Programare Logică
Cursul I

Claudia MUREȘAN
cmuresan@fmi.unibuc.ro, c.muresan@yahoo.com

Universitatea din București
Facultatea de Matematică și Informatică
București

2019–2020, Semestrul II

1. Introducere
2. Inițiere în programarea în Prolog
3. Mnemonic despre proprietățile cuantificatorilor
4. Mnemonic despre funcții
5. Constantele sunt operații fără argumente

Prescurtări uzuale care vor fi folosite în lecții

- **i. e.** (*id est*) = adică
- **a. î.** = astfel încât
- **ddacă** = dacă și numai dacă
- **ș. a. m. d.** = și așa mai departe
- —//— : să se demonstreze că
- ⨳: contradicție

Programarea imperativă:

- programatorul trebuie să–i spună calculatorului, pas cu pas, ce să facă: "parcurge cu un indice această mulțime, la fiecare iterație verifică această condiție..." etc..

Limbaje de programare imperativă: *Pascal, C, Java* etc..

Programarea logică/declarativă:

- programatorul descrie un cadru de lucru, și cere calculatorului să rezolve o cerință în acel cadru;
- pe baza unui backtracking și a altor tehnici de programare încorporate în interpretorul/compilatorul limbajului de programare logică folosit, calculatorul determină proprietățile pe care le poate folosi pentru a rezolva cerința respectivă și ordinea în care trebuie să le aplice.

Limbaje de programare logică/declarativă

- *Prolog* (PROGRAMMING IN LOGIC): bazat pe **predicate**, pe relații între obiecte; utilizat pentru jocuri/deducții logice, în procesarea limbajului natural etc.;

- *CafeObj, Maude:* bazate pe **specificații**, pe descrierea unor tipuri de structuri algebrice; destinate demonstrării automate de proprietăți matematice; utilizate și în verificarea sistemelor software;

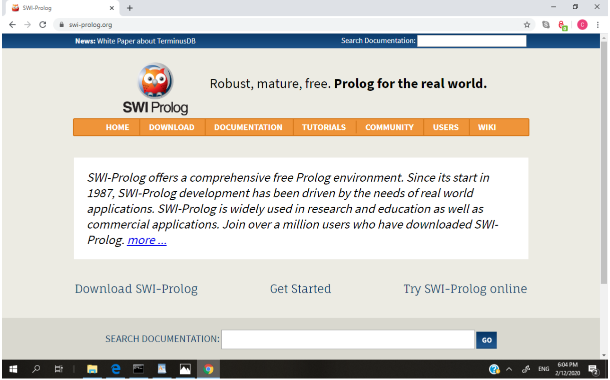
- *Haskell:* bazat pe **funcții** și **recursii**;

recursia este foarte importantă în toate limbajele de programare logică: este principalul mijloc de calcul, înlocuind, de exemplu, instrucțiunile repetitive

- etc..

Siteul SWI–Prolog

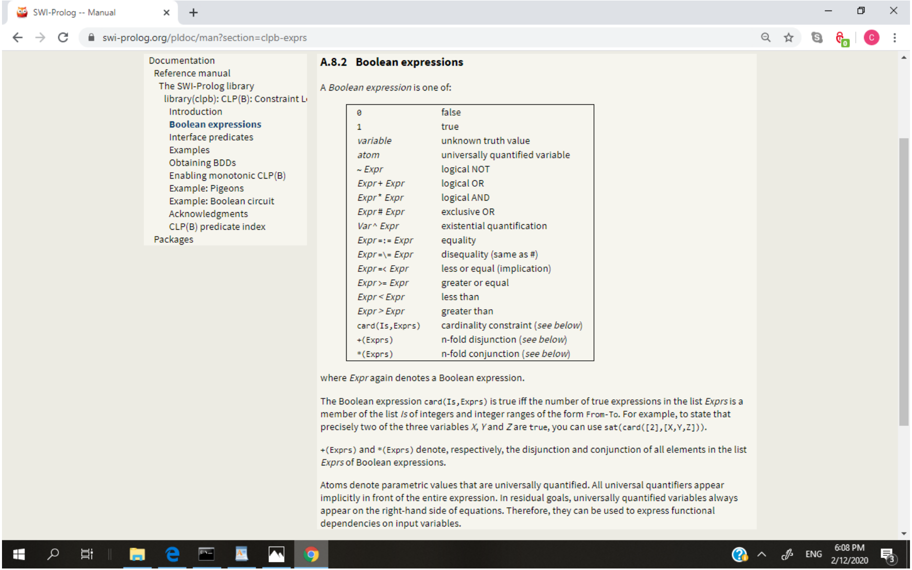
https://www.swi-prolog.org



Din josul paginii de la această adresă:

- se poate descărca SWI–Prolog;
- se poate căuta în documentația SWI–Prolog;
- se poate accesa versiunea online a SWI–Prolog.

