Jednostavno tipizirani lambda-račun

Hren, Miho

Other / Ostalo

Publication year / Godina izdavanja: **2023**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:095271

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-26**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Jednostavno tipizirani λ-račun

Miho Hren

Voditelj: *doc. dr. sc. Ante Đerek*

Suvoditelj: *prof. dr. sc. Mladen Vuković*

Zagreb, svibanj 2023.

Sadržaj

1. Uvod

Na putu prema Curry–Howardovoj korespondenciji, prvo smo se u [1] bavili programiranjem u netipiziranom λ-računu. Zatim smo se u [2] bavili intuicionističkom logikom. Ovaj rad posvećujemo jednostavno tipiziranom λ-računu (engl. *simply typed λ-calculus*, STLC), kao zadnjem koraku prije (propozicijskog) Curry–Howarda.

Na početku ćemo spomenuti par nedostataka netipiziranog λ-računa. Zatim ćemo proučiti vrste matematičkih problema koji se javljaju u STLC-u. Nakon toga dajemo pregled ključnih teorema o STLC-u, kako općenitih, tako vezanih uz normalizaciju i konfluentnost.

Ovaj rad temelji se na knjizi [4] koja je fokusirana na praktični dio STLC-a, dok je teorijski pristup naglašen u knjizi [5].

1.1. Nedostatci netipiziranog λ-računa

Iako je jezik (netipiziranog) λ-računa jednostavan, njime možemo iskazati mnogo toga. Prisjetimo se, λ-term može biti jednog od tri oblika:

1. varijabla (x, y, z, ...),
2. aplikacija MN, za neke terme M i N, ili
3. apstrakcija $\lambda x.M$, za varijablu x i term M.

Može se pokazati da su ova tri pravila dovoljna za prikaz svih izračunljivih funkcija [3]. Međutim, ovaj sustav ima i neintuitivne aspekte, a navodimo njih tri.

Prva neintuitivna pojava jest samoaplikacija, što možemo vidjeti u termu

$$\omega = \lambda x.xx.$$

S programske perspektive, term ω prima neki objekt, a potom primjenjuje taj objekt nad samim sobom. Pitanje samoaplikacije jest filozofsko — na što bismo

mogli odgovoriti “Samoaplikacija je znanje o samom sebi.”, no trebali bismo dublje promisliti.

Usko vezano uz samoaplikaciju jest i problem normalizacije. Ponovimo, za term kažemo da je u normalnoj formi ako ga ne možemo više reducirati. No, term $\Omega = \omega\omega$ se reducira na samoga sebe:

$$\omega\omega = (\lambda x.xx)(\lambda x.xx) \to (\lambda x.xx)(\lambda x.xx) = \omega\omega,$$

pa ga možemo beskonačno mnogo puta reducirati, odnosno možemo beskonačno računati. Upravo ova pojava omogućuje Turing-potpunost netipiziranog λ-računa, što će reći, sve Turing-izračunljive funkcije mogu se prikazati u λ-računu i za svaki λ-term postoji Turingov stroj koji ga računa. Moglo bi se reći da λ-račun opisuje intuitivan pojam izračunljivosti.

Konačno, po teoremu o fiksnoj točki, za proizvoljan term T postoji fiksna točka. Štoviše, postoji kombinator Y koji za ulazni term vraća fiksnu točku tog terma, odnosno vrijedi

$$YT \to T(YT).$$

Teorem o fiksnoj točki protivi se stečenoj matematičkoj intuiciji, primjerice, funkcija $f(x) = x + 1$ nema fiksnu točku — ako bi imala, tada bi bilo $0 = 1$.

Pokušajmo sada riješiti ove probleme neformalnim uvođenjem jednostavnih tipova. Za term M izjavit ćemo da je tipa σ, a pisati $M : \sigma$. U našem modelu postoje samo osnovni tipovi (primjerice nat i bool) i funkcijski tipovi (primjerice $\mathsf{nat} \to \mathsf{nat}$ i $\mathsf{bool} \to (\mathsf{bool} \to \mathsf{bool})$). Pogledajmo sada redukciju:

$$\omega M \equiv (\lambda x.xx)M \to MM.$$

U aplikaciji MM, term M ima ulogu i funkcije i argumenta. Kako je term M funkcija, a prima term tipa σ, tada tip od M mora biti $\sigma \to \alpha$, za neki tip α. No, kako smo prije izjavili da je M tipa σ, tada mora biti $\sigma = \sigma \to \alpha$. Ovu “jednadžbu” možemo beskonačno primjenjivati:

$$\sigma = \sigma \to \alpha = \sigma \to (\sigma \to \alpha) = \cdots = \sigma \to (\sigma \to (\sigma \to (\dots))),$$

čime smo dobili “beskonačan” tip, odnosno funkciju koja prima beskonačno mnogo argumenata, što u našem modelu nije moguće. Kasnije ćemo formalizirati nepostojanje samoaplikacije i pokazati da ne vrijedi teorem o fiksnoj točki, te da svi termi imaju normalnu formu.

1.2. Jednostavni tipovi

Analogno pojmu varijable, za osnovne tipove koristimo *tipske varijable*; prebrojiv skup $\mathbb{V}$. Osnovne tipove označavamo slovima s početka grčkog alfabeta (α, β, γ, itd.), dok jednostavne tipove označavamo slovima s kraja grčkog alfabeta (σ, τ, itd.).

Definicija 1. Skup jednostavnih tipova $\mathbb{T}$ definiramo rekurzivno:

1. svaki osnovni tip je jednostavni tip,
2. ako su σ i τ jednostavni tipovi, tada je i $\sigma \to \tau$ jednostavan tip.

Primjer 1. Za osnovne tipove α i β, možemo konstruirati jednostavne tipove $\alpha \to \alpha$ i $(\alpha \to \beta) \to \alpha \to \beta$. Za "streličaste" tipove (engl. *arrow types*) smatramo da su desno asocijativni, odnosno $\alpha \to (\alpha \to \alpha)$ je isto što i $\alpha \to \alpha \to \alpha$.

