Fundamentele limbajelor de programare

Notițe pentru parțial

Aprilie, 2023

1 Paradigme de definire a funcţiilor

Criteriul/Paradigma	**Abordarea extensională**	**Abordarea intensională**
Definirea unei funcţii	$f : X \to Y$ este o mulțime de perechi f $\subseteq X \times Y$, a.î., $\forall x \in X, \exists y \in Y$ a.î.$(x, y) \in f$	O formula de calcul pentru funcție de ex., $f(x) = x^2 - 1$
Eglitatea funcţională	$f = g \iff f(x) = g(x), \forall x \in X$ Se compară doar rezultatele funcției	$f(x) = x^2 - 1, g(x) = (x - 1)(x + 1), f \neq g$ Se compară formulele de definiție
Puncte forte	Cuprinde și funcții care nu pot fi definite prin formule	Utilă d.p.d.v al implementării.

2 Introducere în $\lambda - calcul$

Exemple de reprezentare ale funcțiilor în $\lambda - calcul$:

Reprezentarea clasică	**λ calcul**
$f(x) = x^2$	$\lambda x.x^2$
$g(x) = (f \circ f)(x)$	$\lambda x.f(f(x))$
$h(f) = f \circ f$ (funcție de nivel înalt)	$\lambda f.\lambda x.f(f(x))$

3 Noțiuni preliminare despre tipuri

Definitia 3.1. *Notăm cu x:X, faptul că un termen x are tipul X.*

Exemplul 3.2. $f = \lambda x.x : X \to X$, $g = \lambda f.\lambda x.f(f(x)) : (X \to X) \to (X \to X)$.

Comparație între versiunile de $\lambda - calcul$, in funcție de prezența tipurilor:

Fără tipuri	**Tipuri simple**	**Tipuri polimorfice**
Nu dăm tipul expresiilor	Dăm tipul expresiilor	Putem specifica tipul $X \to X$, fără a-l preciza explicit pe X
Nu dăm domeniul/codomeniul funcțiilor	Nu putem aplica o funcție, decât dacă se potrivește tipul argumentului	

4 Calculabilitate

Definitia 4.1. *Spunem că o funcție este calculabilă, dacă există o metodă de a calcula f(n), pentru orice n.*

Modele de calculabilitate, echivalente:

(i) Mașina Turing (Turing)

(ii) Funcții recursive (Goedel)

(iii) $\lambda - calcul$ (Church)

5 Noțiuni de $\lambda - calcul$

Definitia 5.1. *Expresiile din $\lambda - calcul$ se numesc termeni.*

Notatia 5.2. *Notăm cu $\mathbb{V}$, mulțimea infinită a variabilelor si cu A alfabetul,* $A = \mathbb{V} \cup \{"(", ")", "\lambda", "."\}$.

Definitia 5.3 (Definiția inductivă a λ-termenilor, scrisă în forma BNF)**.**

$$M, N = x \mid MN \mid (\lambda x.M)$$

Definitia 5.4 (Definiție alternativă a λ-termenilor)**.**
Mulțimea λ-termenilor este cea mai mică submulțime $\Lambda \subseteq A^*$, a.î.:

(i) $\mathbb{V} \subseteq \Lambda$

(ii) Dacă $M, N \in \Lambda$, atunci $MN \in \Lambda$.

(iii) Dacă $x \in \mathbb{V}, M \in \Lambda$, atunci $(\lambda x.M) \in \Lambda$.

Conventia 5.5.

(i) Parantezele exterioare se elimină.

(ii) Aplicația este asociativă la stânga.

(iii) Corpul abstractizătilor se extinde la dreapta, cât de mult posibil.

(iv) Abstractizările se comprimă.

6 Variabile libere și variabile legate

Definitia 6.1. *Terminlogie specifică:*

(i) $\lambda\_.\_$ se numește operator de legare (binder).

(ii) x din $\lambda x.\_$ se numește variabilă de legare (binding).

