

Logică și structuri discrete
Relații. Funcții parțiale

Marius Minea
marius@cs.upt.ro

<http://cs.upt.ro/~marius/curs/1sd/>

24 octombrie 2016

Relații în lumea reală și informatică

O relație (matematică) modelează *legătura* dintre două entități (posibil de *tip* diferit)

relații subiect-obiect: un om a citit o carte

relații umane: copil , părinte , prieten

relații cantitative : egal, mai mic

Transpuse în informatică:

rețele sociale : “prieten”, “follow”, “în cercuri”, etc.

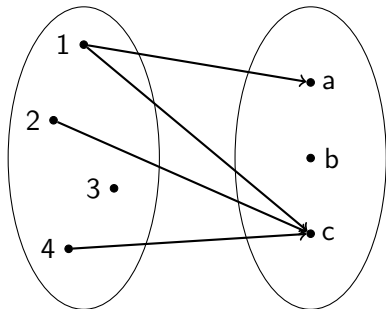
O relație între elementele *aceleiași* mulțimi definește un *graf* (elementele sunt noduri, relația e reprezentată prin muchii)

⇒ relațiile sunt o noțiune cheie în teoria grafurilor

Relații: definiție și exemple

O *relație binară* R între două mulțimi A și B e o mulțime de perechi:
o *submulțime a produsului cartezian* $A \times B$: $R \subseteq A \times B$

Notăm $(x, y) \in R$, sau $x R y$, sau $R(x, y)$ (x e în relație cu y)



$$A = \{1, 2, 3, 4\}, \quad B = \{a, b, c\}$$
$$R = \{(1, a), (1, c), (2, c), (4, c)\}$$

O relație e o noțiune *mai generală* decât o funcție: o funcție asociază *fiecăru* $x \in A$ *un singur* $y \in B$

Într-o relație putem avea:

- 1: are asociate *mai multe* elemente: a, c
- 2: are asociat *un singur* element: c
- 3: nu are asociat *niciun* element din B

Relații (cont.)

Generalizat, putem avea o relație *n-ară* care e o mulțime de *n*-tupluri (din produsul cartezian a *n* mulțimi).

Exemplu: $R \subseteq \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$

$R(x, y, m)$ dacă m e un multiplu comun pentru a și b :

$R(2, 9, 18)$, $R(6, 9, 18)$, $R(2, 9, 36)$, etc.

În general, o relație nu e o noțiune simetrică:

produsul cartezian/perechea sunt noțiuni ordonate, $(x, y) \neq (y, x)$

Există, desigur, relații simetrice (în lumea reală și în matematică)

Reprezentarea unei relații

Explicit, prin mulțimea perechilor (dacă e finită)

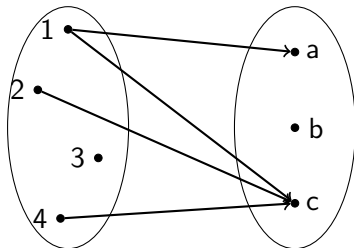
Printr-o *regulă* care leagă elementele:

$$R = \{(x, x^2 + 1) \mid x \in \mathbb{Z}\}$$

Ca matrice booleană / binară, pentru A, B finite,
cu liniile indexate după A , și coloanele după B

$m_{xy} = 1$ dacă $(x, y) \in R$, $m_{xy} = 0$ dacă $(x, y) \notin R$

reprezentare practică dacă A și B nu sunt foarte mari



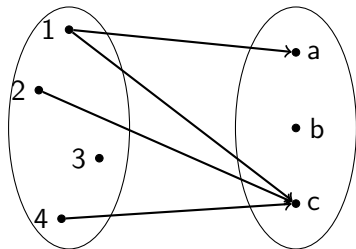
	a	b	c
1	1	0	1
2	0	0	1
3	0	0	0
4	0	0	1

Relația văzută ca funcție

O *relație* $R \subseteq A \times B$ poate fi văzută ca o *funcție* $f_R : A \rightarrow \mathcal{P}(B)$ de la A la mulțimea părților lui B :

$$f_R(x) = \{y \in B \mid (x, y) \in R\}$$

Asociază fiecărui x mulțimea elementelor lui B cu care x e în relație (posibil vidă): $f_R(1) = \{a, c\}$, $f_R(3) = \emptyset$



	a	b	c
1	1	0	1
2	0	0	1
3	0	0	0
4	0	0	1

Un vector de biți/booleani poate reprezenta o mulțime:

$\begin{array}{ccc} a & b & c \\ 1 & 0 & 1 \end{array}$ reprezintă $\{a, c\}$ (prin funcția caracteristică)

Numărul de relații între două mulțimi

Între A și B (finite) există $2^{|A| \cdot |B|}$ relații $R \subseteq A \times B$

Rezultă direct din definiție: o relație e o submulțime $R \subseteq A \times B$.
Deci, $R \in \mathcal{P}(A \times B)$. Dar $|\mathcal{P}(A \times B)| = 2^{|A \times B|} = 2^{|A| \cdot |B|}$.