Definicija 2. Skup pred-tipiziranih (engl. *pre-typed*) $\lambda_{\to}$**-terma**, u oznaci $\Lambda_{\mathbb{T}}$, definiramo rekurzivno:

- svaka varijabla je term;
- ako su M i N termi, tada je i (MN) term;
- ako je x varijabla, σ jednostavan tip i M neki term, tada je $(\lambda x : \sigma.M)$ također term

Primjer 2. Pred-tipizirani $\lambda_{\to}$-termi nisu uvelike drugačiji od netipiziranih λ-terma:

1. x je $\lambda_{\to}$-term jer je varijabla,
2. $\lambda x : \alpha.x$ je $\lambda_{\to}$-term jer je nastao apstrakcijom terma x nad varijablom x tipa α,
3. $(\lambda x : \alpha.x)x$ je $\lambda_{\to}$-term jer je nastao aplikacijom terma $\lambda x : \alpha.x$ na term x,
4. $\lambda x : \alpha.\lambda y : \beta.z$ je $\lambda_{\to}$-term jer je nastao dvostrukom apstrakcijom.

Naziv *pred-tipizirani term* govori nam da je za neke terme potrebna dodatna informacija kako bismo odredili njihov tip. Iz prethodnog primjera, jasno je da je term $\lambda x : \alpha.x$ tipa $\alpha \to \alpha$, međutim tip terma $\lambda x : \alpha.\lambda y : \beta.z$ nije u potpunosti jasan jer nedostaje informacija o varijabli z. Zbog ove pojave ne

možemo promatrati terme same za sebe, već ih moramo promatrati u nekom *kontekstu*.

Prisjetimo se, u netipiziranom λ-računu terme smo konstruirali pomoću varijabla, apstrakcija i aplikacija. U jednostavno tipiziranom λ-računu, a po prethodnom komentaru, konstrukcija više nije jednostavna, zato što smo unijeli informaciju o tipu u term. *Dozvoljene* $\lambda_\to$-terme konstruirat ćemo pomoću *izvoda*.

1.3. Pravila izvoda sistema $\lambda_\to$

Kako bismo mogli govoriti o izvodima, prvo moramo definirati par osnovnih pojmova.

Definicija 3. Izjava (engl. *statement*) je uređeni par (M, σ) gdje je M $\lambda_\to$-term, a σ je jednostavni tip. Pišemo $M : \sigma$, a čitamo "term M je tipa σ". Kažemo još da je u izjavi $M : \sigma$ term M subjekt, a σ je njegov tip.

Deklaracija (engl. *declaration*) je izjava čiji je subjekt varijabla.

Kontekst (engl. *context*) je niz deklaracija s različitim subjektima.

Prosudba (engl. *judgement*) je uređena trojka (Γ, M, σ), gdje je Γ kontekst, M je $\lambda_\to$-term, a σ je jednostavan tip. Pišemo $\Gamma \vdash M : \sigma$, a čitamo "term M tipa σ je izvodljiv iz konteksta Γ".

Definicija 4. Sistem $\lambda_\to$ zadan je sljedećim pravilima izvoda:

$$\frac{}{\Gamma_1, x : \sigma, \Gamma_2 \vdash x : \sigma}\ (var)$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ (appl)$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma . M : \sigma \to \tau}\ (abst)$$

Pravilo (var) govori da smijemo koristiti sve varijable koje znamo "od prije" te da varijable ne mijenjaju tip. Funkcijska interpretacija pravila $(appl)$ je prirodna. Primjerice, za funkciju $f : \mathbb{N} \to \mathbb{R}$ i varijablu $x \in \mathbb{N}$, vrijednost $f(x)$ bit će realan broj, odnosno $f(x) \in \mathbb{R}$. Logičku interpretaciju pravila $(appl)$ vidimo fokusiranjem na tipove, čime dobivamo pravilo eliminacije konjunkcije. Konačno, pravilo $(abst)$ je dualno pravilu $(appl)$. Na primjer, ako za proizvoljni $x \in \mathbb{R}$ vrijedi $g(x) \in \mathbb{Z}$, tada je $g : \mathbb{R} \to \mathbb{Z}$. S logičke perspektive, pravilo $(abst)$ odgovara pravilu introdukcije kondicionala.

Ovdje nećemo definirati pojam izvoda jer je sasvim analogan definiciji izvoda u sistemu prirodne dedukcije uz razliku da su ovdje čvorovi stabla izvoda prosudbe. Gdje će biti potrebno radi dokazivanja, koristit ćemo strukturalnu indukciju po pravilima izvoda, što je zapravo indukcija po visini stabla izvoda.

Primjer 3. Izvedimo term $\lambda x : \alpha.x$ u praznom kontekstu.

$$\dfrac{\dfrac{}{x : \alpha \vdash x : \alpha}\,(var)}{\vdash \lambda x : \alpha.x : \alpha \to \alpha}\,(abst)$$

Na kraju izvoda dobili smo prosudbu $((), \lambda x : \alpha.x, \alpha \to \alpha)$, čime smo potkrijepili tvrdnju s kraja prošlog odjeljka, odnosno term $\lambda x : \alpha.x$ je doista tipa $\alpha \to \alpha$. Napominjemo, ako su tipovi α i β različiti, tada su i termi $I_\alpha \coloneqq \lambda x : \alpha.x$ i $I_\beta \coloneqq \lambda x : \beta.x$ različiti, jer je tip prvog $\alpha \to \alpha$, a tip drugog je $\beta \to \beta$.

Primjer 4. Pokušajmo sada izvesti term $T \coloneqq \lambda x : \alpha.\lambda y : \beta.z$ iz praznog konteksta. U tu svrhu, valja primijetiti da proizvoljan dozvoljen term nastaje primjenom točno jednog pravila izvoda. Također, kako je z varijabla, možemo joj pridijeliti neki tip, recimo σ. Sada ćemo "izvod" pisati obrnutim smjerom, od tvrdnje prema pretpostavkama (kao što se matematika inače radi).

$$\dfrac{\dfrac{\vdash \lambda x : \alpha.\lambda y : \beta.z : \alpha \to (\beta \to \sigma)}{x : \alpha \vdash \lambda y : \beta.z : \beta \to \sigma}\,\text{obrnuti}\ (abst)}{x : \alpha, y : \beta \vdash z : \sigma}\,\text{obrnuti}\ (abst)$$

S ovim obrnutim izvodom više ne možemo nastaviti, jer nam se deklaracija $z : \sigma$ ne javlja u kontekstu. Zaključujemo da term T možemo izvesti samo iz konteksta koji sadrži deklaraciju $z : \sigma$. Na sličan način izveli bismo term $\lambda z : \sigma.T$ iz praznog konteksta.

