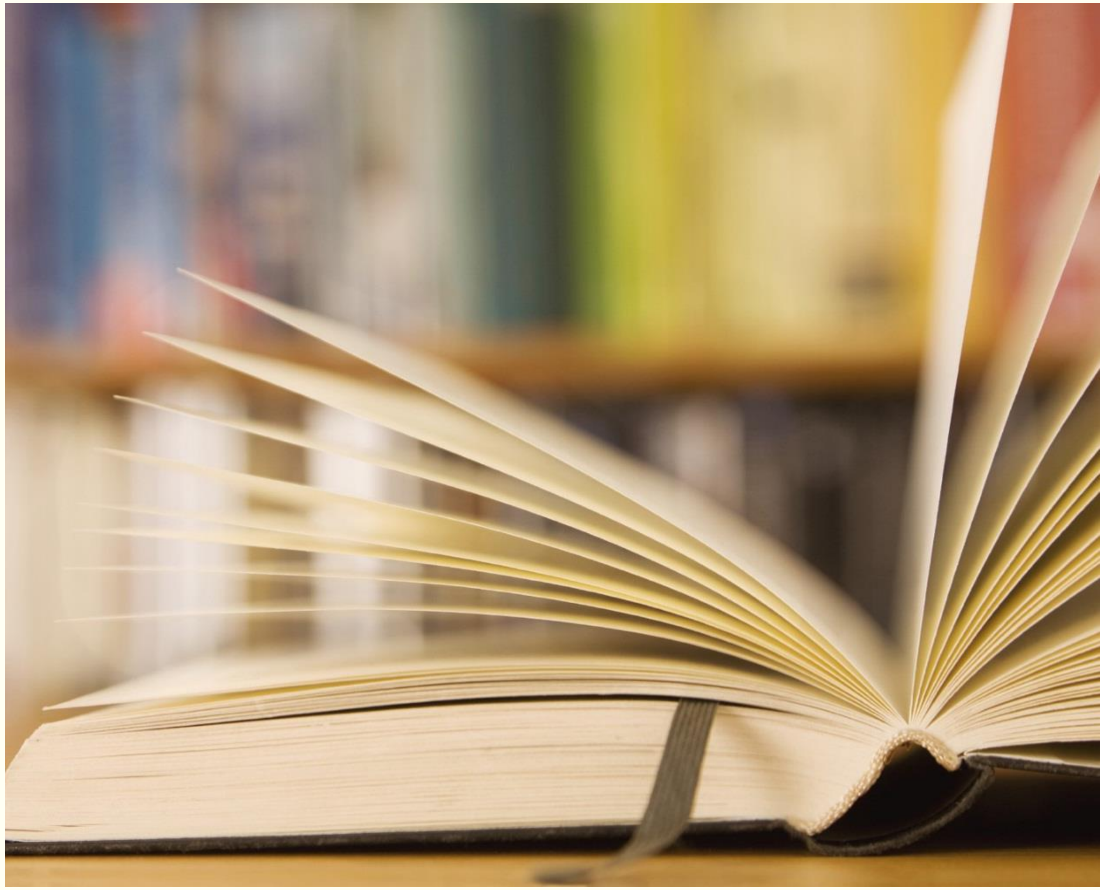
FUNKCIONALNO PROGRAMIRANJE
(MATERIJALI PREUZETI OD DOC. DR MILANE GRBIĆ)

Lambda račun



LAMBDA RAČUN

Lambda račun

- Funkcije u matematici imaju imena.

Suppose $f : \mathbb{R} \rightarrow \mathbb{R}$ is defined by:

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ x^2 sin(1/x^2) & \text{if } x \neq 0 \end{cases}$$

- Funkcije u programskim jezicima imaju imena.

```
int suc(int n)
 { return n + 1;
 }
```

Lambda račun

- Lambda račun možemo da shvatimo kao račun anonimnih (neimenovanih) funkcija.

$$x \mapsto t[x] \qquad \lambda x.\, t[x]$$

  - Metod za predstavljanje funkcija;
  - Skup pravila za sintaksnu transformaciju;
- Istorijat lambda računa:
  - 1930. - Alonzo Church;
  - 1932/1933. godina prva verzija lambda računa;
  - 1941. godina druga verzija lambda računa;

Zadaci:

1. U lambda notaciji zapisati funkciju identiteta.
2. U lambda notaciji napisti funkciju koja vraća dvostruku vrijednost svog argumenta.

