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, тати и деди

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

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 алгебре, са идејама више теорије категорија, док теорија типова има корене у математичкој логици и теоријском рачунарству. Сматра се да ХоТТ представља алетернативно заснивање математике, поступком формализације уз помоћу интерактивних доказивача. Програм заснивања математике у ХоТТ се назива *универзално заснивање* (енгл. *Univalent Foundations*) [9].

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

Постоји пуно разлога за изучавање ХоТТ-а и заснивање математике кроз интерактивне доказиваче теорема, али један од главних је дао један од оснивача, Владимир Војводски (Владимир Воеводский), увидевши пропусте у туђим и својим радовима [**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 Војводски, 2014.)

1.1 Филозофија и историја

Теорију типова је оригинално представио Бертранд Расел [**rus08**] (Bertrand Russell), решавајући парадокс у заснивању математике тог времена. Након њега Алонзо Черч (Alonzo Church), развија *просто типизирани ламбда рачун* (ПТЛР) (енгл. *Simply Typed Lambda Calculus*) [**crc40**, **crc41**]. Николас де Брујн (Nicolaas de Bruijn) инспирисан ПТЛР-ом развија први аутоматски доказивач теорема Automath [**automath**]. Теорију типова даље развија Пер Мартин-Луф (Per Martin-Löf) коју данас знамо као МЛТТ, или *интуиционистичка/конструктивистичка/зависна теорија типова* [**pml75**, **pml82**, **pml84**, **pml98**]. МЛТТ представља основу за друге теорије које је проширују и које имплементирају разни интерактивни доказивачи теорема: Agda [**agda**], Coq [**coq**], Epigram [**epigram**], Idris [**idris**], NuPRL [**nuprl**]. Инспирисани резултатом да теорија типова може да се интерпретира као ∞-групоид [**hs98**], Владимир Војводски [**vlad06**], Стив Ауоди (Steve Awodey) и Мајкл Ворен (Michael Warren) [**aw09**] независно развијају ХоТТ.

МЛТТ, па самим тим и ХоТТ, се заснива на *Брауверовом интуиционистичком покрет* [**brw**] који тврди да се сва математика, укључујући и концепт доказа, изводи из концепта *конструкције* (рачуна/алгоритма/програма класификованог типом). Браувер је сматрао да је математичко резоновање људска активност и да је математика језик у коме се преносе математички концепти. Другим речима, способност извршавања алгоритма у сврху извођења менталне конструкције је фундаментално људска. Због тога, једини начин на који субјекат може доћи до математичке истине је да доживи истинитост, тако што изведе корак-по-корак одговарајућу менталну конструкцију. Случно, једини начин да субјекат дође до математичке неистине је да доживу њену неистиност, тако што схвати да извођење одговарајуће менталне конструкције није могуће. Интуиционистичи покрет даље развијају Колмогоров [**kol32**] и Хејтинг [**hey**] формулисањем интуиционистичке логике.

Интуиционистички покрет глобално искључује *закон искључења трећег* $P \vee \neg P$ [**brw**]. Разлог томе су *слаби контра-примери* (енгл. *weak counterexamples*). Пример једног слабог контра-примера је Голдбахова претпоставка: *Сваки паран број већи од* 2 *се може представити у облику збира два проста.* Конструктивистички не можемо конструисати доказ да је Голдбахова претпоставка тачна, нити да је нетачна, а поред тога немамо ни процедуру одлучи-

вања. Због тога, не можемо тврдити да је Голдбахова претпоставка тачна или нетачна. Општије, не можемо тврдити да важи закон искључења трећег. Приметимо да закон искључења трећег искључујемо само глобално, односно да уколико је за конструкцију неког доказа потребно искористити закон искључења трећег, довољно је навести га у претпоставкама тврђења. Тиме добијамо да теореме које не користе искључење трећег имају снажнији резултат (јер користе мањи скуп претпоставки), док задржавамо и резултат теорема за које је неопходно користити закон искључења трећег.

У сржи теорије типова је појам *типског расуђивања* (енгл. *type judgment*)

$$t : T$$

који читамо као *t је терм типа T* или *терм t настањује тип T* (енгл. *t inhabits T*). Термови и типови се узимају као примитивни појмови који се не дефинишу. По Брауверу, терм t представља начин на који спроводимо конструкцију T. На пример, уколико желимо да конструишемо терм типа B и ако имамо конструисане термове $a : A$ и $f : A \to B$, онда терм $f(a)$ описује начин на који конструишемо терм типа B, односно $f(a) : B$. Често се у HoTT-у, за терм каже *тачка*, а за тип *простор*, па се тако за расуђивање $t : T$ каже да тачка t *припада* простору T.

Термови и типови у теорији типова имају три интерпретације: (1) докази исказа (логичка интерпретација), (2) програми типова (програмерска интерпретација) и (3) пресликавања структура (категоричка интерпретација). На пример, расуђивање $f : A \to B$ се може сматрати као: (1) доказ импликације, (2) функција која за дати улаз типа A враћа излаз типа B и (3) морфизам из објеката A у објекат B. Ову доктрину је поставио Роберт Харпер (Robert Harper) и назвао ју је *рачунарски тринитаризам* (енгл. *computational trinitarianism*) **[rob11]**. Рачунарски тринитаризам подразумева да сваки концепт који се јавља у једној интерпретацији треба да има значење у друге два интерпретације.