De exemplu, dând o căutare după "boolean expressions", găsim:


SWI-Prolog -- Manual

swi-prolog.org/pldoc/man?section=clpb-exprs

A.8.2 Boolean expressions

A *Boolean expression* is one of:

`0`	false
`1`	true
variable	unknown truth value
atom	universally quantified variable
~ *Expr*	logical NOT
Expr + *Expr*	logical OR
Expr * *Expr*	logical AND
Expr # *Expr*	exclusive OR
Var ^ *Expr*	existential quantification
Expr =:= *Expr*	equality
Expr =\= *Expr*	disequality (same as #)
Expr =< *Expr*	less or equal (implication)
Expr >= *Expr*	greater or equal
Expr < *Expr*	less than
Expr > *Expr*	greater than
`card(Is,Exprs)`	cardinality constraint (*see below*)
`+(Exprs)`	n-fold disjunction (*see below*)
`*(Exprs)`	n-fold conjunction (*see below*)

where *Expr* again denotes a Boolean expression.

The Boolean expression `card(Is,Exprs)` is true iff the number of true expressions in the list *Exprs* is a member of the list *Is* of integers and integer ranges of the form `From-To`. For example, to state that precisely two of the three variables *X*, *Y* and *Z* are `true`, you can use `sat(card([2],[X,Y,Z]))`.

`+(Exprs)` and `*(Exprs)` denote, respectively, the disjunction and conjunction of all elements in the list *Exprs* of Boolean expressions.

Atoms denote parametric values that are universally quantified. All universal quantifiers appear implicitly in front of the entire expression. In residual goals, universally quantified variables always appear on the right-hand side of equations. Therefore, they can be used to express functional dependencies on input variables.

Structura unui program în Prolog

Un program în Prolog este format din **clauze**; acestea sunt de trei tipuri:

- **fapte**:

propoziție.
predicat(*listă de variabile si constante*).

acestea semnifică proprietăți intotdeauna adevărate;
predicatele exprimă relații între obiecte;
propozițiile sunt predicatele fără variabile;

- **reguli**:

fapt :- succesiune de fapte.

faptele din partea dreaptă a unei reguli pot fi separate prin virgulă (care reprezintă *conjuncția logică*) sau alți conectori logici (a se vedea slideul anterior);

(semnificația unei reguli)

are loc acest fapt :- dacă au loc aceste fapte.

- **întrebări**: formate din **scopuri**.

Numele de **variabile** în Prolog încep cu *literă mare* sau *underscore* (_).
Variabilă nedenumită: *underscore* (_):

- simbol generic pentru variabile care apar o singură dată într–un fapt sau o regulă;
- sunt folosite doar pentru a indica locații de variabile, nu și pentru a lucra cu acele variabile.

Orice alt nume, inclusiv șiruri de caractere cuprinse între apostrofuri, denumește o constantă sau un predicat.

Sintaxa pentru liste în Prolog:

$[\,]$	lista vidă
$[elem_1, elem_2, \ldots, elem_n]$	lista formată din elementele $elem_1, elem_2, \ldots, elem_n$
$[elem1, elem2, \ldots, elemn \mid T]$	lista cu primele n elemente $elem_1, elem_2, \ldots, elem_n$ și coada T

Exemplu

$$[1, 2, 3, 4, 5] = [1, 2, 3, 4, 5 \mid [\,]] = [1 \mid [2, 3, 4, 5]] = [1, 2, 3 \mid [4, 5]] = [1, 2 \mid [3 \mid [4, 5]]]$$

Exemplu

Cum putem scrie predicate pentru:

- calculul lungimii unei liste;
- concatenarea a două liste?

