Curs 13

2021-2022

Fundamentele limbajelor de programare

Cuprins

λ-calcul

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 mulţime infinită de variabile.
Mulţimea 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 $x \in Var$ ş i $t \in \Lambda T$ atunci $(\lambda x.t) \in \Lambda T$

Lambda termeni

Convenţii:

- se elimină parantezele exterioare
- aplicarea este asociativă la stânga: $t_1\,t_2\,t_3$ 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 $\lambda x.\lambda y.\lambda z.t$

Mulţimea variabilelor libere $FV(t)$

Pentru un λ-termen t mulţimea variabilelor libere este definită astfel:

[Variabilă] $FV(x) = \{x\}$

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

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

Substituţii

Fie t un λ-termen si $x \in Var$.

Definiţie intuitivă

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

Definirea substituţiei

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

α-conversie (α-echivalenţă)

Definim următoarea relaţie peste termeni:

α-conversia $=_\alpha$

[Reflexivitate] $t =_\alpha t$

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

[Tranzitivitate] $t_1 =_\alpha t_2$ şi $t_2 =_\alpha t_3$ 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

$$t\, t_1 =_\alpha t\, t_2,\ t_1 t =_\alpha t_2 t \text{ şi } \lambda x.t_1 =_\alpha \lambda x.t_2$$

Vom lucra modulo α-conversie, doi termeni α-echivalenţi vor fi consideraţi "egali".

Daca $t_1 =_\alpha t_2$, atunci vom folosi notaţia $[u/x]t_1 =_\alpha [u/x]t_2$ pentru a semnala faptul că în t_1 am redenumit variabilele.

α-conversie

Exemplu:

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

  Observăm că $\lambda z.yz =_\alpha \lambda x.yx$

α-conversie

Exemplu:

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

  Observăm că $\lambda z.yz =_\alpha \lambda x.yx$
- $[y/z]\lambda xy.zzx = \lambda x.[y/z]\lambda y.zzx =_\alpha \lambda x.[y/z]\lambda v.zzx = \lambda x.\lambda v.[y/z](zzx) = \lambda xv.yyx$

β-reducţie

β-reducţia este o relaţie pe mulţimea λ-termenilor, lucrand modula α-echivalenta.

β-reducţia $\rightarrow_\beta$, $\overset{*}{\rightarrow}_\beta$

- 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

  $t\ t_1 \rightarrow_\beta t\ t_2$, $t_1 t \rightarrow_\beta t_2 t$ ş i $\lambda x.t_1 \rightarrow_\beta \lambda x.t_2$

β-reducţie

β-reducţia este o relaţie pe mulţimea λ-termenilor, lucrand modula α-echivalenta.

β-reducţia $\rightarrow_\beta$, $\overset{*}{\rightarrow}_\beta$

- 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

  $t\, t_1 \rightarrow_\beta t\, t_2$, $t_1 t \rightarrow_\beta t_2 t$ şi $\lambda x.t_1 \rightarrow_\beta \lambda x.t_2$

- zero sau mai mulţi paş i $\overset{*}{\rightarrow}_\beta \subseteq \Lambda T \times \Lambda T$

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

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

β-reducţie

Să considerăm termenul $(\lambda x.(\lambda y.yx)z)v$

- $(\lambda x.(\lambda y.yx)z)v \rightarrow_\beta (\lambda y.yv)z \rightarrow_\beta zv$

β-reducţie

Să considerăm termenul $(\lambda x.(\lambda y.yx)z)v$

- $(\lambda x.(\lambda y.yx)z)v \rightarrow_\beta (\lambda y.yv)z \rightarrow_\beta zv$
- $(\lambda x.(\lambda y.yx)z)v \rightarrow_\beta (\lambda x.zx)v \rightarrow_\beta zv$

β-reducţie

Să considerăm termenul $(\lambda x.(\lambda y.yx)z)v$

- $(\lambda x.(\lambda y.yx)z)v \rightarrow_\beta (\lambda y.yv)z \rightarrow_\beta zv$
- $(\lambda x.(\lambda y.yx)z)v \rightarrow_\beta (\lambda x.zx)v \rightarrow_\beta zv$

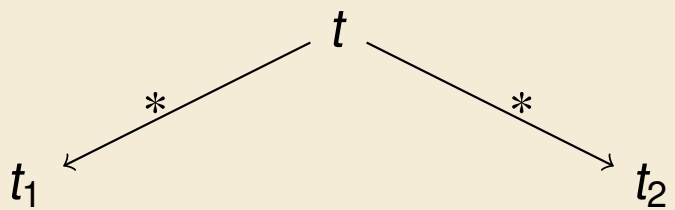
Observăm că un termen poate fi β-redus în mai multe moduri.