Predností upotrebe lambda računa

- $f(x) \rightarrow \quad f\ x$;
- $f\ x\ y \rightarrow \big(f(x)\big)(y)$;
- $\lambda x.\lambda y.t[x,y] \rightarrow \lambda x\ y.t[x,y]$;
- $\lambda x.x\ y \rightarrow \lambda x.(x\ y)$;

$$(\lambda x\ y.\ x + y)\ 1\ 2 = (\lambda y.\ 1 + y)\ 2 = 1 + 2$$

- Vezane (m i y u prvom i drugom izrazu) i slobodne (n u trećem) varijable;

$$\sum_{m=1}^{n} m = \frac{n(n+1)}{2}$$

$$\int_0^x 2y + a\ dy = x^2 + ax$$

```
int suc(int n)
 { return n + 1;
 }
```

Prednosti upotrebe lambda računa

- Promjena imena vezane varijable (alfa konverzija);

$$\int_0^x 2y + a\ dy = x^2 + ax$$

$$\int_0^x 2z + a\ dz = x^2 + ax$$

$$\lambda x.\ E[x]$$

$$\lambda y.\ E[y]$$

$$\int_0^x 2a + a\ da \neq x^2 + ax$$

- Identična imena slobodnih i vezanih varijabli (nije zabranjeno, ali je zbunjujuće):

$$\int_0^x 2x + a\ dx = x^2 + ax$$

Lambda račun kao formalni sistem

- Lambda termovi (izrazi):
  - Varijable;
  - Konstante;
  - Kombinacije – primjena funkcije s na argument t; s i t mogu biti proizvoljni lambda termi. Zapis $s\ t$.
  - Apstrakcija – lambda terma s nad varijablom x, u oznaci $\lambda x. s$.
- Induktivna definicija lambda izraza.
  - Funkcije nad lambda izrazima se definišu rekurzivno.
  - Svojstva lambda izraza se dokazuju strukturalnom indukcijom.
- Sintaksa lambda termova (izraza) u BNF (Backus-Naurovoj formi):

$$Exp = Var \mid Const \mid Exp\ Exp \mid \lambda\, Var.\ Exp$$

Zadaci

3. Da li su sljedeći izrazi lambda izrazi ili ne? Obrazložiti odgovor.

   a) 42
   b) $(+6)$
   c) $\lambda y. * \ 2 \ y$
   d) $\lambda v. \ v \ x \ y$
   e) $(\lambda f. \lambda a. \lambda b. f \ a \ b)(\lambda x. (\lambda y. x))$

Slobodne i vezane varijable

- Za pojavljivanje v u lambda izrazu E kažemo da je vezano ako se javlja u podizrazu E_1 od E u formi $\lambda v. E_1$.
- Ako promjenljiva nije vezana u nekom izrazu, tada je njeno pojavljivanje slobodno.
- Lambda račun se naziva čist ako ne uključuje konstante.
- Ako lambda račun uključuje konstante, onda se naziva primjenjen.

Slobodne i vezane varijable

- Skup slobodnih varijabli u izrazu s označavamo sa $FV(s)$ i definišemo rekurzivno na sljedeći način:

$$
\begin{aligned}
FV(x) &= \{x\} \\
FV(c) &= \emptyset \\
FV(s\ t) &= FV(s) \cup FV(t) \\
FV(\lambda x.\, s) &= FV(s) - \{x\}
\end{aligned}
$$

Slobodne i vezane varijable

- Skup vezanih varijabli u izrazu s označavamo sa $BV(s)$ i definišemo rekurzivno na sljedeći način:

$$\begin{aligned}
BV(x) &= \emptyset \\
BV(c) &= \emptyset \\
BV(s\,t) &= BV(s) \cup BV(t) \\
BV(\lambda x.\, s) &= BV(s) \cup \{x\}
\end{aligned}$$

Zadaci

4. Za lambda izraz

$$s = (\lambda x\, y.\ x)(\lambda x.\, z\, x)$$

odrediti $FV(s)$ i $BV(s)$.

