Paradigme de Programare

S.I. dr. ing. Andrei Olaru
slides: Mihnea Muraru si Andrei Olaru

Catedra de Calculatoare

2013 – 2013, semestrul 2

Cursul 2

Calcul Lambda

Cuprins

Introducere

Calculul Lambda

λ

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

  `http://en.wikipedia.org/wiki/Lambda_calculus`

- Echivalent cu Maşina Turing (v. Teza Church-Turing)
  - Axat pe conceptul matematic de funcţie – totul este funcţie
  - Calculul = evaluarea aplicaţiilor de funcţii, prin substituţie textuală
  - Evaluare = obţinerea unei valori $\rightarrow$ funcţie!
  - Absenţa efectelor laterale şi a stării
  - Model formal

Aplicaţii

ale calculului λ

- Aplicaţii importante în
  - programare
  - demonstrarea formală a corectitudinii programelor,

  datorită modelului simplu de execuţie

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

λ-Expresii

λ-expresii

Definiţie

Definiția 7.1 (λ-expresie).

- Variabilă: o variabilă x este o λ-expresie
- Funcţie: dacă x este o variabilă şi E este o λ-expresie, atunci $\lambda x.E$ este o λ-expresie, reprezentând funcţia anonimă, unară, cu parametrul formal x şi corpul E
- Aplicaţie: dacă F şi A sunt λ-expresii, atunci $(F\ A)$ este o λ-expresie, reprezentând aplicaţia expresiei F asupra parametrului actual A

λ-expresii

Exemple

Exemplul 7.2.

1. $x \rightarrow$ variabila x
2. $\lambda x.x \rightarrow$ funcţia identitate
3. $\lambda x.\lambda y.x \rightarrow$ funcţie selector, cu altă funcţie drept corp!
4. $(\lambda x.x\ y) \rightarrow$ aplicaţia funcţiei identitate asupra parametrului actual y
5. $(\lambda x.(x\ x)\ \lambda x.x)$

- Intuitiv, evaluarea aplicaţiei $(\lambda x.x\ y)$ presupune substituirea lui x, în corp, prin $y \rightarrow$ rezultat y

Apariţii ale variabilelor

Definiţii

Definiția 7.3 (Apariţie legată).

O apariţie x_n a unei variabile x este legată într-o expresie E dacă:

- $E = \lambda x.F$ sau
- $E = \ldots \lambda x_n.F \ldots$ sau
- $E = \ldots \lambda x.F \ldots$ şi x_n apare în F.

Definiția 7.4 (Apariţie liberă).

O apariţie a unei variabile este liberă într-o expresie dacă nu este legată în acea expresie.

- Atenţie! În raport cu o expresie dată!

Variabile

Definiţii

Definiția 7.5 (Variabilă legată).

O variabilă este legată într-o expresie dacă toate apariţiile sale sunt legate în acea expresie.

Definiția 7.6 (Variabilă liberă).

O variabilă este liberă într-o expresie dacă nu este legată în acea expresie i.e. dacă cel puţin o apariţie a sa este liberă în acea expresie.

Definiția 7.7 (Variabilă de legare).

Parametrul formal, x, al funcţiei $\lambda x.E$.

- Atenţie! În raport cu o expresie dată!

[Apariţii ale]variabilelor

Exemplu 1

Exemplul 7.8.

În expresia $E = (\lambda x.x\ x)$, evidenţiem apariţiile lui x:
$(\lambda x_1.\ \underbrace{x_2}_{F}\ x_3)$.

- x_1, x_2 în E
- x_3 în E
- x_2 în F!
- x în E şi F

[Apariţii ale]variabilelor

Exemplu 1

Exemplul 7.8.

În expresia $E = (\lambda x.x\ x)$, evidenţiem apariţiile lui x:
$(\lambda x_1.\ \underbrace{x_2}_{F}\ x_3)$.

- x_1, x_2 legate în E
- x_3 liberă în E
- x_2 liberă în F!
- x liberă în E şi F

[Apariţii ale]variabilelor

Exemplu 2

Exemplul 7.9.

În expresia $E = (\lambda x.\lambda z.(z\ x)\ (z\ y))$, evidenţiem apariţiile: $(\lambda x_1.\underbrace{\lambda z_1.(z_2\ x_2)}_{F}\ (z_3\ y_1))$.

- x_1, x_2, z_1, z_2 în E
- y_1, z_3 în E
- z_1, z_2 în F
- x_2 în F
- x în E, în F
- y în E
- z în E, în F

[Apariţii ale]variabilelor

Exemplu 2

Exemplul 7.9.

