

Paradigme de Programare

Ș.I. dr. ing. Andrei Olaru

Departamentul Calculatoare

2015 – 2016, semestrul 2

Cursul 3: Calcul Lambda

λ

$$(\lambda x. x \quad y) \rightarrow_{\beta} y$$

Cursul 3: Calcul Lambda

- 1 Introducere
- 2 Lambda-expresii
- 3 Reducere
- 4 Evaluare
- 5 Limbajul lambda-0 și incursiune în TDA

Introducere

- ne punem problema dacă putem realiza un calcul sau nu \rightarrow pentru a demonstra trebuie să avem un model simplu al calculului (**cum realizăm calculul**, în mod formal).
- un model de calculabilitate trebuie să fie cât mai simplu, atât ca număr de **operații** disponibile cât și ca **mode de construcție a valorilor**.
- corectitudinea unui program se demonstrează mai ușor dacă limbajul de programare este mai apropiat de mașina teoretică (modelul abstract de calculabilitate).

- **Model de calculabilitate** (Alonzo Church, 1932) – introdus în cadrul cercetărilor asupra fundamentelor matematicii.

[http://en.wikipedia.org/wiki/Lambda_calculus]

- sistem formal pentru exprimarea calculului.
- **Echivalent** cu Mașina Turing (v. Teza Church-Turing)
- Axat pe conceptul matematic de **funcție** – totul este o funcție

- Aplicații importante în
 - **programare**
 - demonstrarea formală a **corectitudinii** programelor, datorită modelului simplu de execuție

- Baza teoretică a numeroase **limbaje**:
LISP, Scheme, Haskell, ML, F#, Clean, Clojure, Scala, Erlang etc.

Lambda-expresii

- 1 $x \rightarrow$ variabila (**numele**) x



1 $x \rightarrow$ variabila (numele) x

2 $\lambda x.x \rightarrow$ funcția identitate



Exemplu



Exemplu

- 1 $x \rightarrow$ variabila (numele) x
- 2 $\lambda x.x \rightarrow$ funcția identitate
- 3 $\lambda x.\lambda y.x \rightarrow$ funcție selector



- 1 $x \rightarrow$ variabila (numele) x
- 2 $\lambda x.x \rightarrow$ funcția identitate
- 3 $\lambda x.\lambda y.x \rightarrow$ funcție selector
- 4 $(\lambda x.x y) \rightarrow$ aplicația funcției identitate asupra parametrului actual y



Exemplu

- 1 $x \rightarrow$ variabila (numele) x
- 2 $\lambda x.x \rightarrow$ funcția identitate
- 3 $\lambda x.\lambda y.x \rightarrow$ funcție selector
- 4 $(\lambda x.x y) \rightarrow$ aplicația funcției identitate asupra parametrului actual y
- 5 $(\lambda x.(x x) \lambda x.x) \rightarrow ?$



Exemplu

- 1 $x \rightarrow$ variabila (numele) x
- 2 $\lambda x.x \rightarrow$ funcția identitate
- 3 $\lambda x.\lambda y.x \rightarrow$ funcție selector
- 4 $(\lambda x.x y) \rightarrow$ aplicația funcției identitate asupra parametrului actual y
- 5 $(\lambda x.(x x) \lambda x.x) \rightarrow ?$



Intuitiv, evaluarea aplicației $(\lambda x.x y)$ presupune substituția textuală a lui x , în corp, prin $y \rightarrow$ rezultat y .

+ λ -expresie

- **Variabilă**: o variabilă x este o λ -expresie;
- **Funcție**: dacă x este o variabilă și E este o λ -expresie, atunci $\lambda x.E$ este o λ -expresie, reprezentând funcția anonimă, unară, cu parametrul formal x și corpul E ;
- **Aplicație**: dacă F și A sunt λ -expresii, atunci $(F A)$ este o λ -expresie, reprezentând aplicația expresiei F asupra parametrului actual A .



$((\lambda x.\lambda y.x z) t)$



$((\lambda x.\lambda y.x z) t)$

\parallel

substituție

\Downarrow

$(\lambda y.z t)$


$$((\lambda x.\lambda y.x z) t)$$
$$\Downarrow$$

substituție

$$\Downarrow$$
$$(\lambda y.z t)$$
$$\Downarrow$$

substituție

$$\Downarrow$$
$$z$$


$$((\lambda x.\lambda y.x z) t)$$
$$\Downarrow$$

substituție

$$\Downarrow$$
$$(\lambda y.z t)$$
$$\Downarrow$$

substituție

$$\Downarrow$$
$$z$$

nu mai este nicio funcție de aplicat



$$((\lambda x.\lambda y.x z) t)$$

$$\Downarrow$$

substituție

$$\Downarrow$$

$$(\lambda y.z t)$$

$$\Downarrow$$

substituție

$$\Downarrow$$

$$z$$

nu mai este nicio funcție de aplicat

· cum știm **ce** reducem, **cum** reducem, în ce **ordine**, și ce **aparitii** ale variabilelor înlocuim?

Reducere

- β -redex: o λ -expresie de forma: $(\lambda x.E A)$
 - E – λ -expresie – este corpul funcției
 - A – λ -expresie – este parametrul actual

- β -redexul se reduce la $E_{[A/x]}$ – E cu toate aparițiile **libere** ale lui x din E înlocuite cu A prin substituție textuală.

+ **Apariție legată** O apariție x_n a unei variabile x este legată într-o expresie E dacă:

- $E = \lambda x.F$ sau
- $E = \dots \lambda x_n.F \dots$ sau
- $E = \dots \lambda x.F \dots$ și x_n apare în F .

+ **Apariție liberă** O apariție a unei variabile este liberă într-o expresie dacă nu este legată în acea expresie.

- Atenție! În raport cu o expresie dată!