Sau, folosind reprezentarea ca matrice, care are $|A| \cdot |B|$ elemente.
Fiecare poate fi ales independent în 2 feluri: 0 sau 1, deci $2^{|A| \cdot |B|}$ combinații.

Sau, considerând funcția corespunzătoare, $f : A \rightarrow \mathcal{P}(B)$.
Numărul de funcții e $|\mathcal{P}(B)|^{|A|} = (2^{|B|})^{|A|} = 2^{|B| \cdot |A|}$

Funcții parțiale

Dacă relaxăm condiția ca o funcție să asocieze o valoare *fiecărui* element, obținem o *funcție parțială* $f : A \rightarrow B$.

O funcție parțială e un *caz particular de relație*:
asociază câte *un singur* element din B (ca funcția)
dar nu neapărat fiecărui element din A .

Funcții parțiale sunt utile

– când domeniul exact al funcției nu e cunoscut

(funcții care nu sunt neapărat calculabile în orice punct).

în conjectura Collatz ($3 \cdot n + 1$), numărul de pași până la 1
ar putea să nu existe (infini) pentru anumiți n

– când domeniul de definiție al funcției e foarte mare sau nelimitat,
dar reprezentăm funcția explicit doar pentru valorile de interes

Exemplu: populația unei localități

posibil să nu știm populația pentru toate localitățile

dacă argumentul e un șir, nu orice șir e nume de localitate

Dicționare: funcții parțiale în ML

Un *dicționar* memorează *perechi* care asociază o *cheie* (primul element) cu o *valoare* (al doilea element)

În ML, folosim modulul Map, parametrizat (ca la mulțimi) cu un modul care definește tipul (ordonat) al cheilor

```
module M = Map.Make(String)
```

crează un modul care reprezintă asocieri între șiruri de caractere (cheia, de tip string) și un tip valoare încă neprecizat

Adăugând o pereche în asociere se instanțiază și tipul valorilor:

```
let m = M.add "x" 5 M.empty
```

adaugă la dicționarul vid o asociere de la șirul "x" la 5

echivalent:

```
let m = M.singleton "x" 5
```

acum, m e un *dicționar* de la șiruri la întregi

Căutăm valoarea asociată unei chei în dicționar:

```
M.find "x" m      returnează întregul 5
```

```
M.find "y" m      generează excepția Not_found
```

Lucrul cu excepții

O *excepție* este o condiție specială care întrerupe calculul normal dacă nu e *tratată*, se abandonează execuția programului altfel, codul de *tratare a excepției* stabilește ce se face

Unele funcții standard *generează* excepții:

```
List.hd [] produce Exception: Failure "hd"  
char_of_int 300 dă Invalid_argument "char_of_int"
```

Operații matematice pot genera excepții:

```
1 / 0 produce Exception: Division_by_zero
```

Aceste excepții (primele 2 cu parametru șir, ultima fără parametru) și altele sunt predefinite (în modulul Pervasives deschis implicit)

Putem genera excepții cu funcția raise (parametru: excepție):

```
raise Not_found sau raise (Failure "gresit") etc.
```

```
failwith "mesaj" e echivalent cu raise (Failure "mesaj")  
invalid_arg "msg" e la fel cu raise (Invalid_argument "msg")
```

Tratarea excepțiilor

Pentru a lucra corect, trebuie să știm ce excepții pot genera funcțiile pe care le folosim le *tratăm* corespunzător

Sintaxa: **try** *expresie* e tot o formă de expresie

with *tipar*

unde *tipar* tratează una sau mai multe excepții și are forma

| *excepție-1* -> *expresie-1* (valoarea în acest caz)

| *excepție-2* -> *expresie-2* (valoarea în cazul 2)

...

Dacă *expresie* se evaluează normal, ea dă rezultatul; altfel, dacă apare *excepție-k* se evaluează *expresie-k* pe toate ramurile, expresiile au același tip cu cea din **try** *expresie*

```
fun x m -> try M.find x m  
         with Not_found -> 0
```

Excepțiile se propagă, terminând fiecare funcție, până când sunt “prinse” de un bloc de tratare – altfel, programul e abandonat.

