

Programare funcțională

Introducere în programarea funcțională folosind Haskell
C11

Claudia Chiriță

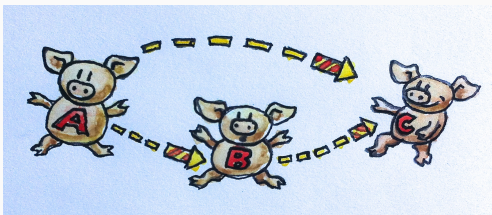
Denisa Diaconescu

Departamentul de Informatică, FMI, UB

Intro în Teoria Categoriilor

O categorie

- A category is an embarrassingly simple concept.
Bartosz Milewski, *Category Theory for Programmers*
- Categorie = obiecte + săgeți
- Ingredient cheie: compunerea de săgeți

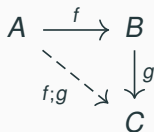


credits: Bartosz Milewski

Categorie

O **categorie** \mathbb{C} constă în

- **Obiecte:** notate A, B, C, \dots . Notăm cu $|\mathbb{C}|$ obiectele lui \mathbb{C}
- **Săgeți:** pentru orice obiecte A și B , există o mulțime de săgeți $\mathbb{C}(A, B)$
 - notăm $f \in \mathbb{C}(A, B)$ cu $f : A \rightarrow B$ sau $A \xrightarrow{f} B$
- **Compunere:** pentru orice săgeți $f : A \rightarrow B$ și $g : B \rightarrow C$ există o săgeată $f; g : A \rightarrow C$



- **Identitate:** pentru orice obiect A există o săgeată $id_A : A \rightarrow A$
- **Axiome:** pentru orice săgeți $f : A \rightarrow B$, $g : B \rightarrow C$, și $h : C \rightarrow D$, avem

$$f; (g; h) = (f; g); h \qquad f; id_B = f = id_A; f$$

Exemplu - categoria de mulțimi

Categoria **Set** are

- **Obiecte:** mulțimi
- **Săgeți:** funcții
- **Compunere:** compunerea de funcții
- **Identitate:** pentru orice mulțime A , funcția identitate $id_A : A \rightarrow A$,
 $id_A(a) = a$
- **Axiome:** ✓

Un **monoid** **M** este o structură $\langle M, +, e \rangle$ astfel încât

- **M** este o mulțime
- **+** : $M \times M \rightarrow M$ este asociativă
(adică $(a + b) + c = a + (b + c)$ pentru orice $a, b, c \in M$)
- **e** $\in M$ este identitate pentru **+**
(adică $e + a = a + e = a$ pentru orice $a \in M$)

Monoizii sunt un concept extrem de puternic:

- **Stau în spatele aritmeticii de bază**
 - și adunarea, și înmulțirea formează un monoid
- **Sunt prezenți peste tot în programare**
 - șiruri de caractere, liste ...

Exemplu - categoria de monoizi

Categoria **Mon** are

- **Obiecte:** monoizi
- **Săgeți:** morfisme de monoizi
(adică funcții care nu "strică" operația de monoid)
- **Compunerea:** compunerea de morfisme de monoizi
- **Identitatea:** pentru orice obiect **M**, $id_M : M \rightarrow M$, $id_M(m) = m$
- **Axiome:** ✓

Exemplu - un monoid ca o categorie

Orice monoid $\mathbf{M} = \langle M, +, e \rangle$ este o categorie cu

- Obiecte: un singur obiect \heartsuit
- Săgeți: elementele mulțimii M (adică $\mathbf{M}(\heartsuit, \heartsuit) = M$)
- Compunerea: operația de monoid $+$
- Identitatea: identitatea monoidului e
- Axiome:

$$f; (g; h) = (f; g); h$$

$$f; id_B = f = id_A; f$$

$$a + (b + c) = (a + b) + c$$

$$a + e = a = e + a$$

Exemplu - Categoria \mathbf{Hask}

- **Obiectele:** tipuri
- **Săgețile:** funcții între tipuri

$f :: a \rightarrow b$

- **Identități:** funcția polimorfică **id**

$\mathbf{id} :: a \rightarrow a$

- **Compunere:** funcția polimorfică **(.)**

$(.) :: (b \rightarrow c) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow c$

Subcategorii ale lui `Hask` date de tipuri parametrizate

- **Obiecte:** o clasă restrânsă de tipuri din `|Hask|`
Exemplu: tipuri de forma `[a]`
- **Săgeți:** toate funcțiile din `Hask` între tipurile obiecte

Exemple:

concat :: `[[a]] -> [a]`

words :: `[Char] -> [String]`

reverse :: `[a] -> [a]`

Exemple

Liste obiecte: tipuri de forma `[a]`

Optiuni obiecte: tipuri de forma `Maybe a`

Funcții de sursă t obiecte: tipuri de forma `t -> a`

De ce categorii?