(iii) N din $\lambda x.N$ se nume'te domeniul de legare al lui x(scope). Toate aparițiile lui x în N sunt legate.

(iv) O variabilă care nu este legată se numește liberă.

(v) Un termen care nu conține variabile libere se numește termen închis, sau combinator.

Notatia 6.2. *Mulțimea variabilelor libere ale unui termen se notează cu FV.*

Definitia 6.3 (Definiția recursivă pe termeni a mulțimii variabilelor libere)**.**

(i) $FV(x) = x$

(ii) $FV(MN) = FV(M) \cup FV(N)$

(iii) $FV(\lambda x.M) = FV(M) \setminus x$

7 Redenumire de variabile

Notatia 7.1. *Fie $x, y \in \mathbb{V}$ și M un termen. Atunci, notăm cu $M\langle y/x\rangle$ termenul obținut prin redenumirea lui x cu y în M.*

(i) $x\langle y/x\rangle \equiv y$

(ii) $z\langle y/x\rangle \equiv z$, dacă $x \neq z$

(iii) $MN\langle y/x\rangle \equiv (M\langle y/x\rangle)(N\langle y/x\rangle)$

(iv) $(\lambda x.M)\langle y/x\rangle \equiv \lambda y.(M\langle y/x\rangle)$

(v) $(\lambda z.M)\langle y/x\rangle \equiv \lambda z.(M\langle y/x\rangle)$, dacă $x \neq z$

Acest tip de redenumire se folosește doar în cazurile în care y nu apare în M.

8 α-echivalență

Definitia 8.1. *α-echivalența este cea mai mică relație de congruență pe λ-termeni, care satisface următoarele:*

$$(\text{REFL})\quad \frac{}{M = M}$$

$$(\text{SYMM})\quad \frac{M = N}{N = M}$$

$$(\text{TRANS})\quad \frac{M = N \quad N = P}{M = P}$$

$$(\text{CONG})\quad \frac{M = P \quad N = Q}{MN = PQ}$$

$$(\xi)\quad \frac{M = N}{\lambda x.M = \lambda x.N}$$

$$(\alpha)\quad \frac{y \notin M}{\lambda x.M = \lambda y.(M\langle y/x\rangle)}$$

Definitia 8.2 (Definiție alternativă)**.**
Numim α-echivalență egalitatea între termeni, modulo redenumiri de variabile legate.

Conventia 8.3. *Convenția Barendregt: Variabilele legate sunt redenumite, pentru a fi distincte.*

9 Substituții

Notatia 9.1. *Fie M,N termeni, $x \in \mathbb{V}$. Notăm cu $M[N/x]$, rezultatul obținut prin înlocuirea lui x cu N în M.*

Reguli pentru substituție:

(i) Inlocuim doar variabilele libere.

(ii) Pentru a evita legarea neintenționată a variabilelor libere, redenumim variabilele legate, înainte de substituție.

(i) $x[N/x] \equiv N$

(ii) $y[N/x] \equiv y$

(iii) $MP[N/x] \equiv (M[N/x])(P[N/x])$

(iv) $(\lambda x.M)[N/x] \equiv \lambda x.M$

(v) $(\lambda y.M)[N/x] \equiv \lambda y.(M[N/x])$, dacă $x \neq y$ și $y \notin FV(N)$.

(vi) $(\lambda y.M)[N/x] \equiv \lambda y'.(M\langle y'/y\rangle[N/x])$, dacă $x \neq y$, $y \in FV(N)$ și y' variabilă nouă.

10 β-reducție

Conventia 10.1. *Doi termeni sunt egali, dacă sunt α-echivalenți.*

Definitia 10.2. *Terminlogie specifică:*

(i) Se numește β-reducție, procesul de evaluare a λ-termenilor, prin "pasarea de argumente funcțiilor".

(ii) Se numește β-redex, un termen de forma $(\lambda x.M)N$.

(iii) Redusul unui termen $(\lambda x.M)N$ este $M[N/x]$.

(iv) Se numește formă normală un λ-termen fără redex-uri.

