λ-kalkul rychle a pochopitelně

Petr "s3rvac" Zemek, s3rvac@seznam.cz
15. května 2007
Fakulta Informačních Technologií, Brno

Abstrakt

Tento článek se snaží co nejpochopitelněji a formou analogií prezentovat základy formálního systému λ-kalkulu, především z pohledu předmětu Principy programovacích jazyků. Výklad je neformální, pro formálnější popis se můžete podívat do opory IPP nebo na wikipedii. Tento článek si neklade za cíl plně nahradit studijní oporu, měl by sloužit především pro rychlejší pochopení λ-kalkulu. Proto se omlouvám za některá případná slovní spojení a zjednodušení.

1 Co to vlastně ten λ-kalkul je?

Jedná se o formální popis, který slouží jako základ pro funkcionální jazyky, takže všechny konstrukce v těchto jazycích jdou přepsat právě na λ-kalkul. Představit si to můžete tak, že byste programovali místo v LISPu v matematice :).

2 A k čemu mi to teda je?

Třeba k získání více bodů na zkoušce právě na otázkách z λ-kalkulu :). Jinak je fajn chápat alespoň ty základy, může to sloužit k lepšímu pochopení fungování funkcionálních jazyků.

3 Ok, tak jdeme na to aneb základní konstrukce

Základní prvky λ-kalkulu jsou tři následující (pojmenování dle IPP).

3.1 Proměnné

Obyčejné proměnné, tak jak je znáte z jiných jazyků, většinou se značí x, y, z.

3.2 Abstrakce

lam x . x + 2

parametr = x
tělo = x+2

Definice funkce – představte si např. funkci $f(x) = x + 2$, tak přesně taková funkce se v λ-kalkulu zapíše takto: $\lambda\, x\, .\, x + 2$. Část mezi λ a tečkou je parametr funkce a za tečkou se nachází tělo funkce.

Funkce v λ-kalkulu může mít jenom jeden parametr, takže pokud jich má mít víc, např. máme funkci $g(x, y) = x - y$, tak se to zapíše $\lambda\, x\, .\, \lambda\, y\, .\, x - y$. Toto se zkracuje na $\lambda\, x\, y\, .\, x - y$ a znamená to, že funkce má dva parametry a při zavolání vrací jejich rozdíl.

3.3 Aplikace

Volání funkce – když si vezmu naši funkci $f(x) = x + 2$, tak ta se zavolá např. s argumentem 3 takto: $f(3)$. Funkce v λ-kalkulu se volají podobně jako v LISPu (který aspoň trochu určitě znáte :), kde se funkce volá takto: $(f\ 3)$, tzn. nejdříve bude název funkce a poté její argumenty. Funkce f se v λ-kalkulu tedy zavolá takto: $(\lambda\, x\, .\, x + 2)\ 3$.

Vysvětlení by mělo být už jasnější. Číslo 3 se dosadí za parametr x a přejde se do těla, tam se ke 3 přičte 2 a výsledkem je 5.

Každý z těchto tří prvků se označuje jako λ-výraz – ty budu značit velkými písmeny, např. E a znamená to, že pod E se může skrývat buď proměnná, abstrakce (definice funkce) nebo aplikace (volání funkce).

4 Volné a vázané proměnné

mezi lambda a tečkou

Proměnná je v λ-výrazu vázaná, pokud se jedná o parametr nějaké funkce, takže např. ve výrazu $(\lambda x\,.\,y\,x$ je x vázaná proměnná. Ostatní proměnné (v minulém příkladu y) jsou volné. Nehledejte v tom nic složitého. Proměnná se vždycky váže na nejbližší lambdu směrem doleva, takže ve výrazu $(\lambda x\,((\lambda x\,.\,x)\,w))$ se proměnná x uprostřed výrazu váže na lambdu co je hned vlevo od ní a ne na tu úplně vlevo! Proměnná w je samozřejmě volná.

5 Zjednodušování zápisu

Tady toho moc vysvětlit nejde, takže se kdyžtak mrkněte na tuto část do opory, já tady jen ve stručnosti shrnu, co je možné zjednodušit na co:

- místo $(A\,B)$ lze napsat $A\,B$
- místo $(\lambda V\,.\,(A\,B\,C))$ lze napsat $\lambda V\,.\,A\,B\,C$
- místo $(\lambda F(\lambda G(\lambda H\,.\,x)))$ lze napsat jen $\lambda F\,G\,H\,.\,x$

Další zjednodušení je možné provést pojmenováním nějakého výrazu pomocí **LET**. Např. $\mathbf{LET\,K} = \lambda x\,y\,.\,x$ a poté můžu **K** používat v jiných výrazech, např. to může být argument při volání funkce atd.

6 Redukce a konverze

A dostáváme se k zřejmě nejobtížnější části, nicméně věřím, že se mi to podaří popsat dostatečně pochopitelně :). Jenom poznámka – to, jestli se říká konverze nebo redukce je jedno, protože někde to je tak a jinde naopak. Pomocí nich je možné měnit λ-výrazy, takže jsou velmi důležité. V λ-kalkulu máme celkem tři druhy konverzí.

