A$? Кључно запажање лежу и томе да типови идентитета нису индуктивни тип, већ да су индуктивна фамилија типова. То значи да индукција путањом тврди да је фамилија типова $x =_A y$, где су x и y слободне тачке простора A , индуктивно дефинисана константном путањом refl_x . Односно, $\sum_{(x,y:A)} (x =_A y)$ је индуктивно генерисан константним путањама у свакој тачки $x : A$. Битно је напоменути да су обе тачке слободне (може само једна бити фиксирана, али не и обе) јер то доводи до доказа о јединствености идентификација.



Слика 2.4: Геометријска репрезентација индукције путањом.

Особине типова идентитета

Лема 1. Нека је A тип у контексту Γ . Тада можемо конструисати функцију

$$\text{inv}_A : \prod_{(x,y:A)} (x =_A y) \rightarrow (y =_A x)$$

индукцијом путање $p : x =_A y$ као $\text{inv}_A(x, x, \text{refl}_x) \equiv \text{refl}_x$. Функцију inv_A називамо инверз путањи. Чесћо, за даћу путању $p : x =_A y$, њен инверз означавамо са $p^{-1} \equiv \text{inv}_A(x, y, p)$.

Доказ. Да би конструисали елемент типа $\prod_{(x,y:A)} (x =_A y) \rightarrow (y =_A x)$, конструисамо функцију

$$f(x) : \prod_{(y:A)} (x =_A y) \rightarrow (y =_A x)$$

за било који елемент $x : A$. По индукцији путање $p : x =_A y$ довољно је конструисати путању

$$f(x, x, \text{refl}_x) : x =_A x$$

за било који елемент $x : A$. Конструкција ове путање је тривијална и због тога узимамо да је $f(x, x, \text{refl}_x) \equiv \text{refl}_x$. Коначно, имамо да је

$$\text{inv}_A(x, x, \text{refl}_x) \equiv \text{refl}_x$$

.

□

Лема 2. Нека је A тип у контексту Γ . Тада можемо конструисати функцију

$$\text{conc}_A : \prod_{(x,y,z:A)} (x =_A y) \rightarrow (y =_A z) \rightarrow (x =_A z)$$

индукцијом њућање $p : x =_A y$ као $\text{conc}_A(x, x, z, \text{refl}_x, q) := q$. Функцију conc_A називамо надовезивање путања. Чесшо, за даље њућање $p : x =_A y$ и $q : y =_A z$, надовезану њућању ојначавамо са $p \cdot q := \text{conc}_A(x, y, z, p, q)$.

Доказ. Прво конструишемо функцију

$$f(x) : \prod_{(y:A)} (x =_A y) \rightarrow \prod_{(z:A)} (y =_A z) \rightarrow (x =_A z)$$

за било који елемент $x : A$. По индукцији путање $p : (x =_A y)$ довољно је конструисати функцију

$$f(x, x, \text{refl}_x) : \prod_{(z:A)} (x =_A z) \rightarrow (x =_A z)$$

за било који елемент $x : A$. Даље, довољно је конструисати функцију

$$f(x, x, \text{refl}_x, z) : (x =_A z) \rightarrow (x =_A z)$$

за било које елементе $x, z : A$. Конструисање ове функције је тривијална и због тога имамо да је $f(x, x, \text{refl}_x, z, q) := q$. Коначно, имамо да је

$$\text{conc}_A(x, x, z, \text{refl}_x, q) := f(x, x, \text{refl}_x, z, q) := q.$$

□

Лема 3. Нека је A шии, нека су елементи $x, y, z, w : A$ и нека су њућање $p : x =_A y$, $q : y =_A z$ и $r : z =_A w$ у контексту Γ . Тада важи:

$$(i) \text{ refl}_x \cdot p = p \text{ и } p \cdot \text{refl}_y = p$$

$$(ii) p^{-1} \cdot p = \text{refl}_y \text{ и } p \cdot p^{-1} = \text{refl}_x$$

$$(iii) (p^{-1})^{-1} = p$$

$$(iv) (p \cdot q) \cdot r = p \cdot (q \cdot r)$$

(i) *Доказ.* Желимо да конструишемо путању

$$\text{unit}_l(p) : \text{refl}_x \cdot p = p,$$

$$\text{unit}_r(p) : p \cdot \text{refl}_y = p.$$

Индукцијом по путањи $p : x =_A y$ довољно је конструисати

$$\text{unit}_l(\text{refl}_x) : \text{refl}_x \cdot \text{refl}_x = \text{refl}_x,$$

$$\text{unit}_r(\text{refl}_x) : \text{refl}_x \cdot \text{refl}_x = \text{refl}_x.$$