Să scriem un predicat *lung*(L, N), care este satisfăcut ddacă:

- L este o listă, N este un număr natural și
- N este lungimea listei L, adică numărul elementelor lui L:

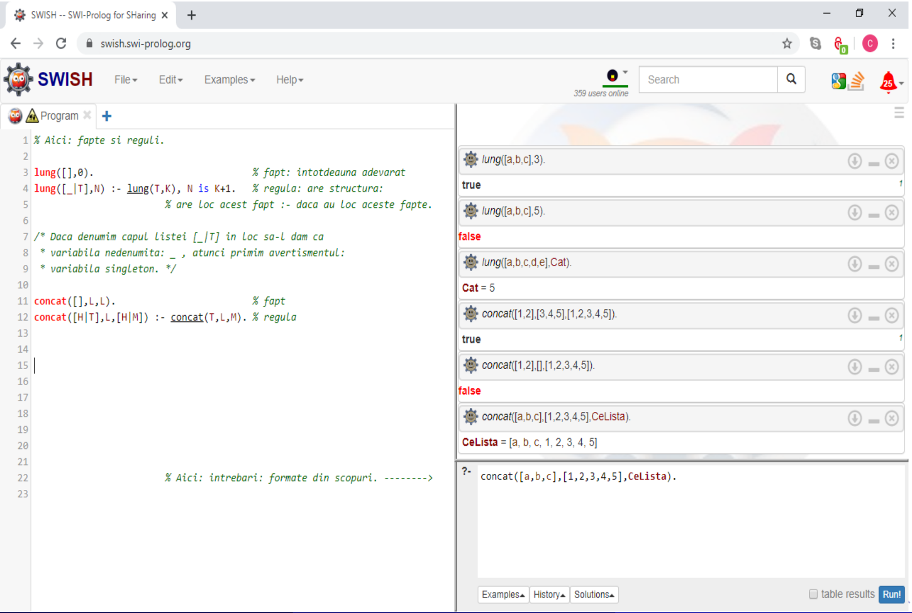
$$\mathrm{lung}([\,], 0).$$
$$\mathrm{lung}([\_|T], N)\ :-\ \mathrm{lung}(T, K),\ N \text{ is } K + 1.$$

Operatorul **is** produce executarea calculului în acea expresie aritmetică.
Înlocuiți **is** cu **=** și vedeți ce obțineți!
Și un predicat *concat*$(L1, L2, L)$, care este satisfăcut ddacă:

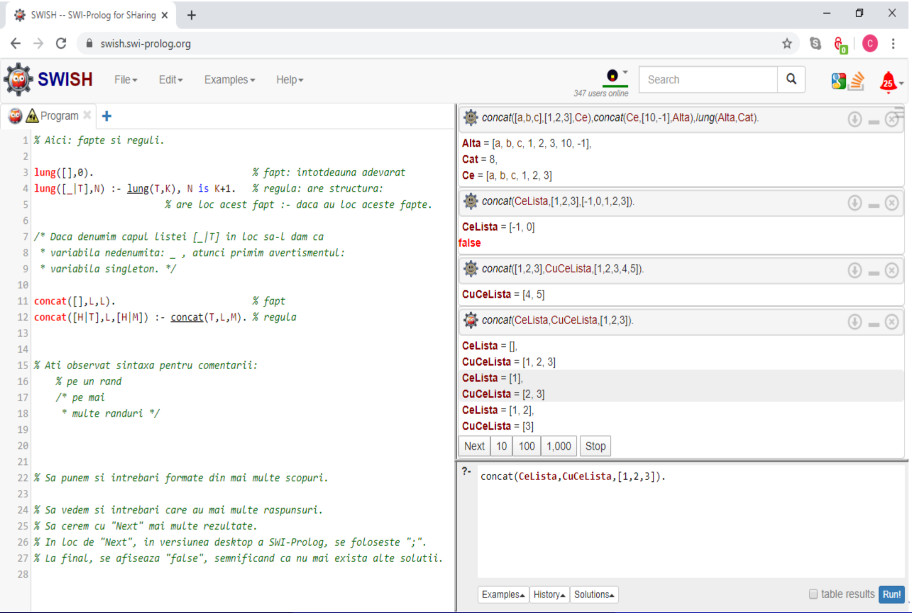
- $L1$, $L2$ și L sunt liste și
- concatenarea listei $L1$ cu $L2$ este L:

$$\mathrm{concat}([\,], L, L).$$
$$\mathrm{concat}([H|T], L, [H|M])\ :-\ \mathrm{concat}(T, L, M).$$