· O apariție **legată în expresie** este o apariție a parametrului formal al unei funcții definite **în expresie**, în corpul funcției; o apariție **liberă** este o apariție a parametrului formal al unei funcții definite **în exteriorul** expresiei, sau nu este parametru formal al niciunei funcții.

- $x_{<1>}$ ← apariție liberă



· O apariție **legată în expresie** este o apariție a parametrului formal al unei funcții definite **în expresie**, în corpul funcției; o apariție **liberă** este o apariție a parametrului formal al unei funcții definite **în exteriorul** expresiei, sau nu este parametru formal al niciunei funcții.

• $x_{<1>}$ ← apariție liberă

• $(\lambda y. x_{<1>} z)$ ← apariție încă liberă, nu o leagă nimeni



· O apariție **legată în expresie** este o apariție a parametrului formal al unei funcții definite **în expresie**, în corpul funcției; o apariție **liberă** este o apariție a parametrului formal al unei funcții definite **în exteriorul** expresiei, sau nu este parametru formal al niciunei funcții.

• $x_{\langle 1 \rangle}$ ← apariție liberă

• $(\lambda y. x_{\langle 1 \rangle} z)$ ← apariție încă liberă, nu o leagă nimeni

• $\lambda x_{\langle 2 \rangle}. (\lambda y. x_{\langle 1 \rangle} z)$ ← $\lambda x_{\langle 2 \rangle}$ leagă apariția $x_{\langle 1 \rangle}$



· O apariție **legată în expresie** este o apariție a parametrului formal al unei funcții definite **în expresie**, în corpul funcției; o apariție **liberă** este o apariție a parametrului formal al unei funcții definite **în exteriorul** expresiei, sau nu este parametru formal al niciunei funcții.

• $x_{\langle 1 \rangle}$ ← apariție liberă

• $(\lambda y. x_{\langle 1 \rangle} z)$ ← apariție încă liberă, nu o leagă nimeni

• $\lambda x_{\langle 2 \rangle}. (\lambda y. x_{\langle 1 \rangle} z)$ ← $\lambda x_{\langle 2 \rangle}$ leagă apariția $x_{\langle 1 \rangle}$

• $(\lambda x_{\langle 2 \rangle}. (\lambda y. x_{\langle 1 \rangle} z) x_{\langle 3 \rangle})$ ← apariția x_3 este liberă – este în exteriorul corpului funcției cu parametrul formal x (λx_2)
corp λx_2

Mod de gândire

· O apariție **legată în expresie** este o apariție a parametrului formal al unei funcții definite **în expresie**, în corpul funcției; o apariție **liberă** este o apariție a parametrului formal al unei funcții definite **în exteriorul** expresiei, sau nu este parametru formal al niciunei funcții.

• $x_{\langle 1 \rangle}$ ← apariție liberă

• $(\lambda y. x_{\langle 1 \rangle} z)$ ← apariție încă liberă, nu o leagă nimeni

• $\lambda x_{\langle 2 \rangle}. (\lambda y. x_{\langle 1 \rangle} z)$ ← $\lambda x_{\langle 2 \rangle}$ leagă apariția $x_{\langle 1 \rangle}$

• $(\lambda x_{\langle 2 \rangle}. (\lambda y. x_{\langle 1 \rangle} z) x_{\langle 3 \rangle})$ ← apariția x_3 este liberă – este în exteriorul corpului funcției cu parametrul formal x (λx_2)
 corp λx_2

• $\lambda x_{\langle 4 \rangle}. (\lambda x_{\langle 2 \rangle}. (\lambda y. x_{\langle 1 \rangle} z) x_{\langle 3 \rangle})$ ← $\lambda x_{\langle 4 \rangle}$ leagă apariția $x_{\langle 3 \rangle}$

Introducere

λ -Expresii

Reducere

Evaluare

λ_0 și TDA

+ | **O variabilă este legată** într-o expresie dacă **toate** aparițiile sale sunt legate în acea expresie.

+ | **O variabilă este liberă** într-o expresie dacă nu este legată în acea expresie i.e. dacă **cel puțin o** apariție a sa este liberă în acea expresie.

- Atenție! În raport cu o **expresie** dată!

În expresia $E = (\lambda x.x x)$, evidențiem aparițiile lui x :

$(\lambda \underset{\langle 1 \rangle}{x} . \underbrace{\underset{\langle 2 \rangle}{x} \underset{\langle 3 \rangle}{x}}_F)$.

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$ în E
- $x_{\langle 3 \rangle}$ în E
- $x_{\langle 2 \rangle}$ în $F!$
- x în E și F



Exemplu

În expresia $E = (\lambda x.x x)$, evidențiem aparițiile lui x :

$(\lambda \underset{\langle 1 \rangle}{x} . \underbrace{\underset{\langle 2 \rangle}{x} \underset{\langle 3 \rangle}{x}}_F)$.

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$ **legate** în E
- $x_{\langle 3 \rangle}$ în E
- $x_{\langle 2 \rangle}$ în $F!$
- x în E și F



Exemplu

În expresia $E = (\lambda x.x x)$, evidențiem aparițiile lui x :

$(\lambda \underset{\langle 1 \rangle}{x} . \underbrace{\underset{\langle 2 \rangle}{x} \underset{\langle 3 \rangle}{x}}_F)$.

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$ **legate** în E
- $x_{\langle 3 \rangle}$ **liberă** în E
- $x_{\langle 2 \rangle}$ în F !
- x în E și F



Exemplu

În expresia $E = (\lambda x.x x)$, evidențiem aparițiile lui x :

$(\lambda \underset{\langle 1 \rangle}{x} . \underbrace{\underset{\langle 2 \rangle}{x} \underset{\langle 3 \rangle}{x}}_F)$.

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$ **legate** în E
- $x_{\langle 3 \rangle}$ **liberă** în E
- $x_{\langle 2 \rangle}$ **liberă** în F !
- x în E și F



Exemplu

În expresia $E = (\lambda x.x x)$, evidențiem aparițiile lui x :

$(\lambda \underset{\langle 1 \rangle}{x} . \underbrace{\underset{\langle 2 \rangle}{x} \underset{\langle 3 \rangle}{x}}_F)$.

