Kartezijanski zatvorene kategorije

Henezi, Nikola

Sveučilište u Zagrebu

PMF - Matematički odjel

Nikola Henezi

Kartezijanski zatvorene kategorije

Diplomski rad

Voditelji rada:
prof. dr. sc. Boris Širola
izv. prof. dr. sc. Mladen Vuković

Ovaj diplomski rad obranjen je dana _____ pred nastavničkim povjerenstvom u sastavu:

1. _____ , predsjednik
2. _____ , član
3. _____ , član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Uvod **ii**

1 Teorija kategorija **1**
1.1 Definicije i osnovni primjeri 1
1.2 Kategorija Hask 6
1.3 Dijagram 7
1.4 Monoik i epik 10
1.5 Kategorijski konstruktori 13
1.5.1 Inicijalni i finalni objekti 13
1.5.2 Produkti i koprodukti 14
1.6 Funktori 19
1.6.1 Maybe 20
1.6.2 Funktori u Haskellu 20

2 Kartezijanski zatvorene kategorije **22**

3 Osvrt na $\lambda-$račun **25**

4 Veza CCC-a i $\lambda-$računa **31**

Bibliografija **35**

Uvod

Teorija kategorija nastala je 1940. godine kada su S. Eilenberg i S. M. Lane pokušali spojiti dvije grane matematike: algebru i topologiju, što je rezultiralo strukturom koje ne odgovara samo algebri i topologiji, već i drugim granama matematike. Sve do 1957. se na teoriju kategorija moglo gledati kao pojednostavljeni jezik za već postojeće matematičke grane. A. Grothendieck te godine koristi teoriju kategorija kako bi došao do značajnih rezultata u algebarskoj geometriji. Od tada se kategorije koriste kako bi se proučavali već postojeći problemi sa drugačijeg stajališta i pokušalo doći do novih saznanja.

Tokom 19. stoljeća su matematičari tražili temelj matematike i vjerovalo se da je to teorija skupova. B. Lawvere je vidio teoriju kategorija kao temelj matematike, pokazavši da je kategorija skupova, zapravo kategorija s nekim lijepim svojstvima i, između ostalog, pokazao je da se logika višeg reda može proučavati u jeziku kategorija. Godine 1980. je J. Lambek pokazao da tipovi i programi imaju strukturu posebne kategorije, što je omogućilo promatranje programa i svojstva programa bez doticanja implementacije. E. Moggi je, do tada, samo teorijski koncept monada preveo u računarstvo i danas je to temeljni alat za kompoziciju komputacija u nekim funcijskim programskim jezicima. Baez i Dolan su pokazali da se teorija kategorija, osim računarstva, može koristiti i u kvantnoj fizici. No njezina korist uočava se i u drugim granama znanosti. Teorija kategorija je zamišljena kao most koji spaja grane matematike i danas se otkrivaju novi načini i grane koje se mogu povezati.

Shultz i Spivak su pokazali da se baze podataka mogu promatrati kao kategorije, a sam funkcijski programski jezik Haskell je inspiriran teorijom kategorija gdje su abstraktni matematički koncepti poput funktora ili monada dobili praktičnu primjenu u svakodnevnom razvoju softvera.

U ovom diplomskom radu dajemo pregled osnovnih koncepata teorije kategorija s praktičnim primjerima, te kroz $\lambda-$račun i programski jezik Haskell demonstriramo praktičnu primjenu teorije kategorija.

U prvom poglavlju uvodimo pojam kategorije, navodimo nekoliko primjera, uvodimo koncept dijagrama i tehniku praćenja dijagrama. Proučavamo strukture strelice i objekata u kategoriji, preslikavanja između kategorija i pokazujemo praktičku primjenu teorija kategorija u funkcijskom programskom jeziku Haskell.

U drugom poglavlju bavimo se kartezijanski zatvorenim kategorijama. To su posebne vrste kategorija koje imaju ekspresivnu moć jednaku $\lambda-$računu. Zatim navodimo nekoliko najbitnijih primjera kartezijanski zatvorenih kategorija.

U trećem poglavlju dajemo proširenje jednostavno tipiziranog λ-račun, definiramo osnovne pojmove i navodimo ključne rezultate.

U četvrtom poglavlju opisujemo konstrukciju kojom se pokazuje veza između $\lambda-$računa i kartezijanski zatvorenih kategorija, te razmatramo primjenu i praktičnost teorija kategorija.

Ovim putem bih se zahvalio Davidu Kaloperu Meršinjakomu za uvod u funkcijsko programiranje i teoriju kategorija, koje je inspiriralo ne samo ovaj diplomski rad, već i uvelike usmjerilo moj profesionalni razvoj. Izv. prof. dr. sc. Mladenu Vukoviću se zahvaljujem na strpljenju, pedantnosti, vodstvu i brojnim primjedbama koje su usmjerila i oblikovale ovaj rad u smislenu cjelinu.

1 Teorija kategorija

U ovoj točki definiramo pojam kategorije i detaljno raspisujemo nekoliko primjera kategorija. Pod točkom "Dijagram" uvodimo koncept dijagrama i tehniku "praćenja dijagrama" koja olakšava neka razmatranja u teoriji kategorija. Iduću točku posvećujemo strelicama i objektima, navodimo nekoliko posebnih slučajava, proučavamo njihove strukture i izdvajamo nekoliko nama zanimljivih primjera. U posljednjoj točki ove poglavlja bavimo se preslikavanjem između kategorija i demonstriramo praktičnu primjenu teorije kategorija u funkcijskom programskom jeziku Haskell.

1.1 Definicije i osnovni primjeri

Teorija kategorije proučava objekte i preslikavanja između njih. Objekti i preslikavanja su primitivni objekti u teoriji kategorija i njih ne definiramo. Objekti ne moraju biti kolekcije elemenata i preslikavanja ne moraju biti funkcije na skupovima. U ovoj točki definiramo kategorije i detaljno raspisujemo nekoliko primjera.

Zbog preglednosti uvodimo novu oznaku za preslikavanje. Neka su A i B dvije klase. Preslikavanje f, koje svakom $a \in A$ pridružuje jedinstveni $b \in B$ označavamo sa $A \xrightarrow{f} B$. Neka je C proizvoljna klasa i $B \xrightarrow{g} C$, tada sa $A \xrightarrow{f} B \xrightarrow{g} C$ označavamo kompoziciju preslikavanja f i g. Ponekad, zbog jednostavnosti nećemo imenovati preslikavanje i pisati ćemo samo $B \to C$.

Definicija 1.1.

Kategorija $\boldsymbol{G}$ sastoji se od:

- klase $Obj_{\boldsymbol{G}}$ čije elemente nazivamo objekti kategorije $\boldsymbol{G}$
- klase $Arw_{\boldsymbol{G}}$ čije elemente nazivamo strelice kategorije $\boldsymbol{G}$
- preslikavanja $Arw_{\boldsymbol{G}} \xrightarrow{source} Obj_{\boldsymbol{G}}$
- preslikavanja $Arw_{\boldsymbol{G}} \xrightarrow{target} Obj_{\boldsymbol{G}}$
- preslikavanja $Obj_{\boldsymbol{G}} \xrightarrow{id} Arw_{\boldsymbol{G}}$ koje svakom $B \in Obj_{\boldsymbol{G}}$ pridružuje strelicu id_B i vrijedi:
$$target(id_B) = source(id_B) = B.$$
Preslikavanje id nazivamo identiteta kategorije $\boldsymbol{G}$.

- preslikavanja $\circ : Arw_{\boldsymbol{G}} - > Arw_{\boldsymbol{G}} - > Arw_{\boldsymbol{G}}$ takvog da za svaki morfizam $f, g \in Arw_{\boldsymbol{G}}$ postoji morfizam $f \circ g$ za koji vrijedi:

$$source(f) = target(g)$$
$$source(f \circ g) = source(g)$$
$$target(f \circ g) = target(f)$$

  Preslikavanje $\circ$ zovemo kompozicija strelica u $\boldsymbol{G}$.

  Preslikavanje identiteta (id) i kompozicija ($\circ$) moraju zadovoljavati:

  - uvjet identiteta: za svaku strelicu $A \xrightarrow{f} B \in Arw_G$ vrijedi:

$$id_B \circ f = f = f \circ id_A$$

  - uvjet asocijativnost: za sve strelice $A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D$ vrijedi:

$$(h \circ g) \circ f = h \circ (g \circ f) \tag{1.1.1}$$

Za $A, B \in Obj_G$ skup svih strelica sa A u B označavamo sa $\boldsymbol{G}[A, B]$. Ukoliko je iz konteksta jasno o kojoj kategoriji se radi, onda ćemo umjesto $Obj_{\boldsymbol{G}}$ i $Arw_{\boldsymbol{G}}$ pisati samo Obj i Arw. Sada ćemo detaljno raspisati nekoliko primjera kategorija.

Primjer 1.1.

Monoid je uređena trojka $(M, \cdot_M, 1_M)$ gdje je M skup, $1_M \in M$, $\cdot_M$ binarna operacija na M za koju vrijedi da za svaki $a, b, c \in M$ imamo:

$$(a \cdot_M b) \cdot_M c = a \cdot_M (b \cdot_M c)$$

$$1_M \cdot_M a = a = a \cdot_M 1_M$$

Neutralni element 1_M i binarnu operaciju $\cdot_M$ uglavnom ćemo pisati kao 1 i $\cdot$ osim ako iz konteksta neće biti jasno na kojem monoidu su definirani. Kada promatramo kategoriju Monoida, tada su objekti skupovi, a za dva monoida M, N definiramo strelicu $M \xrightarrow{\delta} N$ kao funkciju za koju vrijedi da za svaki $a, b \in M$ imamo:

$$\delta(a \cdot b) = \delta(a) \cdot \delta(b)$$

$$\delta(1) = 1$$

i zovemo je **morfizam**. Pokažimo da je kompozicija dva morfizma ponovno morfizam. Neka su $a, b \in Obj_M$ i $M \xrightarrow{f} N \xrightarrow{g} P$, tada vrijedi:

$$(g \circ f)(a \cdot b) = g(f(a \cdot b)) = g(f(a) \cdot f(b)) = g(f(a)) \cdot g(f(b)) = (g \circ f)(a) \cdot (g \circ f)(b)$$

Za monoid M definiramo identitetu id_M kao standardnu funkciju identiteta, to jest za svaki $a \in Obj_M$ vrijedi:

$$id_M(a) = a$$

Pokažimo sada da tako definirane strelice na monoidu zadovoljavaju svojstvo identiteta i asocijativnosti (1.1.1) za kategorije. Neka je $a \in Obj_M$ i $M \xrightarrow{f} N \in Arw_M$. Tada vrijedi:

$$(id_N \circ f)(a) = id_N(f(a)) = f(a) = f(id_M(a)) = (f \circ id_M)(a)$$

te je svojstvo identiteta zadovoljeno.
Neka su M, N, P, R monoidi, $M \xrightarrow{f} N \xrightarrow{g} P \xrightarrow{h} R \in Arw_M$ i $a \in Obj_M$, tada vrijedi:

$$((h \circ g) \circ f)(a) = h(g(f(a))) = (h \circ (g \circ f))(a).$$

Zbog asocijativnosti kompozicije funkcija tu klasu monoida možemo promatrati kao kategorije. Kategoriju monoida označavamo sa ***Mon***.

Definicija 1.2.

Neka je S skup i $\leq_S$ binarna relacija na S. Uređeni par $(S, \leq_S)$ zovemo **parcijalno uređeni skup** ako za svaki $a, b, c \in S$ vrijedi:

- $a \leq_S a$ (refleksivnost)
- ako $a \leq_S b$ i $b \leq_S a$ tada $a = b$ (antisimetričnost)
- ako $a \leq_S b$ i $b \leq_S c$ tada $a \leq_S c$ (tranzitivnost)

Radi kraćeg zapisa pisat ćemo samo $\leq$ umjesto $\leq_S$ i govoriti o parcijalno uređenom skupu S gdje je implicitno definirana binarna relacija $\leq$.