Теорија типова се заснива на идеји о *доказној релевантности* (енгл. *proof relevance*) која тврди да су математичка тврђења, па и сами докази, грађани првог реда, што значи да се различити докази истог исказа могу међусобно поредити. Прецизније, ако су $p_0 : P$ и $p_1 : P$ докази исказа P, односно начини на који можемо конструисати тип P, тада се они могу поредити и у том смислу су *релевантни*. Са друге стране, код *доказно ирелевантних* система битно је само постојање доказа.

Доказивање исказа представља конструисање програма одређеног типа (за детаље видети поглавље 2.5). У том смислу, логика представља област математике, која се бави конструкцијама које су докази. Штавише, по Брауверу, математика је област рачунарства, како су конструкције представљене рачуном/алгоритмом/програмом. Напоменимо да постоје и друге конструкције које нису докази (на пример, природни бројеви).

Још једна карактеристика МЛТТ-а, па самим тим и ХоТТ-а, је да користи *синтетички*, а не *аналитички* приступ. У синтетичком приступу, основни објекти су примитиве чије се особине и релације аксиоматизују, из којих се даље логички дедукују последице. У аналитичком приступу, основни објекти су конструисани од других објеката, а њихове особине и релације су дедуковане из математичког окружења у ком су дефинисани. Често се за пример синтетичног приступа узима Еуклидска геометрија (где су основни објекти тачка, права и раван), а за аналитички приступ Декартова/аналитичка геометрија (где се основни објекти конструишу помоћу скупова).

Интенционални и екстенционални типови? (нешто чуно, проучити ако остане времена).

1.2 Мотивација и циљ рада

Формализација математике представља једану од примарних делатности математичара. Од самог настанка математике, математичари су покушавали да прецизно заснују математику уводећи прецизно дефинисан дедуктивни систем, као и аксиоме теорије које формализују. Један од првих сачуваних примера су Еуклидови Елементи [ee], који су, након Библије, највише проучавано, преписивано и штампано дело у људској историји. Иако су Елементи представљали формализам геометрије тог времена, данас знамо да садрже неколико контрадикција.

Стотинама година касније, многи велики математичари настоје да унапреде поступак формализације у смислу прецизнијег дефинисања дедуктивог система, а и самих метода/алата који се користе при формализацији. Вековима се уводе нови симболи, како би се резоновање у мозгу одвило механички, једноставним погледом. Почетком двадесетог века, математика и разноликост симбола се у тој мери шири да се многи математичари из исте области међусобно не разумеју. Другим речима, користе различите симболе за исте

појмове. Из тог разлога, 1934. године, формира се група математичара под колективним псеудонимом Николас Бурбаки (Nicolas Bourbaki). Бурбаки група, оригинално, има за циљ да формализује уџбенике математичке анализе. Након почетног успеха, објављује низ уџбеника из разних области математике као што су теорија скупова, апстрактна алгебра, топологија, и Лијева алгебра.

Данас се за било коју формализацију која се изводи помоћу *оловке и папира* сматра да је *неформална*. Због тога, модерни поступак формализације подразумева коришћење интерактивних доказивача теорема. Интерактивни доказивачи омогућавају алгоритамску проверу доказа која је од кључног значаја за исправност доказа. Иако интерактивни доказивачи представљају конкретну програмску имплементацију, па могу садржати грешке при имплементацији, те грешке скоро никада неће утицати на то да интерактивни доказивач прихвати неисправан доказ као исправан. Због тога се формализација унутар интективних доказивача сматра као *формална*.

ХоТТ, тренутно, представља најбољи начин формалног заснивања математике. Такође, сматра се да ће имати велики утицај и бити део неких нових покушаја заснивања у будућности. Као такав, вредан је изучавања. Са друге стране, како је МЛТТ у основи ХоТТ-а, циљ овог рада се састоји у:

1. детаљном теоријском и практичном изучавању МЛТТ;
2. формализацији основних објеката и конструкција МЛТТ-а у типски зависном програмском језику Agda.

1.3 Друге формализације

Тренутно постоји неколицина библиотека које за циљ имају да формализују математику у оквиру ХоТТ-а. Прву и основну библиотеку, написану у интерактивном доказивачу Coq, објавио је Владимир Војводски, 2010. године, под називом Foundations [**vlad_found**]. Ова библиотека постала је део проширене библиотеке UniMath [**unimath**], коју је оригинално написало неколико аутора, такође у интерактивном доказивачу Coq. Након успеха ове две библиотеке, настаје пројекат унивалентног заснивања у коме се паралелно развијају две нове библиотека HoTT-Coq [**hottcoq**] и HoTT-Agda [**hottagda**]. Посебне 2013. године, у Институту за напредне студије у Прин-

стону, пише се и објављује прва неформална књига *Homotopy Type Theory: Univalent Foundations of Mathematics*, или краће *The HoTT Book* [9]. По узору на UniMath, од 2021. године, развија се и библиотека Agda-UniMath [**agda-unimath**]. Све четири библиотеке се, и данас, активно развијају.

Једна мана XoTT-а је недостатак конструкције унивалентности. Унивалентност се аксиоматизује, те нема правило израчунавања. Због тога, није могуће конструисати типове за чију се конструкцију користи унивалентност. Другим речима, када систем наиђе на унивалентност неће знати како да је редукује. Као решење овог проблема настаје *коцкаста теорија типова* (енгл. *cubical type theory*) [**cube**]. Коцкаста теорија типова је имплементирана проширењем програмског језика Agda који се назива Cubical Agda. То је омогућило да настане библиотека cubical [**cubi**] која се, такође, активно развија.