Proprietatea de confluenţă ne asigură că vom ajunge întotdeauna la acelaşi rezultat.

Confluenţa β-reducţiei

Teorema Church-Rosser

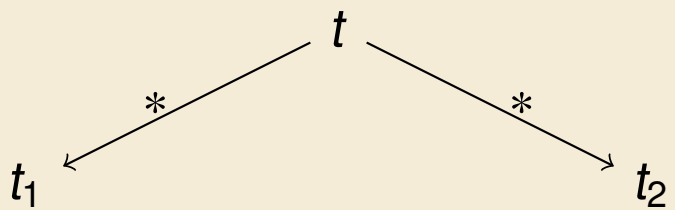
Dacă $t \overset{*}{\rightarrow}_\beta t_1$ şi $t \overset{*}{\rightarrow}_\beta t_2$



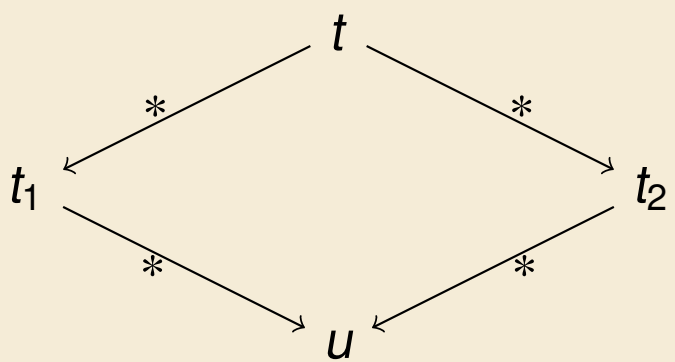
Confluenţa β-reducţiei

Teorema Church-Rosser

Dacă $t \overset{*}{\to}_\beta t_1$ şi $t \overset{*}{\to}_\beta t_2$



atunci există u astfel încât $t_1 \overset{*}{\to}_\beta u$ şi $t_2 \overset{*}{\to}_\beta u$.



β-forma normală

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

Formă normală

- un λ-termen căruia nu i se mai poate aplica reducerea într-un pas $\rightarrow_\beta$ se numeşte *β-formă normală*
- dacă $t \overset{*}{\rightarrow}_\beta u_1$, $t \overset{*}{\rightarrow}_\beta u_2$ şi u_1, u_2 sunt β-forme normale atunci, datorită confluenţei, $u_1 =_\alpha u_2$
- un λ-termen poate avea cel mult o β-formă normală (modulo α-echivalenţă)

β-forma normală

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

Formă normală

- un λ-termen căruia nu i se mai poate aplica reducerea într-un pas $\rightarrow_\beta$ se numeşte *β-formă normală*
- dacă $t \overset{*}{\rightarrow}_\beta u_1$, $t \overset{*}{\rightarrow}_\beta u_2$ şi u_1, u_2 sunt β-forme normale atunci, datorită confluenţei, $u_1 =_\alpha u_2$
- un λ-termen poate avea cel mult o β-formă normală (modulo α-echivalenţă)

Exemplu:

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

β-conversia

Intuitiv, β-conversia extinde β-reducţia în ambele direcţii.

- $(\lambda y.yv)z \rightarrow_\beta zv \leftarrow_\beta (\lambda x.zx)v$

β-conversia

Intuitiv, β-conversia extinde β-reducţia în ambele direcţii.

- $(\lambda y.yv)z \to_\beta zv \leftarrow_\beta (\lambda x.zx)v$
- $(\lambda y.yv)z \leftarrow_\beta (\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda x.zx)v$

β-conversia

Intuitiv, β-conversia extinde β-reducţia în ambele direcţii.

