Curs 6

2020-2021

Fundamentele Limbajelor de Programare

λ-calcul

- În 1929-1932 Church a propus λ-calculul ca sistem formal pentru logica matematică. În 1935 a argumentat că orice funcție calculabilă peste numere naturale poate fi calculată in λ-calcul.
- În 1935, independent de Church, Turing a dezvoltat mecanismul de calcul numit astăzi Mașina Turing. În 1936 și el a argumentat câ orice funcție calculabilă peste numere naturale poate fi calculată de o mașină Turing. De asemenea, a arătat echivalența celor două modele de calcul. Această echivalență a constituit o indicație puternică asupra "universalității" celor două modele, conducând la ceea ce numim astăzi "Teza Church-Turing".

Referințe

λ-calcul: sintaxa

Lambda Calcul - sintaxă

$$
\begin{array}{rll}
t = & x & \text{(variabilă)} \\
& \mid \lambda x.\, t & \text{(abstractizare)} \\
& \mid t\ t & \text{(aplicare)}
\end{array}
$$

λ-calcul: sintaxa

Lambda Calcul - sintaxă

$$
\begin{array}{lll}
t = & x & \text{(variabilă)} \\
 & \mid \lambda x.\, t & \text{(abstractizare)} \\
 & \mid t\ \ t & \text{(aplicare)}
\end{array}
$$

λ-termeni

Fie $Var = \{x, y, z, \ldots\}$ 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 $(\lambda x.t) \in \Lambda T$

Lambda termeni

λ-termeni: exemple

- x, y, z
- (xy), (yx), $(x(yx))$
- $(\lambda x.x)$, $(\lambda x.(xy))$, $(\lambda z.(xy))$, $(\lambda x.(\lambda z.(xy)))$
- $((\lambda x.x)y)$, $((\lambda x.(xz))y)$, $((\lambda x.x)(\lambda y.y))$

Lambda termeni

λ-termeni: exemple

- x, y, z
- (xy), (yx), $(x(yx))$
- $(\lambda x.x)$, $(\lambda x.(xy))$, $(\lambda z.(xy))$, $(\lambda x.(\lambda z.(xy)))$
- $((\lambda x.x)y)$, $((\lambda x.(xz))y)$, $((\lambda x.x)(\lambda y.y))$

Convenții:

- se elimină parantezele exterioare
- aplicarea este asociativă la stînga: $t_1 t_2 t_3$ este $(t_1 t_2)t_3$
- corpul abstractizării este extins la dreapta: $\lambda x.t_1 t_2$ este $\lambda x.(t_1 t_2)$ (nu $(\lambda x.t_1)t_2$)
- scriem $\lambda xyz.t$ în loc de $\lambda x.\lambda y.\lambda z.t$

Lambda termeni / Funcții anonime

λ-termeni: exemple

- x, y, z
- $(xy), (yx), (x(yx))$
- $(\lambda x.x), (\lambda x.(xy)), (\lambda z.(xy)), (\lambda x.(\lambda z.(xy)))$
- $((\lambda x.x)y), ((\lambda x.(xz))y), ((\lambda x.x)(\lambda y.y))$

λ-termeni/ funcții anonime în Haskell

În Haskell, \ e folosit în locul simbolului λ și -> în locul punctului.

$\lambda x.x * x$ este `\x -> x * x`

$\lambda x.x > 0$ este `\x -> x > 0`

Variabile libere și legate

Apariții 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 e legată.

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

Exemplu:

- $\lambda x.x$ este un termen închis
- $\lambda x.xy$ nu este termen închis, x este legată, y este liberă
- în termenul $x(\lambda x.xy)$ prima apariție a lui x este liberă, a doua este legată.

Variabile libere

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] $FV(\lambda x.t) = FV(t) \setminus \{x\}$

Exemplu:

$$
\begin{aligned}
FV(\lambda x.xy) = \quad & FV(xy) \setminus \{x\} \\
& (FV(x) \cup FV(y)) \setminus \{x\} \\
& (\{x\} \cup \{y\}) \setminus \{x\} \\
& \{y\}
\end{aligned}
$$

Variabile libere

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] $FV(\lambda x.t) = FV(t) \setminus \{x\}$

Exemplu:

$$
\begin{aligned}
FV(\lambda x.xy) = \quad & FV(xy) \setminus \{x\} \\
& (FV(x) \cup FV(y)) \setminus \{x\} \\
& (\{x\} \cup \{y\}) \setminus \{x\} \\
& \{y\}
\end{aligned}
$$

$FV(x\lambda x.xy) =$

Variabile libere

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] $FV(\lambda x.t) = FV(t) \setminus \{x\}$

Exemplu:

$$\begin{aligned} FV(\lambda x.xy) = \quad & FV(xy) \setminus \{x\} \\ & (FV(x) \cup FV(y)) \setminus \{x\} \\ & (\{x\} \cup \{y\}) \setminus \{x\} \\ & \{y\} \\ FV(x\lambda x.xy) = \quad & \{x, y\} \end{aligned}$$

Substituții

Fie t un λ-termen $x \in Var$.