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$ **legate** în E
- $x_{\langle 3 \rangle}$ **liberă** în E
- $x_{\langle 2 \rangle}$ **liberă** în F !
- x **liberă** în E și F



Exemplu

Exemplu 2

În expresia $E = (\lambda x. \lambda z. (z x) (z y))$, evidențiem aparițiile:

$$(\lambda_{\langle 1 \rangle} x_{\langle 1 \rangle} . \lambda_{\langle 1 \rangle} z_{\langle 1 \rangle} . (z_{\langle 2 \rangle} x_{\langle 2 \rangle}) (z_{\langle 3 \rangle} y_{\langle 1 \rangle})).$$

$\underbrace{\hspace{10em}}_F$

- $x_{\langle 1 \rangle}, x_{\langle 2 \rangle}, z_{\langle 1 \rangle}, z_{\langle 2 \rangle}$ în E
- $y_{\langle 1 \rangle}, z_{\langle 3 \rangle}$ în E
- $z_{\langle 1 \rangle}, z_{\langle 2 \rangle}$ în F
- $x_{\langle 2 \rangle}$ în F
- x în $E,$ în F
- y în E
- z în $E,$ în F



Exemplu

Exemplu 2

În expresia $E = (\lambda x. \lambda z. (z x) (z y))$, evidențiem aparițiile:

$(\lambda_{\langle 1 \rangle} x_{\langle 1 \rangle} . \lambda_{\langle 1 \rangle} z_{\langle 1 \rangle} . (z_{\langle 2 \rangle} x_{\langle 2 \rangle}) (z_{\langle 3 \rangle} y_{\langle 1 \rangle}))$
 $\underbrace{\hspace{10em}}_F$

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$, $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în E
- $y_{\langle 1 \rangle}$, $z_{\langle 3 \rangle}$ în E
- $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ în F
- $x_{\langle 2 \rangle}$ în F
- x în E , în F
- y în E
- z în E , în F



Exemplu

Exemplu 2

În expresia $E = (\lambda x. \lambda z. (z x) (z y))$, evidențiem aparițiile:

$(\lambda_{\langle 1 \rangle} x_{\langle 1 \rangle} . \lambda_{\langle 1 \rangle} z_{\langle 2 \rangle} . (z_{\langle 2 \rangle} x_{\langle 2 \rangle}) (z_{\langle 3 \rangle} y_{\langle 1 \rangle}))$
 $\underbrace{\hspace{10em}}_F$

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$, $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în E
- $y_{\langle 1 \rangle}$, $z_{\langle 3 \rangle}$ **libere** în E
- $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ în F
- $x_{\langle 2 \rangle}$ în F
- x în E , în F
- y în E
- z în E , în F

E

Exemplu

Exemplu 2

În expresia $E = (\lambda x. \lambda z. (z x) (z y))$, evidențiem aparițiile:

$(\lambda_{\langle 1 \rangle} x_{\langle 2 \rangle} . \lambda_{\langle 1 \rangle} z_{\langle 2 \rangle} . (z_{\langle 2 \rangle} x_{\langle 2 \rangle}) (z_{\langle 3 \rangle} y_{\langle 1 \rangle}))$
 $\underbrace{\hspace{10em}}_F$

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$, $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în E
- $y_{\langle 1 \rangle}$, $z_{\langle 3 \rangle}$ **libere** în E
- $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în F
- $x_{\langle 2 \rangle}$ în F
- x în E , în F
- y în E
- z în E , în F



Exemplu

Exemplu 2

În expresia $E = (\lambda x. \lambda z. (z x) (z y))$, evidențiem aparițiile:

$(\lambda_{\langle 1 \rangle} x_{\langle 1 \rangle} . \lambda_{\langle 1 \rangle} z_{\langle 1 \rangle} . (z_{\langle 2 \rangle} x_{\langle 2 \rangle}) (z_{\langle 3 \rangle} y_{\langle 1 \rangle}))$
 $\underbrace{\hspace{10em}}_F$

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$, $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în E
- $y_{\langle 1 \rangle}$, $z_{\langle 3 \rangle}$ **libere** în E
- $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în F
- $x_{\langle 2 \rangle}$ **liberă** în F
- x în E , în F
- y în E
- z în E , în F



Exemplu

Exemplu 2

În expresia $E = (\lambda x. \lambda z. (z x) (z y))$, evidențiem aparițiile:

$$(\lambda_{\langle 1 \rangle} x_{\langle 2 \rangle} . \lambda_{\langle 1 \rangle} z_{\langle 2 \rangle} . (z_{\langle 2 \rangle} x_{\langle 2 \rangle}) (z_{\langle 3 \rangle} y_{\langle 1 \rangle})).$$

$\underbrace{\hspace{10em}}_F$

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$, $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în E
- $y_{\langle 1 \rangle}$, $z_{\langle 3 \rangle}$ **libere** în E
- $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în F
- $x_{\langle 2 \rangle}$ **liberă** în F
- x **legată** în E , dar **liberă** în F
- y în E
- z în E , în F

E

Exemplu

Exemplu 2

În expresia $E = (\lambda x. \lambda z. (z x) (z y))$, evidențiem aparițiile:

$$(\lambda_{\langle 1 \rangle} x_{\langle 2 \rangle} . \lambda_{\langle 1 \rangle} z_{\langle 2 \rangle} . (z_{\langle 2 \rangle} x_{\langle 2 \rangle}) (z_{\langle 3 \rangle} y_{\langle 1 \rangle})).$$

F

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$, $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în E
- $y_{\langle 1 \rangle}$, $z_{\langle 3 \rangle}$ **libere** în E
- $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în F
- $x_{\langle 2 \rangle}$ **liberă** în F
- x **legată** în E , dar **liberă** în F
- y **liberă** în E
- z în E , în F