Neka su S i R dva parcijalno uređena skupa i neka je $R \xrightarrow{f} S$ preslikavanje za koje vrijedi da za svaki $a, b \in S$ imamo: $a \leq b \implies f(a) \leq f(b)$. Tada kažemo da je f **monotono preslikavanje**. Pokažimo da su monotona preslikavanja zatvorena na kompoziciju. Neka je Q parcijalno uređen skup, $S \xrightarrow{f} R \xrightarrow{g} Q$ monotona preslikavanja i $a, b \in S$ takvi da $a \leq b$. Tada vrijedi:

$$a \leq b \implies f(a) \leq f(b) \implies g(f(a)) \leq g(f(b)),$$

to jest

$$a \leq b \implies (g \circ f)(a) \leq (g \circ f)(b).$$

Identiteta na parcijalno uređenom skupu S definirana je kao standardna funkcijska identiteta. Kao i u prethodnom primjeru lako se pokaže da vrijedi:

$$(id_R) \circ f = f = f \circ id_S.$$

$$(h \circ g) \circ f = h \circ (g \circ f),$$

pa možemo govoriti o kategoriji ***Pos*** gdje su objekti parcijalno uređeni skupovi a strelice monotona preslikavanja. Neke matematičke strukture se mogu promatrati kao poseban slučaj kategorija, a upravo su parcijalno uređeni skupovi jedan takav slučaj.

Primjer 1.2. Neka je $(S, \leq)$ parcijalno uređen skup, tada definiramo kategoriju ***Pos(S, $\leq$)*** na slijedeći način:

- objekti kategorije ***Pos(S, $\leq$)*** su elementi skupa S
- ako $a, b \in S$ i $a \leq b$ tada postoji jedinstvena strelica $a \xrightarrow{(b,a)} b$
- ako za $a, b \in S$ ne vrijedi $a \leq b$, tada ne postoji strelica izmedu a i b.

Primijetimo da za $a, b \in S$ ***Pos(S, $\leq$)***$[a, b]$ je jednočlan skup ukoliko vrijedi $a \leq b$, a u suprotnome je prazan, pa u kategoriji ***Pos(S, $\leq$)*** možemo umjesto $a \xrightarrow{(b,a)} b$ pisati $a \leq b$. Pokažimo sad na primjeru prirodnih brojeva da je ***Pos($\mathbb{N}$, $\leq$)*** kategorija.

Neka je $\mathbb{N}$ skup svih prirodnih brojeva. Tada su objekti kategorije ***Pos($\mathbb{N}$, $\leq$)*** prirodni brojevi, a prema definiciji, za $a, b \in \mathbb{N}$ postoji strelica $a \xrightarrow{(b,a)} b$ ukoliko vrijedi $a \leq b$ i tada je očito $source(b, a) = a$ i $target(b, a) = b$. Svakome $a \in \mathbb{N}$ je pridruženo preslikavanje $a \xrightarrow{id_a} a$ za koji vrijedi:

$$source(id_a) = a = target(id_a) = a$$

Kako je relacija $\leq$ tranzitivna, komponiranje strelica u ***Pos($\mathbb{N}$, $\leq$)*** je dobro definirano. Uvjeti identitete i asocijativnosti su zadovoljeni, budući da je $\leq$ refleksivna i tranzitivna relacija, pa je trivijalno provjeriti da ako je $a \xrightarrow{f} b \in Arw$ proizvoljna strelica da tada vrijedi $f = a \leq b$ i imamo:

$$id_b \circ f = (a \leq b) \leq (b \leq b) = (a \leq b) = f = (a \leq a) \leq (a \leq b) = f \circ id_a$$

Posebno je važno ovdje razlikovati kategoriji svih parcijalno uređenih skupova ***Pos***, gdje su objekti kategorije parcijalno uređeni skupovi i kategoriju nad nekim određenim parcijano uređenim skupom ***Pos(S, $\leq$)***, gdje su objekti kategorije elementi parcijalno uređenog skupa.

Definicija 1.3. Neka je $(S, \leq)$ parcijalno uređen skup i $T \subseteq S$. Za $v \in S$ kažemo da je **gornja međa** skupa T ako za svaki $t \in T$ vrijedi $t \leq v$. **Supremum** skupa T je **najmanja gornja međa** tog skupa.

Primjer 1.3.

Pokažimo sada primjer neke kategorije ***G*** gdje su objekti konačni skupovi, a strelica između objekata ne mora biti uobičajena funkcija.

Neka su A, B konačni skupovi, strelica $A \xrightarrow{f} B$ je proizvoljna funkcija: $f : A \times B \to \mathbb{R}$
Za strelice $A \xrightarrow{f} B \xrightarrow{g} C$ definiramo: $g \circ f : A \times C \to \mathbb{R}$ sa

$$(g \circ f)(a, c) = \sum_{y \in B} f(a, y) g(y, c)$$

Lako se vidi da je to kategorija.

Primjer 1.4.

Neka je $\boldsymbol{G}$ neka kategorija. Tada definiramo kategoriju $\boldsymbol{G}^{op}$ koja ima iste objekte kao i $\boldsymbol{G}$, a za svaku strelicu $A \xrightarrow{f} B$ u G dana je dualna strelica $B \xrightarrow{f^{op}} A$ u $\boldsymbol{G}^{op}$. Za strelice $A \xrightarrow{f} B \xrightarrow{g} C$ kompozicija strelica u $\boldsymbol{G}^{op}$ definirana je sa:

$$f^{op} \circ^{op} g^{op} = (g \circ f)^{op}$$

Pokažimo da tako definirana struktura $\boldsymbol{G}^{op}$ zadovoljava uvjete asocijativnosti i identiteta, to jest da je tako zadana struktura stvarno kategorija. Neka je $A \xrightarrow{f} B \xrightarrow{g} C \xrightarrow{h} D \in Arw_G$, tada redom vrijedi:

$$id_B^{op} \circ^{op} f^{op} = (id_B \circ f)^{op} = (f \circ id_A)^{op} = f^{op},$$

i

$$\begin{aligned}
(h^{op} \circ^{op} g^{op}) \circ^{op} f^{op} &= (h \circ g)^{op} \circ^{op} f^{op} \\
&= ((h \circ g) \circ f)^{op} \\
&= (h \circ (g \circ f))^{op} \\
A &= h^{op} \circ^{op} (g \circ f)^{op} \\
&= h^{op} \circ^{op} (g^{op} \circ^{op} f^{op}),
\end{aligned}$$

Pa je $\boldsymbol{G}^{op} kategorija.$

U kasnijim razmatranjima bit će nam potreban pojam podkategorije, pa ga ovdje definiramo.

Definicija 1.4.

Za kategoriju $\boldsymbol{G}$ kažemo da je podkategorija kategorije $\boldsymbol{C}$ ako vrijedi

- $Obj_{\boldsymbol{G}} \subseteq Obj_{\boldsymbol{C}}$
- za sve $A, B \in \boldsymbol{G}$ vrijedi $\boldsymbol{G}[A, B] \subseteq \boldsymbol{C}[A, B]$
- preslikavanje kompozicija i identitata su jednaki u $\boldsymbol{G}$ i $\boldsymbol{C}$

Kažemo da je podkategorija $\boldsymbol{C}$ **puna** ako za svaki $A, B \in \boldsymbol{G}$ vrijedi $\boldsymbol{G}[A, B] = \boldsymbol{C}[A, B]$

Primijetimo da je puna podkategorija jedinstveno zadana svojim objektima.

1.2 Kategorija Hask

Haskell je funkcijski programski jezik nazvan po logičaru Haskellu Curryju. U Haskellu **tip** intuitivno možemo shvatiti kao skup: `Char` je skup svih Unicode znakova (`'A'` , `')'` , `'^'`), `Bool` sadrži samo dva elementa `{True, False}` , `Integer` je beskonačni skup koji sadrži sve cijele brojeve, `Void` je prazan skup ($\emptyset$). Kada u Haskellu zapišemo da je x Integer, to jest: `x :: Integer` to znači da je x element skupa cijelih brojeva. Poseban tip je `()` koji nazivamo unit i čija je jedina vrijednost `()` .

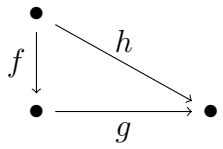
Haskellove funkcije ne možemo poistovjetiti s matematičkim funkcijama jer Haskellove funkcije trebaju izvršiti neki kod - što nije problem ukoliko se može doći do rezultata u konačno mnogo koraka. Ponekad to nije slučaj. Zbog Halting problema ne možemo se ograničiti samo na funkcije kod kojih se može doći do rezultata u konačno mnogo koraka pa se u svaki Haskellov tip dodaje posebna vrijednost: $\perp$ (dno). Ona označava da računanje neće stati, pa tako funkcija $f : Bool \to Bool$ u Haskellu definirana sa: `f :: Bool -> Bool` može vratiti `True` , `False` ili $\perp$. Funkcije koje mogu vratiti $\perp$ zovemo parcijalne funkcije. Zbog specijalnog znaka $\perp$ kategoriju Haskellovih funkcija i tipova razlikujemo od ***Set*** i označavamo sa ***Hask***. Tom distinkcijom dolazimo do nekih komplikacija. Pokazano je da za potrebe ovog rada možemo ignorirati znak $\perp$ i njegove posljedice [4]. Uzevši to u obzir, dajemo pojednostavljenu definiciju kategorije ***Hask***.

Primjer 1.5.

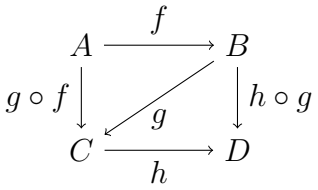
Kategorija ***Hask*** je kategorija Haskellovih tipova i funkcija. U kategoriji ***Hask*** objekti su Haskellovi tipovi koje označavamo velikim slovima: `A, B, C, ...` . Strelice u kategoriji ***Hask*** su Haskellove funkcije koje označavamo malim slovima: `f, g, h, ...` . Strelicu $A \xrightarrow{f} B$ u kategoriji ***Hask*** zapisujemo: `f :: A -> B` . Funkcije `f :: A -> B, g :: A -> B` su jednake ako za svaki `x` vrijedi: `f x = g x` . Kompoziciju $(A \xrightarrow{f \circ g} C)$ zapisujemo: `(f.g) :: A -> C` i definiramo kao standardnu funkcijsku kompoziciju, to jest: `(f.g) x = f (g x)` pa se lako pokaže da vrijedi svojstvo asocijativnosti. Identiteta u kategoriji ***Hask*** dana je funkcijom: `id x = x` , te se lako vidi da vrijedi: `id.f = f = id.f` .

