Kombinatorna logika – Program za apstrakciju

Vlado Skoko, Nikola Ajzenhamer

Univerzitet u Beogradu, Matematički fakultet

12. januar 2016.

Sadržaj

Alfabet

Alfabet kombinatorne logike se sastoji od:

- Konstanti: **K, S**
- Promenljivih: *x, y, z, ...*
- Zagrada: (,)
- Simbola jednakosti: $=$

Termi u kombinatornoj logici

Da bismo formulisali jezik induktivno ćemo uvesti terme jezika:

1. Konstante i promenljive su CL-termi
2. Ako su reči U i V nad ovim alfabetom CL-termi, onda je i reč (UV) takođe CL-term (primena).
3. Ništa više nije CL-term.

Primeri (primer levo se naziva i *kombinator*):

SS(**K**(**SKI**)**I**)**I** (**S**xy)**KI**(**K**xy(zx))

Aksiome u kombinatornoj logici

Formule kojima se bavimo u kombinatornoj logici su oblika U = V, gde su U i V neki CL-termi. Račun je zadat sledećim aksiomama

$$U = U,\ \mathbf{K}UV = U,\ \mathbf{S}UVW = UW(VW).$$

Kombinator **I** nije primitivan u jeziku već ga definišemo sledećom lemom i nadalje ga koristimo kao da nam je dat u alfabetu.

Lema: Kombinator **I** definisan kao

$$\mathbf{I} \stackrel{df}{=} \mathbf{SKK}$$

izvodi sledeće pravilo

$$\mathbf{I}U = U.$$

Apstrakcija

- Apstrakcija je operacija koju možemo primeniti nad termima kombinatorne logike i koja nam mehanički izvodi sve ostale prave kombinatore preko kombinatora **S** i **K**.
- Ona je unarna operacija, što znači da radi na jednom termu. Ukoliko želimo apstrakciju svih promenljivih, operaciju moramo primeniti onoliko puta koliko imamo različitih promenljivih.
- Posledica: Izražajna moć lambda računa je ekvivalentna izražajnoj moći kombinatorne logike.

Pravila za primenu apstakcije

Za proizvoljnu promenljivu x i term M, induktivno definišemo term $\langle x \rangle .M$ u kome se x ne pojavljuje:

(p1) $\langle x \rangle .x \stackrel{df}{=} \mathbf{I}$,

(p2) $\langle x \rangle .M \stackrel{df}{=} \mathbf{K}$M, ako se x ne pojavljuje u M,

(p3) $\langle x \rangle .Ux \stackrel{df}{=}$ U, ako se x ne pojavljuje u U,

(p4) $\langle x \rangle .UV \stackrel{df}{=} \mathbf{S}(\langle x \rangle .U)(\langle x \rangle .V)$, ako se ni (p2) ni (p3) ne može primeniti.

Primer 1

Izvršiti sledeću apstrakciju: $[x, y, z].x(yz)$:

- Tražimo šta je $[z].x(yz)$:

$$[z].x(yz) = \mathbf{S}([z].x)([z].yz) = \mathbf{S}(\mathbf{K}x)y.$$

- Zatim računamo

$$[y].\mathbf{S}(\mathbf{K}x)y = \mathbf{S}(\mathbf{K}x).$$

- Zatim računamo

$$[x].\mathbf{S}(\mathbf{K}x) = \mathbf{S}([x].\mathbf{S})([x].\mathbf{K}x) = \mathbf{S}(\mathbf{KS})\mathbf{K}.$$

Primer 1 (provera)

$$\begin{aligned}\mathbf{S}(\mathbf{KS})\mathbf{K}xyz &= \mathbf{KS}x(\mathbf{K}x)yz = \\ &= \mathbf{S}(\mathbf{K}x)yz = \\ &= (\mathbf{K}x)z(yz) = \\ &= x(yz).\end{aligned}$$

Nazovimo dobijeni kombinator $\mathbf{B}$, tj. $\mathbf{B} = \mathbf{S}(\mathbf{KS})\mathbf{K}$.

Primer 2

Izvršiti sledeću apstrakciju: $[x, y, z].xzy$:

- Tražimo šta je $[z].xzy$:

$$[z].xzy = \mathbf{S}([z].xz)([z].y) = \mathbf{S}x(\mathbf{K}y).$$

- Zatim računamo

$$[y].\mathbf{S}x(\mathbf{K}y) = \mathbf{S}([y].\mathbf{S}x)([y].\mathbf{K}y) = \mathbf{S}(\mathbf{K}(\mathbf{S}x))\mathbf{K}.$$

- Zatim računamo

$$\begin{aligned}[x].\mathbf{S}(\mathbf{K}(\mathbf{S}x))\mathbf{K} &= \mathbf{S}([x].\mathbf{S}(\mathbf{K}(\mathbf{S}x)))([x].\mathbf{K}) = \\ &= \mathbf{S}(\mathbf{S}([x].\mathbf{S})([x].\mathbf{K}(\mathbf{S}x)))(\mathbf{K}\mathbf{K}) = \\ &= \mathbf{S}(\mathbf{S}(\mathbf{K}\mathbf{S})(\mathbf{S}([x].\mathbf{K})([x].\mathbf{S}x)))(\mathbf{K}\mathbf{K}) = \\ &= \mathbf{S}(\mathbf{S}(\mathbf{K}\mathbf{S})(\mathbf{S}(\mathbf{K}\mathbf{K})\mathbf{S}))(\mathbf{K}\mathbf{K}).\end{aligned}$$