6.1 α-konverze („alfa“)

čím / co

Idea: Vyjdařuje myšlenku, že názvy proměnných nejsou důležité. Jedná se o „prachsprostou“ substituci. Ovšem pozor, daná substituce musí být *platná*, což znamená se žádná volná proměnná nesmí stát vázanou (což by ve výsledku znamenalo změnu významu daného výrazu).

Př.: $\lambda V\,.\,E$ můžeme konvertovat na $\lambda W\,.\,E\,[W/V]$, kde jsme provedli substituci W za V (to, jakou substituci jsme provedli, je uvedeno v hranatých závorkách). Skoro nic na tom není, je to asi nejjednodušší konverze. Jen si je třeba dát pozor na platnost substituce (viz. výše).

6.2 β-konverze („beta“)

Idea: Vyjadřuje myšlenku volání (aplikace) funkce. Zde se substituuje proměnná za parametr volané funkce. Opět je třeba dát pozor na platnost substituce.

Př.: $(\lambda V\,.\,W)\,M$ můžeme konvertovat na $W\,[M/V]$, kde jsme jako argument při volání funkce použili M (substituce), tudíž se vykoná tělo funkce a dostaneme vždy W, protože v těle funkce se vůbec V neobjevuje, takže ve výsledku je úplně jedno (v tomto případě), s jakým argumentem tuto funkci voláme.

6.3 η-konverze („eta“)

Idea: Tuto konverzi není třeba umět používat, uvádím ji zde jen pro úplnost (i v opoře je). Vyjadřuje myšlenku ekvivalence dvou funkcí[1].

Př.: $\lambda V\,.\,(E\,V)$ lze konvertovat na E (V nesmí být volné v E).

Výraz, který je možné podle nějaké redukce změnit se nazývá *redex* podle zkratky z „reducible expression“. Jedná-li se o redex podle příslušné konverze, tak se nazývá α-, β-, či η-redexem.

[1] Pro zájemce viz. http://en.wikipedia.org/wiki/Extensionality

7 Kompletní příklad na redukci

Tak, a teď ještě ukážu jeden kompletní příklad (byl na semestrální zkoušce v roce 2004). Zadání je následující: *Výraz v λ-kalkulu* $(\lambda\, x\, y\, .\, x)\, u\, v$ *lze redukovat na...* Máme tedy funkci se dvěma parametry a na pravé straně proměnné, které budou použity jako argumenty dané funkce (budeme danou funkci volat). Nejdříve provedeme β-konverzi, takže první parametr funkce (x) nahradíme proměnnou u a dostaneme $(\lambda\, y\, .\, u)\, v\, [u/x]$. První parametr (argument) máme tedy vyřešený. Poté provedeme další β-konverzi, kde za druhý parametr substituujeme proměnnou v a dostaneme $u\, [v/y]$ (to, co je v tělu funkce), protože druhý parametr nemá na výsledek vliv. Řešením je tudíž u.

8 Rekurze

Někde jsem na toto téma viděl otázku, takže to tady taky napíšu. Rekurzi v λ-kalkulu nelze napsat takto, protože funkce **f** ještě nebyla definována:

$$\mathbf{f} a_1 \ldots a_n = \ldots \mathbf{f} \ldots$$

Musíme použít tzv. *operátor pevného bodu*[2] (toto si zapamatujte), např. **Y** a definice funkce bude vypadat následovně:

$$\mathbf{LET}\ \mathbf{f} = \mathbf{Y}(\lambda\, a_1\ \ldots\ a_n\, .\, \ldots\ \mathbf{f}\ \ldots)$$

9 Nějaké ty drobnosti nakonec...

Zde bych si doporučoval si zapamatovat rozdíl mezi *rovností* a *identičností* dvou výrazů. Dva výrazy jsou *identické*, pokud jsou zapsány úplně stejně (v textové podobě). Takže ke zjištění identičnosti stačí na oba výrazy „poštvat“ strcmp(V_1, V_2) :). Naopak dva výrazy jsou si *rovny*, pokud lze jeden pomocí konverzí převést na druhý.

Dále tu máme pojem *zobecněná substituce*. Ta znamená, že když by při konverzích nastal konflikt proměnných, tak se předtím provede α-konverze (substituce), která proměnné vhodně přejmenuje.

10 Závěr

Tak, to by bylo vše k základům λ-kalkulu :). Doufám, že vám to pomůže ve studiu a především na zkoušce. Ono ten λ-kalkul není zase tak složitý, jenom je třeba si udělat vhodné asociace, např. s nějakým programovacím jazykem. Pak jde všechno jednodušeji. Tak zase někdy...

Reference

[1] Kolář D., *Studijní opora k předmětu IPP, III. část*, 2006

[2] Wikipedia, *Internetová encyklopedie*, http://en.wikipedia.org/wiki/Lambda_calculus

[2] Pro zájemce viz. http://en.wikipedia.org/wiki/Fixed_point_combinator