Relații cu ajutorul dicționarelor

Am văzut că o *relație* $R \subseteq A \times B$ poate fi privită ca o *funcție* $f_R : A \rightarrow \mathcal{P}(B)$ de la A la mulțimea părților lui B :

$$f_R(x) = \{y \in B \mid (x, y) \in R\}$$

Dicționarul va fi atunci de la A la *mulțimi* de elemente din B

```
module M = Map.Make(String) (* dicționar pe siruri *)
```

```
module S = Set.Make(String) (* mulțimea de valori *)
```

```
let addpair m (x, y) =
```

```
  let oldset = try M.find x m (* mulțimea asociată cu x *)
```

```
  with Not_found -> S.empty (* nu e, deci mulțimea vidă *)
```

```
  in M.add x (S.add y oldset) m
```

```
(* creează dicționar din lista *)
```

```
let setmap_of_pairs = List.fold_left addpair M.empty
```

```
setmap_of_pairs [("tms", "arad"); ("tms", "lugoj)];;
```

```
asociază "tms" cu mulțimea {"arad", "lugoj"}
```

Relații binare pe o mulțime

Următoarele proprietăți sunt definite pentru relații binare pe o (aceeași) mulțime X : $R \subseteq X \times X$

reflexivă: pentru orice $x \in X$ avem $(x, x) \in R$

ireflexivă: pentru orice $x \in X$ avem $(x, x) \notin R$

simetrică: pentru orice $x, y \in X$,
dacă $(x, y) \in R$ atunci și $(y, x) \in R$

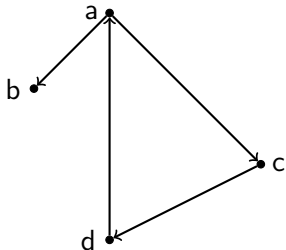
antisimetrică: pentru orice $x, y \in X$,
dacă $(x, y) \in R$ și $(y, x) \in R$, atunci $x = y$

tranzitivă: pentru orice $x, y, z \in X$,
dacă $(x, y) \in R$ și $(y, z) \in R$, atunci $(x, z) \in R$

Relații binare și grafuri

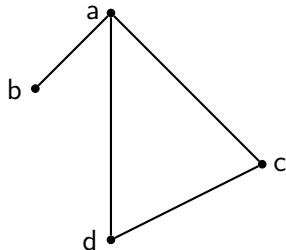
O relație binară pe o mulțime X poate fi reprezentată ca un *graf* cu X ca mulțime de noduri:

graf orientat: relație oarecare



$$R = \{(a,b), (a,c), (c,d), (d,a)\}$$

graf neorientat: relație simetrică



$$R = \{(a,b), (a,c), (a,d), (b,a), (c,a), (c,d), (d,a), (d,c)\}$$

Relații de echivalență

O relație e *de echivalență* dacă e *reflexivă*, *simetrică* și *tranzitivă*

Relația de egalitate e (evident) o relație de echivalență.

Relația de congruență modulo un număr:

$$a \equiv b \pmod{n} \text{ dacă } n \mid a - b \text{ (divide diferența)}$$

Clasa de echivalență a lui x e mulțimea elementelor aflate în relație cu x

$$\{y \mid (y, x) \in R\} \quad \text{notată } \hat{x} \text{ sau } [x]$$

O relație de echivalență pe X definește o *partiție* a lui X
(două clase de echivalență sunt fie identice, fie disjuncte)

Demonstrați!

Relații de ordine stricte și totale

O relație \prec e o *ordine strictă* dacă e *ireflexivă* și *tranzitivă*
nu există x cu $x \prec x$
dacă $x \prec y$ și $y \prec z$ atunci $x \prec z$

Exemple: relațiile $<$ și $>$ între numere (întregi, reale, etc.)
Relația “descendent” între persoane

O relație \preceq e o *ordine totală* dacă e *reflexivă*, *antisimetrică* (dacă $x \preceq y$ și $y \preceq x$ atunci $x = y$), *tranzitivă*, și în plus
oricare două elemente sunt *comparabile*, adică pentru orice x, y
avem $x \preceq y$ sau $y \preceq x$

Exemple: relațiile \leq și \geq între numere (întregi, reale, etc.)