1.3 Dijagram

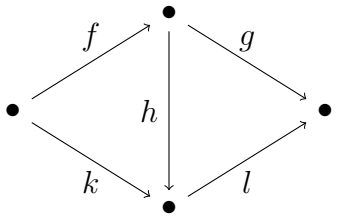
U teoriji kategorija dijagrame koristimo kao reprezentaciju jednadžbi. Za kategoriju $\boldsymbol{G}$ i $A, B \in Obj_G$ već smo vidjeli da preslikavanje $f \in \boldsymbol{G}[A, B]$ označavamo dijagramom: $A \xrightarrow{f} B$. Pogledajmo sljedeći dijagram s tri strelice:



Primijetimo da objektima nismo dali imena (zato što nam to za ovaj primjer nije bitno), ali to ne znači da su ta tri objekta jednaka. Ako vrijedi: $g \circ f = h$. tada kažemo da dijagram **komutira**. Neka su $f, g, h \in Arw_G$. Svojstvo asocijativnosti $(h \circ g) \circ f = h \circ (g \circ f)$ za strelice f, g, h možemo prikazati i slijedećim dijagramom:



Kao i u mnogim granama matematike, u teoriji kategorija ponekad želimo pokazati da su dvije stvari jednake. Rijetko želimo pokazati da su dva objekta jednaka; češće ćemo pokazivati jednakost strelica i za to ćemo koristiti tehniku zvanu praćenje dijagrama (engleski: diagram chasing). Promotrimo slijedeći dijagram:



Dani dijagram ima četiri neimenovana objekta, pet strelica (f, g, h, k, l) i pet strelica nastalih slijedećim kompozicijama:

$$g \circ f, h \circ f, l \circ h \circ f, l \circ h, l \circ k$$

Primijetimo da neke od tih strelica mogu biti jednake. Ovaj dijagram ima tri ćelije: vanjsku (f, k, l, g), lijevi unutarnji trokut (f, h, k) i desni unutarnji trokut (h, g, l). Neke od tih ćelija mogu komutirati:

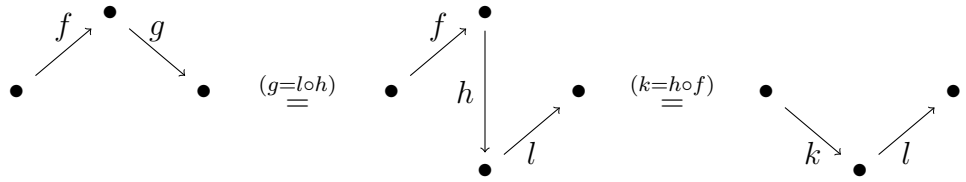
- lijevi trokut komutira ako vrijedi $h \circ f = k$

- desni trokut komutira ako vrijedi $l \circ h = g$
- vanjska ćelija komutira ako vrijedi $g \circ f = l \circ k$

Praćenje dijagrama je proces u kojem pokazujemo da neka ćelija komutira pomoću činjenica da neke druge ćelije komutiraju te nekih drugih svojstava dijagrama.

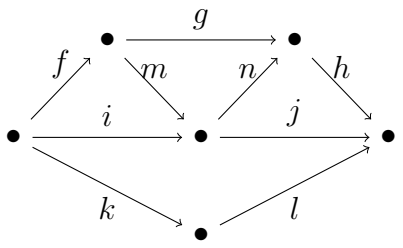
Primjer 1.6.

Pokažimo za prethodni dijagram da ukoliko unutarnji trokuti komutiraju, tada vanjska ćelija komutira, to jest ako vrijedi: $h \circ f = k$ i $l \circ h = g$ tada vrijedi: $g \circ f = l \circ k$. Pokažimo to algebarski: $g \circ f = (l \circ h) \circ f \overset{(1.1.1)}{=} l \circ (h \circ f) = l \circ k$. Tehnikom praćenja dijagrama i uočavanjem da su neke kompozicije jednake imamo:

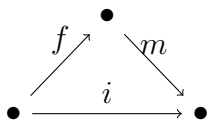


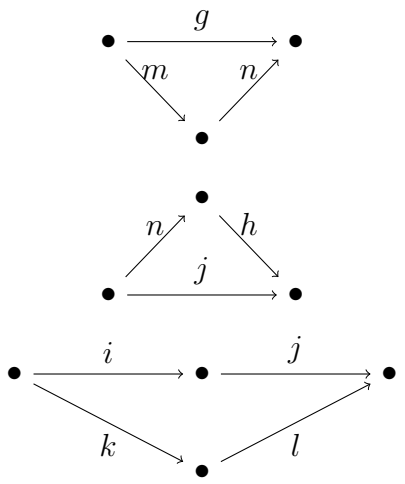
Kod jednostavnijih primjera ne vidi se prednost korištenja tehnike dijagrama. Pokažimo sada na malo kompliciranijem primjeru kako se tehnikom dijagrama intuitivnije može objasniti komutacija dijagrama.

Primjer 1.7.
U ovom primjeru želimo istaknuti prednosti tehnike pračenja dijagrama kod kompleksnijih dijagrama naspram algebraskog pristupa. Zadan je slijedeći dijagram:

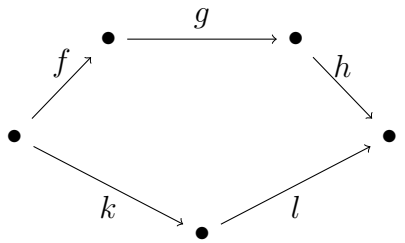


Pretpostavimo da u prethdnom dijagramu komutiraju slijedeća četiri unutarnja trokuta:

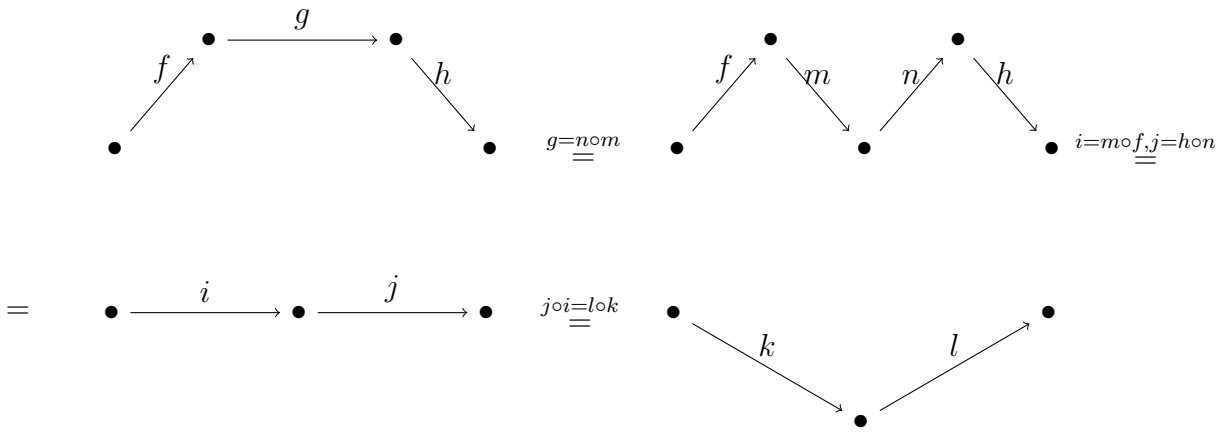




Pokažimo da komutira i vanjska ćelija, koja je prikazana slijedećim dijagramom.



Koristeći tehniku praćenja dijagrama, imamo:



1.4 Monoik i epik

U ovoj točki proširujemo terminologiju i proučavamo posebne slučajeve strelica kako bismo došli do definicije izomorfizma.

Neka je $\boldsymbol{G}$ proizvoljna kategorija te $A, B \in Obj_G$ i $A \xrightarrow{f} B, A \xrightarrow{g} B \in Arw_G$, . Budući da strelice imaju istu domenu i kodomenu, tada pišemo $A \underset{g}{\overset{f}{\rightrightarrows}} B$.

Definicija 1.5.

Neka je $\boldsymbol{G}$ proizvoljna kategorija. Strelicu $B \xrightarrow{m} A \in Arw_G$ zovemo **monoik** ako za svaki par strelica $X \underset{g}{\overset{f}{\rightrightarrows}} B$ vrijedi da $m \circ f = m \circ g$ povlači $f = g$,

Primjer 1.8.

Neka je $\boldsymbol{G}$ neka kategorija čiji objekti su skupovi i neka su strelice $B \xrightarrow{m} A$ funkcije između skupova. Ako je funkcija $m : B \to A$ injektivna, tada je i strelica $B \xrightarrow{m} A$ monoik. Pokažimo to.
Pretpostavimo da vrijedi $m \circ f = m \circ g$ za neki paralelni par strelica $A \underset{g}{\overset{f}{\rightrightarrows}} B$. Kako su f i g funkcije dovoljno je pokazati da za svaki $x \in X$ vrijedi: $f(x) = g(x)$. Imamo: $m(f(x)) = (m \circ f)(x) = (m \circ g)(x) = m(g(x))$. No, kako je m injektivna za $a, b \in B$ imamo: ako vrijedi $m(a) = m(b)$ tada vrijedi $a = b$, pa je m monoik.

Definicija 1.6.

Neka je $\boldsymbol{G}$ proizvoljna kategorija. Strelicu $A \xrightarrow{e} B$ zovemo **epik** ako za svaki par strelica $B \underset{g}{\overset{f}{\rightrightarrows}} X$ vrijedi da $f \circ e = g \circ e$ povlači $f = g$.

Pokažimo sada na sličnom primjeru kao i ranije kako se epik ponaša na kategoriji nad skupovima.

Primjer 1.9.

Neka je $\boldsymbol{G}$ neka kategorija čiji elementi su skupovima i neka su strelice $A \xrightarrow{e} B$ funkcije između skupova. Ako je funkcija $e : A \to B$ surjektivna, da je tada strelica $A \xrightarrow{e} B$ epik. Pokažimo da vrijedi: $f \circ e = g \circ e$ za neki par strelica $B \underset{g}{\overset{f}{\rightrightarrows}} X$. Kako su f i g funkcije dovoljno je pokazati da za svaki $x \in B$ vrijedi: $f(x) = g(x)$. Kako je e surjektivna funkcija znamo da za svaki $b \in B$ postoji $a \in A$ takav da vrijedi: $b = e(a)$. Tada za svaki $b \in B$ vrijedi:

$$f(b) = f(e(a)) = (f \circ e)(a) = (g \circ e)(a) = g(e(a)) = g(b),$$

pa je e epik.

Prethodni primjeri pokazuju da za "dovoljno lijepe" kategorije vrijedi:

$$\text{injekcija} \implies \text{monoik} \qquad \text{i} \qquad \text{surjekcija} \implies \text{epik}.$$

No, za neke kategorije "injektivna strelica" ili "surjektivna strelica" nemaju smisla tako da takva intuitivna interpretacija ima smisla samo za "dovoljno lijepe" kategorije. Pravilnije bi bilo gledati na monoik i epik kao na strelice koje se mogu "poništiti" na jednoj strani. Ako pak strelica ima jednostrani inverz tada dobivamo posebnu klasu monoika i epika.

Definicija 1.7.

Ako za strelice: $B \xrightarrow{s} A, A \xrightarrow{r} B$ vrijedi: $r \circ s = id_B$ tada s nazivamo **sekcija**, a r **retrakcija**. **Bimorfizam** je strelica koja je monoik i epik.

Analogno kao u primjerima 1.8 i 1.9 lako se pokaže da u kategoriji čiji su elementi skupovi, bijektivne funkcije nad skupovima odgovaraju bimorfizmima. Zatim, lako se i pokaže da je svaka sekcija monoik i svaka retrakcija epik.

Definicija 1.8.

Za strelicu $A \xrightarrow{f} B$ kažemo da je **izomorfizam** ako postoji strelica $B \xrightarrow{g} A$ takva da vrijedi:

$$g \circ f = id_A \; i \; f \circ g = id_B$$

Definicija 1.9.

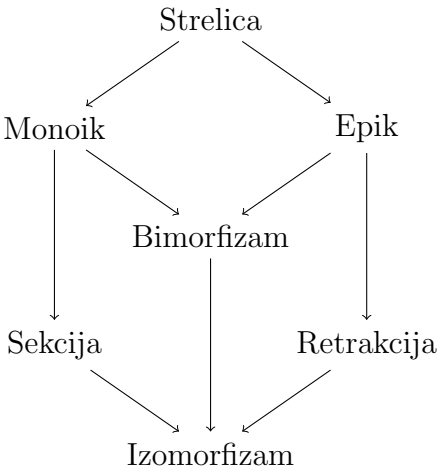
Neka je $\boldsymbol{G}$ proizvoljna kategorija. Za $A, B \in Obj_G$ kažemo da su **izomorfni** ako postoji izomorfizam $A \to B$.

Lako se pokaže da ukoliko je strelica sekcija i epik ili retrakcija i monoik tada je i izomorfizam.

Zbog jednostavnijeg i razumljivijeg pregleda grafički ilustriramo terminologiju definiranu u ovoj točki. Grafički je prikazana veza između svih definiranih pojmova.