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

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

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 промењивих 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:

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

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

Исцрпна листа структурних правила закључивања у интуиционистичкој теорији типова се може наћи у [8]. 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 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 $f : A \to B$ 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 $mapstob(x)$. Тада $b(x)$ настањује 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:

$$\frac{\Gamma, x : A \vdash B(x)\ \text{type}}{\Gamma \vdash \prod_{(x:A)} B(x)\ \text{type}}\ [\prod\text{-form}] \qquad \frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma \vdash \lambda x.b(x) : \prod_{(x:A)} B(x)}\ [\prod\text{-intro}] \qquad \frac{\Gamma \vdash f : \prod_{(x:A)} B(x)}{\Gamma, x : A \vdash f(x) : B(x)}\ [\prod\text{-elim}]$$

$$\frac{\Gamma, x : A \vdash b(x) : B(x)}{\Gamma \vdash (\lambda y.b(y))(x) \equiv b(x) : B(x)}\ [\prod\text{-comp}_1] \qquad \frac{\Gamma \vdash f : \prod_{(x:A)} B(x)}{\Gamma \vdash \lambda x.f(x) \equiv f : \prod_{(x:A)} B(x)}\ [\prod\text{-comp}_2]$$

Специјалан случај типа зависних функција је тип (уобичајених) *функција* $A \to B$. 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* $A \to B$ дефинишемо као:

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

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

Дефиниција 2.3.2. За сваки тип A дефинишемо *функцију идентитета* $\mathsf{id}_A : A \to A$ као $\mathsf{id}_A :\equiv \lambda x.x$.

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

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 конструишемо 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 $\mathbb{0}$, 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 идентитета $x =_A y$.

Тип природних бројева

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

$$\frac{}{\vdash \mathbb{N} \text{ type}}\ [\mathbb{N}\text{-form}] \qquad \frac{}{\vdash 0_{\mathbb{N}} : \mathbb{N}}\ [\mathbb{N}\text{-intro}_{0_{\mathbb{N}}}] \qquad \frac{}{\vdash \mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}}\ [\mathbb{N}\text{-intro}_{\mathsf{succ}_{\mathbb{N}}}]$$

$$\frac{\begin{array}{l}\Gamma, n : \mathbb{N} \vdash P(n) \text{ type} \\ \Gamma \vdash p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}}) \\ \Gamma \vdash p_{\mathsf{succ}_{\mathbb{N}}} : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\mathsf{succ}_{\mathbb{N}}}) : \prod_{(n:\mathbb{N})} P(n)}\ [\mathbb{N}\text{-ind}]$$

$$\frac{\begin{array}{l}\Gamma, n : \mathbb{N} \vdash P(n) \text{ type} \\ \Gamma \vdash p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}}) \\ \Gamma \vdash p_{\mathsf{succ}_{\mathbb{N}}} : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\end{array}}{\Gamma \vdash \mathsf{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\mathsf{succ}_{\mathbb{N}}}, 0_{\mathbb{N}}) \equiv p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}})}\ [\mathbb{N}\text{-comp}_{0_{\mathbb{N}}}^{\mathsf{ind}_{\mathbb{N}}}]$$

$$\frac{\begin{array}{l}\Gamma, n : \mathbb{N} \vdash P(n) \text{ type} \\ \Gamma \vdash p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}}) \\ \Gamma \vdash p_{\mathsf{succ}_{\mathbb{N}}} : \prod_{(n:\mathbb{N})} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))\end{array}}{\Gamma,\ n : \mathbb{N} \vdash \mathsf{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\mathsf{succ}_{\mathbb{N}}}, \mathsf{succ}_{\mathbb{N}}(n)) \equiv p_{\mathsf{succ}_{\mathbb{N}}}(n, \mathsf{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\mathsf{succ}_{\mathbb{N}}}, n)) : P(\mathsf{succ}_{\mathbb{N}}(n))}\ [\mathbb{N}\text{-comp}_{\mathsf{succ}_{\mathbb{N}}}^{\mathsf{ind}_{\mathbb{N}}}]$$

$$\frac{\begin{array}{l}\Gamma \vdash A \text{ type} \\ \Gamma \vdash a_{0_{\mathbb{N}}} : A \\ \Gamma \vdash a_{\mathsf{succ}_{\mathbb{N}}} : \mathbb{N} \to A \to A\end{array}}{\Gamma \vdash \mathsf{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\mathsf{succ}_{\mathbb{N}}}) : \mathbb{N} \to A}\ [\mathbb{N}\text{-rec}_{\mathbb{N}}]$$

$$\frac{\begin{array}{l}\Gamma \vdash A \text{ type} \\ \Gamma \vdash a_{0_{\mathbb{N}}} : A \\ \Gamma \vdash a_{\mathsf{succ}_{\mathbb{N}}} : \mathbb{N} \to A \to A\end{array}}{\Gamma \vdash \mathsf{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\mathsf{succ}_{\mathbb{N}}}, 0_{\mathbb{N}}) \equiv a_{0_{\mathbb{N}}} : A}\ [\mathbb{N}\text{-comp}_{0_{\mathbb{N}}}^{\mathsf{rec}_{\mathbb{N}}}]$$