Relații de ordine parțială

În practică apar adesea relații de ordine care nu sunt totale:

Într-un campionat, după faza pe grupe, avem un clasament în fiecare grupă dar nu și între echipe din grupe diferite

un sistem știe o ordine între mesajele recepționate, dar nu neapărat între momentele trimiterii lor

În expresia $f(x) + g(x)$, f și g se apelează *înainte* de adunare, dar nu neapărat f înainte de g

O relație e o *ordine parțială* (non-strictă), dacă e *reflexivă*, *antisimetrică* și *tranzitivă*

relația de divizibilitate între întregi

relația de incluziune \subseteq pe mulțimea părților

Orice ordine totală e și o ordine parțială (dar nu și reciproc).

Orice ordine parțială induce o ordine strictă, și reciproc:

Definim: $a \prec b$ dacă $a \preceq b$ și $a \neq b$

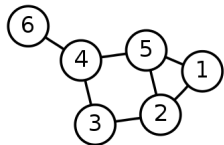
Invers, definim $a \preceq b$ dacă $a \prec b$ sau $a = b$

Noțiunea de punct fix

$x \in X$ e *punct fix* pentru funcția $f : X \rightarrow X$ dacă $f(x) = x$.
(privind f ca o transformare, ea nu îl modifică pe x)

Exemplu: fie un graf $G = (V, E)$, și pentru $X \subseteq V$ funcția
 $f(X) = X \cup \bigcup_{v \in X} \text{vecini}(v)$ (adăugăm la X toți vecinii)

$f(U) = U \Rightarrow$ din nodurile U , urmărind vecinii nu găsim noduri noi



Pornind de la $S_0 = \{6\}$ calculăm $S_1 = f(S_0) = \{6, 4\}$, $S_2 = \{6, 4, 3, 5\}$, $S_3 = \{6, 4, 3, 5, 1, 2\}$, $S_4 = S_3$. Am atins un punct fix: avem toate nodurile care pot fi atinse din 6.

Multe prelucrări repetitive pot fi definite ca transformări care se opresc când atingem un *punct fix*

care sunt toate configurațiile posibile într-un joc?

care sunt toate variabilele de care depinde o variabilă dată? etc.

Existența unui punct fix e legată de *ordini parțiale* și *lattice*.

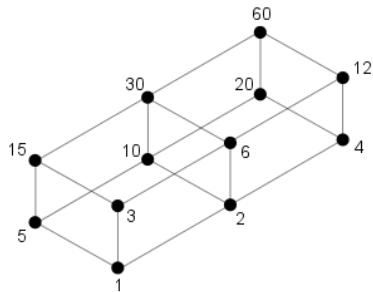
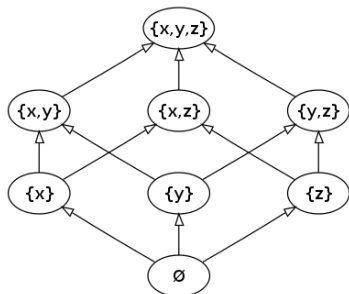
Lattice

O *latice* e o mulțime *parțial ordonată*, în care orice două elemente au un *minorant* și un *majorant*.

(elemente mai mici, respectiv mai mari în ordine decât cele două).

Ex: mulțimea părților unei mulțimi (ordine: \subseteq ; minor./maj.: \cap , \cup)

Ex: mulțimea divizorilor unui număr (c.m.m.d.c, c.m.m.m.c)



Imagine: http://en.wikipedia.org/wiki/File:Hasse_diagram_of_powerset_of_3.svg

http://en.wikipedia.org/wiki/File:Lattice_of_the_divisibility_of_60.svg

Latice complete și puncte fixe

O latice L e *completă* dacă *orice* mulțime $S \subseteq L$ are un cel mai mic majorant (supremum) și un cel mai mare minorant (infimum).

Acestea sunt condiții mai puternice decât pentru o latice oarecare:
pentru orice *submulțime* (chiar infinită, nu doar două elemente)
avem o ordine între majoranți/minoranți, și un *cel mai mic/mare*

⇒ Luând $S = L$, rezultă că L are un element minim și unul maxim
Exemplele din pagina anterioară sunt latice complete.

Teorema de punct fix (Knaster-Tarski):

Fie f o funcție monotonă pe o latice completă.

Atunci mulțimea punctelor fixe a lui f e tot o latice completă.

Corolar:

O funcție monotonă pe o latice completă are un *cel mai mic punct fix* și un *cel mai mare punct fix*.

se obțin pornind de la $0, f(0), f(f(0)), \dots$ resp. $M, f(M), f(f(M)), \dots$
unde 0 și M sunt elementul cel mai mic respectiv cel mai mare

Punctul fix în ML

Dacă există limita șirului $x, f(x), f(f(x)) \dots$ putem scrie:

```
let rec fix f x =  
  let nxt = f x in  
  if nxt = x then x else fix f nxt
```

`fix f x` compară $f(x)$ cu x . Dacă sunt egale, x e punct fix, și e returnat. Dacă nu, reapelăm recursiv cu valoarea $f(x)$. Apelul al n -lea va avea argumentul $f^{n-1}(x)$ și îl compară cu $f^n(x)$. Dacă există n cu $f^{n-1}(x) = f^n(x)$, va fi găsit, $f^{n-1}(x)$ e punct fix.