Zamjena

- Lambda apstrakcija i primjena;

$$(\lambda x\ y.\ x + y)\ 1\ 2 = (\lambda y.\ 1 + y)\ 2 = 1 + 2$$

- Sa $t[s/x]$ označavamo zamjenu varijable x u izrazu t izrazom s.

$$\begin{array}{rcl}
x[t/x] & = & t \\
y[t/x] & = & y \text{ if } x \neq y \\
c[t/x] & = & c \\
(s_1\ s_2)[t/x] & = & s_1[t/x]\ s_2[t/x] \\
(\lambda x.\ s)[t/x] & = & \lambda x.\ s \\
(\lambda y.\ s)[t/x] & = & \lambda y.\ (s[t/x]) \text{ if } x \neq y \text{ and either } x \notin FV(s) \text{ or } y \notin FV(t) \\
(\lambda y.\ s)[t/x] & = & \lambda z.\ (s[z/y][t/x]) \text{ otherwise, where } z \notin FV(s) \cup FV(t)
\end{array}$$

Izračunavanje vrijednosti lambda izraza

- δ *pravila*
  - Aplikacija konstantnih funkcija.
- β *redukcija* (najvažnije, jer predstavlja evaluaciju funkcije E1(v) za zadati argument E2)

$$(\lambda \mathrm{v}.E_1)\ E_2 \longrightarrow_\beta E_1[E_2/\mathrm{v}]$$

  - β *redeks*; $(\lambda \mathrm{x.E})E_1$
- η *redukcija* – nije nam bitna za programerske aspekte lambda računa.
- α *konverzija*
  - Proces promjene imena vezanih varijabli – pokazano ranije.

Zadaci

5. Izračunati vrijednost sljedećih lambda izraza:

$$(\lambda v.v)\ c$$
$$(\lambda v.x\ (v\ c)\ v)\ (a\ y)$$
$$(\lambda v.c)\ (x\ v)$$
$$(\lambda v.v\ c)\ (\lambda x.x\ a)$$
$$(\lambda x.\text{plus}\ x\ 1)\ ((\lambda y.\text{times}\ y\ y)\ 3)$$

Zadaci

6. Izračunati vrijednost lambda izraza:

$$(\lambda f.\lambda x.f\ 4\ x)\ (\lambda y.\lambda x.\ +\ x\ y)\ 3$$

Izračunavanje vrijednosti lambda izraza

- *δ pravila*
  - Aplikacija konstantnih funkcija.
- *β redukcija*

$$(\lambda \mathrm{v}.E_1)\ E_2 \longrightarrow_\beta E_1[E_2/\mathrm{v}]$$

  - *β redeks*; $(\lambda \mathrm{x}.\mathrm{E})E_1$
- *η redukcija*

$$\lambda \mathrm{x}.\mathrm{E}\ \mathrm{x} \longrightarrow_\eta \mathrm{E}$$

označavaju istu funkciju ako se x ne javlja kao slobodno u E; E mora

- *α konverzija*
  - Proces promjene imena vezanih varijabli.
  - Sintaksno ekvivalentni termi se smatraju različitim reprezentacijama iste apstrakcije.
  - Primjer $\lambda \mathrm{x}.\underline{((\lambda \mathrm{y}.\lambda \mathrm{x}.+\ \mathrm{x}\ \mathrm{y})\ \mathrm{x})}$

Jednakost lambda izraza

- Dva lambda terma su jednaka ako se je moguće dobiti jedan iz drugog konačnim nizom α, β i η konverzija.

$$\frac{s \xrightarrow[\alpha]{} t \text{ or } s \xrightarrow[\beta]{} t \text{ or } s \xrightarrow[\eta]{} t}{s = t}$$

$$\frac{}{t = t}$$

$$\frac{s = t}{t = s}$$

$$\frac{s = t \text{ and } t = u}{s = u}$$

$$\frac{s = t}{s\ u = t\ u}$$

$$\frac{s = t}{u\ s = u\ t}$$

$$\frac{s = t}{\lambda x.\ s = \lambda x.\ t}$$

$$\equiv_\alpha$$

$$(\lambda x.\ x)y \equiv_\alpha (\lambda y.\ y)y.$$

Lambda redukcije

$$\frac{s \xrightarrow[\alpha]{} t \text{ or } s \xrightarrow[\beta]{} t \text{ or } s \xrightarrow[\eta]{} t}{s \longrightarrow t}$$

$$\frac{}{t \longrightarrow t}$$

$$\frac{s \longrightarrow t \text{ and } t \longrightarrow u}{s \longrightarrow u}$$

$$\frac{s \longrightarrow t}{s\ u \longrightarrow t\ u}$$

$$\frac{s \longrightarrow t}{u\ s \longrightarrow u\ t}$$

$$\frac{s \longrightarrow t}{\lambda x.\, s \longrightarrow \lambda x.\, t}$$

Normalne forme

- **Definicija:**

Za izraz E kažemo da je u *normalnoj formi* ako ne može dalje da se redukuje, tj. ako više ne sadrži ni jedan redeks.