- Strelica može biti monoik ili epik
- ukoliko je monoik i epik, onda je nužno izomorfizam
- Ukoliko je strelica retrakcija i monoik tada je i izomorfizam
- Ukoliko je strelica sekcij i epik tada je i izomorfizam

- Ukoliko je strelica bimorfizam i sekcija ili retrakcija tada je i izomorfizam
- Ukoliko je strelica sekcija i retrakcija tada je izomorfizam



1.5 Kategorijski konstruktori

U ovoj točki istražujemo neke osnovne kategorijske konstruktore, to jest objekte koji zadovoljavaju neka pravila definirana u teoriji kategorija. Kako u jeziku kategorija ne gledamo unutrašnju strukturu objekata svi koncepti moraju biti definirani pomoću relacija između objekata.

1.5.1 Inicijalni i finalni objekti

Definicija 1.10.

Za objekt S u kategoriji $\boldsymbol{G}$ kažemo da je **inicijalni** ako za svaki objekt A postoji jedinstvena strelica $S \to A$. Za objekt S u kategoriji $\boldsymbol{G}$ kažemo da je **finalni** ako za svaki objekt A postoji jedinstvena strelica $A \to S$.

Lako se pokaže da ukoliko je objekt I finalni objekt u kategoriji $\boldsymbol{G}$ tada je I inicijalni objekt u $\boldsymbol{G}^{op}$.

Primjer 1.10.

Neka je ***Set*** kategorija skupova i neka je $1 = \{x\}$ skup s jednim elementom. Za svaki skup A postoji jedinstvena strelica $A \to 1$ (funkcija koja preslikava sve u x) pa je 1 finalni objekt za ***Set***. Za svaki skup A postoji jedinstvena strelica $\emptyset \to A$ (gdje je $\emptyset$ prazan skup) pa je $\emptyset$ inicijalni objekt za ***Set***.

Primjer 1.11.

`Void` je inicijalni objekt u ***Hask***. Zbog polimorfizma [1] možemo definirati polimorfnu funkciju: `absurd :: Void -> a` gdje je `a` bilo koji tip u ***Hask***.

Primjer 1.12.

Tip `()` je finalni objekt u ***Hask***. Možemo definirati polimorfnu funkciju `unit :: a -> ()` kao: `unit x = ()`

Kategorija $\boldsymbol{G}$ može imati i finalni i inicijalni objekt, a ako ima oboje onda oni ne moraju biti isti. Objekt koji je i finalni i inicijalni ponekad zovemo **nulti** objekt. Lako se pokaže da ukoliko kategorija ima dva inicijalna objekta tada su oni izomorfni. Stoga ima smisla govoriti o inicijalnom objektu kategorije te analogno o finalnom objektu.

[1] Parametrizirani polimorfizam omogućava da se funkcije pišu generalno bez ovisnosti o tipu, opširnije o tome u točki "Funktori". Za sada možemo funkciju `absurd :: -> a` interpretirati kao skup funkcija {Void → a | a je tip u ***Hask***} gdje odabir funkcije ovisi o kodomeni.

1.5.2 Produkti i koprodukti

Produkt u teoriji kategorija je generalizacija Kartezijevog produkta na skupovima. Prisjetimo se, ako su A, B skupovi tada je Kartezijev produkt $A \times B$ definiran sa: $A \times B = \{(a, b) | a \in A, b \in B\}$.

Primijetimo da je uz takvu definiciju Kartezijevog produkta prirodno definirati dvije projekcije: $p_A : A \times B \to A, p_A(a, b) = a$, te $p_B : A \times B \to B, p_B(a, b) = b$.

Definirajmo sada projekcije u kategoriji ***Hask***. Neka su `a` i `b` proizvoljni tipovi u ***Hask***. Tada sa `fst` i `snd` označavamo funkcije koje su definirane na slijedeći način:

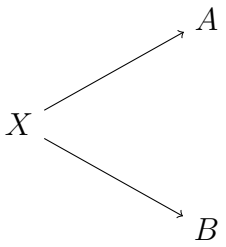
```
fst :: (a, b) -> a
fst (x, y) = x

snd :: (a, b) -> b
snd (x, y) = y
```

Sada smo spremi definirati prvi konstruktor.

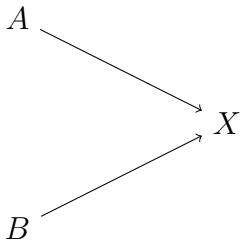
Definicija 1.11.

Neka je $\boldsymbol{G}$ neka kategorije, te $A, B \in Obj_G$. **Grananje prema paru** A, B je objekt $X \in Obj_C$ zajedno sa strelicama $X \xrightarrow{A}$ i $X \xrightarrow{B}$. Dijagramom to možemo prikazati kao:



Definicija 1.12.

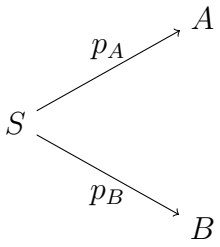
Neka je $\boldsymbol{G}$ neka kategorija te $A, B \in Obj_G$. **Grananje od para** A, B je objekt $X \in Obj_C$ zajedno sa strelicama $A \xrightarrow{X}$ i $B \xrightarrow{X}$. Dijagramom to možemo prikazati kao:



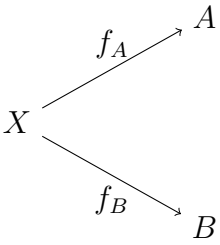
Kada će iz konteksta biti jasno o kojem grananju se radi, govorit ćemo samo o grananju. Sada možemo definirati produkt.

Definicija 1.13.

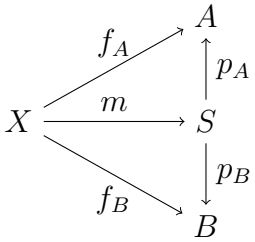
Neka su $A, B \in Obj_G$. **Produkt** od A i B je grananje



sa svojstvom da za svako grananje:



postoji jedinstvena strelica $X \xrightarrow{m} S$ takva da dijagram:



komutira. Tada m zovemo **mediator** za grananje na X.

Primijetimo dvije stvari:

- produkt nije samo objekt, produkt je objekt i dvije strelice
- mediator je jedinstveni za grananje na X

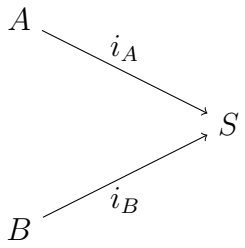
Produkt od A i B, to jest postojanje gore opisanog granjanja ćemo označavati sa $A \times B$. Općenito, definicija produkta se može generalizirati na više od dva člana, ali ona ne doprinosi razmatranju o ovome radu, pa izjednačavamo definiciju binarnog produkta i produkta. Za kategoriju $\boldsymbol{G}$ kažemo da ima produkt (binarni produkt) ukoliko za svaki $A, B \in Obj_G$ postoji produkt (binarni produkt). U kategoriji ***Hask*** možemo definirati funkciju (višeg reda) `factorizer` koja će za bilo koji tip c i dvije projekcije $c \xrightarrow{p} a$ i $c \xrightarrow{q} b$ vratiti jedinstveni mediator. Ovdje je definicija funkcija factorizer.

```
factorizer :: (c -> a) -> (c -> b) -> (c -> (a, b))
factorizer p q = \x -> (p x, q x)
```

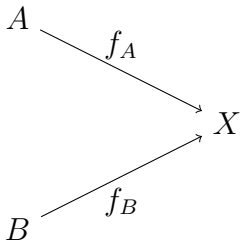
Sada, u slijedećoj definiciji definiramo pojam koprodukta.

Definicija 1.14.

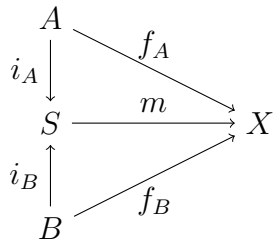
Neka je $\boldsymbol{G}$ neka kategorija. Neka su $A, B \in Obj_G$. **Koprodukt** od A i B je grananje



sa svojstvom da za svako grananje:



postoji jedinstvena strelica $S \xrightarrow{m} X$ takva da dijagram:



komutira.

Koprodukt je dualan pojam produktu, i njegova intepretacija u kategoriji ***Set*** je pojam disjunktne unije. Prisjetimo se; neka je $A = A_i : i \in I$ familija skupova. Disjunktna unija od A je $\bigsqcup_{i \in I} A_i = \bigcup_{i \in I} \{(x, i) : x \in A_i\}$.

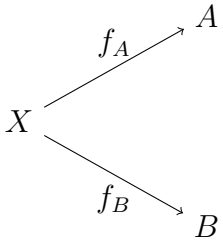
Definicija 1.15.

Za kategoriju ***G*** kažemo da je **Kartezijeva** (skraćeno: "***G*** je **CC**") ako:

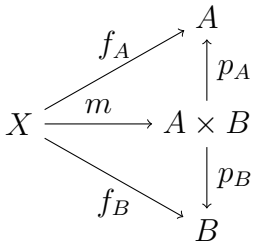
- sadrži inicijalni objekt,
- za svaki par $A, B \in Obj_G$ postoji produkt.

Primjer 1.13.

Za kategoriju ***Set*** već smo pokazali da sadrži inicijalni objekt $\emptyset$. Pokažimo sada da je produkt u kategoriji ***Set***, Kartezijev produkt. Neka su A i B skupovi. Za proizvoljno grananje



na X definiramo $X \xrightarrow{m} A \times B$ kao: $m(x) = (f_A(x), f_B(x))$ za $x \in X$. Lako se pokaže da sljedeći dijagram komutira:



Ostaje nam pokazati jedinstvenost, to jest da je m jedina takva funkcija. Pretpostavimo da postoji neka druga funkcija $X \xrightarrow{h} A \times B$ takva da vrijedi: $p_A \circ h = f_A$ i $p_B \circ h = f_B$. Za $x \in X$ imamo:

$$h(x) = (p_A(h(x), p_B(h(x))) = ((p_A \circ h)(x), (p_B \circ h)(x)) = (f_A(x), f_B(x)) = m(x).$$

Stoga je ***Set*** jedna Kartezijeva kategorija.

Primjer 1.14. Neka su x, y objekti u ***Pos(S, $\leq$)***. Prema definiciji produkta, produkt od x i y je objekt z, zajedno sa strelicama $z \to y$ i $z \to x$. Strelice $z \to y$ i $z \to x$ možemo zapisati i kao $z \leq y$ i $z \leq x$. Definicija produkta zahtjeva i da za bilo koji objekt w iz ***Pos(S, $\leq$)***, $w \to y$ i $w \to x$ postoji strelica $w \to z$, to jest ako $w \leq y$ i $w \leq x$ vrijedi $w \leq z$. S obzirom da je $z \leq y$ i $z \leq x$, z je infimum od x i y. Ovime primjerom smo pokazali da se egistencija produkta u ***Pos(S, $\leq$)*** može svesti na egzistenciju infimuma.

1.6 Funktori

Do sada smo definirali kategorije, posebno i kategoriju ***Hask***, proučavali različite vrste strelica, objekata i pokazali njihove ekvivalente u kategoriji ***Hask***-u. U ovoj točki bavimo se "preslikavanjima" između kategorija i demonstriramo praktičnu primjenu teorije kategorija u Haskellu.

Definicija 1.16.

Neka su $\boldsymbol{C}$ i $\boldsymbol{G}$ kategorije. **Funktor** F iz $\boldsymbol{C}$ u $\boldsymbol{G}$ je preslikavanje koje:

1. svakome objektu $x \in Obj_{\boldsymbol{C}}$ pridružuje jedinstveni objekt $F(x) \in Obj_{\boldsymbol{G}}$
2. svakoj strelici $X \xrightarrow{f} Y \in Arw_{\boldsymbol{C}}$ pridružuje jedinstvenu strelicu $F(X) \xrightarrow{F(f)} F(Y) \in Arw_{\boldsymbol{G}}$ tako da vrijedi:
   (a) $F(id_X) = id_{F(X)}$ za $X \in Obj_{\boldsymbol{C}}$
   (b) $F(g \circ f) = F(g) \circ F(f)$ za sve $X \xrightarrow{f} Y$ i $Y \xrightarrow{g} Z$

Funktori čuvaju strukturu kategorije jer objekti i strelice dobivaju svoje analogone u preslikanoj kategoriji. Od posebne važnosti su funktori sa $\boldsymbol{G}$ u $\boldsymbol{G}$.