Definitia 10.3. *Un pas de β-reducție este cea mai mică relație pe λ-termeni, care satisface următoarele:*

$$(\beta)\quad \frac{}{(\lambda x.M)N \to_\beta M[N/x]}$$

$$(\text{CONG1})\quad \frac{M \to_\beta P}{MN \to_\beta PN}$$

$$(\text{CONG2})\quad \frac{N \to_\beta P}{MN \to_\beta MP}$$

$$(\xi)\quad \frac{M \to_\beta P}{(\lambda x.M) \to_\beta \lambda x.P}$$

Procesul de β-reducție constă în aplicarea repetată a câte unui pas de β-reducție, rezultatul final nefiind afectat de ordinea alegerii redex-urilor.

Notatia 10.4. *Notăm cu* $M \twoheadrightarrow M'$*, faptul că, prin efectuarea a 0 sau mai mulți pași de* β*-reducție asupra lui M, se poate obține M'.*

Definitia 10.5. *Un termen M se numește:*

(i) Slab-normalizabil, dacă $\exists N$ *formă normală, a.î.* $M \twoheadrightarrow N$.

(ii) Puternic-normalizabil, dacă nu există β*-reducții infinite care încep din M.*

Teorema 10.6 (Confluența β-reducției)**.** ***Teorema Church-Rosser***
Dacă $M \twoheadrightarrow M_1$ *și* $M \twoheadrightarrow M_2$*, atunci* $\exists M'$*, a.î.* $M_1 \twoheadrightarrow M'$ *și* $M_2 \twoheadrightarrow M'$.

Consecinta 10.7. *Un* λ*-termen are cel mult o* β*-formă normală (modulo* α*-echivalență).*

10.1 Strategii de evaluare

(i) **Strategia normală (leftmost-outermost)**: se alege, la fiecare pas, redex-ul cel mai din stânga și apoi cel mai din exterior. Avantajul acestei strategii constă în faptul că, dacă există o formă normală pentru termenul evaluat, strategia garantează găsirea acestuia.

(ii) **Strategia aplicativă (leftmost-innermost)**: se alege, la fiecare pas, redex-ul cel mai din stânga și apoi cel mai din interior.

10.1.1 Strategii în programarea funcțională

(i) **Strategia CBN**: folosește strategia normală de β-reducție, fără reduceri în corpul λ-abstractizărilor. Amânăm evaluarea argumentelor (lazy-evaluation).

(ii) **Strategia CBV**: folosește strategia aplicativă de β-reducție, fără reduceri în corpul λ-abstractizărilor. Funcțiile sunt apelate doar prin valoare (după ce argumentele au fost complet evaluate).

Haskell este singurul limbaj de programare funcțional care implementeaza Strategia CBN.

Definitia 10.8. *In acest context, numim valoare orice* λ*-termen pentru care nu există* β*-reducții.*

Exemplul 10.9. $(\lambda x.x)$ este o valoare, în timp ce $(\lambda x.x)1$ nu este.

11 Expresivitatea λ-calculului

11.1 Reprezentarea valorilor booleene

$$T = \lambda xy.x \qquad F = \lambda xy.y \qquad if = \lambda btf.btf$$
$$and = \lambda b_1b_2.if b_1b_2F \qquad or = \lambda b_1b_2.if b_1Tb_2 \qquad not = \lambda b_1.if b_1FT$$

11.2 Reprezentarea numerelor naturale, sub forma numeralilor Church

$$\bar{n} = \lambda fx.f^nx$$
$$Succ = \lambda nfx.f(nfx) \qquad add = \lambda mnfx.mf(nfx) \qquad mul = \lambda mn.m(add\ n)\bar{0} \qquad exp = \lambda mn.m(mul\ n)\bar{1}$$
$$isZero = \lambda nTF.n(\lambda z.F)T$$

12 Puncte fixe

Definitia 12.1 (Definiția generală a punctului fix)**.**
Fie f o funcție. Spunem că x este un ***punct fix*** *al lui f, dacă* $f(x) = x$.