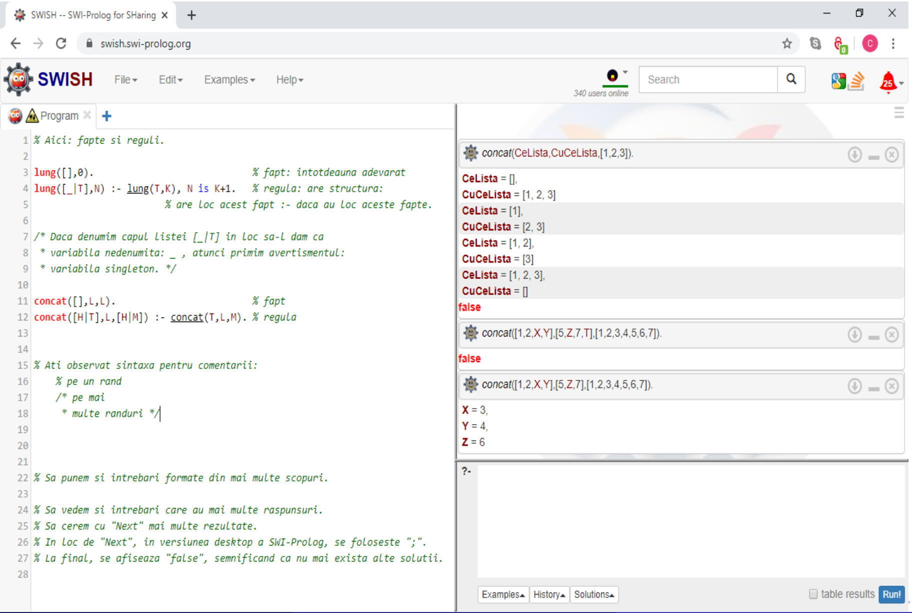
Încărcăm acest program în versiunea online a SWI–Prolog



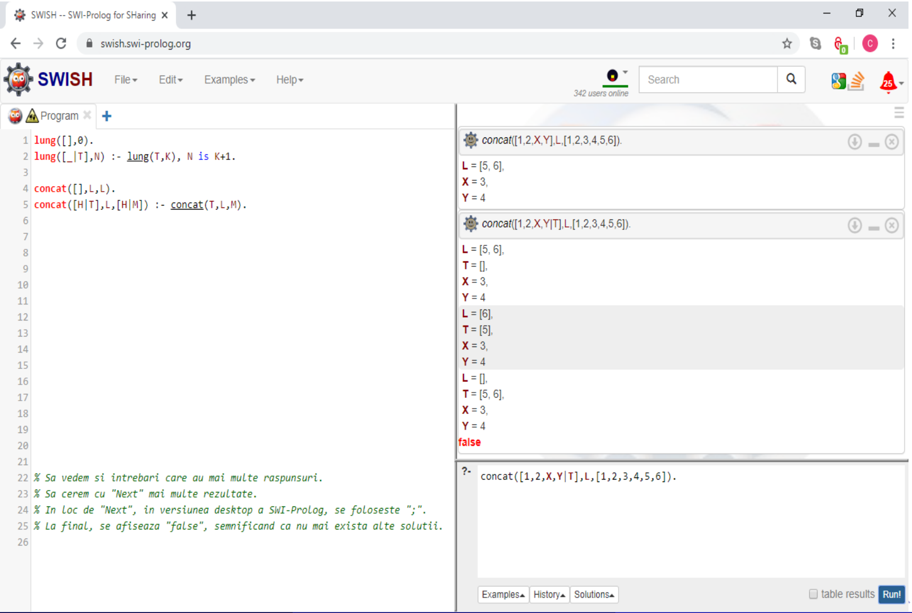
Interogări cu scopuri sau variabile multiple