Обе путање је тривијално конструисати као $\text{refl}_{\text{refl}_x}$. \square

(ii) *Доказ.* Желимо да конструишемо путању

$$\text{inv}_l(p) : p^{-1} \cdot p = \text{refl}_y,$$

$$\text{inv}_r(p) : p \cdot p^{-1} = \text{refl}_x.$$

Индукцијом по путањи $p : x =_A y$ довољно је конструисати путању

$$\text{inv}_l(\text{refl}_x) : \text{refl}_x^{-1} \cdot \text{refl}_x = \text{refl}_x,$$

$$\text{inv}_r(\text{refl}_x) : \text{refl}_x \cdot \text{refl}_x^{-1} = \text{refl}_x.$$

Али како је $\text{refl}_x^{-1} \equiv \text{refl}_x$ претходне путање се свде на оне као и у претходном доказу. Због тога обе путање тривијално конструишемо као $\text{refl}_{\text{refl}_x}$. \square

(iii) *Доказ.* Желимо да конструишемо путању

$$\text{doubleInv}(p) : (p^{-1})^{-1} = p.$$

Индукцијом по путањи $p : x =_A y$ довољно је конструисати путању

$$\text{doubleInv}(\text{refl}_x) : (\text{refl}_x^{-1})^{-1} = \text{refl}_x.$$

Али како је $(\text{refl}_x^{-1})^{-1} \equiv \text{refl}_x^{-1} \equiv \text{refl}_x$ претходна путања се своди на $\text{refl}_x = \text{refl}_x$. Због тога путању тривијално конструишемо као $\text{refl}_{\text{refl}_x}$. \square

(iv) *Доказ.* Желимо да конструишемо путању

$$\text{assoc}_A(p, q, r) : (p \cdot q) \cdot r = p \cdot (q \cdot r).$$

Индукцијом по путањи $p : x =_A y$ довољно је конструисати путању

$$\text{assoc}_A(\text{refl}_x, q, r) : (\text{refl}_x \cdot q) \cdot r = \text{refl}_x \cdot (q \cdot r)$$

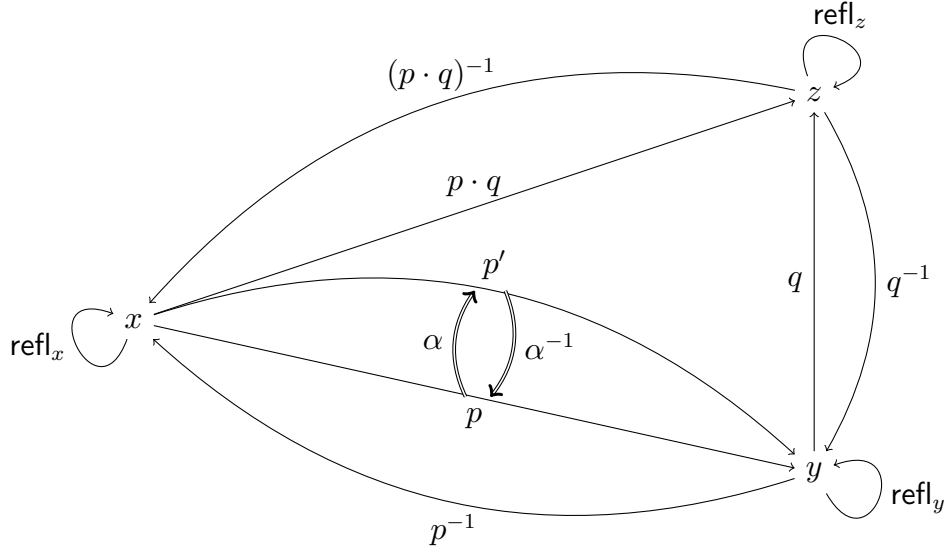
Али како је $\text{refl}_x \cdot q \equiv q$ и $\text{refl}_x \cdot (q \cdot r) \equiv q \cdot r$ претходна путања се своди на

$$\text{assoc}_A(\text{refl}_x, q, r) : q \cdot r = q \cdot r.$$

Због тога путању тривијално конструишемо као $\text{assoc}_A(\text{refl}_x, q, r) \equiv \text{refl}_{q \cdot r}$. \square

Једнакости	Хомотопија	∞ -Групоид
рефлексивност	константна путања	идентички морфизам
симетричност	обртање путања	инверз морфизма
транзитивност	надовезивање путања	композиција морфизама

Табела 2.2: Разне интерпретације особина типова идентитета



Слика 2.5: Групоидална структура типова.