- **Definicija:**

Kažemo da postoji normalna forma za izraz E, ako postoji niz redukcija

$$\mathrm{E} \longrightarrow E_1 \longrightarrow E_2 \longrightarrow \ldots \longrightarrow^{*} \mathrm{E}$$

i izraz $* E$ je u normalnoj formi.

Strategije redukcije

- Teorijska razmatranja i funkcionalno programiranje?
- Izbor narednog redeksa na dva najčešća načina:
  - Najlevlji i najunutrašnjiji redeks (naziva se još redukcija po vrednosti ili aplikativna redukcija) – ne mora da se završi.

$$
\begin{aligned}
& (\lambda x.\, y)\ ((\lambda x.\, x\ x\ x)\ (\lambda x.\, x\ x\ x)) \\
\longrightarrow\ & (\lambda x.\, y)\ ((\lambda x.\, x\ x\ x)\ (\lambda x.\, x\ x\ x)\ (\lambda x.\, x\ x\ x)) \\
\longrightarrow\ & (\lambda x.\, y)\ ((\lambda x.\, x\ x\ x)\ (\lambda x.\, x\ x\ x)\ (\lambda x.\, x\ x\ x)\ (\lambda x.\, x\ x\ x)) \\
\longrightarrow\ & \ldots
\end{aligned}
$$

  - Najlevlji redeks (naziva se još redukcija po nazivu ili lenja redukcija ili normalna redukcija) – sigurno pronalazi normalnu formu ako ona postoji.

$$(\lambda x.\, y)\ ((\lambda x.\, x\ x\ x)\ (\lambda x.\, x\ x\ x)) \longrightarrow y$$

Redeksi i mogući načini redukcije

Definicija: *Krajnje levi* redeks je onaj čije λ (ili primitivni identifikator u slučaju δ redeksa) je tekstualno najlevlje od svih ostalih redeksa u izrazu.

Slično se definiše i *Krajnje desni* redeks.

Definicija: Redeks je *spoljašnji* ako se za njega ne može naći redeks koji ga sadrži.

Definicija: Redeks je *unutrašnji* ako se za njega ne može naći redeks koji je sadržan u njemu.

Redeksi i mogući načini redukcije

1. Kranje levo-spoljašnja (Normalni redosled). Ova redukcija je sigurna. Naredni redeks se u svakom koraku bira počevši od početnog u krajnje levom delu izraza. Kada se primenjuje β-redeks, argument ne mora da bude u normalnoj formi.
2. Kranje levo-unutrašnja (Aplikativni redosled). Ova redukcija je nesigurna. Naredni redeks se u svakom koraku bira počevši od završnog u krajnje levom delu izraza. Kada se primenjuje β-redeks, i argument i apstrakcija moraju da budu u normalnoj formi.
3. Paralelno spoljašnja. Ova redukcija je sigurna. Svi redeksi koji nisu ugnježđeni unutar nekog drugog redeksa se biraju i redukuju istovremeno. Izabrani redeksi ne mogu da se preklapaju.
4. Paralelno unutrašnja. Ova redukcija je nesigurna. Svi redeksi koji ne sadrže ugnježđene redekse se biraju i redukuju istovremeno.
5. Potpuno paralelno. Svi redeksi se istovremeno redukuju. Ova strategija je vrlo komplikovana jer redeksi mogu biti medjusobno ugnježđeni.

Zadaci

1. Izračunati vrijednost lambda izraza

$(\lambda x. + x\ x)((\lambda y. * \ y\ y)(+5\ 5))$

a) normalnim redoslijedom redukcije;

b) aplikativnim redoslijedom redukcije.

2. Izračunati vrijednost lambda izraza

$(\lambda g.\ g + \left(g\ * (g+5)\right))(\lambda f. \lambda x. f\ x\ x)$

a) normalnim redoslijedom redukcije;

b) aplikativnim redoslijedom redukcije.