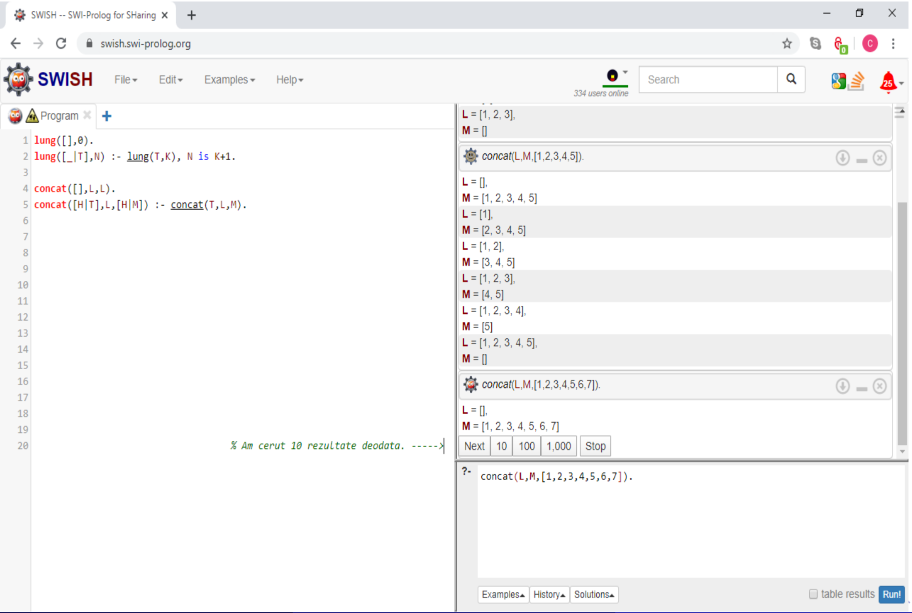
Întrebări cu mai multe răspunsuri posibile



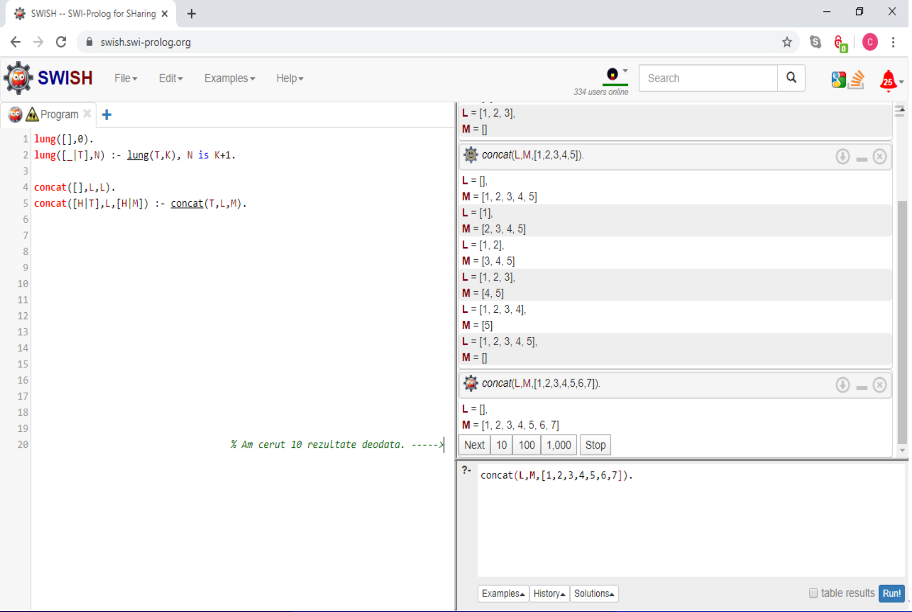
Vom vedea cum găsește Prologul aceste răspunsuri