Definicija 5. Za term M kažemo da je **dozvoljen** (engl. *legal*) ako postoji kontekst Γ i tip σ takvi da vrijedi $\Gamma \vdash M : \sigma$. Za term M kažemo da ga je **moguće tipizirati** (engl. *the term is typable*), ako postoji tip σ takav da vrijedi $\vdash M : \sigma$.

Primjer 5. Term T iz Primjera 4 je dozvoljen, jer u kontekstu $(z : \sigma)$ ima tip $\alpha \to \beta \to \sigma$, ali ga nije moguće tipizirati. Term $\lambda z : \sigma.T$ iz istog primjera je moguće tipizirati, a tip mu je $\sigma \to \alpha \to \beta$. Svaka varijabla je dozvoljen term zbog pravila (var), ali nijednu varijablu nije moguće tipizirati.

Primjer 6. Dokažimo da je term $\lambda x : \alpha.\lambda y : \beta.x$ moguće tipizirati te da mu je tip $\alpha \to \beta \to \alpha$.

$$\dfrac{\dfrac{\dfrac{}{x : \alpha, y : \beta \vdash x : \alpha}\,(var)}{x : \alpha \vdash \lambda y : \beta.x : \beta \to \alpha}\,(abst)}{\vdash \lambda x : \alpha.\lambda y : \beta.x : \alpha \to \beta \to \alpha}\,(abst)$$

Doista, ovim izvodom dobili smo prosudbu $\vdash \lambda x : \alpha.\lambda y : \beta.x : \alpha \to \beta \to \alpha$ čime je tvrdnja dokazana.

Konačno, spomenimo i drugi način tipiziranja terma. Naime, do sada smo kod izvoda *eksplicitno* spominjali tipove varijabli kako bismo došli do konačne prosudbe. Zbog toga smo mijenjali sintaksu λ-terma da bismo dobili $\lambda_{\to}$-terme. Ovakav način tipiziranja zove se Church-tipiziranje (imenovano po Alonzu Churchu) i ono je eksplicitno, apriorno. Drugi način tipiziranja naziva se Curry-tipiziranje (imenovano po Haskellu Curryju), a ono je implicitno, odnosno aposteriorno. U Curry-tipiziranju radimo s običnim λ-termima nad kojima pokušavamo odrediti tip terma bez znanja o tipovima varijabli.

Za programske jezike često se koristi kombinacija Church- i Curry-tipiziranja. Uzmimo kao primjer programski jezik Haskell (koji je također imenovan po Haskellu Curryju). Funkciju identiteta `id` i njen tip možemo definirati eksplicitno:

```
id :: a -> a
id x = x
```

Istu tu funkciju možemo definirati i implicitno:

```
id x = x
```

Za implicitno definirane tipove, Haskell kompajler prilikom kompilacije mora odrediti tip odgovarajuće funkcije.

Napomenimo još da je ovako definirana funkcija `id` polimorfna, odnosno njen tip nije $\alpha \to \alpha$, već $\forall\alpha.\alpha \to \alpha$. Drugim riječima, funkcija `id` je funkcija dva argumenta. Prvi argument je proizvoljan tip α, a drugi argument je neki x tipa α. Dakle, funkcija `id` je zapravo indeksirana familija funkcija $\{\mathsf{id}_\alpha : \alpha \to \alpha\}$. Polimorfne tipove, odnosno terme, ne možemo konstruirati u jednostavno tipiziranom λ-računu, već u sistemu $\lambda 2$, odnosno "polimorfnom λ-računu".

2. Problemi u teoriji tipova

Jednostavno tipizirani λ-račun je podgrana matematike, odnosno logike, koja se naziva teorija tipova (koja je opet podgrana teorije dokaza). Neke druge teorije tipova su račun konstrukcija (implementiran kao Coq) te ranije spomenuti polimorfni λ-račun i njegovo proširenje, Hindley–Milnerov tipski sustav (implementiran kao Haskell). U teorijama tipova općenito postoje termi i pridijeljeni im tipovi, a zajedničko svim teorijama tipova su (formalni) problemi koji se javljaju kao posljedica strukture tipova i terma. Nekoliko njih neformalno opisujemo u nastavku te dajemo primjere navedenih problema u jednostavno tipiziranom λ-računu.

Često prilikom programiranja u strogo statički tipiziranom jeziku kompajler javlja "grešku o tipu". To znači da postoji varijabla koja bi trebala biti jednog tipa, ali je u stvarnosti drugog tipa. Na primjer, za liniju `int a = "a";`, Java kompajler javlja grešku. Naime, varijablu `a` deklarirali smo kao cijeli broj, a potom smo ju definirali kao niz znakova `"a"`, što nema smisla. Kompajler je znao da ovakva definicija nema smisla, jer je proveo *provjeru tipa*. Temeljem cijelog programa (odnosno *konteksta*), kompajler je provjerio je li *tip* `int` adekvatan za vrijednost (odnosno *term*) `"a"`, te se ispostavilo da nije. Dakle, **problem provjere tipa** (engl. *type checking*) je pitanje je li dani term danog tipa u danom kontekstu, formalnije zapisano

$$\mathsf{Kontekst} \overset{?}{\vdash} \mathsf{Term} : \mathsf{Tip}.$$

U nekim slučajevima želimo mogućnost automatskog zaključivanja tipa terma na temelju njegove definicije. Ovaj problem je sličan problemu provjere tipa. Naziva se **problem dodjele tipa** (engl. *type assignment*), a odgovara na pitanje kojeg je tipa dani term u danom kontestu, formalnije zapisano

$$\mathsf{Kontekst} \vdash \mathsf{Term} : ?.$$

Bitna razlika između ovog i problema provjere tipa jest što problem provjere tipa daje da/ne odgovor, dok problem dodjele tipa daje traženi tip, ako on postoji.