Definicija 1.17.

Neka je $\boldsymbol{G}$ proizvoljna kategorija i neka je F funktor sa $\boldsymbol{G}$ u $\boldsymbol{G}$. Tada kažemo da je F **endofunktor** nad $\boldsymbol{G}$.

Definicija 1.18.

Neka je $\boldsymbol{G}$ proizvoljna kategorija te neka je F preslikavanje iz $\boldsymbol{G}$ u $\boldsymbol{G}$ takvo da vrijedi:

- $x = F(x)$ za svaki $x \in Obj_{\boldsymbol{G}}$
- $f = F(f)$ za svaki $f \in Obj_{\boldsymbol{G}}$

Tada je F endofunktor nad $\boldsymbol{G}$ koji zovemo **funktor identitete**.

Funktori u Haskellu usko su povezani s gornjom definicijom funktora. Da bismo mogli objasniti funktore u Haskellu na primjeru napravit ćemo malu digresiju i definirati jedan od najjednostavnijih parametriziranih tipova koji se označava sa `Maybe` .

1.6.1 Maybe

Parametrizirani tip `Maybe` služi za enkapsuliranje opcionalnih vrijednosti. Primjerice, pretpostavimo da želimo napisati funkciju u Haskellu koja će dohvatiti prvi element liste cijelih brojeva. Što će se dogoditi ako je lista prazna?

Tip `Maybe` omogućava nam modeliranje tipa koji može ali i ne mora imati vrijednost. Isto kao što tip `Bool` može imati `True` ili `False` tako `Maybe` može imati vrijednosti `Just a` ili `Nothing`. Primijetimo da, iako formalno `Maybe` može imati dvije vrijednosti, prirodno je vrijednost `Nothing` interpretirati kao da nismo dobili ništa (što može služiti za rješavanje takozvanih rubnih uvjeta).

`Maybe` je parametrizirani konstruktor tipa, što znači da pomoću njega možemo generirati nove tipove. Definicija `Maybe` u Haskellu glasi:

```
data Maybe a = Just a | Nothing
```

Vidimo da pomoću `Maybe a`, gdje je `a` bilo koji tip možemo konstruirati druge tipove koji mogu imati vrijednost ili `Just a` ili `Nothing`. Na primjer `Maybe Integer` može imati vrijednost `Just Integer` ili `Nothing`. Vratimo li se na primjer liste, to znači da naša funkcija koja bi vraćala prvi element liste cijelih brojeva ima tip:

```
head' :: [Integer] -> Maybe Integer
```

Kada primjenimo `head` na praznu lisu dobijemo `Nothing`, dok na nepraznoj dobijemo `Just Int` a. To ilustriramo sa slijedećim izrazima i njegovim rezultatima u Haskellu.

```
>> head' [2, 4, 6, 8, 10]
Just 2
>> head' []
Nothing
```

1.6.2 Funktori u Haskellu

Funktori u Haskellu su preslikavanja iz kategorije ***Hask*** u kategoriju ***Func***, gdje je kategorija ***Func*** podkategorija od kategorije ***Hask***-a. Funktor liste preslikava kategoriju ***Hask*** u kategoriju ***Lst***, gdje kategorija ***Lst*** sadrži samo liste, to jest `[T]` [2] za bilo koji tip `T`. Definicija funktora u Haskellu glasi:

```
class Functor (f :: * -> *) where
  fmap :: (a -> b) -> f a -> f b
```

koji mora zadovoljavati:

```
fmap id = id
fmap (f . g) = fmap f . fmap g
```

Primijetimo da gornja dva uvjeta u Haskellu odgovaraju i zahtjevima nad funktorima u kategorijskom smislu.

[2] `[a]` je konstruktor tipa s jednim parametrom, kao i `Maybe a`

Primjer 1.15.

Pokažimo da je `Maybe a` funktor. Primijetimo da, pošto je `Maybe` konstruktor tipa (s jednim parametrom), bilo koji tip `X` možemo poslati iz ***Hask***-a i dobiti novi tip `Maybe X`. Zaključujemo da `Maybe` preslikava objekte iz jedne kategorije u drugu[3].

Definirajmo sada `fmap` kako bi bili zadovoljeni ostali uvjeti da `Maybe a` bude funktor.

```
instance Functor Maybe where
  fmap f (Just x) = Just (f x)
  fmap f Nothing = Nothing
```

Iz gornje definicije vidimo da za bilo koji $\texttt{A} \xrightarrow{f} \texttt{B} \in Arw_{\textbf{\textit{Hask}}}$ vrijedi: $\texttt{Maybe A} \xrightarrow{\texttt{fmap f}} \texttt{Maybe B}$. Lako se provjeri da sa tako definiranom funkcijom `fmap Maybe a` jest funktor.

Korisna intuicija kod Haskellovih funktora je da oni reprezentiraju tipove preko kojih se može mapirati: to mogu biti liste cijelih brojeva, balansirana stabla, opcionalne vrijednosti (`Maybe a`) ili nešto drugo. Jednostavan primjer mapiranja preko liste cijelih brojeva može se pokazati korištenjem funkcije `double` koja će udvostručiti cijeli broj. Definicija `double` u Haskellu glasi:

```
double :: Double -> Double
double n = n * 2
```

Primijetimo da `double` možemo gledati kao $\texttt{Double} \xrightarrow{\texttt{double}} \texttt{Double}$. Kako je lista cijelih brojeva (`[Double]`) funktor, možemo preslikati `double` u $\texttt{[Double]} \xrightarrow{\texttt{fmap double}} \texttt{[Double]}$. na primjer:

```
>> fmap double [1.0..10.0]
[2.0,4.0,6.0,8.0,10.0,12.0,14.0,16.0,18.0,20.0]
```

Time smo uz pomoć teorije kategorije i funktora eliminirali potrebu za pisanjem petljama koje su neizostavne kod imperativnih programskih jezika. Sama unutarnja implementacija funkcije `fmap` za neki funktor može koristiti petlje, no krajnji korisnik jezika je pošteđen implementacijskih detalja.

[3]Objekti u kategoriji ***Hask*** su tipovi

2 Kartezijanski zatvorene kategorije

U ovoj točki definiramo pojam kartezijanski zatvorene kategorija. To je vrsta kategorije koja ima ekspresivnu moć jednaku tipiziranom lambda računu. Da bi definirali projam zatvorene kategorije nužno je definirati i pojam eksponenta, kategorijskog analogona funkcijskom prostoru između dva skupa. Nakon definicije kartezijanski zatvorenih kategorija navodimo nekoliko najbitnijih primjera.

Definicija 2.1. Neka je $\boldsymbol{G}$ kategorija koji ima produkt i neka su $Z, Y \in Obj_G$. **Eksponencijalni objekt** je objekt Z^Y koji ima svojstvo $Z^Y \times Y \xrightarrow{apply} Z$ ako za bilo koji objekt $X \in Obj_c$ i morfizam $X \times Y \xrightarrow{g} Z$ postoji jedinstveni morfizam $X \xrightarrow{\lambda g} Z^Y$ t.d.

$$X \times Y \xrightarrow{\lambda g \times Y} Z^Y \times Y \xrightarrow{apply} Z \text{ je } g$$

Morfizam *apply* se ponekad u literaturi zove i *eval*. U kategoriji $\boldsymbol{Set}$, eksponencijalan objekt Z^Y je skup svih funkcija $Y \to Z$ gdje jednostavno možemo vidjeti strukturu *apply* funkcije:

$$Z^Y \times Y \xrightarrow{apply} Z \text{ pridružuje uređenom paru } (f, y) \text{ rezultat } f(y).$$

Slično, za preslikavanje $X \times Y \xrightarrow{g} Z$ možemo definirati morfizam $X \xrightarrow{\lambda g} Z^Y$ (tzv. currying) kao:

$$\lambda g(x)(y) = g(x, y)$$

Definicija 2.2. Za kategoriju G kažemo da je **kartezijanski zatvorena kategorija**, skračeno **CCC** (eng. Cartesian closed category), ako vrijedi:

- kategorija $\boldsymbol{G}$ sadrži finalni objekt,
- za svaki par $A, B \in Obj_G$ postoji produkt $A \times B$ u $\boldsymbol{G}$, s projekcijama $\pi_1 : A \times B \to A$ i $\pi_2 : A \times B \to B$
- za svaki par $A, B \in Obj_G$ postoji eksponencijalni objekt B^A u $\boldsymbol{G}$.

Finalni objek u kategoriji $\boldsymbol{G}$ označavamo sa 1.

Primijetimo da smo i gornju definiciju mogli iskazati pomoću prethodno definirane kartezijeve kategorije; kartezijeva kategorija G je kartezijanski zatvorena kategorija, ukoliko za svaki par $A, B \in Obj_G$ postoji eksponencijalni objekt B^A u G.

Primjer 2.1. Pokazali smo da je ***Set*** kategorija i da je $\emptyset$ finalni objekt za Set. Postojanje eksponencijalnog objekta i njegove struktura je opisana u primjeru 1.18. Primijetimo sad još da u ***Set*** postoji i kartezijev produkt.

- Za $A, B \in Obj_{\boldsymbol{Set}}$, $A \times B = \{(a, b) | a \in A, b \in B\}$, to jest $A \times B$ je standardni Kartezijev produkt nad skupovima.

Iz čega zaključujemo da je ***Set*** jedna CCC.

Definicija 2.3. Parcijalno uređeni skup B zovemo **Booleova algebra** ako:

- za sve $x, y \in B$ postoji supremum skupa $\{x, y\}$ kojeg označavamo sa $x \vee y$
- za sve $x, y \in B$ postoji infimum skupa $\{x, y\}$ kojeg označavamo sa $x \wedge y$
- za sve $x, y, z \in b$ vrijedi: $x \vee (y \wedge z) = (x \wedge y) \vee (x \wedge z)$
- B sadrži dva elementa, 0 i 1, gdje je 0 najmanji element u B, a 1 najveći
- za svaki element $x \in B$ postoji $\neg x$ za koji vrijedi: $x \wedge \neg x = 0$ i $x \vee \neg x = 1$. Element $\neg x$ nazivamo **komplement** od x.

Za Booleovu algebru B kažemo još i da je trivijalna ako $1 = 0$.

Budući da je svaka Booleova algebra ujedno i parcijalno uređen skup, označimo s ***Pos(B)*** pripadnu kategoriju Booleove algebre, kao što smo napravili u primjeru 1.2. U sljedećem primjeru pokazujemo da je kategorija ***Pos(B)*** ujedno i CCC.

Primjer 2.2. Pokažimo da kategorija ***Pos(B)*** zadovoljava sve uvjete iz definicije CCC. Finalni objekt je 1, a prema definiciji Booleanove algebre, za svaka dva elementa postoji infimum. Iz primjera 1.14 slijedi da je to dovoljan uvjet za postojanje produkta.

Preostalo nam je za svaki par $A, B \in Obj_{\boldsymbol{Pos(B)}}$ pokazati postojanje eksponencijalnog objekta B^A u ***Pos(B)***. Definiramo da je B^A jednako $\neg A \vee B$. Da bi smo pokazali postojanje *apply* morfizma, moramo pokazati da je i $B^A \wedge A \leq B$. Redom imamo:

$$\begin{aligned} B^A \wedge a \leq b &= (\neg a \vee b) \wedge a \\ &= (\neg a \wedge a) \vee (b \wedge a) \\ &= 0 \vee (b \vee a) \\ &= b \vee a \leq b \end{aligned}$$

Kako bi dokazali egistenciju morfizma λ dokazujemo da vrijedi slijedeće:

za svaki objekt $C \in Obj_{Pos(B)}$ koji ima svojstvo $C \wedge A \leq B$ imamo $C \leq B^A$

U tu svrhu pretpostavimo da za neki objekt $C \in Obj_{\boldsymbol{Pos(B)}}$ vrijedi $C \wedge A \leq B$. Tada imamo

$$\begin{aligned} C &= C \wedge 1 \\ &= C \wedge (A \vee \neg A) \\ &= (C \wedge A) \vee (A \wedge \neg A) \\ &\leq B \vee (A \wedge \neg A) \\ &\leq B \vee (\neg A) = B^A \end{aligned}$$

Čime smo pokazali da je ***Pos(B)*** jedna CCC.