De-Brujinov lambda račun

- Sintaksna i semantička jednakost lambda izraza;
- Uklanjanje imena svih promjenljivih i zamjena brojem koji predstavlja broj λ između pojavljivanja promjenljive u tijelu funkcije i λ za koje je promjenljiva vezana.
- Primjeri:

Standard	de Bruijn
$\lambda x.x$	$\lambda.0$
$\lambda z.z$	$\lambda.0$
$\lambda x.\lambda y.x$	$\lambda.\lambda.1$
$\lambda x.\lambda y.\lambda s.\lambda z.x\ s\ (y\ s\ z)$	$\lambda.\lambda.\lambda.\lambda.3\ 1\ (2\ 1\ 0)$
$(\lambda x.x\ x)\ (\lambda x.x\ x)$	$(\lambda.0\ 0)\ (\lambda.0\ 0)$
$(\lambda x.\lambda x.x)\ (\lambda y.y)$	$(\lambda.\lambda.0)\ (\lambda.0)$

Zadaci

3. Zapisati funkciju sljedbenik u De-Brujinovom obliku.
4. Zapisati sljedeći lambda izraz u De-Brujinovom obliku.

$$\lambda x.\lambda y.\lambda f.f\ (\lambda x.x)(+\ x\ y)$$

LAMBDA RAČUN KAO PROGRAMSKI JEZIK

Predstavljanje podataka u lambda računu

- Notacija:

$$s \triangleq s'$$

'$s =_{def} s'$'.

jednaki po definiciji;

- Uslovni izraz; 'if E then E_1 else E_2' as 'COND E E_1 E_2'
- Varijable sa lijeve strane trebaju biti apstrakovane.

$$\text{fst } p \triangleq p \text{ true}$$

$$\text{fst} \triangleq \lambda p.\, p \text{ true}$$

Logičke vrijednosti i operatori

- Logičke vrijednosti:

$$\begin{aligned} \text{true} &\triangleq \lambda x\ y.\, x \\ \text{false} &\triangleq \lambda x\ y.\, y \end{aligned}$$

- Uslovni izraz:

$$\text{if } E \text{ then } E_1 \text{ else } E_2 \triangleq E\ E_1\ E_2$$

Logičke vrijednosti i operatori

$$\begin{aligned} \text{if true then } E_1 \text{ else } E_2 &= \text{true } E_1\ E_2 \\ &= (\lambda x\ y.\ x)\ E_1\ E_2 \\ &= E_1 \end{aligned}$$

$$\begin{aligned} \text{if false then } E_1 \text{ else } E_2 &= \text{false } E_1\ E_2 \\ &= (\lambda x\ y.\ y)\ E_1\ E_2 \\ &= E_2 \end{aligned}$$

Logičke vrijednosti i operatori

$$
\begin{aligned}
\text{not } p &\triangleq \text{if } p \text{ then false else true} \\
p \text{ and } q &\triangleq \text{if } p \text{ then } q \text{ else false} \\
p \text{ or } q &\triangleq \text{if } p \text{ then true else } q
\end{aligned}
$$

Zadaci:

5. Izračunati:
   a) $\neg T$
   b) $\neg F$

6. Izračunati:
   a) $\wedge TT$
   b) $\wedge FT$

7. Izračunati:
   a) $\vee TT$
   b) $\vee FT$

Parovi i torke

- Uređen par

$$(E_1, E_2) \triangleq \lambda f.\ f\ E_1\ E_2$$

- Prvi, drugi element para:

$$\begin{aligned} \text{fst}\ p &\triangleq p\ \text{true} \\ \text{snd}\ p &\triangleq p\ \text{false} \end{aligned}$$

- Pokazati da je validna defincija!

Parovi i torke

- n-torka

$$(E_1, E_2, \ldots, E_n) = (E_1, (E_2, \ldots E_n))$$

- Uređena četvorka

$$\begin{aligned}
(p, q, r, s) &= (p, (q, (r, s))) \\
&= \lambda f.\ f\ p\ (q, (r, s)) \\
&= \lambda f.\ f\ p\ (\lambda f.\ f\ q\ (r, s)) \\
&= \lambda f.\ f\ p\ (\lambda f.\ f\ q\ (\lambda f.\ f\ r\ s)) \\
&= \lambda f.\ f\ p\ (\lambda g.\ g\ q\ (\lambda h.\ h\ r\ s))
\end{aligned}$$