Haskell kompajler rješava ovaj problem ako ne navedemo tip funkcije. Ako pak navedemo tip funkcije, Haskell kompajler će rješavati problem provjere tipa. Neki funkcijski jezici, primjerice Haskell i Agda, nude programeru mogućnost da prilikom programiranja koriste "tipizirane rupe" (engl. *typed hole*). Time programer može, za danu varijablu u programu, zatražiti pomoć od kompajlera da mu pokaže kojeg bi tipa ta varijabla trebala biti, što je još jedna instanca problema dodjele tipa.

Problem dodjele tipa možemo generalizirati tako da je i kontekst nepoznat. Time smo dobili **problem dobre tipiziranosti** (engl. *well-typedness*), koji odgovara na pitanje postoji li kontekst i term takav da je u tom kontekstu dani term tog tipa, formalnije zapisano

$$? \vdash \mathsf{Term} : ?.$$

Ovaj problem svodi se na pitanje je li dani term dozvoljen. S druge strane, na pitanje je li dani term moguće tipizirati odgovara problem dodjele tipa uz prazan kontekst.

Posljednji problem kojeg razmatramo javlja se u dokazivanju teorema pomognuto računalom. Naime, matematičke teoreme moguće je reprezentirati tipovima, a jednom kad znamo tipsku reprezentaciju teorema, dovoljno je pronaći term tog tipa čime je dokazano da je teorem istinit (zašto je tome tako, preopširno je pitanje za ovaj rad). Ovdje se javlja **problem nastanjenosti tipa** (engl. *type inhabitation*), koji odgovara na pitanje postoji li term koji je danog tipa u danom kontekstu, formalnije zapisano

$$\mathsf{Kontekst} \vdash ? : \mathsf{Tip}.$$

Za dokazivanje teorema konteksta mora biti prazan, inače dokazujemo neku implikaciju koja ima traženi teorem kao konzekvens. U praksi algoritam za problem nastanjenosti tipa implementiraju samo dokazivači teorema jer u "svakodnevnom" programiranju nema potrebe za takvim alatom. Štoviše, današnji dokazivači teorema (primjerice Coq i Agda, zapravo asistenti u dokazivanju teorema) pretežito koriste algoritme provjere tipova, a za pronalazak terma se oslanjaju na "stvarnog" dokazivača teorema, čovjeka. Jedna od posljedica "snage" teorija tipova koje se koriste pri dokazivanju teorema je njihova neodlučivost, pa algoritmi za nastanjenost tipa niti ne postoje.

3. Osnovna svojstva

U ovom poglavlju dokazujemo jednostavna, ali fundamentalna, svojstva sistema $\lambda_{\rightarrow}$. Prvo dokazujemo lemu o generiranju koju ćemo koristiti u mnogim drugim dokazima. Zatim dokazujemo lemu o slobodnim varijablama, koja tvrdi da ne možemo zaključiti nešto što nije u kontekstu. Konačno dokazujemo jedinstvenost tipova, a kao jednostavan korolar i nepostojanje samoaplikacije.

Lema 1. (O generiranju.) Pretpostavimo da vrijedi $\Gamma \vdash M : \sigma$.

1. Ako je M varijabla x, onda je $(x : \sigma) \in \Gamma$.
2. Ako je M aplikacija PQ, onda je $\Gamma \vdash P : \tau \rightarrow \sigma$ i $\Gamma \vdash Q : \tau$ za neki tip τ.
3. Ako je M apstrakcija $\lambda x : \tau.P$, onda je $\sigma = \tau \rightarrow \rho$ za neki tip ρ te vrijedi $\Gamma, x : \tau \vdash P : \rho$.

Dokaz. Oblik terma M ovisi o posljednjem koraku izvoda. Uvidom u pravila izvoda sistema $\lambda_{\rightarrow}$ vidimo da varijable, aplikacije i apstrakcije možemo izvesti samo redom pravilima (var), $(appl)$ i $(abst)$. Primjerice, aplikaciju PQ očito možemo izvesti samo pomoću pravila $(appl)$, pa po definiciji pravila mora biti $\Gamma \vdash P : \tau \rightarrow \sigma$ i $\Gamma \vdash Q : \sigma$ za neki tip τ. Analogno i u ostalim slučajevima. Napominjemo da u slučaju apstrakcije možemo bez smanjenja općenitosti pretpostaviti da x nije u domeni od Γ. □

Lema o generiranju potvrđuje da pravila izvoda sistema $\lambda_{\rightarrow}$ možemo primjenjivati ne samo unaprijed, nego i unatrag. Time smo opravdali primjere izvoda u ranijim poglavljima gdje smo ispitivali dozvoljenost terma.

Lema 2. (O slobodnim varijablama.) Ako $\Gamma \vdash M : \sigma$, onda $FV(M) \subseteq dom^1(\Gamma)$.

[1]Domena konteksta, u oznaci $dom(\Gamma)$ je skup svih subjekata koji se javljaju u deklaracijama konteksta.

Dokaz. Česta je praksa tvrdnje nad rekurzivno definiranim strukturama dokazivati matematičkom indukcijom. Tako postupamo i ovdje, a radi ilustracije indukciju po M ovdje provodimo u potpunosti.

Slučaj kada je M varijabla ujedno je i baza indukcije. Pretpostavimo $\Gamma \vdash x : \sigma$. Po lemi o generiranju je $(x : \sigma) \in \Gamma$, a kako je $FV(x) = \{x\}$, očito je $FV(x) \subseteq dom(\Gamma)$.

U slučaju aplikacije, neka je $\Gamma \vdash PQ : \sigma$. Po lemi o generiranju vrijedi $\Gamma \vdash P : \tau \to \sigma$ i $\Gamma \vdash Q : \tau$. Po pretpostavci indukcije imamo $FV(P) \subseteq dom(\Gamma)$ i $FV(Q) \subseteq dom(\Gamma)$. Kako je $FV(PQ) = FV(P) \cup FV(Q)$, očito mora biti $FV(PQ) \subseteq dom(\Gamma)$.