$$\frac{\begin{array}{l}\Gamma \vdash A \text{ type} \\ \Gamma \vdash a_{0_{\mathbb{N}}} : A \\ \Gamma \vdash a_{\mathsf{succ}_{\mathbb{N}}} : \mathbb{N} \to A \to A\end{array}}{\Gamma, n : \mathbb{N} \vdash \mathsf{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\mathsf{succ}_{\mathbb{N}}}, \mathsf{succ}_{\mathbb{N}}(n)) \equiv a_{\mathsf{succ}_{\mathbb{N}}}(n, \mathsf{rec}_{\mathbb{N}}(a_{0_{\mathbb{N}}}, a_{\mathsf{succ}_{\mathbb{N}}}, n)) : A}\ [\mathbb{N}\text{-comp}_{\mathsf{succ}_{\mathbb{N}}}^{\mathsf{rec}_{\mathbb{N}}}]$$

По правилу $\mathbb{N}$-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 $\mathsf{succ}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$ и она говори да ће $\mathsf{succ}_{\mathbb{N}}(n)$ бити канонични терм

типа $\mathbb{N}$ ако је $n : \mathbb{N}$ канонични терм. Због тога су $0_{\mathbb{N}}, \mathsf{succ}_{\mathbb{N}}(0_{\mathbb{N}}), \mathsf{succ}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(0_{\mathbb{N}})), \ldots$ канонични термови који настањују 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 $\mathsf{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\mathsf{succ}_{\mathbb{N}}}) : \prod_{(n:\mathbb{N})} P(n)$ потребно је конструисати елемент $p_{0_{\mathbb{N}}} : P(0_{\mathbb{N}})$ (*база индукције*) и $p_{\mathsf{succ}_{\mathbb{N}}} : \prod_{n:\mathbb{N}} P(n) \to P(\mathsf{succ}_{\mathbb{N}}(n))$ (*индуктивни корак*). Даље, за сваки од конструктора треба увести правило израчунавања у складу са зависном функцијом $\mathsf{ind}_{\mathbb{N}}(p_{0_{\mathbb{N}}}, p_{\mathsf{succ}_{\mathbb{N}}}) : \prod_{(n:\mathbb{N})} P(n)$. Због тога имамо два правила израчунавања $\mathbb{N}\text{-comp}_{0_{\mathbb{N}}}$ и $\mathbb{N}\text{-comp}_{\mathsf{succ}_{\mathbb{N}}}$.

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

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

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

Заиста, за овако дефинисану операцију сабирања важи:

$$m +_{\mathbb{N}} 0_{\mathbb{N}} \equiv m;$$
$$m +_{\mathbb{N}} \mathsf{succ}_{\mathbb{N}}(n) \equiv \mathsf{succ}_{\mathbb{N}}(m +_{\mathbb{N}} n).$$

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

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

Такође, за овако дефинисану операцију множења важи:

$$m \times_{\mathbb{N}} 0_{\mathbb{N}} \equiv 0_{\mathbb{N}};$$
$$m \times_{\mathbb{N}} \mathsf{succ}_{\mathbb{N}}(n) \equiv (m +_{\mathbb{N}} (m \times_N n)).$$

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

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

$$f(0_{\mathbb{N}}) \equiv \Phi_{0_{\mathbb{N}}};$$
$$f(\mathsf{succ}_{\mathbb{N}}(n)) \equiv \Phi_{\mathsf{succ}_{\mathbb{N}}},$$

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

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

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

Празни тип

Празни тип $\mathbb{0}$ је дегенерисани пример индуктивног типа кога не настањује ни један елемент. Прецизније, празни тип $\mathbb{0}$ дефинишемо следећом спецификацијом.

$$[\mathbb{0}\text{-form}]\ \frac{}{\vdash \mathbb{0}\ \text{type}} \quad [\mathbb{0}\text{-ind}]\ \frac{\Gamma, 0 \vdash P(x)\ \text{type}}{\Gamma \vdash \mathsf{ind}_{\mathbb{0}} : \prod_{(x:\mathbb{0})} P(x)} \quad [\mathbb{0}\text{-rec}]\ \frac{\Gamma \vdash A\ \text{type}}{\Gamma \vdash \mathsf{rec}_{\mathbb{0}} : \mathbb{0} \to A}$$

Како празан тип $\mathbb{0}$ 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 $\mathbb{0}$ постоји елемент $\mathsf{ind}_{\mathbb{0}} : \prod_{(x:\mathbb{0})} P(x)$. Чешће се користи правило рекурзије које тврди да уколико конструишемо елемент $x : \mathbb{0}$, онда можемо да конструишемо елемент $\mathsf{rec}_{\mathbb{0}}(x) : A$ било ког типа A. Правило рекурзије за празни тип $\mathbb{0}$ се обично назива и *правило контрадикције* или *правило противречности*.

Дефиниција 2.4.1. За сваки тип A дефинишемо тип *негације* od A као $\neg A := A \to \mathbb{0}$. Поред тога, кажемо да је тип A *празан* ако његову негацију настањује неки елемент, тј. $\mathsf{empty}(A) := A \to \mathbb{0}$.