Exemplu

Exemplu 2

În expresia $E = (\lambda x. \lambda z. (z x) (z y))$, evidențiem aparițiile:

$$(\lambda_{\langle 1 \rangle} x_{\langle 1 \rangle} . \lambda_{\langle 1 \rangle} z_{\langle 1 \rangle} . (z_{\langle 2 \rangle} x_{\langle 2 \rangle}) (z_{\langle 3 \rangle} y_{\langle 1 \rangle})).$$

$\underbrace{\hspace{10em}}_F$

- $x_{\langle 1 \rangle}$, $x_{\langle 2 \rangle}$, $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în E
- $y_{\langle 1 \rangle}$, $z_{\langle 3 \rangle}$ **libere** în E
- $z_{\langle 1 \rangle}$, $z_{\langle 2 \rangle}$ **legate** în F
- $x_{\langle 2 \rangle}$ **liberă** în F
- x **legată** în E , dar **liberă** în F
- y **liberă** în E
- z **liberă** în E , dar **legată** în F



Exemplu

Variabile libere (*free variables*)

- $FV(x) = \{x\}$
- $FV(\lambda x.E) = FV(E) \setminus \{x\}$
- $FV((E_1 E_2)) = FV(E_1) \cup FV(E_2)$

Variabile legate (*bound variables*)

- $BV(x) = \emptyset$
- $BV(\lambda x.E) = BV(E) \cup \{x\}$
- $BV((E_1 E_2)) = BV(E_1) \setminus FV(E_2) \cup BV(E_2) \setminus FV(E_1)$

+ **O expresie închisă** este o expresie care **nu** conține variabile libere.

Ex) Exemplu

- $(\lambda x.x \ \lambda x.\lambda y.x) \rightarrow$ închisă
- $(\lambda x.x \ a) \rightarrow$ deschisă, deoarece a este liberă
- Variabilele **libere** dintr-o λ -expresie pot sta pentru alte λ -expresii – $\lambda x.((+ x) 1)$.
- Înaintea evaluării, o expresie trebuie adusă la forma **închisă**.
- Procesul de înlocuire trebuie să se **termine**.

+ | **β -reducere:** Evaluarea expresiei $(\lambda x.E A)$, cu E și A λ -expresii, prin **substituirea textuală** a tuturor aparițiilor **libere** ale parametrului **formal** al funcției, x , din corpul acesteia, E , cu parametrul **actual**, A :

$$(\lambda x.E A) \rightarrow_{\beta} E_{[A/x]}$$

+ | **β -redex** Expresia $(\lambda x.E A)$, cu E și A λ -expresii – o expresie pe care se poate aplica β -reducerea.

Ex

Exemplu

- $(\lambda x.x y) \rightarrow_{\beta} x_{[y/x]} \rightarrow y$
- $(\lambda x.\lambda x.x y)$
- $(\lambda x.\lambda y.x y)$

Ex

Exemplu

- $(\lambda x.x\ y) \rightarrow_{\beta} x_{[y/x]} \rightarrow y$
- $(\lambda x.\lambda x.x\ y) \rightarrow_{\beta} \lambda x.x_{[y/x]} \rightarrow \lambda x.x$
- $(\lambda x.\lambda y.x\ y)$

Ex

Exemplu

- $(\lambda x.x\ y) \rightarrow_{\beta} x_{[y/x]} \rightarrow y$
- $(\lambda x.\lambda x.x\ y) \rightarrow_{\beta} \lambda x.x_{[y/x]} \rightarrow \lambda x.x$
- $(\lambda x.\lambda y.x\ y) \rightarrow_{\beta} \lambda y.x_{[y/x]} \rightarrow \lambda y.y$

Ex

Exemplu

- $(\lambda x.x y) \rightarrow_{\beta} x_{[y/x]} \rightarrow y$
- $(\lambda x.\lambda x.x y) \rightarrow_{\beta} \lambda x.x_{[y/x]} \rightarrow \lambda x.x$
- $(\lambda x.\lambda y.x y) \rightarrow_{\beta} \lambda y.x_{[y/x]} \rightarrow \lambda y.y$ **Greșit!** Variabila liberă y devine **legată**, schimbându-și semnificația.
 $\rightarrow \lambda y^{(a)}.y^{(b)}$

Ex

Exemplu

- $(\lambda x.x y) \rightarrow_{\beta} x_{[y/x]} \rightarrow y$
- $(\lambda x.\lambda x.x y) \rightarrow_{\beta} \lambda x.x_{[y/x]} \rightarrow \lambda x.x$
- $(\lambda x.\lambda y.x y) \rightarrow_{\beta} \lambda y.x_{[y/x]} \rightarrow \lambda y.y$ **Greșit!** Variabila liberă y devine **legată**, schimbându-și semnificația.
 $\rightarrow \lambda y^{(a)}.y^{(b)}$

Care este problema?

- **Problemă:** în expresia $(\lambda x.E A)$:
 - variabilele libere din A nu au nume comune cu variabilele legate din E : $FV(A) \cap BV(E) = \emptyset$
→ reducere întotdeauna **corectă**
 - există variabilele libere din A care au nume comune cu variabilele legate din E : $FV(A) \cap BV(E) \neq \emptyset$
→ reducere **potențial greșită**
- **Soluție:** redenumirea variabilelor legate din E , ce coincid cu cele libere din A → **α -conversie**.

- **Problemă:** în expresia $(\lambda x.E A)$:
 - variabilele libere din A nu au nume comune cu variabilele legate din E : $FV(A) \cap BV(E) = \emptyset$
→ reducere întotdeauna **corectă**
 - există variabilele libere din A care au nume comune cu variabilele legate din E : $FV(A) \cap BV(E) \neq \emptyset$
→ reducere **potențial greșită**
- **Soluție:** redenumirea variabilelor legate din E , ce coincid cu cele libere din A → **α -conversie**.