U slučaju apstrakcije, neka je $\Gamma \vdash (\lambda x : \tau.P) : \sigma$. To je po lemi o generiranju $\Gamma, x : \tau \vdash P : \rho$, gdje je ρ takav da vrijedi $\sigma = \tau \to \rho$. Ovdje vrijedi pretpostavka indukcije, odnosno $FV(P) \subseteq dom(\Gamma) \cup \{x\}$. Tada je $FV(\lambda x : \tau.P) = FV(P) \setminus \{x\}$, pa iz ove činjenice i prethodne rečenice slijedi $FV(\lambda x : \tau.P) \subseteq dom(\Gamma)$.

Time je tvrdnja dokazana indukcijom po M. U nastavku rada cijelu indukciju provodimo samo gdje je prikladno. □

Lema 3. (O jedinstvenosti tipova.) Ako vrijedi $\Gamma \vdash M : \tau$ i $\Gamma \vdash M : \sigma$, onda je $\tau = \sigma$.

Dokaz. Traženu tvrdnju dokazujemo indukcijom po duljini terma M. Razmatramo slučajeve kada je M redom varijabla, aplikacija, i apstrakcija.

Za varijablu x imamo $\Gamma \vdash x : \tau$ i $\Gamma \vdash x : \sigma$. Po lemi o generiraju je onda $(x : \tau) \in \Gamma$ i $(x : \sigma) \in \Gamma$. Jer je Γ niz deklaracija s različitim subjektima, a u deklaracijama $(x : \tau)$ i $(x : \sigma)$ javlja se isti subjekt, mora biti $\tau = \sigma$.

Za aplikaciju PQ imamo $\Gamma \vdash PQ : \tau$ i $\Gamma \vdash PQ : \sigma$, pa po lemi o generiranju vrijedi $\Gamma \vdash P : \rho_1 \to \tau$ i $\Gamma \vdash Q : \rho_1$ te $\Gamma \vdash P : \rho_2 \to \sigma$ i $\Gamma \vdash Q : \rho_2$. Po pretpostavci indukcije na izvod terma Q tada vrijedi $\rho_1 = \rho_2$, pa je i $\rho_1 \to \tau = \rho_2 \to \tau$. Po pretpostavci indukcije na izvod terma P imamo $\rho_1 \to \tau = \rho_2 \to \sigma$, iz čega slijedi $\rho_2 \to \tau = \rho_2 \to \sigma$, pa mora biti i $\tau = \sigma$.

Za aplikaciju $\lambda x : \rho.P$ imamo $\Gamma \vdash (\lambda x : \rho.P) : \tau$ i $\Gamma \vdash (\lambda x : \rho.P) : \sigma$. Kako je riječ o apstrakciji, mora biti $\tau = \rho \to \mu_1$ i $\sigma = \rho \to \mu_2$. Opet po lemi o generiranju imamo $\Gamma, x : \rho \vdash P : \mu_1$ i $\Gamma, x : \rho \vdash P : \mu_2$, pa po pretpostavci indukcije na izvod terma P mora biti $\mu_1 = \mu_2$, a samim time je i $\rho \to \mu_1 = \rho \to \mu_2$, odnosno $\tau = \sigma$. □

Korolar 1. Ne postoje kontekst Γ, term M i tip σ takvi da vrijedi $\Gamma \vdash MM : \sigma$.

Dokaz. Pretpostavimo suprotno, odnosno da takvi Γ, M i σ postoje. Tada je $\Gamma \vdash MM : \sigma$, pa po lemi o generiranju imamo $\Gamma \vdash M : \tau \to \sigma$ i $\Gamma \vdash M : \tau$. Po lemi o jedinstvenosti tipova tada tipovi τ i $\tau \to \sigma$ moraju biti jednaki, što je očita kontradikcija. $\square$

4. Redukcija i normalizacija

U ovom poglavlju želimo ispitati računalnu snagu sistema $\lambda_{\to}$. U tu svrhu definiramo relaciju β-redukcije i pokazujemo da je sistem $\lambda_{\to}$ značajno slabiji od netipiziranog λ-računa.

Definicija 6. Relacija **jednokoračne** β**-redukcije**, u oznaci $\to_\beta$, je najmanja relacija na skupu dozvoljenih $\lambda_{\to}$-terma koja ima iduća svojstva:

1. $(\lambda x : \sigma.M)N \to_\beta M[x := N]$,
2. ako vrijedi $M \to_\beta N$, onda vrijedi i
   - $ML \to_\beta NL$,
   - $LM \to_\beta LN$,
   - $(\lambda x : \sigma.M) \to_\beta (\lambda x : \sigma.N)$,

   za svaki term L i svaki jednostavni tip σ.

Relacija β-**redukcije**, u oznaci $\twoheadrightarrow_\beta$, je refleksivno i tranzitivno zatvorenje jednokoračne β-redukcije.

Napomena. Terme oblika $(\lambda x : \sigma.M)N$ zovemo *redeksima* (od engl. *reducible expression, redex*).

Kao i u slučaju netipiziranog λ-računa, relacija β-redukcije hvata intuitivan pojam računanja. Uzmimo na primjer funkciju sljedbenika $s : \mathbb{N} \to \mathbb{N}$. Znamo da je $s(x) = x + 1$, pa je $s(1) = 1 + 1 = 2$. Kada smo tražili vrijednost $s(1)$, prvo smo trebali uvrstiti $x = 1$ u tijelo funkcije s. Nakon toga smo trebali primijeniti definiciju binarnog operatora $+$, a na kraju smo dobili vrijednost 2. Jasno je da termi[1] $s(1)$, $1 + 1$ i 2 reprezentiraju isti objekt, no očito su oni sintaksno različiti. Računanje je upravo zaključivanje da su dva sintaksno različita objekta semantički jednaka, a u jednostavno tipiziranom λ-računu, ovaj pojam je formaliziran u

[1] U općenitom smislu.

relaciji β-redukcije. Ovdje valja primijetiti da je term 2 "jednostavniji" od terma $1 + 1$ jer u njemu nemamo što računati.