3 Osvrt na $\lambda-$račun

Sredinom 1930-tih, neovisno jedan o drugome, A. Church i A. Turing dali su negativan odgovor na poznati *Entscheidungsproblem*, koji je D. Hilbert zadao 1928. godine. A. Church je došao do negativnog odgovora pomoću $\lambda-$računa, a A. Turing pomoću Turingovih strojeva.

Na Turingovim strojevima bazira se Von Neumannovo računalo, koje je temelj današnje arhitekture računala. Imperativni jezici, poput C-a, Fortrana ili Pascala temeljeni su na načinu na kojem se daju upute Turingovim strojevima, odnosno slijed instrukcija. Suprotno od njih, funkcijski programski jezici, poput ML-a, Mirande ili Haskella bazirani su na $\lambda-$računu i nemaju, još, direktni analogon na arhitekturi računala.

Kako nam netipizirani $\lambda-$račun nije bitan, budući da se ne može povući paralela s teorijom kategorija, u ovoj točki dajemo pregled pojmova iz $\lambda-$računa čije je razumijevanje nužno kako bi se mogla uspostaviti bijekcija s kartezijanskim zatvorenim kategorijama. Detaljniji pregled netipiziranog $\lambda-$računa može se pronaći u [2] i [8]. Dajemo proširenu definiciju jednostavno tipiziranog $\lambda-$računa kako bi se moglo jasnije, i tehnički jednostavnije, opisati veza između $\lambda-$računa i kartezijanski zatvorenih kategorija. Definiramo pojmove kao što su: jednostavni tipovi, termi $\lambda-$računa, supstitucija i slobodne varijable, te dajemo dedukcijska pravila za modificirani jednostavni tipizirani $\lambda-$račun.

Prije nego što krenemo sa definicijama tipiziranog $\lambda-$računa, nužno je definirati alfabet pomoću kojih pišemo izraze λ-računa.

Definicija 3.1 (Alfabet $\lambda-$računa)**.** Alfabet $\lambda-$računa sastoji se od slijedećih simbola:

- oznaka varijabli: $a, b, c, ..$
- apstraktora: λ
- zagrade: ()
- točke: .

Sada ćemo definirati što su tipovi u $\lambda-$računu i njihove oznake.

Definicija 3.2 (Jednostavni tipovi)**.** Neka je T neki prebrojiv skup čije elemente nazivamo *tipske varijable*. Neka su $\rightarrow$ i $\times$ simboli koji nisu elementi skupa T. Skup jednostavnih tipova $T_{\rightarrow}$ definiramo kao najmanji skup riječi nad alfabetom $T \cup \{\rightarrow, \times\}$ koji je nadskup od T, te za svake dvije riječi $A, B \in T_{\rightarrow}$ vrijedi $A \rightarrow B, A \times B \in T_{\rightarrow}$.

Simbol $\to$ intuitivno označava funkciju, pa uvjet da za svaki $A, B \in T$ vrijedi $A \to B$ sugerira nam da za bilo koja dva tipa, postoji tip koji odgovara tipu funkcije iz A u B. Tipovi oblika $A \times B$ namjerno grafički sugeriraju sličnost kategorijskom konceptu produkta, a u idućem poglavlju ćemo vidjeti zašto je to tako.

Zbog jednostavnosti ćemo umjesto $T_\to$ sa T označavati skup jednostavnih tipova. Ako vrijedi $a \in T$, tada pišemo $a : T$ ili a^T i kažemo da je a tipa T. Svaki konačan niz oblika $a_1 : T_1, a_2 : T_2, ..., a_n : T_n, n \in \mathbb{N}$ gdje su $a_1, a_2, ..., a_n$ različite varijable zovemo kontekst i označavamo sa Γ.

Kako je sama definicija tipa različita od uobičajene nužno je proširiti i definiciju terma. Definirajmo sad terme jednostavno tipiziranog $\lambda-$računa.

Definicija 3.3 (Termi jednostavno tipiziranog lambda računa T.)**.** Neka je T skup jednostavnih tipova i neka je $1 \in T$. Tada terme jednostavno tipiziranog $\lambda-$računa T definiramo na sljedeći način:

- postoji term () tipa 1 koji nazivamo jedinica (eng. unit)
- za svaki $A \in T$ postoji prebrojivo varijabli tipa A, a svaka varijabla tipa A je ujedno i term tipa A
- ako su a i b termi tipa A i B, tada postoji term (a, b) tipa $A \times B$
- ako je c term tipa $A \times B$, tada postoji term $p_1(c)$ tipa A i term $p_2(c)$ tipa B
- ako su $t^{A\to B}$ i s^A termi tipa $A \to B$ i A, tada je $t^{A\to B}s^A$ term tipa B
- ako je t^B term tipa B i x^A varijabla tipa A, tada je $(\lambda x^A.t^B)^{A\to B}$ term tipa $A \to B$.

Terme jednostavno tipiziranog lambda računa T označavamo sa λ_T. Term oblika ab nazivamo *aplikacija*, a term oblika $(\lambda a.b)$ *apstrakcija*. Za terme $a, b \in \lambda_T$ ćemo sa $a \equiv b$ označavati grafičku jednakost dva terma.

Prilikom uspostavljanja bijekcije između tipiziranog $\lambda-$računa i kartezijanskih zatvorenih kategorija, term () i tip 1 će nam služiti za reprezentaciju finalnog objekta u kategoriji, dok će projekcije p_1 i p_2 odgovarati kategorijskoj definiciji produkta.

Želimo naglasiti da su sljedeće definicije jednake onima u netipiziranom lambda računu.

Definicija 3.4 (Slobodne varijable)**.** Neka je $a \in \lambda_T$ proizvoljan term. Skup $FV(t)$ slobodnih varijabli terma t definiramo rekurzivno:

- $FV(x) := \{x\}$
- $FV(ts) := FV(t) \cap FV(s)$
- $FV(\lambda x.t) := FV(t) \setminus \{x\}$

Za varijablu x kažemo da je *slobodna varijabla* u termu t, ako je $x \in FV(t)$. U suprotnom kažemo da je x *vezana* u termu t.

Definicija 3.5 (Supstitucija)**.** Neka su s i t termi, te x varijabla. Term dobiven supstitucijom varijable x s termom s u termu t, u oznaci $t[x/s]$, rekurzivno definiramo ovako:

- $x[x/s] := s$
- $y[x/s] := y$ za $y \not\equiv x$
- $(t_1 t_2)[x/s] := t_1[x/s]t_2[x/s]$
- $(\lambda x.t)[x/s] := \lambda x.t$
- $(\lambda y.t)[x/s] := \lambda y.t[x/s]$ za $y \not\equiv x$

Primjer 3.1. Na primjeru terma $f := (\lambda xy.xyz)(x)(y)$ demonstriramo određivanje skupa slobodnih varijabli, te neke supstitucije. Radimo supstituciju terma a za varijablu x u termu f.

$$\begin{aligned} f[x/a] &= (\lambda xy.xyz)(x)(y)[x/a] \\ &= (\lambda xy.xyz)[x/a](x)[x/a](y)[x/a] \\ &= (\lambda xy.xyz)(a)(y) \end{aligned}$$

Svaka jednakost u gornjem računu je dobivena direktnom primjenom definicije 3.5. Analogno, primjenom definicije, dobivamo i skup $FV(f)$ slobodnih varijabli u f.

$$\begin{aligned} FV(t) &= FV((\lambda xy.xyz)(x)(y)) \\ &= FV(\lambda xy.xyz) \cap FV(x) \cap FV(y) \\ &= (FV(xyz) \setminus \{x, y\}) \cup \{x\} \cup \{y\} \\ &= (FV(x) \cup FV(y) \cup FV(z) \setminus \{x, y\}) \cup \{x\} \cup \{y\} \\ &= ((\{x\} \cup \{y\} \cup \{z\}) \setminus \{x, y\}) \cup \{x\} \cup \{y\} \\ &= (\{x, y, z\} \setminus \{x, y\}) \cup \{x\} \cup \{y\} \\ &= \{z\} \cup \{x\} \cup \{y\} \\ &= \{x, y, z\} \end{aligned}$$

U prethodnom primjeru vidjeli smo da imamo dva sintaktički različita terma: $(\lambda xy.xyz)(x)(y)$ i $(\lambda xy.xyz)(a)(y)$. Želimo da njihovo semantičko značenje bude identično. Primijetimo također da vrijedi sljedeće:

$$FV((\lambda xy.xyz)(a)(y)) = \{a, y, z\} \neq \{x, y, z\} = FV(f).$$

U tu svrhu definiramo konverzije, posebno α i β konverzije, koje možemo promatrati kao zamjenu varijable, aplikaciju funkcije i micanje apstrakcije.

Definicija 3.6 (Konverzija)**.** Neka je λ_T skup terma i $conv \in T \times T$ binarna relacija. Ako za $t, s \in T$ vrijedi $t \, conv \, s$ tada kažemo da t konvertira u s, gdje t zovemo *redeksom* (eng. redex), s *kontraktum* (eng. conversum) od t. Zamjenu redeksa s

kontraktumom zovemo *konverzijom t u s*. Pišemo $t \prec_1 s$ ako je s dobiven od t u jednoj konverziji. Relaciju $\prec$ definiramo kao tranzitivno zatvorenje relacije $\prec_1$, a relaciju $\preceq$ kao tranzitivno i refleksivno zatvorenje $\prec$. Analogno definiramo relacije $\succ$ i $\succeq$.

Navodimo sada dva primjera konverzija: $\alpha-$konverzija i $\beta-$redukcija. Neka su $A, B \in T$. Najjednostavniji tip konverzije u $\lambda-$računu je $\alpha-$konverzija, koju definiramo kao:

$$conv_\alpha = \{(\lambda x^A.x^A, \lambda y^A.y^A) : x, y \in A\}.$$

Intuitivno, $\alpha-$konverzija je zamjena simbola koji reprezentira neki term, a najčešće se koristi kako bi se izbjegla kolizija imena.

$\beta-$redukcija je zamjena terma u tijelu funkcije s termima argumentima funkcije, a najčešće se može promatrati kao aplikaciju funkcije na sintatičkoj razini. Uobičajeno se $\beta-$redukcija definira kao:

$$conv_{\beta 1} = \{(\lambda x^A.t^B, t^B[x^A/s^A]) : x, s \in A, t \in B\}.$$

No, budući da smo uveli dodatan konstrukt za tipove u jezik, moramo proširiti uobičajenu definiciju. U tu svrhu definiramo dvije nove relacije:

$$conv_{\beta 2} = \{(p_1(x, y), x) : x, y \in A\} \text{ i } conv_{\beta 3} = \{(p_2(x, y), y) : x, y \in A\}.$$

Konačno, definiramo $\beta-$redukciju kao uniju tih relacije, to jest

$$conv_\beta = conv_{\beta 1} \cup conv_{\beta 2} \cup conv_{\beta 3}$$

Na idućem primjeru pokazujemo kako možemo β-redukciju intuitivno shvatiti kao aplikaciju funkcije na sintaktičkoj razini.

Primjer 3.2. Neka je $g := (\lambda xy.xyz)(a)(y)$. Tada prema definiciji $\beta-$redukcije imamo

$$(\lambda xy.xyz)(a)(y)\, conv_\beta\, (\lambda y.(xyz[x/a]))(y) = (\lambda y.ayz)(y) conv_\beta\, ayz[y/y] = ayz$$

Sada ćemo definirati i semantičku jednakost terma, to jest jednakost po kojoj će termi f i g biti jednaki.