- $(\lambda y.yv)z \to_\beta zv \leftarrow_\beta (\lambda x.zx)v$
- $(\lambda y.yv)z \leftarrow_\beta (\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda x.zx)v$

β-conversia $=_\beta$

- $=_\beta \subseteq \Lambda T \times \Lambda T$
  $t_1 =_\beta t_2$ dacă există $n \geq 0$ şi $u_0, \ldots, u_n$ astfel încât
  $t_1 =_\alpha u_0$, $u_n =_\alpha t_2$ şi, 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-reducţia în ambele direcţii.

- $(\lambda y.yv)z \to_\beta zv \leftarrow_\beta (\lambda x.zx)v$
- $(\lambda y.yv)z \leftarrow_\beta (\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda x.zx)v$

β-conversia $=_\beta$

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

Exemplu: $(\lambda y.yv)z =_\beta (\lambda x.zx)v$

β-conversia

β-conversia $=_\beta$

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

β-conversia

β-conversia $=_\beta$

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

Observaţii

- $=_\beta$ este o relaţie de echivalenţă

β-conversia

β-conversia $=_\beta$

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

Observaţii

- $=_\beta$ este o relaţie de echivalenţă
- pentru t_1, t_2 λ-termeni şi u_1, u_2 β-forme normale
  dacă $t_1 \overset{*}{\to}_\beta u_1$, $t_2 \overset{*}{\to}_\beta u_2$ şi $u_1 =_\alpha u_2$ 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 $n \geq 0$ şi $u_0, \ldots, u_n$ astfel încât
  $t_1 =_\alpha u_0$, $u_n =_\alpha t_2$ şi, pentru orice i, $u_i \to_\beta u_{i+1}$ sau $u_{i+1} \to_\beta u_i$

Observaţii

- $=_\beta$ este o relaţie de echivalenţă
- pentru t_1, t_2 λ-termeni şi u_1, u_2 β-forme normale
  dacă $t_1 \overset{*}{\to}_\beta u_1$, $t_2 \overset{*}{\to}_\beta u_2$ şi $u_1 =_\alpha u_2$ atunci $t_1 =_\beta t_2$

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

Expresivitatea λ-calculului

Expresivitatea λ-calculului

Vom arăta cum putem exprima în lambda calcul

- Booleeni
- Numere naturale

Expresivitatea λ-calculului

Vom arăta cum putem exprima în lambda calcul

- Booleeni
- Numere naturale

Intuiție: tipurile de date sunt codificate de capabilități.

- Booleeni - capabilitatea de a alege între două alternative
- Numere naturale - capabilitatea de a itera de un număr dat de ori

Booleeni

Intuiție: Capabilitatea de a alege între două alternative.

Codificare: Un Boolean este o funcție cu 2 argumente reprezentând ramurile unei alegeri.

Booleeni

Intuiție: Capabilitatea de a alege între două alternative.

Codificare: Un Boolean este o funcție cu 2 argumente reprezentând ramurile unei alegeri.

Incepem prin a defini doi λ-termeni pentru a coda "true" şi "false":

- $\mathbf{T} = \lambda xy.x$
  din cele două alternative o alege pe prima
- $\mathbf{F} = \lambda xy.y$
  din cele două alternative o alege pe a doua

Negaţia

- $\mathbf{T} = \lambda xy.x$
- $\mathbf{F} = \lambda xy.y$
- $\mathbf{not} = \lambda b.b\ \mathbf{F}\ \mathbf{T}$

Negaţia

- $\mathbf{T} = \lambda xy.x$
- $\mathbf{F} = \lambda xy.y$
- $\mathbf{not} = \lambda b.b\ \mathbf{F}\ \mathbf{T}$

Avem urmatoarele β-reducţii:

- $\mathbf{not}\ \mathbf{T} = (\lambda b.b\ \mathbf{F}\ \mathbf{T})\mathbf{T} \rightarrow_\beta \mathbf{T}\ \mathbf{F}\ \mathbf{T} = (\lambda xy.x)\ \mathbf{F}\ \mathbf{T} \rightarrow_\beta \mathbf{F}$

Negaţia

- $\mathbf{T} = \lambda xy.x$
- $\mathbf{F} = \lambda xy.y$
- $\mathbf{not} = \lambda b.b\ \mathbf{F}\ \mathbf{T}$

Avem urmatoarele β-reducţii:

- $\mathbf{not}\ \mathbf{T} = (\lambda b.b\ \mathbf{F}\ \mathbf{T})\mathbf{T} \rightarrow_\beta \mathbf{T}\ \mathbf{F}\ \mathbf{T} = (\lambda xy.x)\ \mathbf{F}\ \mathbf{T} \rightarrow_\beta \mathbf{F}$
- $\mathbf{not}\ \mathbf{F} = (\lambda b.b\ \mathbf{F}\ \mathbf{T})\mathbf{F} \rightarrow_\beta \mathbf{F}\ \mathbf{F}\ \mathbf{T} = (\lambda xy.y)\ \mathbf{F}\ \mathbf{T} \rightarrow_\beta \mathbf{T}$

Conjunctia

- **T** $= \lambda xy.x$
- **F** $= \lambda xy.y$
- **not** $= \lambda b.b$ **F T**
- **and** $= \lambda ab.ab$**F**

Conjunctia

- $\mathbf{T} = \lambda xy.x$
- $\mathbf{F} = \lambda xy.y$
- $\mathbf{not} = \lambda b.b\ \mathbf{F}\ \mathbf{T}$
- $\mathbf{and} = \lambda ab.ab\mathbf{F}$

Avem urmatoarele β-reducţii:

- $\mathbf{and}\ \mathbf{T}\ \mathbf{T} = (\lambda ab.ab\mathbf{F})\ \mathbf{T}\ \mathbf{T} \rightarrow_\beta \mathbf{T}\ \mathbf{T}\ \mathbf{F} = (\lambda xy.x)\mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}$

Conjunctia

- $\mathbf{T} = \lambda xy.x$
- $\mathbf{F} = \lambda xy.y$
- $\mathbf{not} = \lambda b.b\ \mathbf{F}\ \mathbf{T}$
- $\mathbf{and} = \lambda ab.ab\mathbf{F}$

Avem urmatoarele β-reducţii:

- $\mathbf{and}\ \mathbf{T}\ \mathbf{T} = (\lambda ab.ab\mathbf{F})\ \mathbf{T}\ \mathbf{T} \rightarrow_\beta \mathbf{T}\ \mathbf{T}\ \mathbf{F} = (\lambda xy.x)\mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}$
- $\mathbf{and}\ \mathbf{T}\ \mathbf{F} = (\lambda ab.ab\mathbf{F})\ \mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}\ \mathbf{F}\ \mathbf{F} = (\lambda xy.x)\mathbf{F}\ \mathbf{F} \rightarrow_\beta \mathbf{F}$

Conjunctia

- $\mathbf{T} = \lambda xy.x$
- $\mathbf{F} = \lambda xy.y$
- $\mathbf{not} = \lambda b.b\ \mathbf{F}\ \mathbf{T}$
- $\mathbf{and} = \lambda ab.ab\mathbf{F}$

Avem urmatoarele β-reducţii:

- $\mathbf{and}\ \mathbf{T}\ \mathbf{T} = (\lambda ab.ab\mathbf{F})\ \mathbf{T}\ \mathbf{T} \rightarrow_\beta \mathbf{T}\ \mathbf{T}\ \mathbf{F} = (\lambda xy.x)\mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}$
- $\mathbf{and}\ \mathbf{T}\ \mathbf{F} = (\lambda ab.ab\mathbf{F})\ \mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}\ \mathbf{F}\ \mathbf{F} = (\lambda xy.x)\mathbf{F}\ \mathbf{F} \rightarrow_\beta \mathbf{F}$
- $\mathbf{and}\ \mathbf{F}\ \mathbf{T} = (\lambda ab.ab\mathbf{F})\ \mathbf{F}\ \mathbf{T} \rightarrow_\beta \mathbf{F}\ \mathbf{T}\ \mathbf{F} = (\lambda xy.y)\mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{F}$

Conjunctia

- $\mathbf{T} = \lambda xy.x$
- $\mathbf{F} = \lambda xy.y$
- $\mathbf{not} = \lambda b.b\ \mathbf{F}\ \mathbf{T}$
- $\mathbf{and} = \lambda ab.ab\mathbf{F}$

Avem urmatoarele β-reducţii:

- $\mathbf{and}\ \mathbf{T}\ \mathbf{T} = (\lambda ab.ab\mathbf{F})\ \mathbf{T}\ \mathbf{T} \rightarrow_\beta \mathbf{T}\ \mathbf{T}\ \mathbf{F} = (\lambda xy.x)\mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}$
- $\mathbf{and}\ \mathbf{T}\ \mathbf{F} = (\lambda ab.ab\mathbf{F})\ \mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}\ \mathbf{F}\ \mathbf{F} = (\lambda xy.x)\mathbf{F}\ \mathbf{F} \rightarrow_\beta \mathbf{F}$
- $\mathbf{and}\ \mathbf{F}\ \mathbf{T} = (\lambda ab.ab\mathbf{F})\ \mathbf{F}\ \mathbf{T} \rightarrow_\beta \mathbf{F}\ \mathbf{T}\ \mathbf{F} = (\lambda xy.y)\mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{F}$
- $\mathbf{and}\ \mathbf{F}\ \mathbf{F} = (\lambda ab.ab\mathbf{F})\ \mathbf{F}\ \mathbf{F} \rightarrow_\beta \mathbf{F}\ \mathbf{F}\ \mathbf{F} = (\lambda xy.y)\mathbf{F}\ \mathbf{F} \rightarrow_\beta \mathbf{F}$

Disjunctia

- **T** $= \lambda xy.x$
- **F** $= \lambda xy.y$
- **not** $= \lambda b.b$ **F T**
- **and** $= \lambda ab.ab$**F**
- **or** $=$

Disjunctia

- **T** $= \lambda xy.x$
- **F** $= \lambda xy.y$
- **not** $= \lambda b.b$ **F T**
- **and** $= \lambda ab.ab$**F**
- **or** $= \lambda ab.a$**T**b

Disjunctia

- $\mathbf{T} = \lambda xy.x$
- $\mathbf{F} = \lambda xy.y$
- $\mathbf{not} = \lambda b.b\ \mathbf{F}\ \mathbf{T}$
- $\mathbf{and} = \lambda ab.ab\mathbf{F}$
- $\mathbf{or} = \lambda ab.a\mathbf{T}b$

Avem urmatoarele β-reducţii:

- $\mathbf{or}\ \mathbf{T}\ \mathbf{T} = (\lambda ab.a\mathbf{T}b)\ \mathbf{T}\ \mathbf{T} \rightarrow_\beta \mathbf{T}\ \mathbf{T}\ \mathbf{T} = (\lambda xy.x)\mathbf{T}\ \mathbf{T} \rightarrow_\beta \mathbf{T}$
- $\mathbf{or}\ \mathbf{T}\ \mathbf{F} = (\lambda ab.a\mathbf{T}b)\ \mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}\ \mathbf{T}\ \mathbf{F} = (\lambda xy.x)\mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{T}$
- $\mathbf{or}\ \mathbf{F}\ \mathbf{T} = (\lambda ab.a\mathbf{T}b)\ \mathbf{F}\ \mathbf{T} \rightarrow_\beta \mathbf{F}\ \mathbf{T}\ \mathbf{T} = (\lambda xy.y)\mathbf{T}\ \mathbf{T} \rightarrow_\beta \mathbf{T}$
- $\mathbf{or}\ \mathbf{F}\ \mathbf{F} = (\lambda ab.a\mathbf{T}b)\ \mathbf{F}\ \mathbf{F} \rightarrow_\beta \mathbf{F}\ \mathbf{T}\ \mathbf{F} = (\lambda xy.y)\mathbf{T}\ \mathbf{F} \rightarrow_\beta \mathbf{F}$

Conditional

Putem sa definim

- **if T** $u\ w\ =\ u$
- **if F** $u\ w\ =\ w$

Observati ca u si w sunt orice λ-termeni, nu doar Booleeni.

Conditional

Putem sa definim

- **if T** $u\ w\ =\ u$
- **if F** $u\ w\ =\ w$

Observati ca u si w sunt orice λ-termeni, nu doar Booleeni.

Avem

$$\textbf{if} = \lambda buw.buw$$

unde b este un boolean.