Notatia 12.2. *Fie M și N termeni. Spunem că* $M =_\beta M'$*, dacă N poate fi obținut în 0 sau mai mulți pași direcți sau inverși de* β*-reducție din M.*

Exemplul 12.3. $(\lambda y.yv)z =_\beta (\lambda x.zx)v$, deoarece avem:

$$(\lambda y.yv)z \rightarrow_\beta zv_\beta \leftarrow (\lambda x.zx)v,$$
unde cu $_\beta \leftarrow$ am notat inversa relației de β-reducție.

Definitia 12.4 (Definiția punctului fix în λ-calcul).
Dacă F și M sunt λ-termeni, spunem că M este un ***punct fix*** *al lui F, dacă $FM =_\beta M$.*

Teorema 12.5. *In λ-calcul fără tipuri, orice termen are un punct fix, și anume $M = (\lambda x.F(xx))(\lambda x.F(xx))$.*

Definitia 12.6. ***Combinatorii de puncte fixe*** *sunt termeni închiși care construiesc un punct fix pentru un termen arbitrar.*

Exemplul 12.7. Combinatorul de punct fix al lui **Curry**: $\mathbf{Y} = \lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
Pentru orice termen F, $\mathbf{Y}$ este un punct fix al lui F, deoarece $\mathbf{Y}F \twoheadrightarrow F(\mathbf{Y}F)$.

Exemplul 12.8. Combinatorul de punct fix al lui **Turing**: $\mathbf{\Theta} = (\lambda xy.y(xxy))(\lambda xy.y(xxy))$
Pentru orice termen F, $\mathbf{\Theta}$ este un punct fix al lui F, deoarece $\mathbf{\Theta}F \twoheadrightarrow F(\mathbf{\Theta}F)$.

12.1 Rezolvarea de ecuații în λ-calcul

Putem rezolva ecuații, cu ajutorul punctelor fixe (conform Teoremei 12.5, în λ-calcul fără tipuri, orice termen are un punct fix).

Exemplul 12.9. Considerăm funcția factorial, definită astfel: $fact\ n = if(isZero\ n)(\bar{1})(mul\ n(fact(pred\ n)))$.
Rescriem ecuația, astfel: $fact = (\lambda fn.if(isZero\ n)(\bar{1})(mul\ n(f(pred\ n))))fact$.
Notăm termenul $\lambda fn.if(isZero\ n)(\bar{1})(mul\ n(f(pred\ n)))$ cu F și ecuația devine: $fact = F\ fact$ - o ecuație de punct fix, pe care o putem rezolva, luând:

$$fact = \mathbf{Y}F = \mathbf{Y}(\lambda fn.if(isZero\ n)(\bar{1})(mul\ n(f(pred\ n))))$$

13 λ-calcul cu tipuri simple

Observatia 13.1. *Limitări ale λ-calculului fără tipuri:*

(i) Sunt permise aplicări de forma MM, care sunt contraintuitive.

(ii) Nu este garantată exsitenâ formelor normale pentru λ-termeni, ceea ce poate conduce la calcule infinite.

(iii) Orice λ-termen are un punct fix, ceea ce nu este valabil pentru funcții oarecare.

Notatia 13.2. *Notăm cu $\mathbb{V}$ mulțimea infinită a* ***tipurilor variabilă****.*
De obicei, vom folosi simbolurile $\alpha, \beta, \gamma \ldots$, pentru tipuri variabilă, dar, uneori, folosim și A, B, ...

Definitia 13.3. *(i) Mulțimea* ***tipurilor simple*** *este definită prin:*

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T}$$

(a) ***Tipuri variabilă****: Dacă $\alpha \in \mathbb{V}$, atunci $\alpha \in \mathbb{T}$. (reprezintă tipuri de bază)*

(b) ***Tipuri săgeată****: Dacă $\sigma, \tau \in \mathbb{T}$, atunci $\sigma \to \tau \in \mathbb{T}$ (reprezintă tipuri pentru funcții).*

Observatia 13.4. *Parantezele din tipurile săgeată sunt asociative la dreapta.*

Notatia 13.5. *Not m cu $M : \sigma$, faptul că termenul M are tipul σ.*

Observatia 13.6. *Orice variabilă are un tip unic.*

13.1 Termeni și tipuri

(i) (**Variabilă**) $x : \sigma$

(ii) (**Aplicare**) Dacă $M : \sigma \to \tau$ și $N : \sigma$, atunci $MN : \tau$.