(Des)compunerea este esența programării

- Am de rezolvat problema P
- O descompun în subproblemele P_1, \dots, P_n
- Rezolv problemele P_1, \dots, P_n cu programele p_1, \dots, p_n
 - Eventual aplicând recursiv procedura de față
- Compun rezolvările p_1, \dots, p_n într-o rezolvare p pentru problema inițială

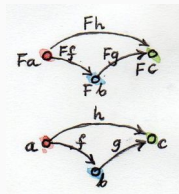
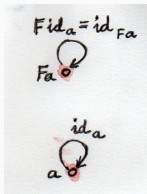
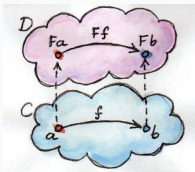
Categoriile rezolvă problema compunerii

- Ne forțează să abstractizăm datele
- Se poate acționa asupra datelor doar prin săgeți
- Forțează un stil de compunere independent de structura obiectelor

Functori

Date fiind două categorii \mathbb{C} și \mathbb{D} , un functor $F : \mathbb{C} \rightarrow \mathbb{D}$ este dat de

- O funcție $F : |\mathbb{C}| \rightarrow |\mathbb{D}|$ de la obiectele lui \mathbb{C} la cele ale lui \mathbb{D}
- Pentru orice $A, B \in |\mathbb{C}|$, o funcție $F : \mathbb{C}(A, B) \rightarrow \mathbb{D}(F(A), F(B))$ compatibilă cu identitățile și cu compunerea
 - $F(id_A) = id_{F(A)}$ pentru orice $A \in |\mathbb{C}|$
 - $F(f; g) = F(f); F(g)$ pentru orice $f : A \rightarrow B, g : B \rightarrow C$



Bartosz Milewski — Functors

Functori în Haskell

În Haskell o instanță **Functor** f este dată de

- Un tip $f\ a$ pentru orice tip a
(deci f trebuie să fie tip parametrizat)
- Pentru orice două tipuri a și b , o funcție

$$\text{fmap} :: (a \rightarrow b) \rightarrow (f\ a \rightarrow f\ b)$$

- Compatibilă cu identitățile și cu compunerea

$$\text{fmap}\ \text{id} == \text{id}$$
$$\text{fmap}\ (g \ .\ h) == \text{fmap}\ g \ .\ \text{fmap}\ h$$
$$\text{pentru orice } h :: a \rightarrow b \text{ și } g :: b \rightarrow c$$

Correspondența Curry-Howard

Teoria Tipurilor	Logică
tipuri	formule
termeni	demonstrații
<i>inhabitation</i> a tipului A	demonstrație a lui A

Correspondența Curry-Howard

Teoria Tipurilor	Logică
tipuri	formule
termeni	demonstrații
<i>inhabitation</i> a tipului A	demonstrație a lui A
tip produs	conjunție
tip funcție	implicație

Coreșpondența Curry-Howard

Teoria Tipurilor	Logică
tipuri	formule
termeni	demonstrații
<i>inhabitation</i> a tipului <i>A</i>	demonstrație a lui <i>A</i>
tip produs	conjunție
tip funcție	implicație
tip sumă	disjunție
tipul void	false
tipul unit	true

Să investigăm mai mult!

Obiect inițial și obiect final

Într-o categorie \mathbb{C} ,

- un obiect T se numește **terminal** dacă pentru orice obiect A există o unică săgeată

$$\tau_A : A \rightarrow T$$

În **Hask**, obiectul terminal este `()`.

Pentru orice tip a , avem `unit :: a -> ()`.

- un obiect I se numește **inițial** dacă pentru orice obiect A există o unică săgeată

$$\iota_A : I \rightarrow A$$

În **Hask**, obiectul inițial este **`Void`**.

Pentru orice tip a , avem `absurd :: Void -> a`.

De ce obiect final?

Fie \mathbb{C} o categorie cu obiect terminal T .

Avem următoarea interpretare:

- **Formulele propoziționale** sunt obiectele lui \mathbb{C}
- **Constanta \top (true)** este obiectul terminal T
- O demonstrație a lui A este o săgeată $f : T \rightarrow A$
- O demonstrație a lui A din ipotezele B este o săgeată $f : B \rightarrow A$

Produce

Fie A și B obiecte într-o categorie \mathbb{C} .