Акције над путањама

Лема 4. Нека су A и B типови, и нека је $f : A \rightarrow B$ функција у контексту Γ . Тада можемо конструисати функцију

$$\text{ap}_f : \prod_{(x,y:A)} (x =_A y) \rightarrow (f(x) =_B f(y))$$

индукцијом путање $p : x =_A y$ као $\text{ap}_f(\text{refl}_x) = \text{refl}_{f(x)}$. Функцију ap_f називамо акција над путањама функције $f : A \rightarrow B$.

Доказ. Индукцијом по путањи $p : x =_A y$ треба конструисати путању

$$\text{ap}_f(x, x, \text{refl}_x) : f(x) =_B f(x).$$

Тривијално конструисамо ову путању као $\text{ap}_f(x, x, \text{refl}_x) \equiv \text{refl}_{f(x)}$. □

Лема 5. Нека су A, B и C типови, нека су елементи $x, y, z : A$ и нека су путање $p : x =_A y$ и $q : y =_A z$ у контексту Γ . Тада важи:

$$(i) \text{ap}_f(p \cdot q) = \text{ap}_f(p) \cdot \text{ap}_f(q)$$

$$(ii) \text{ap}_f(p^{-1}) = \text{ap}_f(p)^{-1}$$

$$(iii) \text{ap}_g(\text{ap}_f(p)) = \text{ap}_{g \circ f}(p)$$

$$(iv) \text{ap}_{\text{id}_A}(p) = p$$

Доказ. Доказ изостављамо како је сличан претходним. \square

Транспорт

Лема 6. Нека је A тип и B фамилија типова над A у контексту Γ . Тада можемо конструисати функцију

$$\text{tr}_B : \prod_{(x,y:A)} (x =_A y) \rightarrow B(x) \rightarrow B(y)$$

индукцијом упућање $p : x =_A y$ као $\text{tr}_B(\text{refl}_x) \equiv \text{id}_{B(x)}$. Функцију tr_B називамо транспорт над B .

Доказ. Индукцијом по путањи $p : x =_A y$ треба конструисати функцију

$$\text{tr}_B(x, x, \text{refl}_x) \rightarrow B(x) \rightarrow B(x).$$

Тривијално конструисамо ову путању као $\text{tr}_B(x, x, \text{refl}_x) \equiv \text{id}_{B(x)}$. \square

Друге врсте једнакости

Дефиниција 2.7.1. *Простор кодова* над природним бројевима \mathbb{N} се може дефинисати као бинарна релација $\text{code}_{\mathbb{N}} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathcal{U}_0$ тако да задовољава следеће расуђивачке једнакости:

$$\text{code}_{\mathbb{N}}(0_{\mathbb{N}}, 0_{\mathbb{N}}) \equiv \mathbb{1}$$

$$\text{code}_{\mathbb{N}}(0_{\mathbb{N}}, \text{succ}_{\mathbb{N}}(m)) \equiv \mathbb{0}$$

$$\text{code}_{\mathbb{N}}(\text{succ}_{\mathbb{N}}(n), 0_{\mathbb{N}}) \equiv \mathbb{0}$$

$$\text{code}_{\mathbb{N}}(\text{succ}_{\mathbb{N}}(n), \text{succ}_{\mathbb{N}}(m)) \equiv \text{code}_{\mathbb{N}}(n, m)$$

Лема 7. *Простор кодова је рефлексивна релација, тј. можемо конструисати функцију*

$$\text{reflcode}_{\mathbb{N}} : \prod_{(n:\mathbb{N})} \text{code}_{\mathbb{N}}(n, n).$$

Доказ. Функцију конструишемо индукцијом по $n : \mathbb{N}$ као

$$\begin{aligned}\text{reflcode}_{\mathbb{N}}(0_{\mathbb{N}}) &::= \star \\ \text{reflcode}_{\mathbb{N}}(\text{succ}_{\mathbb{N}}(n)) &::= \text{reflcode}_{\mathbb{N}}(n).\end{aligned}$$

□

Лема 8. За било које природне бројеве $n, m : \mathbb{N}$ важи $m =_{\mathbb{N}} n \rightarrow \text{code}_{\mathbb{N}}(m, n)$ и $\text{code}_{\mathbb{N}}(m, n) \rightarrow m =_{\mathbb{N}} n$.