Приметимо да је *двојна негација* од A дефинисана као $\neg\neg A := (A \to \mathbb{0}) \to \mathbb{0}$. Због тога, не мора да важи $\neg\neg A \to A$, 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.

$$[\mathbb{1}\text{-form}]\ \frac{}{\vdash \mathbb{1}\ \text{type}} \qquad [\mathbb{1}\text{-intro}_\star]\ \frac{}{\vdash \star : \mathbb{1}}$$

$$[\mathbb{1}\text{-ind}]\ \frac{\Gamma, x : \mathbb{1} \vdash P(x)\ \text{type} \quad \Gamma \vdash p_\star : P(\star)}{\Gamma \vdash \mathsf{ind}_{\mathbb{1}}(p_\star) : \prod_{(x:\mathbb{1})} P(x)} \qquad [\mathbb{1}\text{-comp}]\ \frac{\Gamma, 1 \vdash P(x)\ \text{type} \quad \Gamma \vdash p_\star : P(\star)}{\Gamma \vdash \mathsf{ind}_{\mathbb{1}}(p_\star, \star) \equiv p_\star : P(\star)}$$

$$[\mathbb{1}\text{-rec}]\ \frac{\Gamma \vdash A\ \text{type} \quad \Gamma \vdash a : A}{\Gamma \vdash \mathsf{rec}_{\mathbb{1}}(a) : \mathbb{1} \to A}$$

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 $\mathsf{ind}_{\mathbb{1}}(p_\star) : \prod_{(x:\mathbb{1})} P(x)$ 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, $\mathsf{ind}_{\mathbb{1}}(p_\star, \star) \equiv p_\star : P(\star)$.

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 $\mathsf{rec}_{\mathbb{1}}(a) : \mathbb{1} \to A$.

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

У хомотопној теоријити 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 $\mathsf{inl}(a) : A + B$) или из типа B (ако $b : B$, онда $\mathsf{inr}(b) : A + B$).

$$\frac{\Gamma \vdash A \text{ type}, B \text{ type}}{\Gamma \vdash A + B \text{ type}}\ [+\text{-form}] \qquad \frac{}{\Gamma \vdash \mathsf{inl} : A \to A + B}\ [+\text{-intro}_{\mathsf{inl}}] \qquad \frac{}{\Gamma \vdash \mathsf{inr} : B \to A + B}\ [+\text{-intro}_{\mathsf{inr}}]$$

$$[+\text{-ind}]\ \frac{\begin{array}{l}\Gamma, z : A + B \vdash P(z) \text{ type} \\ \Gamma \vdash p_{\mathsf{inl}} : \prod_{(a:A)} P(\mathsf{inl}(a)) \\ \Gamma \vdash p_{\mathsf{inr}} : \prod_{(b:B)} P(\mathsf{inr}(b))\end{array}}{\Gamma \vdash \mathsf{ind}_+(p_{\mathsf{inl}}, p_{\mathsf{inr}}) : \prod_{(z:A+B)} P(z)}$$

$$[+\text{-comp}]\ \frac{\begin{array}{l}\Gamma, z : A + B \vdash P(z) \text{ type} \\ \Gamma \vdash p_{\mathsf{inl}} : \prod_{(a:A)} P(\mathsf{inl}(a)) \\ \Gamma \vdash p_{\mathsf{inr}} : \prod_{(b:B)} P(\mathsf{inr}(b))\end{array}}{\begin{array}{l}\Gamma, a : A \vdash \mathsf{ind}_+(p_{\mathsf{inl}}, p_{\mathsf{inr}}, \mathsf{inl}(a)) \equiv p_{\mathsf{inl}}(a) : P(\mathsf{inl}(a)) \\ \Gamma, b : B \vdash \mathsf{ind}_+(p_{\mathsf{inl}}, p_{\mathsf{inr}}, \mathsf{inr}(b)) \equiv p_{\mathsf{inr}}(b) : P(\mathsf{inr}(b))\end{array}}$$

$$[+\text{-rec}]\ \frac{\begin{array}{l}\Gamma \vdash type \\ \Gamma \vdash f : A \to X \\ \Gamma \vdash g : B \to X\end{array}}{\Gamma \vdash \mathsf{rec}_+(f, g) : A + B \to X}$$

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

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 $f : A \to X$ и за сваку функцију $g : B \to X$ имамо функцију $\mathsf{rec}_+(f, g) : A + B \to X$. Из правила индукције, за свако $f : A \to X$ и за свако $g : B \to Y$, имамо функцију $f + g : A + B \to X + Y$.

Специјални случај типа копроизвода је *буловски тип* $2 := \mathbb{1}+\mathbb{1}$, чије јединe елементе дефинишемо као $\mathsf{true} :\equiv \mathsf{inl}(\star)$ и $\mathsf{false} :\equiv \mathsf{inr}(\star)$. 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*.

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

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

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 настањивати *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.

$$\frac{\Gamma, x : A \vdash B(x)\ \text{type}}{\Gamma \vdash \sum_{(x:A)} B(x)\ \text{type}}\ [\textstyle\sum\text{-form}] \qquad \frac{\Gamma, x : A \vdash y(x) : B(x)}{\Gamma \vdash (x, y(x)) : \sum_{(x:A)} B(x)}\ [\textstyle\sum\text{-intro}]$$

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

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

Тип зависних парова $\sum_{(x:A)} B(x)$ има један конструктор помоћу кога се могу формирати елементи који га настањују, 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 $\mathsf{ind}_{\sum}(f) : \prod_{p:\sum_{(x:A)} B(x)} P(p)$

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 $\mathsf{ind}_{\sum}(f,(x,y)) \equiv f(x,y) : P((x,y))$.

