Lambda kalkul

Z FITwiki

Obsah

Pojmy

Lambda kalkul je formální systém a výpočetní model používaný v teoretické informatice a matematice pro studium funkcí a rekurze. Jeho autory jsou Alonzo Church a Stephen Cole Kleene.

λ − *kalkul* - 3 typy výrazov:

- **proměnné** - obyčajné premenné - definujú väzbu s okolím
- **abstrakce** (definice funkce) - reprezentuje funkciu s jednou viazanou premennou (hlavička) a telom, ktoré je tvorené λ − *vyrazom* (např. $\lambda x.x+2$). Pokiaľ nejaká operácia vyžaduje viac parametrov, robí sa to vnorením abstrakcií (např. $\lambda xy.x-y$)
- **aplikace** (volání funkce) - ak máme 2 výrazy E1, E2, tak ak je to inak možné, môžeme aplikovať jeden na druhý. Pri aplikácii v poradí (E1,E2) sa E1 nazýva operátor (rator) a E2 operand (rand) (např. $(\lambda x.x+2)3$)

Odstránenie zbytočných zátvoriek:

aplikácie: zľava asociatívne - (...(E1,E2)...)En) => E1 E2 ... En
abstrakcia: dosah hlavičky siaha tak ďaleko, ako je to možné - (λV.(E1...En)) => λV.E1...En
zreťazenie bezprostredne vnorených abstrakcií: (λV1(... (λVn.(E)) => λV1 ... Vn.E

Voľné a viazané premenné - premenná je viazaná najbližšou hlavičkou naľavo od svojho výskytu. Pokiaľ je v tejto hlavičke uvedená, je viazaná, inak je voľná.

Konfluence - jakákoli posloupnost konverzí vede ke stejnému výsledku - ten je pak normální formou. (?)

Typovany lambda kalkul - ptali se me na to a taky jsem vubec nevedel. Nasel jsem, ze diky pouziti program. jazyku bez typu, jako je napr. PHP, je potreba zavest kontext, do ktereho promenna patri.

Konverzie

Sú 3 pravidlá pre konverziu (redukciu, modifikáciu) λ-výrazov na iné λ-výrazy.

- α-konverzia - výraz $\lambda V.E$ sa redukuje na $\lambda V'.E[V'/V]$. $E[V'/V]$ je substitúcia premennej V' za voľné výskyty premennej V vo výraze E, pričom substitúcia musí byť platná. Teda pri substitúcii sa žiadna voľná premenná nestane viazanou. V podstate sa premenuje *V* v hlavičke a v *E* sa nahradia všetky výskyty *V* okrem tých, ktoré sa nachádzajú v podvýraze v tvare λ*V.Ex*.
  - správně $\lambda x.xy \to_\alpha \lambda z.zy$
  - špatně $\lambda x.xy \to_\alpha \lambda y.yy$, $\lambda x.xy \to_\alpha \lambda x.xz$
- β-konverzia - ľubovľná aplikácia tvaru $(\lambda V.E_1)E_2$ sa redukuje na $E_1[E_2/V]$, pričom substitúcia musí byť platná. Ide vlastne o dosadenie výrazu E_2 za premennú V. Dá sa to chápať aj ako predanie parametra do funkcie.
  - správně $(\lambda xy.xy)(xy) \to_\alpha (\lambda xz.xz)(xy) \to_\beta (\lambda z.(xy)z) = \lambda z.xyz$
  - špatně $(\lambda xy.xy)(xy) \to_\beta \lambda y.(xy)y$
- η-konverzia ľubovľná abstrakcia tvaru $\lambda V.(EV)$ může být redukována na E, kde V nie je voľné v E
  - správně $\lambda x.(uv)x \to_\eta uv$, $\lambda x.xy \to_\eta \lambda fx.(xy)f$
  - špatně $\lambda x.(xy)x \to_\eta xy$ (protože V (zde konkrétně x) je volné v E (zde konkrétně (xy), což nesmí být))

Rovnosť

- 2 výrazy sú identické, pokiaľ sú zapísané rovnakou postupnosťou znakov
- 2 výrazy sú rovné, pokiaľ sa dajú konverziami previesť na identické výrazy