Ex | Exemplu

$$(\lambda x.\lambda y.x y) \rightarrow_{\alpha} (\lambda x.\lambda z.x y) \rightarrow_{\beta} \lambda z.x_{[y/x]} \rightarrow \lambda z.y$$

+ **α -conversie:** Redenumirea sistematică a variabilelor **legate** dintr-o funcție: $\lambda x.E \rightarrow_{\alpha} \lambda y.E_{[y/x]}$. Se impun două condiții.



Exemplu

- $\lambda x.y \rightarrow_{\alpha} \lambda y.y_{[y/x]} \rightarrow \lambda y.y$
- $\lambda x.\lambda y.x \rightarrow_{\alpha} \lambda y.\lambda y.x_{[y/x]} \rightarrow \lambda y.\lambda y.y$

+ **α -conversie:** Redenumirea sistematică a variabilelor **legate** dintr-o funcție: $\lambda x.E \rightarrow_{\alpha} \lambda y.E_{[y/x]}$. Se impun două condiții.



Exemplu

- $\lambda x.y \rightarrow_{\alpha} \lambda y.y_{[y/x]} \rightarrow \lambda y.y \rightarrow$ **Greșit!**
- $\lambda x.\lambda y.x \rightarrow_{\alpha} \lambda y.\lambda y.x_{[y/x]} \rightarrow \lambda y.\lambda y.y$

+ **α -conversie:** Redenumirea sistematică a variabilelor **legate** dintr-o funcție: $\lambda x.E \rightarrow_{\alpha} \lambda y.E_{[y/x]}$. Se impun două condiții.



Exemplu

- $\lambda x.y \rightarrow_{\alpha} \lambda y.y_{[y/x]} \rightarrow \lambda y.y \rightarrow$ **Greșit!**
- $\lambda x.\lambda y.x \rightarrow_{\alpha} \lambda y.\lambda y.x_{[y/x]} \rightarrow \lambda y.\lambda y.y \rightarrow$ **Greșit!**

+ **α -conversie:** Redenumirea sistematică a variabilelor **legate** dintr-o funcție: $\lambda x.E \rightarrow_{\alpha} \lambda y.E_{[y/x]}$. Se impun două condiții.



Exemplu

- $\lambda x.y \rightarrow_{\alpha} \lambda y.y_{[y/x]} \rightarrow \lambda y.y \rightarrow$ **Greșit!**
- $\lambda x.\lambda y.x \rightarrow_{\alpha} \lambda y.\lambda y.x_{[y/x]} \rightarrow \lambda y.\lambda y.y \rightarrow$ **Greșit!**

∴ Condiții

- y **nu** este o variabilă liberă, existentă deja în E
- orice apariție liberă în E **rămâne** liberă în $E_{[y/x]}$

Exemple



Exemplu

- $\lambda x.(x y) \rightarrow_{\alpha} \lambda z.(z y) \rightarrow$ Corect!
- $\lambda x.\lambda x.(x y) \rightarrow_{\alpha} \lambda y.\lambda x.(x y)$
- $\lambda x.\lambda y.(y x) \rightarrow_{\alpha} \lambda y.\lambda y.(y y)$
- $\lambda x.\lambda y.(y y) \rightarrow_{\alpha} \lambda y.\lambda y.(y y)$

- $\lambda x.(x y) \rightarrow_{\alpha} \lambda z.(z y) \rightarrow$ Corect!
- $\lambda x.\lambda x.(x y) \rightarrow_{\alpha} \lambda y.\lambda x.(x y) \rightarrow$ Greșit! y este liberă în $\lambda x.(x y)$
- $\lambda x.\lambda y.(y x) \rightarrow_{\alpha} \lambda y.\lambda y.(y y)$
- $\lambda x.\lambda y.(y y) \rightarrow_{\alpha} \lambda y.\lambda y.(y y)$

- $\lambda x.(x y) \rightarrow_{\alpha} \lambda z.(z y) \rightarrow$ Corect!
- $\lambda x.\lambda x.(x y) \rightarrow_{\alpha} \lambda y.\lambda x.(x y) \rightarrow$ Greșit! y este liberă în $\lambda x.(x y)$
- $\lambda x.\lambda y.(y x) \rightarrow_{\alpha} \lambda y.\lambda y.(y y) \rightarrow$ Greșit! Apariția liberă a lui x din $\lambda y.(y x)$ devine legată, după substituire, în $\lambda y.(y y)$
- $\lambda x.\lambda y.(y y) \rightarrow_{\alpha} \lambda y.\lambda y.(y y)$

- $\lambda x.(x y) \rightarrow_{\alpha} \lambda z.(z y) \rightarrow$ Corect!
- $\lambda x.\lambda x.(x y) \rightarrow_{\alpha} \lambda y.\lambda x.(x y) \rightarrow$ Greșit! y este liberă în $\lambda x.(x y)$
- $\lambda x.\lambda y.(y x) \rightarrow_{\alpha} \lambda y.\lambda y.(y y) \rightarrow$ Greșit! Apariția liberă a lui x din $\lambda y.(y x)$ devine legată, după substituire, în $\lambda y.(y y)$
- $\lambda x.\lambda y.(y y) \rightarrow_{\alpha} \lambda y.\lambda y.(y y) \rightarrow$ Corect!

+ | **Pas de reducere:** O secvență formată dintr-o α -conversie și o β -reducere, astfel încât a doua se produce **fără coliziuni**:

$$E_1 \rightarrow E_2 \equiv E_1 \rightarrow_{\alpha} E_3 \rightarrow_{\beta} E_2.$$

+ | **Secvență de reducere:** Succesiune de zero sau mai mulți pași de reducere:

$$E_1 \rightarrow^* E_2.$$

Reprezintă un element din închiderea reflexiv-tranzitivă a relației \rightarrow .