Правило индукције нам омогућава да дефинишемо функције у наставку.

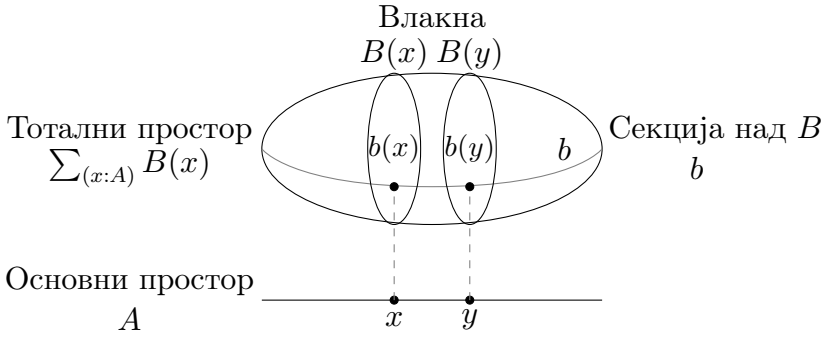
Дефиниција 2.4.3. Нека је B фамилија типова над A. Тада елемент $\mathsf{pr}_1 : \sum_{(x:A)} B(x) \to A$ *пројекције на први елемент* дефинишемо као:

$$\mathsf{pr}_1((a,b)) :\equiv a, \tag{2.1}$$

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

$$\mathsf{pr}_2((a,b)) :\equiv b. \tag{2.2}$$

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* (енгл. *carry*), и како је супротна правилу $\sum$-ind, правило $\sum$-ind често наивамо *одкаријевање* (енгл. *uncarry*).



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:

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

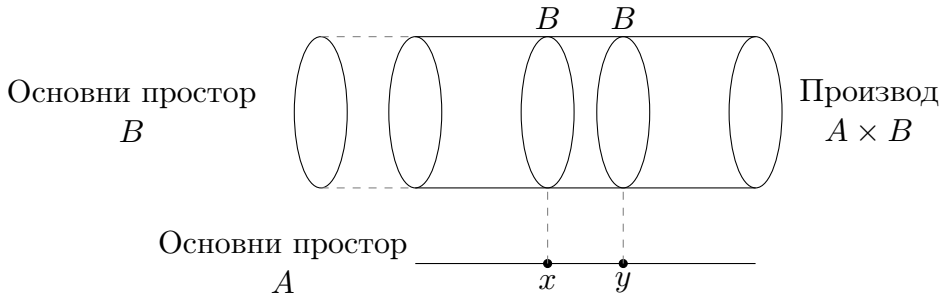
Такође, *пројекцију на први елемент* $\mathsf{fst} : A \times B \to A$ и *пројекцију на други елемент* $\mathsf{snd} : A \times B \to B$ дефинишемо као:

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

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

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

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



Слика 2.2: Геометријска репрезентација типа независних парова.

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

2.5 Искази као типови

Интерпретација *искази као типови* (енгл. *propositions as types*) 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 нетачан, већ да представља колекцију својих

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

Табела 2.1: Кари-Хауардова интерпретација

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 $A \to B$, тј. треба конструисати елемент типа B који користи претпоставку дату постојањем елемента 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 $\mathbb{2}$, тј. P не мора да буде предикат. Поред тога, тип $\sum_{(x:A)} P(x)$ можемо схватити као тип свих елемената $x : A$ za koje $P(x)$.

Како ова два типа дају више информација у односу на традиционално значење исказа $A \vee B$ и $\exists x.P(X)$, користе 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 $B : A \to \mathcal{U}$.

Релимо да типови који могу да се формирају из празног контекста настањују универзум $\mathcal{U}$ (то су, на пример, $\mathbb{0}$, $\mathbb{1}$, 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 $B : A \to \mathcal{U}$, 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*

$$\mathcal{U}_0, \quad \mathcal{U}_1, \quad \mathcal{U}_2, \quad \ldots \tag{2.3}$$

за коју важе следећа правила:

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

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 $\mathcal{U}_0$ і $\mathcal{U}_1$, најмања горња граница $\mathcal{U}_0 \sqcup \mathcal{U}_1$ је $\mathcal{U}_1$.

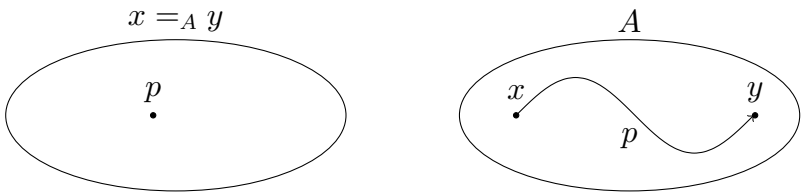
2.7 Типови идентитета

Подсетимо се да из дефиниције операције $+_{\mathbb{N}}$ важи $m +_{\mathbb{N}} \mathsf{0}_{\mathbb{N}} \equiv m$. Природно се намеће питање: Да ли важи $\mathsf{0}_{\mathbb{N}} +_{\mathbb{N}} m \equiv m$? 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?* Одговор на ово питање нам дају *типови идентитета* (енгл. *identity types*).