Definiție intuitivă

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

Exemple: Dacă x, y, z sunt variabile distincte atunci

- $[y/x]\lambda z.x = \lambda z.y$
- $[(\lambda z.zw)/x](\lambda y.x) = \lambda y.\lambda z.zw$

Substituții

Exemple: Dacă x, y, z sunt variabile distincte atunci

- $[y/x]\lambda z.x = \lambda z.y$
- Cine este $[y/x]\lambda y.x$?

Substituții

Exemple: Dacă x, y, z sunt variabile distincte atunci

- $[y/x]\lambda z.x = \lambda z.y$
- Cine este $[y/x]\lambda y.x$?
  Dacă folosim definiția intuitivă obținem
  $[y/x]\lambda y.x = \lambda y.y$ ceea ce este greșit!

Substituții

Exemple: Dacă x, y, z sunt variabile distincte atunci

- $[y/x]\lambda z.x = \lambda z.y$
- Cine este $[y/x]\lambda y.x$?
  Dacă folosim definiția intuitivă obținem
  $[y/x]\lambda y.x = \lambda y.y$ ceea ce este greșit!

  Cum procedăm pentru a repara greșeala? Observăm că $\lambda y.x$ desemneaza o funcție constantă, aceeași funcție putând fi reprezentată prin $\lambda z.x$. Aplicarea corectă a substituției este:
  $[y/x]\lambda y.x = [y/x]\lambda z.x = \lambda z.y$

Substituții

Exemple: Dacă x, y, z sunt variabile distincte atunci

- $[y/x]\lambda z.x = \lambda z.y$
- Cine este $[y/x]\lambda y.x$?
  Dacă folosim definiția intuitivă obținem
  $[y/x]\lambda y.x = \lambda y.y$ ceea ce este greșit!

  Cum procedăm pentru a repara greșeala? Observăm că $\lambda y.x$ desemneaza o funcție constantă, aceeași funcție putând fi reprezentată prin $\lambda z.x$. Aplicarea corectă a substituției este:
  $[y/x]\lambda y.x = [y/x]\lambda z.x = \lambda z.y$

  Avem libertatea de a redenumi variabilele legate!

α-conversie (α-echivalență)

α-conversia $=_\alpha$

[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
$tt_1 =_\alpha tt_2$, $t_1 t =_\alpha t_2 t$ și $\lambda x.t_1 =_\alpha \lambda x.t_2$

α-conversie (α-echivalență)

α-conversia $=_\alpha$

[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
$tt_1 =_\alpha tt_2$, $t_1 t =_\alpha t_2 t$ și $\lambda x.t_1 =_\alpha \lambda x.t_2$

Exemplu:
$[xy/x](\lambda y.yx) =_\alpha [xy/x](\lambda z.zx) =_\alpha \lambda z.z(xy)$

Substituții

Definirea substituției

Rezultatul substituirii lui x cu u în t este definit astfel:

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

α-conversia este compatibilă cu substituția

- $t_1 =_\alpha t_2$ și $u_1 =_\alpha u_2$ implică $[u_1/x]t_1 =_\alpha [u_2/x]t_2$

Vom lucra modulo α-conversie, doi termeni α-echivalenți vor fi considerați "egali".

β-reducție

β-reducția este o relație pe mulțimea α-termenilor.

β-reducția $\rightarrow_\beta$, $\stackrel{*}{\rightarrow}_\beta$

- un singur pas $\rightarrow_\beta \subseteq \Lambda T \times \Lambda T$

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

  [Compatibilitatea] $t_1 \rightarrow_\beta t_2$ implică

  $tt_1 \rightarrow_\beta tt_2$, $t_1 t \rightarrow_\beta t_2 t$ și $\lambda x.t_1 \rightarrow_\beta \lambda x.t_2$

- zero sau mai mulți pași $\stackrel{*}{\rightarrow}_\beta \subseteq \Lambda T \times \Lambda T$

  $t_1 \stackrel{*}{\rightarrow}_\beta t_2$ dacă există $n \geq 0$ și $u_0, \ldots, u_n$ astfel încât

  $t_1 =_\alpha u_0 \rightarrow_\beta u_1 \rightarrow_\beta \cdots \rightarrow_\beta u_n =_\alpha t_2$

Confluența β-reducției

Teorema Church-Rosser

Dacă $t \stackrel{*}{\rightarrow}_\beta t_1$ și $t \stackrel{*}{\rightarrow}_\beta t_2$ atunci există u astfel încât $t_1 \stackrel{*}{\rightarrow}_\beta u$ și $t_2 \stackrel{*}{\rightarrow}_\beta u$.