În expresia $E = (\lambda x.\lambda z.(z\ x)\ (z\ y))$, evidenţiem apariţiile: $(\lambda x_1.\underbrace{\lambda z_1.(z_2\ x_2)}_{F}\ (z_3\ y_1))$.

- x_1, x_2, z_1, z_2 legate în E
- y_1, z_3 libere în E
- z_1, z_2 legate în F
- x_2 liberă în F
- x legată în E, dar liberă în F
- y liberă în E
- z liberă în E, dar legată în F

Determinarea variabilelor libere şi legate

O abordare formală

Variabile libere (*free variables*)

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

Variabile legate (*bound variables*)

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

Expresii închise

Definiția 7.10 (Expresie închisă).

Expresie ce nu conţine variabile libere.

Exemplul 7.11.

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

Reducere

β-reducere

Definiţii

Definiția 8.1 (β-reducere).

Evaluarea expresiei $(\lambda x.E\ A)$, prin substituirea tuturor apariţiilor libere ale parametrului **formal** al funcţiei, x, din corpul acesteia, E, cu parametrul **actual**, A:
$(\lambda x.E\ A) \rightarrow_\beta E_{[A/x]}$.

Definiția 8.2 (β-redex).

Expresia $(\lambda x.E\ A)$ – o expresie pe care se poate aplica β-reducerea.

β-reducere

Exemple

Exemplul 8.3.

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

β-reducere

Exemple

Exemplul 8.3.

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

β-reducere

Exemple

Exemplul 8.3.

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

β-reducere

Exemple

Exemplul 8.3.

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

β-reducere

Exemple

Exemplul 8.3.

- $(\lambda x.x\ y) \rightarrow_\beta x_{[y/x]} \rightarrow y$
- $(\lambda x.\lambda x.x\ y) \rightarrow_\beta \lambda x.x_{[y/x]} \rightarrow \lambda x.x$
- $(\lambda x.\lambda y.x\ y) \rightarrow_\beta \lambda y.x_{[y/x]} \rightarrow \lambda y.y$ Greşit! Variabila liberă y devine legată, schimbându-şi semnificaţia. $\rightarrow \lambda y^{(a)}.y^{(b)}$ Care este problema? (formal)

β-reducere

Coliziuni

- Problemă: în expresia $(\lambda x.E\ A)$:
  - $FV(A) \cap BV(E) = \emptyset \rightarrow$ reducere întotdeauna corectă
  - $FV(A) \cap BV(E) \neq \emptyset \rightarrow$ reducere potenţial greşită
- Soluţie: redenumirea variabilelor legate din E, ce coincid cu cele libere din A.

Exemplul 8.4.

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

β-reducere

Coliziuni

- Problemă: în expresia $(\lambda x.E\ A)$:
  - $FV(A) \cap BV(E) = \emptyset \rightarrow$ reducere întotdeauna corectă
  - $FV(A) \cap BV(E) \neq \emptyset \rightarrow$ reducere potenţial greşită
- Soluţie: redenumirea variabilelor legate din E, ce coincid cu cele libere din A.

Exemplul 8.4.

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

α-conversie

Definiţie

Definiția 8.5 (α-conversie).

Redenumirea sistematică a variabilelor legate dintr-o funcţie: $\lambda x.E \rightarrow_\alpha \lambda y.E_{[y/x]}$. Se impun două condiţii.

Exemplul 8.6.

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

α-conversie

Definiţie

Definiția 8.5 (α-conversie).

Redenumirea sistematică a variabilelor legate dintr-o funcţie: $\lambda x.E \rightarrow_\alpha \lambda y.E_{[y/x]}$. Se impun două condiţii.

Exemplul 8.6.

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

α-conversie

Definiţie

Definiția 8.5 (α-conversie).

Redenumirea sistematică a variabilelor legate dintr-o funcţie: $\lambda x.E \rightarrow_\alpha \lambda y.E_{[y/x]}$. Se impun două condiţii.

Exemplul 8.6.

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

α-conversie

Definiţie

Definiția 8.5 (α-conversie).

Redenumirea sistematică a variabilelor legate dintr-o funcţie: $\lambda x.E \to_\alpha \lambda y.E_{[y/x]}$. Se impun două condiţii.

Exemplul 8.6.