Zobecnená substitúcia

- Zobecnená substitúcia platí pre všetky výrazy (nemusí sa skúmať platnosť substitúcie)
- je definovaná rekurzívne cez E vo výraze E[E'/V] ako:

Výraz E	Výsledok substitúcie E[E'/V]
V	E'
V' (V <> V')	V'
E1E2	E1[E'/V]E2[E'/V]
λV.E1	λV.E1
λV'.E1 (V <> V' a V' nie je voľné v E')	λV'.E1[E'/V]
λV'.E1 (V <> V' a V' je voľné v E')	λX.E1[X/V'][E'/V] (X nie je voľná v E' alebo E1)

Normálna forma

- Výraz je v normálnej forme, pokiaľ neobsahuje žiadne β- ani η- redexy (výrazy, ktoré je možné zmeniť podľa β- alebo η-konverzie). Takže jediná konverzia, ktorú je možné vykonať je α-konverzia
- Výraz sa prevedie do normálnej formy opakovanou redukciou (konverziou) najlavejšieho β- alebo η- redexu, prípadne α-konverzii na zabránenie neplatnej substitúcie.

Operátor pevného bodu

Operátor pevného bodu nám slouží k definování rekurze a plati pren YE=E(YE)

- Definice: $LET\ Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$
- Bottom $LET \bot = Y(\lambda fx.f)$

Bottom je výraz, který bude na výstup neustále produkovat sám sebe. V podstatě modeluje nekonečnou smyčku v programu.

Reprezentace pravdivostních hodnot a operací nad nimi

Hodnoty true a false si můžeme definovat následujícím způsobem:

- LET **True** = $\lambda xy.x$
- LET **False** = $\lambda xy.y$

Po aplikaci β redukce nám tedy zůstane jen **y** nebo **x**.

Pak operace **NOT** nad nimi je definovaná následujícím způsobem:

LET **Not** = $\lambda t.t\ False\ True$

Ukázka Not True:

Not True =
$(\lambda t.t\ False\ True)\ True \rightarrow_\beta True\ False\ True \rightarrow (\lambda xy.x)\ False\ True \rightarrow_\beta (\lambda y.False)\ True \rightarrow_\beta False$

Další operace:

- LET **And** = $\lambda uv.uv\ False$
- LET **Or** = $\lambda uv.u\ True\ v$

Reprezentace čísel a operací nad nimi

Pro reprezentaci čísel v lambda kalkulu se používají Peanova (Pínova) čísla:

- LET 0 = $\lambda fn.n$
- LET 1 = $\lambda fn.fn$
- LET 2 = $\lambda fn.f(fn) = \lambda fn.f^2n$

Nad těmito čísly existují tyto operace (na státnice podle mě stačí znát **succ** a **iszero**, ale pro jistotu sem uvádím i některé další):

- LET **succ** = $\lambda xgm.xg(gm)$
- LET **iszero** = $\lambda m.m(\lambda v.False)\ True$

- LET **prev** = $\lambda xgm.snd\ (x\ (prefn\ g)(True,\ m))$
- LET **add** = $\lambda abgm.ag(bgm)$
- LET **sub** = $\lambda ab.b\ prev\ a$

Pomocné funkce:

- LET **(?:)** = $\lambda ctf.ctf$ (ternární operátor)
- LET **(_,_)** = $\lambda fse.efs$ (datový typ dvojice)
- LET **fst** = $\lambda p.p\ True$ (vybere první z dvojice)
- LET **snd** = $\lambda p.p\ False$ (vybere druhý z dvojice)
- LET **prefn** = $\lambda fp.(fst\ p\ ?\ (False,\ snd\ p) : (False,\ f\ (snd\ p)))$

Další materiál

1. Starý materiál od s3rvace [1] (https://fituska.eu/download/file.php?id=5440)
2. FLP ofi přednáška [2] (https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/FLP-IT/lectures/Lambda%20kalkul.pdf?cid=7369)
3. Wikipedia [3] (http://cs.wikipedia.org/wiki/Lambda_kalkul)

Citováno z „http://wiki.fituska.eu/index.php?title=Lambda_kalkul&oldid=13411“
Kategorie: Státnice 2016 | Státnice FPR | Funkcionální a logické programování

- Stránka byla naposledy editována 16. 6. 2016 v 13:58.