(iii) (**Abstractizare**) Dacă $x : \sigma$ și $M : \tau$, atunci $\lambda x.M : \sigma \to \tau$.

Definitia 13.7. *M are tip (este typeable), dacă există un tip σ, a.î. $M : \sigma$.*

Exemplul 13.8. Exemplu de termen care nu are tip: xx (deoarece x ar trebui să aibă concomitent tipurile $\sigma \to \tau$ și τ, ceea ce este imposibil).

13.1.1 Church-typing vs. Curry-typing

Church-typing (asociere implicită)	**Curry-typing (asociere explicită)**
Dăm tipurile variabilelor, la introducere	Nu prescriem tipurile variabilelor, trebuie deduse

Observatia 13.9. *Asocierea implicită are proprietatea de a găsi cele mai generale tipuri posibile. Totuși, în continuare, vom folosi Church-typing, marcând tipurile* ***variabilelor legate****, după introducerea lor cu o abstractizare, și menționând tipurile* ***variabilelor libere*** *într-un* ***context****.*

13.2 Sistem de deducție pentru Church $\lambda \to$

Definitia 13.10. *Mulțimea λ-termenilor cu pre-tipuri este:*

$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\Lambda_{\mathbb{T}}$$

Definitia 13.11. *Terminologie:*

(i) O ***afirmație*** *este o expresie de forma $M : \sigma$, unde $M \in \Lambda_{\mathbb{T}}$ și $\sigma \in \mathbb{T}$. M se numește* ***subiect*** *și σ* ***tip****.*

(ii) O ***declarație*** *este o afirmație în care subiectul este o variabilă $(x : \sigma)$.*

(iii) Un ***context*** *este o listă de declarații cu subiecți diferiți.*

(iv) O ***judecată*** *este o expresie de forma $\Gamma \vdash M : \sigma$, unde Γ este un context și $M : \sigma$ este o afirmație.*

(v) Un termen M în calculul Church $\lambda \to$ este ***legal****, dacă există un context Γ și un tip σ, a.î. $\Gamma \vdash M : \sigma$.*

Sistemul de deducție:

$$(\textsc{var}) \quad \frac{}{\Gamma \vdash x : \sigma} \text{ if } x : \sigma \in \Gamma$$

$$(\textsc{app}) \quad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

$$(\textsc{abs}) \quad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash (\lambda x : \sigma.M) : \sigma \to \tau}$$

13.3 Probleme din teoria tipurilor

Următoarele probleme sunt decidabile, pentru calculul Church $\lambda \to$:

(i) Type checking = verificarea faptului că, date fiind un context, un termen și un tip dat, termenul poate avea tipul respectiv în acel context: *context* $\vdash$ *term* : *type*

(ii) Typability = veriricarea legalității: ? $\vdash$ *term* :?

(iii) Type assignment = typability în care se dă și contextul: *context* $\vdash$ *term* :?

(iv) Term finding (inhabitation) = dându-se un context și un tip, trebuie stabilit dacă există un termen cu acel tip, în contextul dat: *context* $\vdash$? : *type*

13.4 Limitări ale λ-calculului cu tipuri simple

(i) Nu avem recursie nelimitată (deoarece combinatorii de punct fix nu sunt typeable). Avem doar **recursie primitvă** (cu număr cunoscut de pași).