- $\lambda x.y \to_\alpha \lambda y.y_{[y/x]} \to \lambda y.y \to$ Greşit!
- $\lambda x.\lambda y.x \to_\alpha \lambda y.\lambda y.x_{[y/x]} \to \lambda y.\lambda y.y \to$ Greşit!

Condiţii:

- y nu este liberă în E
- o apariţie liberă în E rămâne liberă în $E_{[y/x]}$

α-conversie

Exemple

Exemplul 8.7.

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

α-conversie

Exemple

Exemplul 8.7.

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

α-conversie

Exemple

Exemplul 8.7.

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

α-conversie

Exemple

Exemplul 8.7.

- $\lambda x.(x\ y) \rightarrow_\alpha \lambda z.(z\ y) \rightarrow$ Corect!
- $\lambda x.\lambda x.(x\ y) \rightarrow_\alpha \lambda y.\lambda x.(x\ y) \rightarrow$ Greşit! y este liberă în $\lambda x.(x\ y)$
- $\lambda x.\lambda y.(y\ x) \rightarrow_\alpha \lambda y.\lambda y.(y\ y) \rightarrow$ Greşit! Apariţia 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!

Definiția 8.8 (Pas de reducere).

O secvenţă formată dintr-o α-conversie şi o β-reducere, astfel încât a doua se produce fără coliziuni:
$E_1 \to E_2 \equiv E_1 \to_\alpha E_3 \to_\beta E_2$.

Definiția 8.9 (Secvenţă de reducere).

Succesiune de zero sau mai mulţi paşi de reducere:
$E_1 \to^* E_2$. Reprezintă un element din închiderea reflexiv-tranzitivă a relaţiei $\to$.

Reducere
Proprietăţi

- $E_1 \to E_2 \implies E_1 \to^* E_2$
- $E \to^* E$
- $E_1 \to^* E_2 \wedge E_2 \to^* E_3 \implies E_1 \to^* E_3$

Exemplul 8.10.

- $((\lambda x.\lambda y.(y\ x)\ y)\ \lambda x.x) \to (\lambda z.(z\ y)\ \lambda x.x) \to (\lambda x.x\ y) \to y$
- $((\lambda x.\lambda y.(y\ x)\ y)\ \lambda x.x) \to^* y$

Forme normale

Întrebări

1. Când se termină calculul? Se termină întotdeauna?
   → NU
2. Comportamentul depinde de secvenţa de reducere?
   → DA
3. Dacă se termină, obţinem întotdeauna acelaşi rezultat?
   → DA
4. Dacă rezultatul este unic, cum îl obţinem?
   → Reducere stânga-dreapta

Forme normale

Definiții

Definiția 9.1 (Formă normală).

Formă a unei expresii, ce nu mai conţine β-redecşi i.e. care nu mai poate fi redusă.

Definiția 9.2 (Formă normală funcţională – FNF).

$\lambda x.F$, chiar dacă F conţine β-redecşi.

- FNF utilizată în programare, corpul unei funcţii fiind evaluat de-abia în momentul aplicării

Exemplul 9.3.

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

Terminarea reducerii (reductibilitate)

Exemplu și definiție

Exemplul 9.4.

$\Omega = (\lambda x.(x\ x)\ \lambda x.(x\ x)) \to (\lambda x.(x\ x)\ \lambda x.(x\ x)) \to^* \ldots$

Ω nu admite o secvenţă de reducere, care se termină.

Definiția 9.5 (Expresie reductibilă).

Expresie ce admite o secvenţă de reducere, care se termină.

Secvenţe de reducere

Exemplul 9.6.

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

$\rightarrow y$

$\rightarrow E \rightarrow y$

$\rightarrow E \rightarrow E \rightarrow y$

...

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

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

Secvenţe de reducere

Exemplul 9.6.

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

$\rightarrow y$

$\rightarrow E \rightarrow y$

$\rightarrow E \rightarrow E \rightarrow y$

...

$\overset{n}{\rightarrow}{}^{*}\ y,\ n \geq 0$

$\overset{\infty}{\rightarrow}{}^{*}\ \ldots$

- E are o secvenţă de reducere, care nu se termină, dar are forma normală y. E este reductibilă, Ω nu.
- Lungimea secvenţelor de reducere, care se termină, este nemărginită.

Unicitatea formei normale

Rezultate

Teorema 9.7 (Church-Rosser / diamantului).

Dacă $E \to^* E_1$ şi $E \to^* E_2$, atunci există E_3 astfel încât $E_1 \to^* E_3$ şi $E_2 \to^* E_3$.

$$\begin{array}{ccccc} & & \to^* E_1 & \to^* & \\ E & & & & E_3 \\ & & \to^* E_2 & \to^* & \end{array}$$

Corolarul 9.8.