Numere naturale

Intuiție: Capabilitatea de a itera o funcție de un număr de ori peste o valoare inițială

Codificare: Un număr natural este o funcție cu 2 argumente

- s funcția care se iterează
- z valoarea inițială

0 ::= $\lambda s\ z.z$ — s se iterează de 0 ori, deci valoarea inițială

Numere naturale

Intuiție: Capabilitatea de a itera o funcție de un număr de ori peste o valoare inițială

Codificare: Un număr natural este o funcție cu 2 argumente

- s funcția care se iterează
- z valoarea inițială

0 ::= $\lambda s\ z.z$ — s se iterează de 0 ori, deci valoarea inițială

1 ::= $\lambda s\ z.s\ z$ — funcția iterată o dată aplicată valorii inițiale

Numere naturale

Intuiție: Capabilitatea de a itera o funcție de un număr de ori peste o valoare inițială

Codificare: Un număr natural este o funcție cu 2 argumente

- s funcția care se iterează
- z valoarea inițială

0 ::= $\lambda s\ z.z$ — s se iterează de 0 ori, deci valoarea inițială

1 ::= $\lambda s\ z.s\ z$ — funcția iterată o dată aplicată valorii inițiale

2 ::= $\lambda s\ z.s(s\ z)$ — s iterată de 2 ori, aplicată valorii inițiale

Numere naturale

Intuiție: Capabilitatea de a itera o funcție de un număr de ori peste o valoare inițială

Codificare: Un număr natural este o funcție cu 2 argumente

- s funcția care se iterează
- z valoarea inițială

0 ::= $\lambda s\ z.z$ — s se iterează de 0 ori, deci valoarea inițială

1 ::= $\lambda s\ z.s\ z$ — funcția iterată o dată aplicată valorii inițiale

2 ::= $\lambda s\ z.s(s\ z)$ — s iterată de 2 ori, aplicată valorii inițiale

...

8 ::= $\lambda s\ z.s(s(s(s(s(s(s\ z))))))$

...

Numere naturale

Intuiție: Capabilitatea de a itera o funcție de un număr de ori peste o valoare inițială

Codificare: Un număr natural este o funcție cu 2 argumente

- s funcția care se iterează
- z valoarea inițială

0 ::= $\lambda s\ z.z$ — s se iterează de 0 ori, deci valoarea inițială

1 ::= $\lambda s\ z.s\ z$ — funcția iterată o dată aplicată valorii inițiale

2 ::= $\lambda s\ z.s(s\ z)$ — s iterată de 2 ori, aplicată valorii inițiale

...

8 ::= $\lambda s\ z.s(s(s(s(s(s(s\ z))))))$

...

Observație: **0** $=$ **false**

Operații aritmetice de bază

succ ::= $\lambda n\ s\ z.s\ (n\ s\ z)$

Operații aritmetice de bază

$$\textbf{succ} ::= \lambda n\ s\ z.s\ (n\ s\ z)$$

$$\textbf{succ}\ \textbf{0} = (\lambda n\ s\ z.s\ (n\ s\ z))\textbf{0} \rightarrow_\beta \lambda s\ z.s\ (\textbf{0}\ s\ z) \rightarrow_\beta \lambda s\ z.s\ z = \textbf{1}$$

Operații aritmetice de bază

$$\textbf{succ} ::= \lambda n\ s\ z.s\ (n\ s\ z)$$

$$\textbf{succ}\ \textbf{0} = (\lambda n\ s\ z.s\ (n\ s\ z))\textbf{0} \rightarrow_\beta \lambda s\ z.s\ (\textbf{0}\ s\ z) \rightarrow_\beta \lambda s\ z.s\ z = \textbf{1}$$

$$\textbf{plus} ::= \lambda m\ n\ s\ z.m\ s\ (n\ s\ z)$$

Operații aritmetice de bază

$$\textbf{succ} ::= \lambda n\ s\ z.s\ (n\ s\ z)$$

$$\textbf{succ}\ \textbf{0} = (\lambda n\ s\ z.s\ (n\ s\ z))\textbf{0} \rightarrow_\beta \lambda s\ z.s\ (\textbf{0}\ s\ z) \rightarrow_\beta \lambda s\ z.s\ z = \textbf{1}$$

$$\textbf{plus} ::= \lambda m\ n\ s\ z.m\ s\ (n\ s\ z)$$

$$\begin{aligned}\textbf{plus}\ \textbf{3}\ \textbf{2} &= (\lambda m\ n\ s\ z.m\ s\ (n\ s\ z))\ \textbf{3}\ \textbf{2} \rightarrow_\beta \lambda s\ z.\textbf{3}\ s\ (\textbf{2}\ s\ z) \\ &\rightarrow_\beta \lambda s\ z.s(s(s(\textbf{2}\ s\ z))) \rightarrow_\beta \lambda s\ z.s(s(s(s(s(z))))) = \textbf{5}\end{aligned}$$