Brojevi

- Nula, sljedbenik nule, sljedbenik sljedbenika nule…
- $0 := \lambda s.(\lambda z.z) = \lambda sz.z$
- $1 := \lambda sz.s(z) = \lambda sz.s\ z$
- $2 := \lambda sz.s\ (s(z)) = \lambda sz.s\ (sz)$
- $3 := \lambda sz.s(s\ (s(z))) = \lambda sz.s\big(s\ (sz)\big)$
- …
- 6:=?
- Zašto ovakav način definisanja brojeva?

$$n \triangleq \lambda f\ x.\ f^n\ x$$

Funkcija sljedbenik, sabiranje, množenje..

- Funkcija sljedbenik

$$\mathrm{SUC} \triangleq \lambda n\ f\ x.\ n\ f\ (f\ x)$$

- Odredimo sljedbenik broja n

$$\begin{aligned}
\mathrm{SUC}\ n &= (\lambda n\ f\ x.\ n\ f\ (f\ x))(\lambda f\ x.\ f^n\ x) \\
&= \lambda f\ x.\ (\lambda f\ x.\ f^n\ x) f\ (f\ x) \\
&= \lambda f\ x.\ (\lambda x.\ f^n\ x)(f\ x) \\
&= \lambda f\ x.\ f^n\ (f\ x) \\
&= \lambda f\ x.\ f^{n+1}\ x \\
&= n+1
\end{aligned}$$

Zadaci:

8. Izračunati sljedbenik 0.
9. Izračunati sljedbenik 1.

Funkcija koja provjerava da li je broj jednak nuli

$$\text{ISZERO } n \triangleq n\ (\lambda x.\ \text{false})\ \text{true}$$

$$\text{ISZERO } 0 = (\lambda f\ x.\ x)(\lambda x.\ \text{false})\ \text{true} = \text{true}$$

$$\begin{aligned}
\text{ISZERO } (n+1) &= (\lambda f\ x.\ f^{n+1} x)(\lambda x.\ \text{false})\text{true} \\
&= (\lambda x.\ \text{false})^{n+1}\ \text{true} \\
&= (\lambda x.\ \text{false})((\lambda x.\ \text{false})^{n}\ \text{true}) \\
&= \text{false}
\end{aligned}$$

Sabiranje, množenje

$$
\begin{aligned}
m + n &\triangleq \lambda f\ x.\ m\ f\ (n\ f\ x) \\
m * n &\triangleq \lambda f\ x.\ m\ (n\ f)\ x
\end{aligned}
$$

$$
\begin{aligned}
m + n &= \lambda f\ x.\ m\ f\ (n\ f\ x) \\
&= \lambda f\ x.\ (\lambda f\ x.\ f^m\ x)\ f\ (n\ f\ x) \\
&= \lambda f\ x.\ (\lambda x.\ f^m\ x)\ (n\ f\ x) \\
&= \lambda f\ x.\ f^m\ (n\ f\ x) \\
&= \lambda f\ x.\ f^m\ ((\lambda f\ x.\ f^n\ x)\ f\ x) \\
&= \lambda f\ x.\ f^m\ ((\lambda x.\ f^n\ x)\ x) \\
&= \lambda f\ x.\ f^m\ (f^n\ x) \\
&= \lambda f\ x.\ f^{m+n} x
\end{aligned}
$$

Zadaci:

10. Izračunati zbir brojeva 2 i 3 u lambda notaciji.

Sabiranje, množenje

$$\begin{aligned} m + n &\triangleq \lambda f\ x.\ m\ f\ (n\ f\ x) \\ m * n &\triangleq \lambda f\ x.\ m\ (n\ f)\ x \end{aligned}$$

$$\begin{aligned} m * n &= \lambda f\ x.\ m\ (n\ f)\ x \\ &= \lambda f\ x.\ (\lambda f\ x.\ f^m\ x)\ (n\ f)\ x \\ &= \lambda f\ x.\ (\lambda x.\ (n\ f)^m\ x)\ x \\ &= \lambda f\ x.\ (n\ f)^m\ x \\ &= \lambda f\ x.\ ((\lambda f\ x.\ f^n\ x)\ f)^m\ x \\ &= \lambda f\ x.\ ((\lambda x.\ f^n\ x)^m\ x \\ &= \lambda f\ x.\ (f^n)^m\ x \\ &= \lambda f\ x.\ f^{mn} x \end{aligned}$$