⋮ Reducere

- $E_1 \rightarrow E_2 \implies E_1 \rightarrow^* E_2$ – un pas este o secvență
- $E \rightarrow^* E$ – zero pași formează o secvență
- $E_1 \rightarrow^* E_2 \wedge E_2 \rightarrow^* E_3 \implies E_1 \rightarrow^* E_3$ – tranzitivitate



Exemplu

$$\begin{aligned} & ((\lambda x. \lambda y. ((y x) y) \lambda x. x) \rightarrow (\lambda z. (z y) \lambda x. x) \rightarrow (\lambda x. x y) \rightarrow y \\ & \implies \\ & ((\lambda x. \lambda y. ((y x) y) \lambda x. x) \rightarrow^* y \end{aligned}$$

Evaluare

· Dacă am vrea să construim o mașină de calcul care să aibă ca program o λ -expresie și să aibă ca operație de bază pasul de reducere, ne punem câteva întrebări:

- 1 Când se **termină** calculul? Se termină **întotdeauna**?
- 2 Comportamentul **depinde** de secvența de reducere?
- 3 Dacă mai multe secvențe de reducere se termină, obținem întotdeauna **același** rezultat?
- 4 Dacă rezultatul este unic, **cum** îl obținem?

Forme normale

Cum știm că s-a terminat calculul?

· Calculul **se termină** atunci când expresia nu mai poate fi redusă \rightarrow expresia nu mai conține β -redecși.

+ **Forma normală** a unei expresii este o formă (la care se ajunge prin **reducere**, care **nu** mai conține β -redecși i.e. care **nu** mai poate fi redusă.

+ **Forma normală funcțională – FNF** este o formă $\lambda x.F$, în care F poate **conține** β -redecși.

Ex | Exemplu

$(\lambda x.\lambda y.(x y) \lambda x.x) \rightarrow_{FNF} \lambda y.(\lambda x.x y) \rightarrow_{FN} \lambda y.y$



Exemplu

$\Omega = (\lambda x.(x x) \lambda x.(x x)) \rightarrow (\lambda x.(x x) \lambda x.(x x)) \rightarrow^* \dots$

Ω **nu** admite nicio secvență de reducere care se termină.

+ **Expresie reductibilă** este o expresie care admite (cel puțin o) secvență de reducere care se termină.

· expresia Ω **nu** este reductibilă.

Secvențe de reducere și terminare

 λ

Dar!

$$E = (\lambda x.y \ \Omega)$$

$$\rightarrow y \quad \text{sau}$$

$$\rightarrow E \rightarrow y \quad \text{sau}$$

$$\rightarrow E \rightarrow E \rightarrow y \quad \text{sau...}$$

 \vdots

$$\xrightarrow{n^*} y, n \geq 0$$

$$\xrightarrow{\infty^*} \dots$$

Exemplu

Dar!

$$E = (\lambda x.y \ \Omega)$$

$$\rightarrow y \quad \text{sau}$$

$$\rightarrow E \rightarrow y \quad \text{sau}$$

$$\rightarrow E \rightarrow E \rightarrow y \quad \text{sau...}$$

$$\vdots$$
$$\xrightarrow{n^*} y, n \geq 0$$

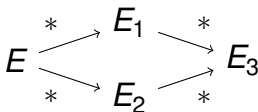
$$\xrightarrow{\infty^*} \dots$$

Exemplu

Exemplu

- E are o secvență de reducere care **nu** se termină;
- dar E are **forma normală** $y \Rightarrow E$ este reductibilă;
- lungimea secvențelor de reducere ale E este **nemărginită**.

T | **Teorema Church-Rosser / diamantului** Dacă $E \rightarrow^* E_1$ și $E \rightarrow^* E_2$, atunci **există** E_3 astfel încât $E_1 \rightarrow^* E_3$ și $E_2 \rightarrow^* E_3$.



C | **Corolar** Dacă o expresie este reductibilă, forma ei normală este **unică**. Ea corespunde **valorii** expresiei.

Unicitatea formei normale

 λ

Exemplu



Exemplu

$(\lambda x. \lambda y. (x\ y)) (\lambda x. x\ y)$



$(\lambda x. \lambda y. (x y) (\lambda x. x y))$

- $\rightarrow \lambda z. ((\lambda x. x y) z) \rightarrow \lambda z. (y z) \rightarrow_{\alpha} \lambda a. (y a)$
- $\rightarrow (\lambda x. \lambda y. (x y) y) \rightarrow \lambda w. (y w) \rightarrow_{\alpha} \lambda a. (y a)$



$(\lambda x. \lambda y. (x y) (\lambda x. x y))$

- $\rightarrow \lambda z. ((\lambda x. x y) z) \rightarrow \lambda z. (y z) \rightarrow_{\alpha} \lambda a. (y a)$
- $\rightarrow (\lambda x. \lambda y. (x y) y) \rightarrow \lambda w. (y w) \rightarrow_{\alpha} \lambda a. (y a)$

- Forma normală corespunde unei **clase** de expresii, echivalente sub **redenumiri** sistematice.
 - **Valoarea** este un anumit membru al acestei clase de echivalență.
- \Rightarrow Valorile sunt **echivalente** în raport cu **redenumirea**.

Modalități de reducere

 λ

Cum putem *organiza* reducerea?

+ **Reducere stânga-dreapta:** Reducerea celui mai superficial și mai din stânga β -redex.

Ex | Exemplu

$((\lambda x.x \ \lambda x.y) \ (\lambda x.(x \ x) \ \lambda x.(x \ x))) \rightarrow (\lambda x.y \ \Omega) \rightarrow y$

+ **Reducere dreapta-stânga:** Reducerea celui mai adânc și mai din dreapta β -redex.