Putem rescrie, folosind o funcție ajutătoare cu doar un parametru (nu mai trebuie repetat la apel parametrul `f`):

```
let fix f =  
  let rec fix1 x =  
    let nxt = f x in  
    if nxt = x then x else fix1 nxt  
  in fix1
```

Inversa și compunerea de relații

Inversa unei relații $R \subseteq A \times B$ e relația $R^{-1} \subseteq B \times A$, cu $(y, x) \in R^{-1}$ dacă și numai dacă $(x, y) \in R$

$$R^{-1} = \{(y, x) \mid (x, y) \in R\}$$

Fie două relații $R_1 \subseteq A \times B$ și $R_2 \subseteq B \times C$.

Compunerea $R_2 \circ R_1 \subseteq A \times C$ e relația

$$R_2 \circ R_1 = \{(x, z) \mid \text{există } y \in B \mid (x, y) \in R_1 \text{ și } (y, z) \in R_2\}$$

La fel ca la funcții, scriem $R_2 \circ R_1$ și vedem că pentru $x \in A$ găsim întâi $y \in B$ și apoi $z \in C$.

Se poate vedea simplu că $(R \circ S)^{-1} = S^{-1} \circ R^{-1}$

Pentru o relație de echivalență, $R = R^{-1}$ (de ce? arătați!)

R e tranzitivă dacă și numai dacă $R \circ R \subseteq R$ (de ce? arătați!)

Pentru o relație binară $R \subseteq A \times A$, se notează $R^2 = R \circ R$, etc.

Închiderea tranzitivă a unei relații

Dintr-o relație R , putem defini o nouă relație, prin “intermediari”, ca în condiția de tranzitivitate.

Ex.: într-un graf, avem *muchii* și *drumuri* (relații între noduri):

Un drum e format din una sau mai multe muchii:

$drum(X, Y)$ dacă $muchie(X, Y)$

sau dacă $muchie(X, Z)$ și $muchie(Z, Y)$

sau dacă $muchie(X, Z)$ și $muchie(Z, U)$ și $muchie(U, Y)$...

Relația *drum* include relația *muchie* și e tranzitivă.

Sau, fie relația $copil(X, Y)$ (X e copilul lui Y):

Definim relația $desc(X, Y)$ dacă $copil(X, Y)$ (1)

descendent: $desc(X, Z)$ dacă $desc(X, Y)$ și $desc(Y, Z)$ (2)

Relația *desc* include relația *copil* (1) și e tranzitivă (2).

Închiderea tranzitivă a unei relații $R \subseteq A \times A$ e relația *tranzitivă minimală* R^+ astfel ca $R \subseteq R^+$

Putem calcula $R^+ = \bigcup_{k=1}^{\infty} R^k = R \cup R^2 \cup \dots$

$$R^2 = R \circ R$$

$$R^3 = R \circ R \circ R$$

Închiderea tranzitivă (cont.)

Un exemplu (fără cicluri):

$\text{copil}(\text{ana}, \text{ion}), \text{copil}(\text{lia}, \text{ion}), \text{copil}(\text{ion}, \text{mara}), \text{copil}(\text{mara}, \text{eva})$.

copil^2 : $\text{desc}(\text{ana}, \text{mara}), \text{desc}(\text{lia}, \text{mara}), \text{desc}(\text{ion}, \text{eva})$

copil^3 : $\text{desc}(\text{ana}, \text{eva}), \text{desc}(\text{lia}, \text{eva})$.

Nu sunt descendenți de ordin > 3 .

Deci, relația $\text{desc} = \text{copil} \cup \text{copil}^2 \cup \text{copil}^3$

Putem defini $f(X) = R \cup (X \circ R)$. Atunci $f(R) = R \cup R^2$ și prin

inducție, $f^n(R) = \bigcup_{k=1}^{n+1} R^k$.

f e monotonă: dacă $X \subseteq Y$, $X \circ R \subseteq Y \circ R$ și $f(X) \subseteq f(Y)$.

Deci f are un punct fix minimal, tocmai închiderea tranzitivă R^+ , iar pentru o mulțime finită calculul are un număr finit de pași.