Zadaci:

11. Izračunati proizvoda brojeva 4 i 3 u lambda notaciji.

Stepenovanje

- $a^b = ba$
- Primjer;

KARIJEVE FUNKCIJE

Karijeve funkcije

- Funkcije sa više argumenata moguće je definisati koristeći funkciju kao povratnu vrijednost:

add' uzima kao argument cijeli broj x i vraća kao rezultat funkciju add' x. Slično, ova funkcija uzima cijeli broj y i vraća kao rezultat funkciju $x + y$.

```
add'     :: Int → (Int → Int)
add' x y = x+y
```

Karijeve funkcije

```
add  :: (Int,Int) → Int

add’ :: Int → (Int → Int)
```

- add i add’ imaju isti konačan rezultat, ali add uzima oba argumenta istovremeno, dok add’ uzima jedan po jedan argument.
- Funkcije koje uzimaju jedan po jedan argument zovu se Karijeve funkcije, u čast rada Haskell Curry-ja nad ovim funkcijama.

Karijeve funkcije

- Funkcije sa više od dva argumenta mogu se definisati kao Karijeve korištenjem ugnježdenih funkcija kao povratnih vrijednosti:

mult uzima argument x i vraća funkciju mult x, koja slično uzima argument y i vraća funkciju mult x y, koja na kraju uzima argument z i vraća rezultat $x * y * z$.

```
mult         :: Int → (Int → (Int → Int))
mult x y z = x*y*z
```

Karijeve funkcije

- Zašto su Karijeve funkcije korisne?
  - Karijeve funckije su fleksibilnije nego funkcije nad torkama jer se parcijalnom primjenom Karijevh funkcija mogu dobiti razne korisne funkcije.

- Konvencije u zapisu Karijevih funkcija
  - Strelica → je desno asocijativna.

Int → (Int → (Int → Int))

```
Int → Int → Int → Int
```

  - Primjena funkcija je lijevo asocijativna.

((mult x) y) z

```
mult x y z
```

TIPOVI

Tipovi

- Tipovi – različite vrste podataka (prirodni brojevi, logičke vrijednosti, funkcije);
- Svrha?
- Zašto dodati tipove i u lambda račun?
  - Logički aspekt: Raselov paradoks; Primjena funkcija samo na podatke iz domena;
  - Programerski aspekt: Generisanje efikasnijeg koda; Lakše otkrivanje grešaka; netipizirani jezici; slabo tipizirani jezici; dinamički tipizirani jezici; statički tipizirani jezici;
- Statički tipiziran jezik i fleksibilnost koda?
  - Polimorfizam;
  - Određivanje tipa izraza;

Tipizirani lambda račun

- Svaki lambda term ima tip.
- Term s može biti primjenje na term t ako se odgovarajući tipovi poklapaju.
  - Primjer?
- Strogo tipiziran jezik;
  - Da li je C strogo tipiziran?
- Notacija
  - $t : \sigma$ u značenju da je t tipa σ;
  - $f : \sigma \to \tau$
  - Tipovi su skupovi kojima pripada određen objekat; $t : \sigma$ tj. $t \in \sigma$.

Vrste tipova

- Šta su tačno tipovi?
- Prosti tipovi (bool, int);
- Složeni tipovi (*composite type*) formiramo na osnovu konstruktora tipova (*type constructor*);
- C – skup prostih tipova, skup T_{yC} baziran na skupu C
  - Primjer?

$$\frac{\sigma \in C}{\sigma \in Ty_C}$$

$$\frac{\sigma \in Ty_C \quad \tau \in Ty_C}{\sigma \to \tau \in Ty_C}$$

Vrste tipova

- Tipovske varijable;
- Konstruktore za tipove (ne samo na osnovu funkcija);
  - Npr.

$$\frac{\sigma \in Ty_C \quad \tau \in Ty_C}{\sigma \times \tau \in Ty_C}$$

- Proizvoljan skup konstruktora proizvoljne arnosti;

$$(\alpha_1, \ldots, \alpha_n)con$$

- Tip ne može biti jednak nekom svom pravom podskupu.