Operații aritmetice de bază

$$\textbf{succ} ::= \lambda n\ s\ z.s\ (n\ s\ z)$$

$$\textbf{succ 0} = (\lambda n\ s\ z.s\ (n\ s\ z))\textbf{0} \rightarrow_\beta \lambda s\ z.s\ (\textbf{0}\ s\ z) \rightarrow_\beta \lambda s\ z.s\ z = \textbf{1}$$

$$\textbf{plus} ::= \lambda m\ n\ s\ z.m\ s\ (n\ s\ z)$$

$$\begin{aligned}\textbf{plus 3 2} &= (\lambda m\ n\ s\ z.m\ s\ (n\ s\ z))\ \textbf{3 2} \rightarrow_\beta \lambda s\ z.\textbf{3}\ s\ (\textbf{2}\ s\ z) \\ &\rightarrow_\beta \lambda s\ z.s(s(s(\textbf{2}\ s\ z))) \rightarrow_\beta \lambda s\ z.s(s(s(s(s(z))))) = \textbf{5}\end{aligned}$$

$$\textbf{mult} ::= \lambda m\ n\ s.m\ (n\ s)$$

Scaderea este mai complicata deoarece nu exista numere negative.

Perechi

Intuiție: Capabilitatea de a aplica o funcție componentelor perechii

Codificare: O funcție cu 3 argumente reprezentând componentele perechii și funcția ce vrem să o aplicăm lor.

pair ::= $\lambda x\ y.\lambda f.f\ x\ y$
Constructorul de perechi

Exemplu: **pair**$x\ y \to_\beta^2 \lambda f.f\ x\ y$

perechea (x, y) reprezintă capabilitatea de a aplica o funcție de două argumente lui x si apoi lui y.

Operații pe perechi

$\texttt{pair} ::= \lambda x\ y.\lambda f.f\ x\ y$

$\texttt{pair}\ xy \equiv_\beta f\ x\ y$

$\texttt{fst} ::= \lambda p.p\ \mathit{true}$ — *true* alege prima componentă

$\texttt{fst}\ (\texttt{pair}\ x\ y) \to_\beta \texttt{pair}\ x\ y\ \mathit{true} \to_\beta^3 \mathit{true}\ x\ y \to_\beta^2 x$

$\texttt{snd} ::= \lambda p.p\ \mathit{false}$ — *false* alege a doua componentă

$\texttt{snd}\ (\texttt{pair}\ x\ y) \to_\beta \texttt{pair}\ x\ y\ \mathit{false} \to_\beta^3 \mathit{false}\ x\ y \to_\beta^2 y$

Liste

Intuiție: Capabilitatea de a agrega o listă

Codificare: O funcție cu 2 argumente:
funcția de agregare și *valoarea inițială*
Lista $[3, 5]$ este reprezentată prin $a\ 3\ (a\ 5\ i)$

Liste

Intuiție: Capabilitatea de a agrega o listă

Codificare: O funcție cu 2 argumente:
funcția de agregare și *valoarea inițială*
Lista $[3, 5]$ este reprezentată prin $a\ 3\ (a\ 5\ i)$

`null` ::= $\lambda a\ i.i$ — lista vidă

`cons` ::= $\lambda x\ l.\lambda a\ i.a\ x\ (l\ a\ i)$
Constructorul de liste

Exemplu: $\texttt{cons}\ 3\ (\texttt{cons}\ 5\ \texttt{null}) \to_\beta^2 \lambda a\ i.a\ 3\ (\texttt{cons}\ 5\ \texttt{null}\ a\ i) \to_\beta^4 \lambda a\ i.a\ 3\ (a\ 5\ (\texttt{null}\ a\ i)) \to_\beta^2 \lambda a\ i.a\ 3\ (a\ 5\ i)$

Lista $[3, 5]$ reprezintă capabilitatea de a agrega elementele 3 si apoi 5 dată fiind o funcție de agregare a și o valoare implicită i.

Pe săptămâna viitoare!