Spunem că

$$A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B$$

este **produs al lui A și B** dacă pentru orice

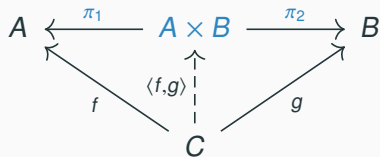
$$A \xleftarrow{f} C \xrightarrow{g} B$$

există o unică săgeată

$$\langle f, g \rangle : C \rightarrow A \times B$$

astfel încât

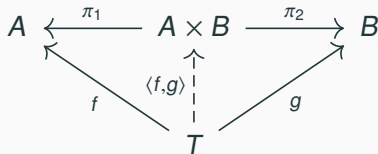
$$\langle f, g \rangle; \pi_1 = f \quad \langle f, g \rangle; \pi_2 = g$$



Produce

Fie \mathbb{C} o categorie cu obiect terminal T și produse.

Fie A și B două obiecte în \mathbf{C} .



$$\frac{f : T \rightarrow A \quad g : T \rightarrow B}{\langle f, g \rangle : T \rightarrow A \times B}$$

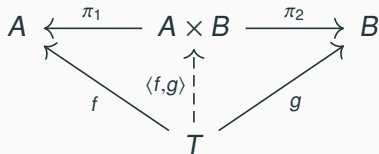
$$\frac{\langle f, g \rangle : T \rightarrow A \times B}{\langle f, g \rangle ; \pi_1 : T \rightarrow A}$$

$$\frac{\langle f, g \rangle : T \rightarrow A \times B}{\langle f, g \rangle ; \pi_2 : T \rightarrow B}$$

Produce

Fie \mathbb{C} o categorie cu obiect terminal T și produse.

Fie A și B două obiecte în \mathbf{C} .



$$\frac{f: T \rightarrow A \quad g: T \rightarrow B}{\langle f, g \rangle: T \rightarrow A \times B}$$

$$\frac{\langle f, g \rangle: T \rightarrow A \times B}{\langle f, g \rangle; \pi_1: T \rightarrow A}$$

$$\frac{\langle f, g \rangle: T \rightarrow A \times B}{\langle f, g \rangle; \pi_2: T \rightarrow B}$$

Vă aduce aminte de ceva?

Trei perspective

λ -calcul cu tipuri

$$\frac{a : A \quad b : B}{\langle a, b \rangle : A \times B} (\times_I)$$

$$\frac{p : A \times B}{fst\ p : A} (\times_{E_1})$$

$$\frac{p : A \times B}{snd\ p : B} (\times_{E_2})$$

Deducție naturală

$$\frac{A \quad B}{A \& B} (\&_I)$$

$$\frac{A \& B}{A} (\&_{E_1})$$

$$\frac{A \& B}{B} (\&_{E_2})$$

O Categorie*

$$\frac{f : T \rightarrow A \quad g : T \rightarrow B}{\langle f, g \rangle : T \rightarrow A \times B}$$

$$\frac{\langle f, g \rangle : T \rightarrow A \times B}{\langle f, g \rangle ; \pi_1 : T \rightarrow A}$$

$$\frac{\langle f, g \rangle : T \rightarrow A \times B}{\langle f, g \rangle ; \pi_2 : T \rightarrow B}$$

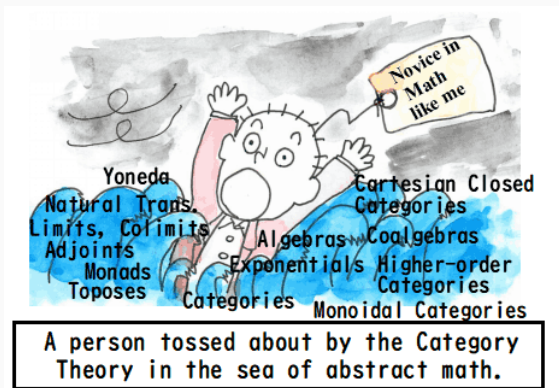
* O categorie cu obiect terminal T și produse

Corespondența Curry-Howard-Lambek

Teoria tipurilor	Logică	Teoria Categoriilor
tipuri	formule	obiecte
termeni	demonstrații	săgeți
<i>inhabitation</i> a tipului A	demonstrație a lui A	săgeată $f : T \rightarrow A$
tip funcție	implicație	?
tip produs	conjunție	produs
tip sumă	disjunție	coprodus
tipul void	fals	obiect inițial
tipul unit	true	obiect terminal

Categorii Cartezian Închise

O categorie cu **obiect terminal**, **produse** și **exponențiali** se numește o **categorie cartezian închisă**. *Cartesian Closed Category (CCC)*



Mulțumim că ați participat la acest curs.

Baftă la examen!