Ex | Exemplu

$(\lambda x.(\lambda x.x \ \lambda x.y) \ (\lambda x.(x \ x) \ \lambda x.(x \ x))) \rightarrow$
 $(\lambda x.(\lambda x.x \ \lambda x.y) \ \underline{\Omega}) \rightarrow \dots$

T | **Teorema normalizării** Dacă o expresie este reductibilă, evaluarea **stânga-dreapta** a acesteia se termină.

- Teorema normalizării (normalizare = aducere la forma normală) **nu** garantează terminarea evaluării oricărei expresii, ci doar a celor **reductibile!**
- Dacă expresia este ireductibilă, **nicio** reducere nu se va termina.

- 1 Când se **termină** calculul? Se termină **întotdeauna**?
→ se termină cu **forma normală [funcțională]**. **NU** se termină decât dacă expresia este **reductibilă**.
- 2 Comportamentul **depinde** de secvența de reducere?
→ **DA**.
- 3 Dacă mai multe secvențe de reducere se termină, obținem întotdeauna **același** rezultat?
→ **DA**.
- 4 Dacă rezultatul este unic, **cum** îl obținem?
→ Reducere **stânga-dreapta**.
- 5 Care este valoarea expresiei?
→ Forma normală [funcțională] (**FNF**).

- **+** **Evaluare aplicativă** (*eager*) – corespunde reducerii **dreapta-stânga**. Parametrii funcțiilor sunt evaluați **înaintea** aplicării funcției.
- **+** **Evaluare normală** (*lazy*) – corespunde reducerii **stânga-dreapta**. Parametrii funcțiilor sunt evaluați **la cerere**.
- **+** **Funcție strictă** – funcție cu evaluare **aplicativă**.
- **+** **Funcție nestrictă** – funcție cu evaluare **normală**.

- Evaluarea **aplicativă** prezentă în majoritatea limbajelor: C, Java, Scheme, PHP etc.

Ex | Exemplu

$(+ (+ 2 3) (* 2 3)) \rightarrow (+ 5 6) \rightarrow 11$

- Nevoie de funcții **nestricte**, chiar în limbajele aplicative: if, and, or etc.

Ex | Exemplu

$(\text{if } (< 2 3) (+ 2 3) (* 2 3)) \rightarrow (< 2 3) \rightarrow \#t \rightarrow (+ 2 3) \rightarrow 5$

Limbajul lambda-0 și incursiune în TDA

- Am putea crea o mașină de calcul folosind calculul λ – mașină de calcul **ipotetică**;
- Mașina folosește limbajul $\lambda_0 \equiv$ calcul lambda;
- **Programul** \rightarrow λ -expresie;
 - + Legări top-level de expresii la nume.
- **Datele** \rightarrow λ -expresii;
- Funcționarea mașinii \rightarrow **reducere** – substituție textuală
 - evaluare normală;
 - terminarea evaluării cu forma normală funcțională;
 - se folosesc numai expresii închise.

· Scrieri **prescurtate**:

- $\lambda x_1.\lambda x_2.\dots.\lambda x_n.E \rightarrow \lambda x_1 x_2 \dots x_n.E$
- $((\dots((E A_1) A_2) \dots) A_n) \rightarrow (E A_1 A_2 \dots A_n)$

Ex | Exemplu

$\lambda x.\lambda y.(x y) \rightarrow \lambda x y.(x y)$

Tipuri de date

Cum reprezentăm datele? Cum interpretăm valorile?

- Putem reprezenta toate datele prin funcții cărora, **convențional**, le dăm o semnificație **abstractă**.

Ex) Exemplu

$$T \equiv_{\text{def}} \lambda x. \lambda y. x \equiv \lambda x y. x \qquad F \equiv_{\text{def}} \lambda x. \lambda y. y \equiv \lambda x y. y$$

- Pentru aceste **tipuri de date abstracte (TDA)** creăm operatori care transformă datele în mod coerent cu interpretarea pe care o dăm valorilor.

Ex) Exemplu

$$\text{not} \equiv_{\text{def}} \lambda x. (x F T)$$

$$(\text{not } T) \rightarrow (\lambda x. (x F T) T) \rightarrow (T F T) \rightarrow F$$

+ **Tip de date abstract – TDA** – Model matematic al unei **mulțimi** de valori și al **operațiilor** valide pe acestea.

∴ Componente

- **constructori de bază**: cum se generează valorile;
- **operatori**: ce se poate face cu acestea;
- **axiome**: cum lucrează operatorii / ce restricții există.

· Constructori: $\left| \begin{array}{l} T : \rightarrow Bool \\ F : \rightarrow Bool \end{array} \right.$

· Operatori: $\left| \begin{array}{l} not : Bool \rightarrow Bool \\ and : Bool^2 \rightarrow Bool \\ or : Bool^2 \rightarrow Bool \end{array} \right.$

· Axiome: $\left| \begin{array}{l} not : not(T) = F \\ \quad \quad not(F) = T \\ and : and(T, a) = a \\ \quad \quad and(F, a) = F \\ or : or(T, a) = T \\ \quad \quad or(F, a) = a \end{array} \right.$



Intuiție: **selecția** între cele două valori, *true* și *false*

- $T \equiv_{\text{def}} \lambda x y. x$
- $F \equiv_{\text{def}} \lambda x y. y$
- Comportament de **selector**:
 - $(T a b) \rightarrow (\lambda x y. x a b) \rightarrow a$
 - $(F a b) \rightarrow (\lambda x y. y a b) \rightarrow b$

- $not \equiv_{\text{def}} \lambda x.(x F T)$
 - $(not T) \rightarrow (\lambda x.(x F T) T) \rightarrow (T F T) \rightarrow F$
 - $(not F) \rightarrow (\lambda x.(x F T) F) \rightarrow (F F T) \rightarrow T$

- $and \equiv_{\text{def}} \lambda x y.(x y F)$
 - $(and T a) \rightarrow (\lambda x y.(x y F) T a) \rightarrow (T a F) \rightarrow a$
 - $(and F a) \rightarrow (\lambda x y.(x y F) F a) \rightarrow (F a F) \rightarrow F$