Primer 2 (provera)

$$\begin{aligned}
\mathbf{S}(\mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KK})\mathbf{S}))(\mathbf{KK})xyz &= \mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KK})\mathbf{S})x)(\mathbf{KK}x)yz = \\
&= \mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KK})\mathbf{S})x\mathbf{K}yz = \\
&= \mathbf{KS}x(\mathbf{S}(\mathbf{KK})\mathbf{S}x)\mathbf{K}yz = \\
&= \mathbf{S}(\mathbf{S}(\mathbf{KK})\mathbf{S}x)\mathbf{K}yz = \\
&= \mathbf{S}(\mathbf{KK}x(\mathbf{S}x))\mathbf{K}yz = \\
&= \mathbf{S}(\mathbf{K}(\mathbf{S}x))\mathbf{K}yz = \\
&= \mathbf{K}(\mathbf{S}x)y(\mathbf{K}y)z = \\
&= \mathbf{S}x(\mathbf{K}y)z = xz(\mathbf{K}yz) = xzy.
\end{aligned}$$

Nazovimo dobijeni kombinator $\mathbf{C}$, tj.
$\mathbf{C} = \mathbf{S}(\mathbf{S}(\mathbf{KS})(\mathbf{S}(\mathbf{KK})\mathbf{S}))(\mathbf{KK})$.

Primer 3

Izvršiti sledeću apstrakciju: $[x, y].xyy$:

- Tražimo šta je $[y].xyy$:

$$[y].xyy = \mathbf{S}([y].xy)([y].y) = \mathbf{S}x\mathbf{I}.$$

- Zatim računamo

$$[x].\mathbf{S}x\mathbf{I} = \mathbf{S}([x].\mathbf{S}x)([x].\mathbf{I}) = \mathbf{SS}(\mathbf{KI}).$$

Primer 3 (provera)

$$\begin{aligned}\mathbf{SS}(\mathbf{KI})xy &= \mathbf{S}x((\mathbf{KI}x)y = \\ &= \mathbf{S}x\mathbf{I}y = \\ &= xy(\mathbf{I}y) = \\ &= xyy.\end{aligned}$$

Nazovimo dobijeni kombinator $\mathbf{W}$, tj. $\mathbf{W} = \mathbf{SS}(\mathbf{KI})$.

Primena kombinatorne logike

- Modelovanje nestriktnih funkcionalnih programskih jezika i hardvera
  - Unlambda
  - Kombinatori **B**, **C**, **K**, **I**, **S**, i **W** mogu se naći u Haskell-u kao `<$>`, `flip`, `pure`, `id`, `<*>`, i `join`, redom.
  - *Point-free* programiranje – u definiciji funkcija se ne imenuju argumenti, nego se funkcije predstavljaju kao kompozicije drugih funkcija. Suma liste se rekurzivno može napisati:

```
suma (x:xs) = x + suma xs
suma [] = 0,
```

    što se može zameniti

```
suma = foldr (+) 0.
```

- Redukcija grafova
- Parsiranje prirodnih jezika

Tehnologije

Tehnologije koje smo koristili prilikom dizajniranja programa su:

- Funkcionalna paradigma – Haskell
  - Haskell platforma
  - Biblioteka za rad sa stringovima (`Data.Strings`)
  - Biblioteka za rad sa grafičkim korisničkim interfejsom (`Graphics.UI.WX`)

Opis programa

- `charAt` – vraća karakter koji se nalazi na poziciji u niski
- `indexOf` (`indexOf_pom`) – vraća poziciju karaktera u niski ako postoji, a -1 inače
- `oslobodiZagrade` – zamenjuje sva pojavljivanja `(c)` sa `c` u niski, gde je `c` konstanta ili promenljiva
- `skiniZagrade` – briše zagrade sa početka i kraja niske
- `razdelnik` – vraća poziciju početka sledećeg terma (u apstrakciji, zdesna)
- `razdelnikSpreda` – vraća poziciju početka sledećeg terma (u primeni, sleva)

Opis programa (nastavak)

- kombinatorne funkcije (`i`, `k`, `s`) – implementiraju konstante **I**, **K** i **S**, redom
- `apstrakcija` – vrši apstrakciju izraza po jednoj promenljivoj
- `standardnaApstrakcija` – vrši apstrakciju izraza za svaku promenljivu i vraća postupak apstrakcije
- `standardnaApstrakcijaFinal` – vrši apstrakciju izraza za svaku promenljivu i vraća rezultat apstrakcije
- `primena` (`primenaPom`) – vrši primenu dobijenog kombinatora nad promenljivama
- `gui` – definiše korisnički interfejs programa
- `main` – pokreće program

Literatura

1. J. Roger Hindley, Jonathan P. Seldin, *Lambda-calculus and combinators, an introduction*, Cambridge University Press, 2008
2. Cem Bozsahin, *Combinatory Logic and Natural Language Parsing*, Middle East Technical University, Ankara
3. *Hackage*, `https://hackage.haskell.org/`
4. *Combinatory logic*, Wikipedia, `https://en.wikipedia.org/wiki/Combinatory_logic`
5. *Tacit programming*, Wikipedia, `https://en.wikipedia.org/wiki/Tacit_programming`
6. *Practical application of SKI calculus and BCKW*, `http://stackoverflow.com/questions/3689347/practical-application-of-ski-calculus-and-bckw`