Definicija 3.7 (Jednakost po konverziji)**.** Neka je T skup jednostavnih tipova, $s, t \in \lambda_T$ i $=_{conv} \in \lambda_T \times \lambda_T$. Kažemo da je *term t jednak termu s u odnosu na konverziju* $=_{conv}$ i pišemo $t =_{conv} s$ ako postoji konačan niz terma $t_o, t_1, t_2, .., t_n,.$ tako da vrijedi sljedeće:

- $t_0 \equiv t$
- $t_n \equiv s$
- $t_i\ conv_\beta\ t_{i+1}$ ili $t_{i+1}\ conv_\beta\ t_i$ za $i \in 0, 1, .., n$

Definicija 3.8 (Normalna forma)**.** Neka je $t \in \lambda_T$ term. Za term t kažemo da je u *normalnoj formi* ako ne postoji term $s \in \lambda_T, s \neq t$ takav da vrijedi $t\ conv_\beta\ s$, to jest ako se na term t ne može primijeniti $\beta-$redukcija. Term t ima normalnu formu ako vrijedi $t \succeq s$ i s je u normalnoj formi.

Primjer 3.3. Primijetimo da term $(\lambda x.x)(\lambda x.x)$ nije u normalnoj formi, budući da možemo primjeniti $\beta-$redukciju na sljedeći način:

$$(\lambda x.x)(\lambda x.x) conv_\beta (\lambda x.x[x/(\lambda x.x)]) = (\lambda x.(\lambda x.x)) = (\lambda x.x).$$

Term $(\lambda x.x)$ je u normalnoj formi. U primjeru 3.2 imamo da se na term $(\lambda xy.xyz)(a)(y)$ može primijeniti $\beta-$redukcija, pa nije niti on u normalnoj formi, dok term ayz jeste.

Normalne forme su nam veoma bitne kod uspostavljanja ekvivalenosti terma. Primjerice, želimo da termi $(\lambda xy.xyz)(a)(y)$ i ayz budu ekvivalentni. Normalna forma nas intuitivno upućuje na klasu ekvivalencije terma. No, prije nego što možemo iskazati nešto takvog, veoma je bitno pokazati da redoslijed primjene konverzije nije bitan za konačan rezultat. Da bi to ostvarili dajemo definiciju konfluentne relacije i iskazujemo tereom čiji se dokaz može pronaći u [2].

Definicija 3.9 (Konfluentne relacije / Church-Rosser)**.** Za relaciju R kažemo da je *konfluenta* (eng. confluent) ako za svaki $t_0, t_1, t_2 \in \lambda_T$ za koji vrijedi $t_0 R t_1$ i $t_0 R T_2$, postoji $t_3 \in \lambda_T$ takav da $t_1 R t_3$ i $t_2 R t_3$.

Teorem 3.1. Neka su $t, t' \in \lambda_T$ termi i $\preceq \in \lambda_T \times \lambda_T$ konfluentna relacija. Tada vrijedi:

$$t = t' \text{ ako i samo ako postoji term } t'' \in \lambda_T \text{ takav da } t \preceq t'' \text{ i } t' \preceq t''$$

Posebno, za neki term $a : T$ sa $E(a)$ označavamo klasu ekvivalencija terma a definiranu kao: $E(a) := \{x : T | \exists x' : T \text{ takav da } a \preceq x' \text{ i } x \preceq x'\}$.

Definirajmo sada pravila dedukcije u jednostavno tipiziranom $\lambda-$računu.

Definicija 3.10. Neka je $a \in \lambda_T$ proizvoljan term. Pišemo $\Gamma \vdash a : A$ ako možemo pomoću pravila dedukcije zaključiti da vrijedi $t : A$ uz neki kontekst Γ. Navodimo pravila dedukcije jednostavno tipiziranog lambda računa:

$$\frac{a : A \in \Gamma}{\Gamma \vdash a : A} \qquad \frac{\Gamma \vdash f : A \to B \qquad \Gamma \vdash a : A}{\Gamma \vdash fa : B} \qquad \frac{\Gamma, a : A \vdash b : B}{\Gamma \vdash \lambda a.b : A \to B}$$

Zatim, imamo i pravila dedukcije za produkt:

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B}{\Gamma \vdash (a, b) : A \times B} \qquad \frac{\Gamma \vdash c : A \times B}{\Gamma \vdash\colon p_1(c) : A} \qquad \frac{\Gamma \vdash c : A \times B}{\Gamma \vdash\colon p_2(c) : B}$$

Posebno, imamo trivijalno pravilo dedukcije za jedinicu:

$$\frac{}{\Gamma \vdash : () : 1}$$

4 Veza CCC-a i $\lambda-$računa

Postojanje velike sličnosti između tipova u programskom jeziku i propozicija u intuicionističkoj logici uočili su H. Curry i W. Howard. Pokazali su da tipovi u programskom jeziku odgovaraju propozicijama u intutionističkoj logici i da vrijednosti tipova odgovaraju dokazima propozicija. Danas nam je ta veza poznata kao *Curry-Howard* izomorfizam. T. Coquand je 1988. napisao je račun konstrukcija [3] (eng. *Calculus of constructions*) koji se može promatrati kao proširenje tipiziranog $\lambda-$računa s veoma eskpresivnim sustavom tipova iz kojeg je kasnije nastao programski jezik Gallina čije je najpoznatija implementacija COQ, interaktivni dokazivač teorema. COQ je dizajniran za pisanje matematičkih dokaza, formalne specifikacije programa i provjeru svojstava programa. Koristeći Curry-Howardov izomorfizam COQ-ov algoritam za provjeravanje tipova (eng. *type checker*) može se provjeriti pravilnost dokaza. Jedan od najpoznatijih korištenja COQ-a je formalizacija dokaza *problema četiri boja* teorema čime je možda i najbolje demonstrirana praktična primjena Curry-Howard izomorfizma.

U kartezijanski zatvorenim kategorijama funkcija s dvije varijable $f : X \times Y \to Z$ uvijek se može interpretirati kao funkcija jedne varijable $f' : X \to Z^Y$. Taj proces, zvan *currying*, doveo je do saznanja da se jednostavno tipizirani $\lambda-$račun može interpretirati kao kartezijanski zatvorena kategorija. Vezu između kartezijanski zatvorenih kategorija i $\lambda-$računa prvi je opisao 1980. godine J. Lambek u [5], poznatu kao *Curry-Howard-Lambek* izomorfizam. U ovom poglavlju opisujemo konstrukciju kojom se pokazuje veza između tipiziranog $\lambda-$računa i kartezijanskih zatvorenih kategorija. Formalni dokaz može se pronaći u [6]. Prvo opisujemo strukturu kartezijanski zatvorene kategorije $\boldsymbol{Ł}$, kategoriju čiji unutarnji jezik je $\lambda-$račun. Navodimo primjer u kojem pokazujemo kako jedna jednostavna funkcija nad prirodnim brojevima izgleda u $\lambda-$računu, kako izgleda njezin analogon u teoriji kategorija i kratko razmatramo jedno od najvećih praktičnih prednosti teorije kategorija, a to je ugrađenost kompozicije.

Kada pišemo jednostavno tipizirani $\lambda-$račun mislimo na proširenje jednostavno tipiziranog λ-računa opisano u prošlom poglavlju. U ovom poglavlju, kao i u prethodnom, sa T ćemo označavati skup jednostavnih tipova.

Definicija 4.1. Sada opisujemo strukturu kategorije $\boldsymbol{Ł}$.

- objekti kategorije $\boldsymbol{Ł}$ su tipovi od T
- za objekte $A, B \in Obj_L$ strelica $A \xrightarrow{f} B$ je klasa ekvivalencije terma tipa B, s jednom slobodnom varijablom tipa A, gdje je $source(f) = A$, $target(f) = B$. Posebno, za $A \in Obj_{\boldsymbol{Ł}}$ je strelica identitete id_A, gdje je $a : A$ pripadna slobodna varijabla, jednaka $E(a)$

- neka su $A \xrightarrow{f} B$ $B \xrightarrow{g} C$ dvije strelice u $\boldsymbol{Ł}$ i neka su $a : A$ i $b : B$ neke pripadne slobodne varijable. Kompozicija $g \circ f$ je klasa ekvivalencije dobivena zamjenom slobodne varijable b u termu g sa termom f.

Definicija strelica pomoću klasa ekvivalencija je sasvim prirodna, budući da želimo da dva terma, koji predstavljaju aplikaciju funkcije, $f = \lambda x^A.y^B$ i $g = \lambda c^A.x^B$ budu jednaka i u kategorijskom smislu, to jest da vrijedi $E(f) = E(g)$. Primijetimo također da je i strelica identitete dobro definirana, budući da za terme $a, b : A$ trivijalno vrijedi $E(a) = E(b)$. Stoga možemo strelicu identitete promatrati kao skup svih varijabli nekog tipa.

Primijetimo da je kompozicija strelica dobro definirana budući da je to zapravo supstitucija terma. Ako je $b : B$ term s jednom slobodnom varijablom $x : A$, a $c : C$ term s jednom slobodnom varijablom $y : B$, tada možemo pretpostaviti da se term b može zamijeniti za varijablu y u termu c. U suprotnom, možemo zamijeniti varijablu x s nekom slobodnom varijablom u termu c i ponoviti supstituciju.

Jednostavno je pokazati da je $\boldsymbol{Ł}$ kategorija. Svakome $A \in Obj_L$ pridružujemo strelicu $A \xrightarrow{id_A} A$. Tada trivijalno vrijedi $source(id_A) = A = target(id_A)$. Uvjet identitete, da za svaku strelicu $A \xrightarrow{f} B \in Arw_{\boldsymbol{Ł}}$ vrijedi $id_B \circ f = f = f \circ id_A$, svodi se na dokazivanje jednakost klasa ekvivalencija. Pokažimo da vrijedi $id_B \circ f = f$. Neka je sa $b : B$ označena slobodna varijabla u id_B, a sa $a : A$ slobodna varijabla u f. Kompozicija $id_B \circ f$ je klasa ekvivalencije dobivena zamjenom varijable b s termom f u termu id_B. No to je zapravo direktna primjena supstitucije $id_B[b/f]$, a to je prema definiciji 3.5 jednako termu f. Analogno se pokazuje da vrijedi $f = f \circ id_A$. Uvjet asocijativnosti je zadovoljen budući da, prema Church-Rosser teoremu, redoslijed zamjena varijabli ne utječe na krajnji rezultat.

Propozicija 4.1. Kategorija $\boldsymbol{Ł}$ je CCC.

Dokaz nećemo pisati; J. Lambek i P. J. Scott su na konstruktivan način dokazali da je tako definirana kategorija ujedno i CCC[6]. No, primijetimo da je jedinica 1 finalni objekt u kategoriji $\boldsymbol{Ł}$, budući da za svaki $A \in Obj_{Ł}$ postoji trivijalna strelica $A \to 1$. Jedinstvenost se jednostavno pokaže. Pretpostavimo da postoje dvije različite strelice $A \xrightarrow{f} 1$ i $A \xrightarrow{g} 1$. Neka je $1_f : 1$ term s jednom slobodnom varijablom $a_f : A$ i neka je $1_g : 1$ term s jednom slobodnom varijablom $a_g : A$. Kako je tip 1 trivijalan, to jest sadrži samo jednu vrijednost (), zaključujemo da vrijedi $1_f = () = 1_g$, pa posebno vrijedi i $E(1_f) = E(1_g)$. Za objekte $A, B \in Obj_{Ł}$, postoji tip $A \times B$ u T, pa postoji i $A \times B \in Obj_{Ł}$ i projekcije p_1 i p_1, što odgovara kategorijskoj definiciji produkta.

