Fundamentele Limbajelor de Programare

Seminar & Laborator
Lambda calcul

3 Februarie 2022

Lambda-calculul (sau λ-calculul) a fost dezvoltat între 1929-1932 de Alonzo Church și a fost propus ca sistem formal pentru logica matematică. În 1935, a demonstrat că orice funcție calculabilă peste numerele naturale, poate fi calculată în λ-calcul. Tot în 1935, independent de Church, Alan Turing a dezvoltat mecanismul numit astăzi mașina Turing, iar în 1936 a argumentat și el că orice funcție calculabilă peste numerele naturale poate fi calculată de o mașină Turing și, în plus, a arătat echivalența celor două modele de calcul (λ-calcul și mașini Turing), ducând la ceea e numim astăzi *Teza Church-Turing*.

Sintaxa λ-calculului:

$$t = x \mid \lambda x.t \mid t\ t$$

λ-termeni. Fie $Var = \{x, y, z, ...\}$ o mulțime infinită de variabile. Mulțimea λ-termenilor ΛT este definită inductiv, astfel:

[Variabilă] $Var \subseteq \Lambda T$
[Aplicare] dacă $t_1, t_2 \in \Lambda T$ atunci $(t_1\ t_2) \in \Lambda T$
[Abstractizare] dacă $x \in Var$ și $t \in \Lambda T$, atunci $(\lambda x.t) \in \Lambda T$

Convenții de scriere.

- în scrierea λ-termenilor vom elimina parantezele exterioare și vom scrie, de exemplu, xy în loc de (xy), sau $\lambda x.xy$ în loc de $(\lambda x.(xy))$;
- aplicarea este asociativă la stânga: $t_1t_2t_3$ este $(t_1t_2)t_3$;
- corpul abstractizării este extins la dreapta: $\lambda x.t_1t_2$ este $\lambda x.(t_1t_2)$;
- scriem $\lambda xyz.t$, în loc de $\lambda x.\lambda y.\lambda z.t$.

1 Variabile libere și legate

Variabile libere și legate. Pentru un termen $\lambda x.t$ spunem că:

- aparițiile variabilei x în t sunt legate *(bound)*;
- λx este legătura *(binder)*, iar t este domeniul *(scope)* legării;
- o apariție a unei variabile este liberă *(free)* dacă apare într-o poziție în care nu este legată.

Un termen fără variabile libere se numește **închis** *(closed)*.

Mulțimea variabilelor libere FV(t). Pentru un λ-termen t, mulțimea variabilelor libere este definită astfel:

[Variabilă] $FV(x) = \{x\}$

[Aplicare] $FV(t_1 t_2) = FV(t_1) \cup FV(t_2)$

[Abstractizare] $FV(\lambda x.t) = FV(t) - \{x\}$

Exercițiul 1 *Calculați mulțimea variabilelor libere pentru următorii λ-termeni:*

a. $\lambda x.xy$

b. $x\lambda x.xy$

c. $x(\lambda xy.xyz)(\lambda v.yv)$

d. $\lambda t.((\lambda xyz.yzx)t)$

2 Substituții

Fie t un λ-termen și $x \in Var$. Pentru un λ-termen u vom nota prin $[u/x]t$ rezultatul înlocuirii tuturor aparițiilor libere ale lui x cu u în t.

[Variabilă] $[u/x]x = u$

[Variabilă] $[u/x]y = y$ dacă $x \neq y$

[Aplicare] $[u/x](t_1 t_2) = [u/x]t_1[u/x]t_2$

[Abstractizare] $[u/x]\lambda y.t = \lambda y.[u/x]t$ unde $x \neq y$ și $y \notin FV(u)$

Variabilele legate pot fi redenumite.

Exercițiul 2 *Aplicați substituțiile indicate în următorii λ-termeni:*

a. $[y/x]\lambda z.x$

b. $[y/x]\lambda y.x$

c. $[\lambda z.z/x](\lambda x.yx)$

d. $[\lambda z.z/x](\lambda y.yx)$

3 α-conversie (α-echivalență)

α-conversia $=_\alpha$ este relația binară care satisface următoarele proprietăți:

[Reflexivitate] $t =_\alpha t$

[Simetrie] $t_1 =_\alpha t_2$ implică $t_2 =_\alpha t_1$

[Tranzitivitate] $t_1 =_\alpha t_2$ și $t_2 =_\alpha t_3$ implică $t_1 =_\alpha t_3$

[Redenumire] $\lambda x.t =_\alpha \lambda y.[y/x]t$ dacă $y \notin FV(t)$

[Compatibilitate] $t_1 =_\alpha t_2$ implică $tt_1 =_\alpha tt_2$, $t_1t =_\alpha t_2t$ și $\lambda x.t_1 =_\alpha \lambda x.t_2$

Compatibilitatea cu substituția.

$$t_1 =_\alpha t_2 \text{ și } u_1 =_\alpha u_2 \text{ implică } [u_1/x]t_1 =_\alpha [u_2/x]t_2$$

Exercițiul 3 *Verificați care dintre α-conversiile următoare sunt adevărate:*

a. $\lambda x.x =_\alpha \lambda y.y$

b. $\lambda x.y =_\alpha \lambda y.x$

c. $\lambda x.xy =_\alpha \lambda x.xz$

d. $\lambda x.xy =_\alpha \lambda y.yy$

e. $x\lambda x.xy =_\alpha x\lambda z.zy$

f. $x\lambda x.xy =_\alpha y\lambda x.xy$

4 β-reducția

β-reducția este o relație definită pe mulțimea α-termenilor, $\beta \subseteq \Lambda T \times \Lambda T$, unde

[Aplicarea] $(\lambda x.y)u \to_\beta [u/x]t$

[Compatibilitatea] $t_1 \to_\beta t_2$ implică $tt_1 \to_\beta tt_2$, $t_1t \to_\beta t_2t$ și $\lambda x.t_1 \to_\beta \lambda x.t_2$

Închiderea reflexivă și tranzitivă a acestei relații se notează $\to_\beta^* \subseteq \Lambda T \times \Lambda T$, iar $t_1 \to_\beta^* t_2$ dacă există $n \geq 0$ și $u_0, ..., u_n$ asfel încât $t_1 =_\alpha u_0 \to_\beta u_1 \to_\beta ... \to_\beta u_n =_\alpha t_2$.

Exercițiul 4 *Să se aplice două β-reducții succesive asupra λ-termenului $(\lambda x.(\lambda y.yx)z)v$. Este o singură variantă de aplicare?*

Un termen poate fi β-redus în mai multe moduri, iar proprietatea de confluență ne asigură că vom ajunge întotdeauna la același rezultat. (forma normală este unică, modulo α-echivalență)

5 Implementarea în Haskell a λ-calculului

În cadrul acestui laborator, vom implementa λ-calculul în Haskell. Pentru aceasta, vom defini următorul tip de date:

```
data Term = V Variable
    | App Term Term
    | Lam Variable Term
```

Exercițiul 5 *Scrieți o instanță a clasei* `Show` *pentru tipul de date* `Term` *și testați pe cel puțin un exemplu.*

```
lambdaExp :: Term
lambdaExp = App (Lam "x" (Lam "y" (App (App (V "x") (V "y")) (V "z")))) (V "y")
```

Exercițiul 6 *Definiți o funcție* `freeVars :: Term -> [Variable]`*, care primește un λ-termen și returnează lista variabilelor libere din termenul respectiv.*

```
freeVars :: Term -> [Variable]
freeVars = undefined
```