Definicija 7. Za dozvoljen term M kažemo da je u β-normalnoj formi ako se ne može dalje reducirati.

Znamo da u netipiziranom λ-računu postoje termi čija normalna forma ne postoji, što konceptualno omogućuje beskonačno računanje. Poznato je i da postoji strategija redukcije koja reducira svaki term na njegovu normalnu formu, ako ona postoji. Međutim, u jednostavno tipiziranom λ-računu, vrijedi i jača tvrdnja.

Kako bi dokazali da za svaki $\lambda_{\rightarrow}$ term postoji normalna forma prvo definiramo neke pomoćne pojmove i dokazujemo jednu lemu.

Definicija 8. Stupanj tipa, u oznaci $\delta(\tau)$ definiramo rekurzivno na način:

- $\delta(\alpha) = 0$, za sve tipske varijable α,
- $\delta(\tau \rightarrow \sigma) = 1 + \max(\delta(\tau), \delta(\sigma))$, za sve jednostavne tipove τ i σ.

Stupanj redeksa $\Delta \equiv (\lambda x : \tau.M)N$, u oznaci $\delta(\Delta)$, je stupanj tipa $\tau \rightarrow \sigma$, gdje je σ tip terma M.

Stupanj terma M definiramo kao $\delta(M) = \max\{\delta(\Delta) \mid \Delta \text{ je redeks u } M\}$. Ako M ne sadrži redekse, tada je $\delta(M) = 0$.

Lema 4. Neka je M term. Redukcija najdesnijeg redeksa stupnja $\delta(M)$ u M smanjuje broj redeksa stupnja $\delta(M)$ ili smanjuje stupanj terma.

Dokaz. [2] Neka je n_M broj redeksa stupnja $\delta(M)$ u M, te neka je Δ najdesniji od njih. Term M' dobivamo redukcijom redeksa Δ u M. Ako tom redukcijom nisu nastali novi redeksi, dokaz je gotov. Inače razmatramo načine na koje su novi redeksi mogli nastati.

Ako je $\Delta \equiv (\lambda x : \sigma.P)Q$ i term P ima više od jedne slobodne pojave varijable x, tada se svaki redeks iz Q pojavljuje više od jednom u M'. Međutim, kako je Δ najdesniji redeks stupnja $\delta(M)$ u M, svi redeksi u Q su stupnja strogo manjeg od $\delta(M)$, pa su takvi i novonastali redeksi.

U slučaju da P ima nula ili jednu slobodnu pojavu varijable x novi redeksi mogu nastati samo tako da je neka ne-apstrakcija A u okolini AB redukcijom Δ postala apstrakcija.

Ako je A varijabla koja je redukcijom Δ postala apstrakcija, to je jedino moguće kada imamo $\Delta \equiv (\lambda x : \sigma. \dots xP \dots)(\lambda y : \tau.Q)$. Term Q je nekog tipa μ

[2] U dokazu smatramo da je zadan neki kontekst te da svaki term ima tip.

pa mora biti $\sigma = \tau \to \mu$ i $P : \tau$. Redukcijom dobivamo $\ldots (\lambda y : \tau.Q)P \ldots$, čiji je stupanj $\delta(\tau \to \mu)$ koji je očito manji od stupnja redeksa Δ.

Ako A nije varijabla, tada je nužno A neka aplikacija. Kad bi Δ bio pravi podterm od A, redukcijom Δ term A ne bi postao apstrakcija. Kada bi A bio pravi podterm od Δ, isto tako A ne bi postao apstrakcija. Dakle, mora biti $A \equiv \Delta$. Kako B mora imati neki tip, recimo μ, tada mora biti $\Delta : \mu \to \sigma$ za neki tip σ. Kako je Δ aplikacija, a ima streličasti tip, to je moguće samo ako je lijevi aplikant dvostruka apstrakcija, a desni proizvoljni term, ili pak je lijevi aplikant identiteta, a desni je jednostruka apstrakcija.

U prvom slučaju imamo $\Delta \equiv (\lambda x : \tau.\lambda y : \mu.R)P$, gdje je $R : \sigma$ i $P : \tau$. Tada redukcijom Δ dobivamo term $(\lambda y : \mu.R_1)B$, gdje je $R_1 : \sigma$, a stupanj tako dobivenog redeksa je $\delta(\mu \to \sigma) < \delta(\tau \to \mu \to \sigma) = \delta(\Delta)$.

U drugom slučaju imamo $\Delta \equiv (\lambda x : \tau.x)(\lambda y : \mu.P)$. Tada mora biti $\tau = \mu \to \sigma$ i $P : \sigma$. Redukcija stvara redeks $(\lambda y : \mu.P)B$ stupnja $\delta(\mu \to \sigma) < \delta((\mu \to \sigma) \to (\mu \to \sigma)) = \delta(\Delta)$.

U svakom slučaju, novonastali redeksi su stupnja manjeg od $\delta(M)$, a redeks Δ nestaje. Ako je Δ bio jedini redeks stupnja $\delta(M)$, tada je stupanj novonastalog terma manji od $\delta(M)$. Inače se broj redeksa stupnja $\Delta(M)$ smanjio za jedan. □

Teorem 1. Svaki $\lambda_\to$-term ima normalnu formu.

Dokaz. Neka je M proizvoljan term. Označimo $\delta_M = \delta(M)$ i neka je n_M broj redeksa stupnja δ_M u termu M.

Tvrdnju dokazujemo indukcijom po leksikografski uređenim parovima $\mathbf{m}_M = (\delta_M, n_M)$. Za $\mathbf{m} = (0, 0)$ tvrdnja slijedi trivijalno jer je tada M već u normalnoj formi. Pretpostavimo da tvrdnja vrijedi za sve $\mathbf{m}' < \mathbf{m}_M$. Sa Δ označimo najdesniji redeks stupnja δ_M u M. Po Lemi 4 reduciranjem redeksa Δ u M dobivamo term M' za kojeg je ili $\delta_{M'} < \delta_M$ ili $n_{M'} < n_M$. U oba slučaja je $\mathbf{m}_{M'} < \mathbf{m}_M$, pa za M' vrijedi pretpostavka indukcije, odnosno M' ima normalnu formu N. Tada očito $M \twoheadrightarrow_\beta N$, pa i M ima normalnu formu. □

Već ovdje vidimo značajnu razliku između netipiziranog i jednostavno tipiziranog λ-računa. Pokazuje se da niti redukcijska strategija nije važna, odnosno redekse možemo reducirati proizvoljnim redoslijedom i svejedno na kraju dobivamo normalnu formu terma.