Kaže se i da je jednostavno tipizirani $\lambda-$račun unutarnji jezik kartezijanski zatvorenih kategorija. Opišimo kako se tipizirani $\lambda-$račun može interpretirati u jeziku kategorija pomoću interpretacijske funkcije ϕ. Najjednostavnija je interpretacija jednostavnih tipova:

- $\forall A \in T, \phi(A) = A \in Obj_C$,

- $\phi(1) = 1_C$,
- $\forall A, B \in T, \phi(A \to B) = \phi(B)$,
- $\forall A, B \in T, \phi(A \times B) = \phi(A) \times \phi(B)$,

gdje vidimo da svaki tip u jednostavno tipiziranom $\lambda-$računu odgovara objektu, a jedinica finalnom objektu kategorije. Analogno, tipovi konstruirani simbolima $\to$ i $\times$ odgovaraju eksponencijalnom objektu i produktu. Posebno, svaka varijabla $a : A$ se interpretira kao morfizam $1 \xrightarrow{\phi(a)} \phi(A)$. Nadalje, interpretaciju proširujemo na kontekst. Kontekst $\Gamma = a_1 : A_1, a_2 : A_2, ..., a_n : A_n$ interpretiramo kao objekt $\phi(A_1) \times \phi(A_2) \times ... \times \phi(A_n)$, a prazan kontekst interpretiramo kao finalni objekt, 1. Tvrdnje iz jednostavno tipiziranog $\lambda-$računa, naravno, imaju interpretaciju u kartezijanski zatvorenim kategorijama. Tvrdnja $\Gamma \vdash a : A$ se interpretira kao morfizam $\phi(\Gamma) \xrightarrow{\phi(\Gamma \vdash a:A)} \phi(A)$. Interpretacije vezane uz produkt su:

- $\phi(\Gamma \vdash (a, b) : A \times B)$ se interpretira kao $\phi(\Gamma) \xrightarrow{\phi(\Gamma \vdash a:A, \Gamma \vdash b:B)} \phi(A) \times \phi(B)$
- $\phi(\Gamma \vdash p_1(a) : A)$ se interpretira kao $\phi(\Gamma) \xrightarrow{p_1 \circ \phi(\Gamma \vdash a:A \times B)} \phi(A)$
- $\phi(\Gamma \vdash p_2(b) : B)$ se interpretira kao $\phi(\Gamma) \xrightarrow{p_2 \circ \phi(\Gamma \vdash b:A \times B)} \phi(B)$

Preostalo nam je još interpretirati apstrakciju i aplikaciju funkcije. Interpretacija aplikacije funkcije $\phi(\Gamma \vdash fa : B)$ je $\phi(\Gamma) \xrightarrow{e \circ (\phi(\Gamma \vdash f:A \to B), \phi(\Gamma \vdash a:A))} \phi(B)$, gdje je strelica $\phi(A \to B) \times \phi(A) \xrightarrow{e} \phi(B)$ morfizam *apply* za pripadni eksponencijalni objekt $\phi(B)^{\phi(A)}$. Konačno, apstrakciju funkcije $\phi(\Gamma \vdash \lambda a.b : A \to B)$ interpretiramo kao $\phi(\Gamma) \xrightarrow{\phi(\Gamma, a:A \vdash b:B}) \phi(B)^{\phi(A)}$. Primijetimo da su gore opisane interpretacije u skladu s intuicijom; aplikacijom funkcije $f : A \to B$ na term $a : A$ dobijemo objekt $\phi(B) \in Obj_{\mathbb{L}}$, dok apstrakcijom $\lambda x.t$ termova tipa $x : A$ i $t : B$ dobijemo objekt $\phi(B)^{\phi(A)}$. Detaljnija definicija interpretacijske funkcije može se pronaći u [6].

Na primjeru jedne jednostavnije funkcije nad prirodnim brojevima pokazat ćemo kako ona izgleda definirana u jeziku $\lambda-$računa i teoriji kategorija.

Primjer 4.1. Neka je $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ definirana kao:

$$f(x, y) = x^2 + 3xy.$$

U jednostavnom tipiziranom $\lambda-$računu funkciju možemo zapisati kao:

$$f = \lambda x.\lambda y.x^2 + 3xy,$$

što je veoma slično tradicionalnom zapisu. Ukoliko bi tu funkciju pokušali zapisati u jeziku kategorija, dobili bi:

$$\mathbb{N} \times \mathbb{N} \xrightarrow{<p_1, p_1, p_1, p_2, 3>} \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \xrightarrow{* \times * \times id} \mathbb{N} \times \mathbb{N} \times \mathbb{N} \xrightarrow{id \times *} \mathbb{N} \times \mathbb{N} \xrightarrow{+} \mathbb{N},$$

što je dosta kompleksan zapis cijelog procesa računanja. Prva strelica $< p_1, p_1, p_1, p_2, 3 >$ simbolizira odvajanje dva inicijalna argumenta u uređenu petorku (3 reprezentira

konstantu, broj 3), druga strelica $* \times * \times id$ simbolizira množenje izraza $3xy$, treća strelica $id \times *$ simbolizira kvadriranje x^2 i konačno, strelica + simbolizira zbrajanje dvije varijable. Primijetimo da se gornja formula može alternativno zapisati kao:

$$p_1^2 + 3p_1p_2$$

gdje p_1 i p_2 označavaju pripadne projekcije. Vidimo da, barem na ovom jednostavnom primjeru, jedina očita razlika između zapisa je pišemo li p_1 i p_2 umjesto x i y. No, prava razlika je u tome pišemo li proces ili rezultat, budući da je iz kategorijskog oblika vidljivo da se dva množenja u funkciji mogu izvesti paralelno. Kategorijski pristup čini se prirodan za istraživanje takvih pitanja.

U kategorijskom pristupu ne postoji mogućnost kolizije varijabli, budući da one ne postoje i samim time ih ne treba imenovati. Ipak, jedna od najvećih prednosti teorija kategorija je ugrađenost kompozicije. Budući da je u samoj definiciji kategorije jasno kako nas zanimaju objekti i veze između tih objekata (morfizmi), a ne unutarnja struktura tih objekata, kategorije nam omogućavaju da se u potpunosti fokusiramo na kompoziciju morfizama. Mnogi najkorisniji programski paketi u programskom jeziku Haskell direktno su inspirirani konceptima u kategoriji. Programski paket *pipes* [19] na praktičan način omogućava pisanje programa s kompleksnim tokovima podataka i definira sučelja koja se jednostavno mogu komponirati s već postojećim programima. Postoji još mnogo primjera koji su, barem djelomično, inspirirani teorijom kategorija, a posebno se još ističe programski paket *lens* [20] koji, kroz puno naprednije koncepte nego što su navedeni u ovom radu, omogućava manipulaciju sa strukturama podataka na puno ekspresivniji način nego što je to moguće u imperativnim programskim jezicima.

Bibliografija

[1] M. Barr, C. Wells, *Category Theory for Computing Science*, 1998, http://www.tac.mta.ca/tac/reprints/articles/22/tr22.pdf

[2] H. P. Barendregt, *The Lambda Calculus — Its Syntax and Semantics*, North-Holland, Amsterdam, 1985

[3] T. Coquand, H. Gerard, *The Calculus of Constructions*, Information and Computation (p.95-120), 1998

[4] N. A. Danielson, J. Hughes, P. Jansson, J. Gibbonsy, *Fast and Loose Reasoning is Morally Correct*, ACM, New York, 2006

[5] J. Lambek, *From λ-calculus to Cartesian closed categories*, Academic Press, London, 1980

[6] J. Lambek, P. J. Scott *Introduction to Higher-Order Categorical Logic*, Cambridge University Press, 1968

[7] B. C. Pierce, *Types and Programming Languages*, MIT Press, 2002

[8] L. Rožić, *Funkcijsko programiranje*, 2014 https://www.math.pmf.unizg.hr/sites/default/files/pictures/rozic-funkcijsko-programiranje.pdf

[9] H. Simmons, *An Introduction to Category theory*, 2010, http://www.cs.man.ac.uk/~hsimmons/zCATS.pdf

[10] https://softwarefoundations.cis.upenn.edu/plf-current/MoreStlc.html, pristupljeno 25.07.2018.

[11] http://www.andrew.cmu.edu/user/awodey/catlog/notes/notes2.pdf, pristupljeno 11.08.2018.

[12] http://scienceblogs.com/goodmath/2006/08/10/from-lambda-calculus-to-cartes/, pristupljeno 29.07.2018.

[13] http://keio-ocw.sfc.keio.ac.jp/letters/01B-001_e/lecture_contents/e-2-2.pdf, pristupljeno 29.07.2018.

[14] https://en.wikibooks.org/wiki/Haskell/Category_theory, pristpljeno 28.02.2018

[15] https://wiki.haskell.org/Type,
pristupljeno 28.02.2018.

[16] http://bartoszmilewski.com/2014/11/24/types-and-functions/,
pristupljeno 25.02.2018.

[17] http://bartoszmilewski.com/2015/01/07/products-and-coproducts/,
pristupljeno 25.02.2018.

[18] http://bartoszmilewski.com/2015/01/20/functors/,
pristupljeno 25.02.2018.

[19] https://hackage.haskell.org/package/pipes-4.3.9,
pristupljeno 15.08.2018

[20] https://hackage.haskell.org/package/lens,
pristupljeno 15.08.2018

Sažetak

U ovom diplomskom radu proučavamo teoriju kategorija s posebnim fokusom na kartezijanski zatvorene kategorije, jednostavno tipizirani $\lambda-$račun i opisujemo vezu između njih koja je poznata kao *Curry-Howard-Lambekov* izomorfizam.

Dan je uvod u teoriju kategorija, te kroz brojne primjere definiramo osnovne pojmove nužne za razumijevanje *Curry-Howard-Lambekov* izomorfizma. Koristimo programski jezik *Haskell* za demonstraciju praktične primjene teorija kategorija i lakše razumijevanje nekih koncepata.

Definirane su kartezijanski zatvorene kategorije, vrsta kategorije koja ima ekspresivnu moć jednaku tipiziranom $\lambda-$računu i pokazano je nekoliko primjera kartezijanski zatvorenih kategorija.

Definirano je proširenje jednostavno tipiziranog $\lambda-$računa s pravilima dedukcije i osnovni pojmovi vezani za tipizirani $\lambda-$račun.

U posljednjem poglavlju je opisana konstrukcija kojom se pokazala veza između tipiziranog $\lambda-$računa i kartezijanski zatvorenih kategorija.

Rad završavamo s primjerom kojim se pokazuje razmjenjivost funkcija u λ-računu s projekcijama u kartezijanski zatvorenim kategorijama. Dajemo kratki ostvrt na prednosti kategorijskog pristupa u programiranju pozivajući se na nekoliko najčešće korištenih programskih paketa.

Summary

In this master thesis we study category theory with special focus on cartesian closed categories and simply typed $\lambda-$calculus. We describe the relationship between them, known as *Curry-Howard-Lambek* isomorphism.

An introduction to the category theory and neccesary terms required for understanding the *Curry-Howard-Lambek* isomorphism are given. Programming language Haskell is used to demonstrate practical applications of category theory.

We define cartesian closed categories, type of category which has expressive power equivalent to the simply typed $\lambda-$calculus and demonstrate few examples of cartesian closed categories.

In the last chapter we define a construction that describes relationship between simply typed $\lambda-$calculus and cartesian closed categories.

Master thesis is finalized with an example which demonstrates interchangeability of functions in simply typed $\lambda-$caluclus with projections in cartesian closed categories. A short overview with advantages of categorical approach to computation is given, demonstrating practicality with few well known programming libraries.

Životopis

Rođen sam 17. srpnja 1990. godine u Koprivnici. Prvih šest razreda osnovne škole završavam u Osnovnoj školi Malešnica, a zadnja dva razreda u Osnovnoj školi Ludbreg. Završavam Elektrostrojasku školu u Varaždinu 2009. godine, a 2010. godine upisujem studij matematike na Prirodoslovno matematičkom fakultetu u Zagrebu, gdje 2014. godine stječem akademski naziv sveučilišnog prvostupnika matematike. Iste godine upisujem sveučilišni diplomski studij Računarstva i matematika na Prirodoslovnom matematičkom fakultetu koji završavam 2018. godine.