Čerčovo i Karijevo tipiziranje

- Čerčovo tipiziranje – ekspilitno
  - Svaki term je jednog tipa.

$$\frac{}{v : \sigma}$$

$$\frac{\text{Constant } c \text{ has type } \sigma}{c : \sigma}$$

$$\frac{s : \sigma \to \tau \quad t : \sigma}{s\, t : \tau}$$

$$\frac{v : \sigma \quad t : \tau}{\lambda v.\, t : \sigma \to \tau}$$

Čerčovo i Karijevo tipiziranje

- Karijevo tipiziranje – impilcitno
  - Term može a i ne mora da bude nekog tipa, ako je nekog tipa, može da bude više različitih tipova.
  - Polimorfizam;
  - Notacija - u datom kontekstu

$$\Gamma \vdash t : \sigma$$

Formalna pravila tipiziranosti

$$\frac{v : \sigma \in \Gamma}{\Gamma \vdash v : \sigma}$$

$$\frac{\text{Constant } c \text{ has type } \sigma}{c : \sigma}$$

$$\frac{\Gamma \vdash s : \sigma \to \tau \qquad \Gamma \vdash t : \sigma}{\Gamma \vdash s\ t : \tau}$$

$$\frac{\Gamma \cup \{v : \sigma\} \vdash t : \tau}{\Gamma \vdash \lambda v.\, t : \sigma \to \tau}$$

Formalna pravila tipiziranosti

- Term je datog tipa ako se to može izvesti iz prethodnih pravila.
- Na primjer, funkcija identiteta

Na osnovu pravila za varijable

$$\{x : \sigma\} \vdash x : \sigma$$

Na osnovu pravila za apstrakcije

$$\emptyset \vdash \lambda x.\, x : \sigma \to \sigma$$

- Zašto je potreban kontekst u Karijevom tipiziranju?

Polimorfizam

- Polimorfizam vs overload vs subtyping?

$$\text{if } (\lambda x.\, x) \text{ true then } (\lambda x.\, x)\ 1 \text{ else } 0$$

COND b t1 t2

$$\dfrac{\dfrac{\dfrac{\{x : bool\} \vdash x : bool}{\vdash (\lambda x.\, x) : bool \to bool} \qquad \dfrac{}{\vdash \text{true} : bool}}{\vdash (\lambda x.\, x) \text{ true} : bool} \qquad \dfrac{\dfrac{\{x : int\} \vdash x : int}{\vdash (\lambda x.\, x) : int \to int} \qquad \dfrac{}{\vdash 1 : int}}{\vdash (\lambda x.\, x)\ 1 : int} \qquad \dfrac{}{\vdash 0 : int}}{\vdash \text{if } (\lambda x.\, x) \text{ true then } (\lambda x.\, x)\ 1 \text{ else } 0 : int}$$

Opštiji tipovi

- Za svaki tipiziran izraz postoji opštiji tip i svi mogući tipovi izraza su instace tog opštijeg tipa.
- Tipovske varijable – tipovi mogu biti formirani primjenom konstruktora tipa na tipove varijabli ili konstanti.
- Zamjene – npr. $(\sigma \to bool)[(\sigma \to \tau)/\sigma] = (\sigma \to \tau) \to bool.$
- Formalno

$$
\begin{array}{rcl}
\alpha_i[\tau_1/\alpha_1, \ldots, \tau_n/\alpha_k] & = & \tau_i \text{ if } \alpha_i \neq \beta \text{ for } 1 \leq i \leq k \\
\beta[\tau_1/\alpha_1, \ldots, \tau_n/\alpha_k] & = & \beta \text{ if } \alpha_i \neq \beta \text{ for } 1 \leq i \leq k \\
(\sigma_1, \ldots, \sigma_n)con[\theta] & = & (\sigma_1[\theta], \ldots, \sigma_n[\theta])con
\end{array}
$$

Opštiji tipovi

- σ je opštiji tip od σ' u oznaci $\sigma \preccurlyeq \sigma'$, ako postoji skup zamjena θ takav da je $\sigma' = \sigma\theta$.
  - Refleksivna relacija;
- Na primjer:

$$\begin{aligned} \alpha &\preceq \sigma \\ \alpha \to \alpha &\preceq \beta \to \beta \\ \alpha \to bool &\preceq (\beta \to \beta) \to bool \end{aligned}$$

$$\begin{aligned} \beta \to \alpha &\preceq \alpha \to \beta \\ \alpha \to \alpha &\not\preceq (\beta \to \beta) \to \beta \end{aligned}$$