Teorem 2. Svaka redukcijska strategija reducira proizvoljan $\lambda_\to$-term na njegovu β-normalnu formu. Kažemo još da je relacija $\twoheadrightarrow_\beta$ *jako normalizirajuća.*

Dokaz. Izlazi iz opsega ovog rada, a može se pronaći u [5]. □

Za posljedicu ovo teorema imamo da jednostavno tipizirani λ-račun gubi računalnu snagu. Neformalno, za neke izračunljive funkcije ne znamo ako stanu ili računaju beskonačno. S druge strane, "računanje" s $\lambda_{\to}$-termima uvijek staje, pa očito ne možemo simulirati jače sustave pomoću sistema $\lambda_{\to}$. Na kraju ovog poglavlja još ćemo pokazati da ne postoji kombinator fiksne točke.

Kao i većina tvrdnji prije i sljedeća lema se lako dokazuje indukcijom po duljini terma. Uočimo sličnosti leme s *let* izrazima u funkcijskim jezicima. Naime, izraz poput `let x = Q in P` je posve ekvivalentan izrazu `P` u kojemu su sve pojave izraza `x` zamijenjene izrazom `Q`.

Lema 5. Ako $\Gamma, x : \tau \vdash P : \sigma$ i $\Gamma \vdash Q : \tau$, onda $\Gamma \vdash P[x := Q] : \sigma$.

Lema 6. (O redukciji subjekta.) Ako $\Gamma \vdash M : \sigma$ i $M \twoheadrightarrow_\beta N$, onda $\Gamma \vdash N : \sigma$.

Dokaz. Indukcijom po broju primjena jednokoračnih β-redukcija u izvodu.

Za bazu indukcije, pretpostavimo da je M neki redeks oblika $(\lambda x : \tau.P)Q$. Po lemi o generiranju mora biti $\Gamma \vdash Q : \tau$ i $\Gamma, x : \tau \vdash P : \sigma$, no tada iz Leme 5 slijedi $\Gamma \vdash P[x := Q] : \sigma$, odnosno $\Gamma \vdash N : \sigma$. U ostalim slučajevima tvrdnja vrijedi kao posljedica leme o generiranju. Primjerice, ako je M aplikacija UV i vrijedi $UV \twoheadrightarrow_\beta UT$, to po lemi o generiranju imamo $\Gamma \vdash U : \tau \to \sigma$ i $\Gamma \vdash V : \tau$. Sada po pretpostavci indukcije slijedi $\Gamma \vdash T : \tau$, pa primjenom pravila (*appl*) dobivamo traženu tvrdnju. □

Korolar 2. U sistemu $\lambda_{\to}$ ne postoji kombinator[3] fiksne točke.

Dokaz. Pretpostavimo suprotno, da takav kombinator postoji, i označimo ga s M. Po definiciji kombinatora fiksne točke, za svaki dozvoljen term F mora vrijediti $MF \twoheadrightarrow_\beta F(MF)$. Označimo s Γ_1 i Γ_2 kontekste takve da vrijedi $\Gamma_1 \vdash M : \sigma$ i $\Gamma_2 \vdash F : \varphi$. Tada je $\Gamma_1, \Gamma_2 \vdash MF : \tau$, gdje je τ takav da je $\sigma = \varphi \to \tau$. Po lemi o redukciji subjekta tada mora vrijediti $\Gamma_1, \Gamma_2 \vdash F(MF) : \tau$. Po lemi o generiranju vrijedi $\Gamma_1, \Gamma_2 \vdash F : \tau \to \tau$. Primijetimo da je sada term F specifičnog tipa, poimence $\tau \to \tau$. No, nisu svi dozvoljeni termi tipa tog oblika, primjer jednog takvog vidjeli smo u Primjeru 6. Dakle, imamo kontradikciju s pretpostavkom da je F proizvoljan dozvoljen term, pa kombinator fiksne točke ne može postojati. □

[3]Term bez slobodnih varijabli.

5. Konfluentnost

U ovom kratkom poglavlju pokazujemo dva standardna rezultata iz teorije tipova. Prvi rezultat je Church–Rosserovo svojstvo sistema $\lambda_{\to}$. Drugi rezultat je jedinstvenost normalne forme, koja slijedi iz Church–Rosserovog svojstva i teorema o postojanju normalne forme. Ovi rezultati zajedno nam govore da je rezultat "računanja" u sistemu $\lambda_{\to}$ jedinstven i ne ovisi o redoslijedu redukcija.

Definicija 9. Neka je $\to$ binarna relacija na skupu S te neka je $\twoheadrightarrow$ njeno refleksivno i tranzitivno proširenje.

Kažemo da relacija $\to$ ima **slabo Church–Rosser svojstvo** ako za sve $M, N, P \in S$ takve da vrijedi $M \to N$ i $M \to P$ postoji $Q \in S$ za kojeg vrijedi $N \twoheadrightarrow Q$ i $P \twoheadrightarrow Q$. Dijagramatski prikazano:

$$\begin{array}{ccc} M & \xrightarrow{\ \beta\ } & P \\ \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle \beta} \\ N & \xrightarrow{\ \beta\ } & Q \end{array}$$

Kažemo da relacija $\to$ ima **jako Church–Rosser svojstvo** ako za sve $M, N, P \in S$ takve da vrijedi $M \twoheadrightarrow N$ i $M \twoheadrightarrow P$ postoji $Q \in S$ za kojeg vrijedi $N \twoheadrightarrow Q$ i $P \twoheadrightarrow Q$. Dijagramatski prikazano:

$$\begin{array}{ccc} M & \xrightarrow{\ \beta\ } & P \\ \downarrow{\scriptstyle \beta} & & \downarrow{\scriptstyle \beta} \\ N & \xrightarrow{\ \beta\ } & Q \end{array}$$

Često se u literaturi Church–Rosser svojstvo naziva konfluentnost ili dijamantno svojstvo. Jaka inačica ima slabije zahtjeve na elemente. Dovoljno je da M i N te M i P budu u refleksivnom i tranzitivnom proširenju originalne relacije.