Exemplu:

β-forma normală

Intuitiv, o formă normală este un termen care nu mai poate fi redus (sau punctul final al unui calcul).

Formă normal ă

- un λ-termen căruia nu i se mai poate aplica reducerea într-un pas $\to_\beta$ se numește *β-formă normală*
- dacă $t \overset{*}{\to}_\beta u_1$, $t \overset{*}{\to}_\beta u_2$ și u_1, u_2 sunt η-forme normale atunci, datorită confluenței, $u_1 =_\alpha u_2$
- un λ-termen poate avea cel mult o β-formă normală (modulo α-echivalență)

β-forma normală

Formă normal ă

- un λ-termen căruia nu i se mai poate aplica reducerea într-un pas $\to_\beta$ se numește *β-formă normală*
- dacă $t \stackrel{*}{\to}_\beta u_1$, $t \stackrel{*}{\to}_\beta u_2$ și u_1, u_2 sunt η-forme normale atunci, datorită confluenței, $u_1 =_\alpha u_2$
- un λ-termen poate avea cel mult o β-formă normală (modulo α-echivalență)

Exemplu:

- zv este β-formă normală pentru $(\lambda x.(\lambda y.yx)z)v$
  $(\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda y.yv)z \to_\beta zv$
- există termeni care **nu** pot fi reduși la o β-formă normală, de exemplu $(\lambda x.xx)(\lambda x.xx)$

β-conversia

Intuitiv, β-conversia extinde β-reducția în ambele direcții.

β-conversia $=_\beta$

- $=_\beta \subseteq \Lambda T \times \Lambda T$
  $t_1 =_\beta t_2$ dacă există $n \geq 0$ și $u_0, \ldots, u_n$ astfel încât
  $t_1 =_\alpha u_0$, $u_n =_\alpha t_2$ și, pentru orice i, $u_i \to_\beta u_{i+1}$ sau $u_{i+1} \to_\beta u_i$

β-conversia

Intuitiv, β-conversia extinde β-reducția în ambele direcții.

β-conversia $=_\beta$

- $=_\beta \subseteq \Lambda T \times \Lambda T$
  $t_1 =_\beta t_2$ dacă există $n \geq 0$ și $u_0, \ldots, u_n$ astfel încât
  $t_1 =_\alpha u_0$, $u_n =_\alpha t_2$ și, pentru orice i, $u_i \to_\beta u_{i+1}$ sau $u_{i+1} \to_\beta u_i$

Observații

- $=_\beta$ este o relație de echivalență

β-conversia

Intuitiv, β-conversia extinde β-reducția în ambele direcții.

β-conversia $=_\beta$

- $=_\beta \subseteq \Lambda T \times \Lambda T$
  $t_1 =_\beta t_2$ dacă există $n \geq 0$ și $u_0, \ldots, u_n$ astfel încât
  $t_1 =_\alpha u_0$, $u_n =_\alpha t_2$ și, pentru orice i, $u_i \to_\beta u_{i+1}$ sau $u_{i+1} \to_\beta u_i$

Observații

- $=_\beta$ este o relație de echivalență
- pentru t_1, t_2 λ-termeni și u_1, u_2 β-forme normale
  dacă $t_1 \stackrel{*}{\to}_\beta u_1$, $t_2 \stackrel{*}{\to}_\beta u_2$ și $u_1 =_\alpha u_2$ atunci $t_1 =_\beta t_2$

β-conversia

β-conversia $=_\beta$

- $=_\beta \subseteq \Lambda T \times \Lambda T$
  $t_1 =_\beta t_2$ dacă există $n \geq 0$ și $u_0, \ldots, u_n$ astfel încât
  $t_1 =_\alpha u_0$, $u_n =_\alpha t_2$ și, pentru orice i, $u_i \to_\beta u_{i+1}$ sau $u_{i+1} \to_\beta u_i$
- $=_\beta$ este o relație de echivalență

β-conversia

β-conversia $=_\beta$

- $=_\beta \subseteq \Lambda T \times \Lambda T$
  $t_1 =_\beta t_2$ dacă există $n \geq 0$ și $u_0, \ldots, u_n$ astfel încât
  $t_1 =_\alpha u_0$, $u_n =_\alpha t_2$ și, pentru orice i, $u_i \rightarrow_\beta u_{i+1}$ sau $u_{i+1} \rightarrow_\beta u_i$
- $=_\beta$ este o relație de echivalență
- pentru t_1, t_2 λ-termeni și u_1, u_2 β-forme normale
  dacă $t_1 \stackrel{*}{\rightarrow}_\beta u_1$, $t_2 \stackrel{*}{\rightarrow}_\beta u_2$ și $u_1 =_\alpha u_2$ atunci $t_1 =_\beta t_2$

β-conversia reprezintă "egalitatea prin calcul", iar β-reducția (modulo α-conversie) oferă o procedură de decizie pentru aceasta.

Pe săptămâna viitoare!