(ii) Tipurile pot fi prea restrictive $\to$ soluția adoptată: **cuantificatori de tipuri** ($\lambda x.x : \Pi\alpha.\alpha \to \alpha$)

13.5 Alte tipuri

Adăugăm tipurile Unit, Void $\times$, $+$, la mulțimea de tipuri , termenii construiți cu ele, la mulțimea λ-termenilor cu pre-tipuri, și, pentru fiecare tip nou adăugat, introducem regulile de inferență corespunzătoare:

$$\mathbb{T} = \mathbb{V} \mid \mathbb{T} \to \mathbb{T} \mid Unit \mid Void \mid \mathbb{T} \times \mathbb{T} \mid \mathbb{T} + \mathbb{T}$$
$$\Lambda_{\mathbb{T}} = x \mid \Lambda_{\mathbb{T}}\Lambda_{\mathbb{T}} \mid \lambda x : \mathbb{T}.\Lambda_{\mathbb{T}} \mid unit \mid \langle \Lambda_{\mathbb{T}}, \Lambda_{\mathbb{T}} \rangle \mid fst\Lambda_{\mathbb{T}} \mid snd\Lambda_{\mathbb{T}} \mid Left\Lambda_{\mathbb{T}} \mid \Lambda_{\mathbb{T}} \mid case\ \Lambda_{\mathbb{T}} of \Lambda_{\mathbb{T}}; \Lambda_{\mathbb{T}}$$

$$(\textsc{unit})\quad \frac{}{\Gamma \vdash unit : Unit}$$

$$(\times_I)\quad \frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N \rangle : \sigma, \tau}$$

$$(\times_{E_1})\quad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash fstM : \sigma}$$

$$(\times_{E_2})\quad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash sndM : \tau}$$

$$(+_{I_1})\quad \frac{\Gamma \vdash M : \sigma}{\Gamma \vdash Left\ M : \sigma + \tau}$$

$$(+_{I_2})\quad \frac{\Gamma \vdash M : \tau}{\Gamma \vdash Right\ M : \sigma + \tau}$$

$$(+_E)\quad \frac{\Gamma \vdash M : \sigma + \tau \quad \Gamma \vdash M_1 : \sigma \to \gamma \quad \Gamma \vdash M_2 : \tau \to \gamma}{\Gamma \vdash case\ M\ of\ M_1; M_2 : \gamma}$$

13.6 Corespondența Curry-Howard

Definitia 13.12. *O formulă este **tautologie**, dacă are valoarea 1, pentru toate evaluările posibile.*

Definitia 13.13.

(i) Corectitudine = sintaxa implica semantica

(ii) Completitudine = sintaxa coincide cu semantica

λ-calcul cu tipuri	**Deducție naturală**
$(\times_I)\quad \frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \langle M, N \rangle : \sigma, \tau}$	$(\wedge_I)\quad \frac{\Gamma \vdash \sigma \quad \Gamma \vdash \tau}{\Gamma \vdash \wedge\sigma\tau}$
$(\times_{E_1})\quad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash fstM : \sigma}$	$(\wedge_{E_1})\quad \frac{\Gamma \vdash \sigma \wedge \tau}{\Gamma \vdash \sigma}$
$(\times_{E_2})\quad \frac{\Gamma \vdash M : \sigma \times \tau}{\Gamma \vdash sndM : \tau}$	$(\wedge_{E_2})\quad \frac{\Gamma \vdash \sigma \wedge \tau}{\Gamma \vdash \tau}$
$(\to_I)\quad \frac{\Gamma \cup \{x : \sigma\} \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}$	$(\supseteq_I)\quad \frac{\Gamma \cup \{\sigma\} \vdash \tau}{\Gamma \vdash \sigma \to \tau}$
$(\to_E)\quad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$	$(\supseteq_E)\quad \frac{\Gamma\sigma \supseteq \tau \quad \Gamma \vdash \sigma}{\Gamma \vdash \tau}$

Teoria tipurilor	**Logică**
tipuri	formule
termeni	demonstrații
Inhabitation pentru tipul σ	demonstrație pentru σ
tip produs	conjuncție
tip sumă	disjuncție
tip funcție	implicație
tip Void	fals
tip Unit	adevăr