- $or \equiv_{\text{def}} \lambda x y.(x T y)$
 - $(or T a) \rightarrow (\lambda x y.(x T y) T a) \rightarrow (T T a) \rightarrow T$
 - $(or F a) \rightarrow (\lambda x y.(x T y) F a) \rightarrow (F T a) \rightarrow a$

- Intuiție: pereche \rightarrow funcție ce așteaptă **selectorul**, pentru a-l aplica asupra membrilor
- $pair \equiv_{\text{def}} \lambda x y z.(z x y)$
 - $(pair a b) \rightarrow (\lambda x y z.(z x y) a b) \rightarrow \lambda z.(z a b)$
- $fst \equiv_{\text{def}} \lambda p.(p T)$
 - $(fst (pair a b)) \rightarrow (\lambda p.(p T) \lambda z.(z a b)) \rightarrow (\lambda z.(z a b) T) \rightarrow (T a b) \rightarrow a$
- $snd \equiv_{\text{def}} \lambda p.(p F)$
 - $(snd (pair a b)) \rightarrow (\lambda p.(p F) \lambda z.(z a b)) \rightarrow (\lambda z.(z a b) F) \rightarrow (F a b) \rightarrow b$



Intuiție: listă \rightarrow pereche (*head*, *tail*)

- $nil \equiv_{\text{def}} \lambda x. T$
- $cons \equiv_{\text{def}} pair$
 - $(cons\ e\ L) \rightarrow (\lambda x\ y\ z. (z\ x\ y)\ e\ L) \rightarrow \lambda z. (z\ e\ L)$
- $car \equiv_{\text{def}} fst$ $cdr \equiv_{\text{def}} snd$



Intuiție: număr \rightarrow listă cu lungimea egală cu valoarea numărului

- $zero \equiv_{\text{def}} nil$
- $succ \equiv_{\text{def}} \lambda n. (cons\ nil\ n)$
- $pred \equiv_{\text{def}} cdr$

• vezi și [http://en.wikipedia.org/wiki/Lambda_calculus#Encoding_datatypes]

- Modalitate de exprimare a **intenției** programatorului;
- **Documentare**: ce operatori acționează asupra căror obiecte;
- Reprezentarea **particulară** a valorilor de tipuri diferite: 1, “Hello”, #t etc.;
- **Optimizarea** operațiilor specifice;
- Prevenirea **erorilor**;
- Facilitarea verificării **formale**;

- Un număr, o listă sau un arbore, posibil desemnate de **aceeași** valoare!
- Valori și operatori reprezentați de funcții, semnificația fiind dependentă de **context**.
- Valoare **aplicabilă** asupra unei alte valori \rightarrow operator!

- Incapacitatea Mașinii λ de a
 - interpreta **semnificația** expresiilor;
 - asigura **corectitudinea** acestora (dpdv al tipurilor).
- Delegarea celor două aspecte **programatorului**;
- **Orice** operatori aplicabili asupra **oricăror** valori;
- Construcții eronate **acceptate** fără avertisment, dar calcule terminate cu
 - valori **fără** semnificație *sau*
 - expresii care **nu** sunt valori (nu au asociată o semnificație), dar sunt **ireductibile**

→ **instabilitate**.

- **Flexibilitate** sporită în reprezentare;
- Potrivită în situațiile în care reprezentarea **uniformă** obiectelor, ca liste de simboluri, este convenabilă.

... vin cu prețul unei dificultăți sporite în **depanare**, **verificare** și **mentenanță**

· Cum realizăm recursivitatea în λ_0 , dacă nu avem nume de funcții?

- **Textuală**: funcție care se autoapelează, folosindu-și numele;
- **Semantică**: ce **obiect** matematic este desemnat de o funcție recursivă, cu posibilitatea construirii de funcții recursive **anonime**.

- Lungimea unei liste:

length $\equiv_{\text{def}} \lambda L.(\text{if } (\text{null } L) \text{ zero } (\text{succ } (\underline{\text{length}} (\text{cdr } L))))$

- Cu ce **înlocuim** zona subliniată, pentru a evita recursivitatea textuală?

- Putem primi ca **parametru** o funcție echivalentă computațional cu *length*?

Length $\equiv_{\text{def}} \lambda f L.(\text{if } (\text{null } L) \text{ zero } (\text{succ } (f (\text{cdr } L))))$

- $(\text{Length } \text{length}) = \text{length} \rightarrow \text{length}$ este un **punct fix** al lui *Length*!

- Cum **obținem** punctul fix?

Combinator de punct fix

mai multe la

http://en.wikipedia.org/wiki/Lambda_calculus#Recursion_and_fixed_points



Exemplu

$Fix = \lambda f.(\lambda x.(f (x x)) \lambda x.(f (x x)))$

- $(Fix F) \rightarrow (\lambda x.(F (x x)) \lambda x.(F (x x))) \rightarrow (F (\lambda x.(F (x x)) \lambda x.(F (x x)))) \rightarrow (F (Fix F))$
- $(Fix F)$ este un **punct fix** al lui F .
- Fix se numește **combinator de punct fix**.
- $length \equiv_{\text{def}} (Fix Length) \rightarrow (Length (Fix Length)) \rightarrow \lambda L.(if (null L) zero (succ ((Fix Length) (cdr L))))$
- Funcție recursivă, **fără** a fi textual recursivă!

- Baza formală a calculului λ
- expresie λ , β -redex, variabile și apariții legate vs. libere, expresie închisă, α -conversie, β -reducere
- FN și FNF, reducere, reductibilitate, evaluare aplicativă și normală
- TDA și recursivitate pentru calcul lambda

+ Dați feedback la acest curs aici:

[<http://goo.gl/forms/SjDsW06v5J>]

Introducere

λ -Expresii

Reducere

Evaluare

λ_0 și TDA