Доказ. Прво конструишемо

$$\text{encode}_{\mathbb{N}} : \prod_{(m, n : \mathbb{N})} m =_{\mathbb{N}} n \rightarrow \text{code}_{\mathbb{N}}(m, n).$$

Индукцијом по путањи $p : m =_{\mathbb{N}} n$ треба конструисати

$$\text{encode}_{\mathbb{N}}(m, m, \text{refl}_m) : \text{code}_{\mathbb{N}}(m, m).$$

Што смо конструисали у претходној леми, тако да $\text{encode}_{\mathbb{N}}(m, m, \text{refl}_m) ::= \text{reflcode}_{\mathbb{N}}(m)$. Даље конструишемо

$$\text{decode}_{\mathbb{N}} : \prod_{(m, n : \mathbb{N})} \text{code}_{\mathbb{N}}(m, n) \rightarrow m =_{\mathbb{N}} n$$

индукцијом по $m : \mathbb{N}$ и $n : \mathbb{N}$. У случају када су оба природна броја нуле, онда $\text{decode}_{\mathbb{N}}(0_{\mathbb{N}}, 0_{\mathbb{N}}, c) : 0_{\mathbb{N}} =_{\mathbb{N}} 0_{\mathbb{N}}$ конструишемо као $\text{decode}_{\mathbb{N}}(0_{\mathbb{N}}, 0_{\mathbb{N}}, c) ::= \text{refl}_{0_{\mathbb{N}}}$. У случају када је тачно један од њих нула, тада конструишемо елемент типа $0 \rightarrow m =_{\mathbb{N}} n$. Овај елемент је тривијално конструисати правилом индукције празног типа. На крају, у случају када су оба различита од нуле, треба конструисати

$$\text{code}_{\mathbb{N}}(\text{succ}_{\mathbb{N}}(m), \text{succ}_{\mathbb{N}}(n)) \rightarrow \text{succ}_{\mathbb{N}}(m) =_{\mathbb{N}} \text{succ}_{\mathbb{N}}(n).$$

Ову конструкцију изводимо на следећи начин:

$$\begin{aligned}\text{code}_{\mathbb{N}}(\text{succ}_{\mathbb{N}}(m), \text{succ}_{\mathbb{N}}(n)) &::= \text{code}_{\mathbb{N}}(m, n) && (\text{деф. 2.7.1}) \\ &\rightarrow m =_{\mathbb{N}} n && (\text{decode}_{\mathbb{N}}(m, n)) \\ &\rightarrow \text{succ}_{\mathbb{N}}(m) =_{\mathbb{N}} \text{succ}_{\mathbb{N}}(n). && (\text{ap}_{\text{succ}_{\mathbb{N}}})\end{aligned}$$

Коначно, завршавамо конструкцију са

$$\text{decode}_{\mathbb{N}}(\text{succ}_{\mathbb{N}}(m), \text{succ}_{\mathbb{N}}(n), c) ::= \text{ap}_{\text{succ}_{\mathbb{N}}}(\text{decode}_{\mathbb{N}}(m, n, c)).$$

□

Глава 3

Агда

Глава 4

Закључак

Биографија аутора

Вук Стефановић Караџић (*Тршић, 26. октобар/6. новембар 1787. — Беч, 7. фебруар 1864.*) био је српски филолог, реформатор српског језика, сакупљач народних умотворина и писац првог речника српског језика. Вук је најзначајнија личност српске књижевности прве половине XIX века. Стекао је и неколико почасних доктората. Учествовао је у Првом српском устанку као писар и чиновник у Неготинској крајини, а након слома устанка преселио се у Беч, 1813. године. Ту је упознао Јернеја Копитара, цензора словенских књига, на чији је подстицај кренуо у прикупљање српских народних песама, реформу ћирилице и борбу за увођење народног језика у српску књижевност. Вуковим реформама у српски језик је уведен фонетски правопис, а српски језик је потиснуо славеносрпски језик који је у то време био језик образованих људи. Тако се као најважније године Вукове реформе истичу 1818., 1836., 1839., 1847. и 1852.