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–Хауардовој интерпретацији представља неки тип, а како зависи од два елемента типа A мора бити фамилија типова. Исказне једнакости другачије називамо *типови идентитета*, и обележавамо као $\mathsf{Id}_A : A \to A \to \mathcal{U}$. За два конкретна елементе $x, y : A$, тип $\mathsf{Id}_A(x, y)$ 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 *путање* 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 посматрати као тип *идентификације* 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 контексту Γ, онда можемо формирати тип идентитета $x =_A y$ кога ће настањивати путање, еквиваленције или идентификације. Основна идентификација коју можемо да конструишемо је *рефлексија*:

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

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

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

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

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

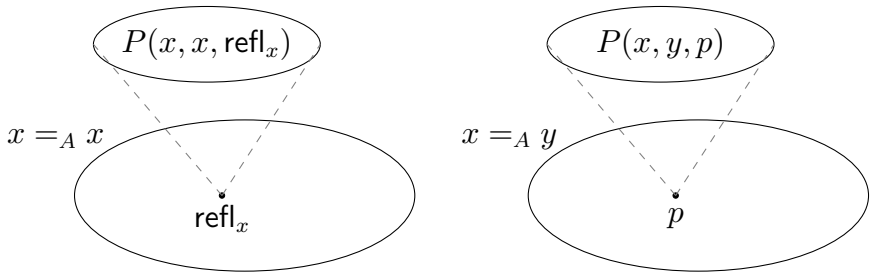
Индукција путање

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 $\mathsf{ind}_= : \prod_{(x,y:A)} \prod_{(p:x=_A y)} P(x, y, p)$ уколико постоји функција $f : \prod_{(x:A)} P(x, x, \mathsf{refl}_x)$. Како постоји само један конструктор $\mathsf{refl}_x : x =_A x$, постоји само једно правило израчунавања које треба да се сложи са правилом индукције. Због тога правило израчунавања тврди $\mathsf{ind}_=(x, x, \mathsf{refl}_x) \equiv f(x) : P(x, x, \mathsf{refl}_x)$. Формално, правило индукције и правило израчунавања је дато следећом спецификацијом:

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

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

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 $P(x, x, \mathsf{refl}_x)$ за било коју тачку $x : A$? Кључно запажање лежи у 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 такви $x : A$. 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*

$$\mathsf{inv}_A : \prod_{(x,y:A)} (x =_A y) \to (y =_A x)$$

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

Доказ. Да би констурисали елемент типа $\prod_{(x,y:A)}(x =_A y) \to (y =_A x)$, конструишемо функцију

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

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

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

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

$$\mathsf{inv}_A(x, x, \mathsf{refl}_x) :\equiv \mathsf{refl}_x.$$

□

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

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

индукцијом путање $p : x =_A y$ *као* $\mathsf{conc}_A(x, x, z, \mathsf{refl}_x, q) :\equiv q$. *Функцију* conc_A *називамо* надовезивање путања. *Често, за дате путање* $p : x =_A y$ *и* $q : y =_A z$, *надовезану путању означавамо са* $p \cdot q :\equiv \mathsf{conc}_A(x, y, z, p, q)$.

Доказ. Прво конструишемо функцију

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

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

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

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

$$f(x, x, \mathsf{refl}_x, z) : (x =_A z) \to (x =_A z)$$

за било које елементе $x, z : A$. Конструисање ове функције је тривијално и због тога имамо да је $f(x, x, \mathsf{refl}_x, z, q) :\equiv q$. Коначно, тражена конструкција је

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

□

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

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

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

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

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

(i) *Доказ.* Желимо да конструишемо путању

$$\mathsf{unit}_{\mathsf{l}}(p) : \mathsf{refl}_x \cdot p = p,$$
$$\mathsf{unit}_{\mathsf{r}}(p) : p \cdot \mathsf{refl}_y = p.$$

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

$$\mathsf{unit}_{\mathsf{l}}(\mathsf{refl}_x) : \mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x,$$
$$\mathsf{unit}_{\mathsf{r}}(\mathsf{refl}_x) : \mathsf{refl}_x \cdot \mathsf{refl}_x = \mathsf{refl}_x.$$

Обе путање је тривијално конструисати као $\mathsf{refl}_{\mathsf{refl}_x}$. □

(ii) *Доказ.* Желимо да конструишемо путању

$$\mathsf{inv}_{\mathsf{l}}(p) : p^{-1} \cdot p = \mathsf{refl}_y,$$
$$\mathsf{inv}_{\mathsf{r}}(p) : p \cdot p^{-1} = \mathsf{refl}_x.$$

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

$$\mathsf{inv}_{\mathsf{l}}(\mathsf{refl}_x) : \mathsf{refl}_x^{-1} \cdot \mathsf{refl}_x = \mathsf{refl}_x,$$
$$\mathsf{inv}_{\mathsf{r}}(\mathsf{refl}_x) : \mathsf{refl}_x \cdot \mathsf{refl}_x^{-1} = \mathsf{refl}_x.$$