Dacă o expresie este reductibilă, forma ei normală este unică. Ea corespunde valorii expresiei.

Unicitatea formei normale

Exemple

Exemplul 9.9.

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

- $\to \lambda z.((\lambda x.x\ y)\ z) \to \lambda z.(y\ z) \to_\alpha \lambda a.(y\ a)$
- $\to (\lambda x.\lambda y.(x\ y)\ y) \to \lambda w.(y\ w) \to_\alpha \lambda a.(y\ a)$

- Forma normală corespunde unei clase de expresii, echivalente sub redenumiri sistematice.
- Valoarea este un anumit membru al acestei clase.

Modalităţi de reducere

Definiția 9.10 (Reducere stânga-dreapta).

Reducerea celui mai superficial şi mai din stânga β-redex.

Exemplul 9.11.

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

Definiția 9.12 (Reducere dreapta-stânga).

Reducerea celui mai adânc şi mai din dreapta β-redex.

Exemplul 9.13.

$$((\lambda x.x\ \lambda x.y)\ \underline{(\lambda x.(x\ x)\ \lambda x.(x\ x))}) \to (\lambda x.y\ \underline{\Omega}) \to \ldots$$

Ce modalitate alegem?

Teorema 9.14 (Normalizării).

Dacă o expresie este reductibilă, evaluarea stânga-dreapta a acesteia se termină.

- Teorema normalizării nu garantează terminarea evaluării oricărei expresii, ci doar a celor reductibile!
- Dacă expresia este ireductibilă, nicio reducere nu se va termina.

Evaluare

Ordini de evaluare

Diferă în diferite limbaje

Definiția 10.1 (Evaluare aplicativă – *eager*).

Corespunde reducerii dreapta-stânga. Parametrii funcţiilor sunt evaluaţi înaintea aplicării funcţiei.

Definiția 10.2 (Funcţie strictă).

Funcţie cu evaluare aplicativă.

Definiția 10.3 (Evaluare normală – *lazy*).

Corespunde reducerii stânga-dreapta. Parametrii funcţiilor sunt evaluaţi la cerere.

Definiția 10.4 (Funcţie nestrictă).

Funcţie cu evaluare normală.

În practică

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

Exemplul 10.5.

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

- Nevoie de funcţii nestricte, chiar în limbajele aplicative: `if`, `and`, `or` etc.

Exemplul 10.6.

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

Transferul parametrilor

- Evaluare aplicativă
  - *Call by value*
  - *Call by sharing*
  - *Call by reference*
  - *Call by copying*

- Evaluare normală
  - *Call by name*
  - *Call by need*

Call by value

Exemplul 10.7.

```
1  // C sau Java
2  void f(int x) {
3      x = 3;
4  }
```

```
1  // C
2  void g(struct str s) {
3      s.member = 3;
4  }
```

Efectul liniilor 3 este invizibil la apelant.

- Evaluarea parametrilor înaintea aplicaţiei funcţiei şi transferul unei copii a valorii acestuia
- Modificări locale invizibile la apelant
- C, C++, tipurile primitive în Java

Call by sharing

- Variantă a *call by value*
- Trimiterea unei referinţe la obiect
- Modificări locale asupra referinţei invizibile la apelant
- Modificări locale asupra obiectului referit vizibile la apelant
- Scheme, tipurile referinţă în Java
- Diferenţă faţă de C, unde o structură trimisă ca parametru este complet copiată

Call by reference

- Trimiterea unei referinţe la obiect
- Modificări locale asupra referinţei şi obiectului referit vizibile la apelant
- Folosirea & în C++

Call by name

- Argumente neevaluate în momentul aplicării funcţiei → substituţie directă (textuală) în corpul funcţiei
- Evaluare parametrilor la cerere, de fiecare dată când este nevoie de valoarea acestora

Call by need

- Variantă a *call by name*
- Evaluarea unui parametru doar la prima utilizare a acestuia
- Memorarea valorii unui parametru deja evaluat şi returnarea acesteia în cazul utilizării repetate a aceluiaşi parametru
- în Haskell

Sfârșitul cursului 2

Ce am învățat

· Baza formală a calculului λ, expresie λ, legat vs. liber, expresie închisă, α-conversie, β-reducere, FN și FNF, reductibilitate, evaluare aplicativă și normală, call by $*$.