Propozicija 1. Ako relacija $\to_\beta$ ima slabo, onda ima i jako Church–Rosser svojstvo.

Dokaz. Pretpostavimo da $\rightarrow_\beta$ ima slabo Church–Rosser svojstvo. Neka je M dozvoljen $\lambda_\rightarrow$-term. Po Teoremu 1, M ima normalnu formu. Ako je normalna forma terma M jedinstvena, dokaz je trivijalan.

Inače, neka su N_1 i N_2 dvije različite normalne forme terma M - terme s različitim normalnim formama zvat ćemo *nejednoznačnim*. Jer su normalne forme različite, to moraju postojati M_1 i M_2 takvi da vrijedi $M \rightarrow_\beta M_1 \twoheadrightarrow_\beta N_1$ i $M \rightarrow_\beta M_2 \twoheadrightarrow_\beta N_2$.

Ako je $M_1 = M_2$, onda smo pronašli nejednoznačan term M_1 takav da $M \rightarrow_\beta M_1$. Ako pak je $M_1 \neq M_2$, po pretpostavci slabog Church–Rosser svojstva postoji N_3 takav da $M_1 \twoheadrightarrow_\beta N_3$ i $M_2 \twoheadrightarrow_\beta N_3$. Bez smanjenja općenitosti možemo pretpostaviti da je N_3 u normalnoj formi. Sad kako su N_1, N_2 i N_3 u normalnoj formi, a N_1 i N_2 su različiti, to je N_3 različita od N_1 ili N_2. Tada je M_1, odnosno M_2, nejednoznačan term na kojeg se M reducira.

Time smo pokazali da za svaki nejednoznačan term M postoji nejednoznačan term M' takav da $M \rightarrow_\beta M'$, pa nejednoznačne terme možemo beskonačno reducirati, što je kontradikcija s Teoremom 2. Dakle, svaki term mora biti jednoznačan, a u tom slučaju tvrdnja vrijedi po početku dokaza. □

Teorem 3. Relacija $\rightarrow_\beta$ ima slabo Church–Rosser svojstvo.

Skica dokaza. Primijetimo da se dva redeksa u termu nikad ne preklapaju: ili su disjunktni ili je jedan podterm drugog. Ako su disjunktni, možemo reducirati prvog pa drugog, ili drugog pa prvog, a u oba slučaja dolazimo do istog terma. Ako je jedan podterm drugog, shema reduciranja je malo kompliciranija. Naime, reduciranjem vanjskog redeksa možemo dobiti više kopija unutarnjeg redeksa, pa njih sve treba reducirati. S druge strane, redukcijom prvo unutarnjeg, a zatim vanjskog redeksa, dobivamo isti rezultat kao u obrnutom slučaju. Formalniji dokaz proveli bismo indukcijom po broju primjena jednokoračne β-redukcije u izvodu. □

Korolar 3. Relacija $\rightarrow_\beta$ ima jako Church–Rosser svojstvo.

Dokaz. Slijedi iz Propozicije 1 i Teorema 3. □

Korolar 4. Svaki $\lambda_\rightarrow$-term ima jedinstvenu normalnu formu.

Dokaz. Neka je M dozvoljen term i neka su N_1 i N_2 dvije njegove normalne forme (koje postoje po Teoremu 1). Po definiciji normalne forme vrijedi $M \twoheadrightarrow_\beta N_1$ i $M \twoheadrightarrow_\beta N_2$. Po jakom Church–Rosser svojstvu onda postoji N_3 takav da vrijedi

$N_1 \twoheadrightarrow_\beta N_3$ i $N_2 \twoheadrightarrow_\beta N_3$. No, kako su termi N_1 i N_2 već u normalnoj formi, to mora biti $N_1 = N_3 = N_2$. $\square$

6. Zaključak

U ovom radu prikazali smo jednostavno tipizirani λ-račun. Motivirali smo uvođenje tipova u netipizirani λ-račun za sprečavanje nekih neintuitivnih pojava kao što su samoaplikacija i teorem o fiksnoj točki. Definirali smo sintaksu i pravila izvoda sistema $\lambda_{\to}$. Zatim smo napravili kratak pregled formalnih problema koji se javljaju u općenitim teorijama tipova, a čije primjere smo vidjeli u prethodnim poglavljima. Potom smo uveli relaciju β-redukcije kao formalizaciju pojma računanja, i dokazali smo neka osnovna svojstva našeg računa. Dokazali smo da je β-redukcija jako normalizirajuća te da ima Church–Rosserovo svojstvo.

Iz dokazanih rezultata može se pokazati da jednostavno tipizirani λ-račun nije opći model izračunavanja kao što je to netipizirani λ-račun ili parcijalno rekurzivne funkcije. Dodatno, u jednostavno tipiziranom λ-računu možemo programirati, no lakše je programirati u nekim proširenjima, kao što je polimorfni λ-račun. U smjeru matematičke logike, autor preporučuje proučavanje Curry–Howardove korespondencije i njenog utjecaja na razvoj programskih jezika i dokazivača teorema.

7. Literatura

[1] Hren, M. Osnove λ-računa. Seminarski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2021.

[2] Hren, M. Intuicionistička propozicijska logika. Seminarski rad, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Matematički odsjek, 2022.

[3] Lovnički, S. Interpreter za λ-račun. Diplomski rad, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Matematički odsjek, 2018.

[4] Nederpelt, R. i Geuvers, H. *Type Theory and Formal Proof: An Introduction.* Cambridge University Press, 2014.

[5] Sørensen, M. i Urzyczyn, P. *Lectures on the Curry-Howard Isomorphism.* Elsevier Science, 2006.