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

(iii) *Доказ.* Желимо да конструишемо путању

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

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

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

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

(iv) *Доказ.* Желимо да конструишемо путању

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

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

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

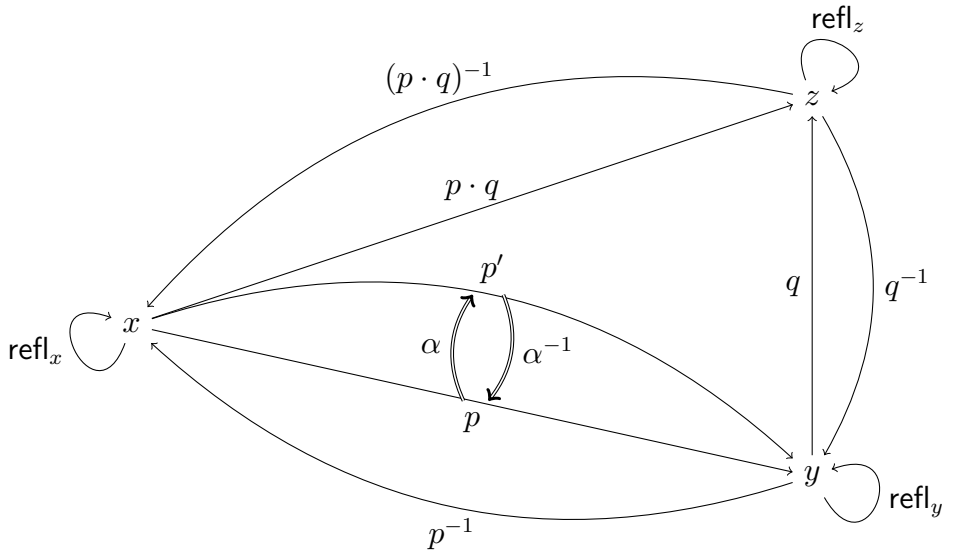
Али како је $\mathsf{refl}_x \cdot q \equiv q$ и $\mathsf{refl}_x \cdot (q \cdot r) \equiv q \cdot r$ претходна путања се своди на

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

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

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 $f : A \to B$ функција у контексту* Γ. *Тада можемо конструисати функцију*

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

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

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

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

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

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

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

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

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

Доказ. Доказ изостављамо како је сличан претходним. □

Транспорт

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

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

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

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

$$\mathsf{tr}_B(x, x, \mathsf{refl}_x) \to B(x) \to B(x).$$

Тривијално конструишемо ову путању као $\mathsf{tr}_B(x, x, \mathsf{refl}_x) :\equiv \mathsf{id}_{B(x)}$. □

Друге врсте једнакости

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

$$\begin{aligned}
\mathsf{code}_{\mathbb{N}}(0_{\mathbb{N}}, 0_{\mathbb{N}}) &\equiv \mathbb{1} \\
\mathsf{code}_{\mathbb{N}}(0_{\mathbb{N}}, \mathsf{succ}_{\mathbb{N}}(m)) &\equiv \mathbb{0} \\
\mathsf{code}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(n), 0_{\mathbb{N}}) &\equiv \mathbb{0} \\
\mathsf{code}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(n), \mathsf{succ}_{\mathbb{N}}(m)) &\equiv \mathsf{code}_{\mathbb{N}}(n, m)
\end{aligned}$$

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

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

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

$$\mathsf{reflcode}_{\mathbb{N}}(0_{\mathbb{N}}) :\equiv \star$$
$$\mathsf{reflcode}_{\mathbb{N}}(\mathsf{succ}_{\mathbb{N}}(n)) :\equiv \mathsf{reflcode}_{\mathbb{N}}(n).$$

□

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

Доказ. Прво конструишемо

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

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

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

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

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

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

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

Ову конструкцију изводимо на следећи начин:

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

Коначно, завршавамо конструкцију са

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

□

Глава 3

Агда

Глава 4

Закључак

Литература

[1] Guillaume Brunerie и др. *Homotopy Type Theory in Agda*. URL: `https://github.com/HoTT/HoTT-Agda`.

[2] Cyril Cohen и др. *Cubical Type Theory: a constructive interpretation of the univalence axiom*. 2016. arXiv: `1611.02108 [cs.LO]`.

[3] Thierry Coquand, Simon Huber и Anders Mörtberg. *On Higher Inductive Types in Cubical Type Theory*. 2018. arXiv: `1802.01170 [cs.LO]`.

[4] Martín Hötzel Escardó. „Introduction to univalent foundations of mathematics with Agda". У: *arXiv preprint arXiv:1911.00580* (2019).

[5] Jackson Macor. *A brief introduction to type theory and the univalence axiom*. 2015.

[6] Per Martin-Löf и Giovanni Sambin. *Intuitionistic type theory*. Св. 9. Bibliopolis Naples, 1984.

[7] Ulf Norell. „Dependently typed programming in Agda". У: *Proceedings of the 4th international workshop on Types in language design and implementation*. 2009, стр. 1–2.

[8] Egbert Rijke. *Introduction to Homotopy Type Theory*. 2022. arXiv: `2212.11082 [math.LO]`.

[9] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: `https://homotopytypetheory.org/book`, 2013.

[10] Andrea Vezzosi, Anders Mörtberg и Andreas Abel. „Cubical Agda: A dependently typed programming language with univalence and higher inductive types". У: *Journal of Functional Programming* 31 (2021), e8.

[11] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson и др. *UniMath — a computer-checked library of univalent mathematics*. available at `http://unimath.org`. DOI: `10.5281/zenodo.7848572`. URL: `https://doi.org/10.5281/zenodo.7848572`.

[12] Philip Wadler. „Propositions as Types". У: *Commun. ACM* 58.12 (Нов. 2015), стр. 75–84. ISSN: 0001-0782. DOI: `10.1145/2699407`. URL: `https://doi.org/10.1145/2699407`.

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.