Vedeți și celelalte opțiuni ale SWI–Prolog online

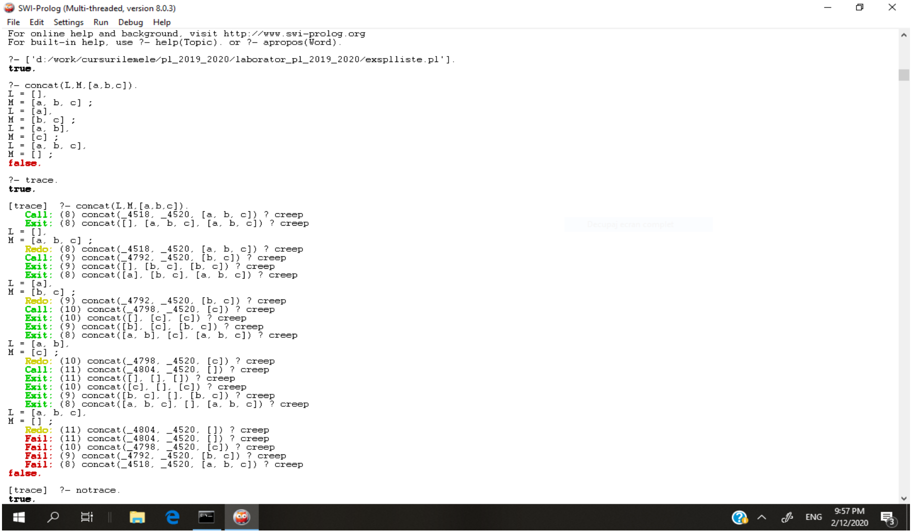


Inclusiv opțiuni pentru editare rapidă



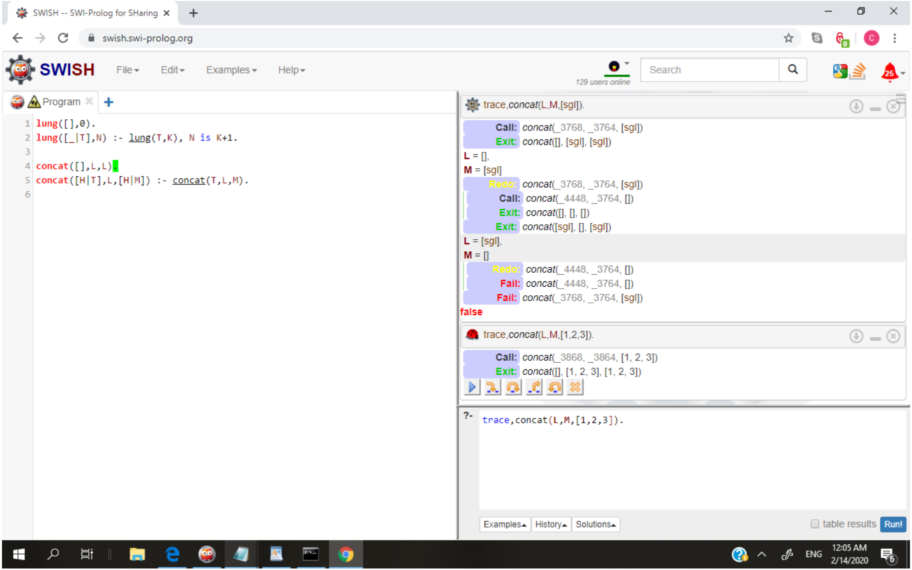
Și acum în versiunea desktop a SWI–Prolog

Am scris cele doua predicate de mai sus într–un fișier text cu extensia *.pl*:



Comanda *trace* produce detalierea pașilor urmați de Prolog pentru a rezolva interogările. Renunțare la această afișare detaliată: *notrace*.

trace și săgeată "Step into" în SWI–Prolog online



Unificare (orientativ) – detalii într–un curs următor

Două funcții **unifică** ddacă:

- acele funcții coincid
  și
- au aceleași argumente:

$$f(arg_1, \ldots, arg_n) = g(a_1, \ldots, a_k) \Longleftrightarrow \begin{cases} f = g \ (\Longrightarrow n = k) \text{ și} \\ arg_1 = a_1, \\ \vdots \\ arg_n = a_n. \end{cases}$$

Prolog–ul încearcă să unifice scopurile din interogare cu predicatele din:

- fapte;
- membrii stângi ai regulilor – dacă o astfel de unificare reușeste, atunci următorul scop este membrul drept al aceleiași reguli, cu argumentele rezultate în urma unificării.

Considerăm interogarea:

```
?- concat(L1,L2,[a,b,c]).
```

Observați din pașii afișati cu *trace* că primul lucru executat de Prolog este redenumirea variabilelor din interogare în nume de forma *_număr*.

$\mathrm{concat} \neq \mathrm{lung}$, așadar $\mathrm{concat}(\mathrm{L1}, \mathrm{L2}, [\mathrm{a}, \mathrm{b}, \mathrm{c}])$ nu unifică:

- nici cu $\mathrm{lung}([\,], 0)$,
- nici cu $\mathrm{lung}([\mathrm{H}|\mathrm{T}], \mathrm{N})$.

În schimb, concat(L1,L2,[a,b,c]) unifică:

cu $\mathrm{concat}([\,], \mathrm{L}, \mathrm{L})$, pentru $\mathbf{L1} = [\,]$ și $\mathbf{L2} = \mathbf{L} = [\mathbf{a}, \mathbf{b}, \mathbf{c}]$ – aceasta este *prima soluție* a interogării,

dar și:

cu $\mathrm{concat}([\mathrm{H}|\mathrm{T}], \mathrm{L}, [\mathrm{H}|\mathrm{M}])$, pentru $\mathbf{H} = \mathbf{a}$, $\mathbf{L1} = [\mathbf{a}|\mathbf{T}]$, $\mathbf{L2} = \mathbf{L}$ și $\mathbf{M} = [\mathbf{b}, \mathbf{c}]$,

așadar **următorul scop** este:

```
?- concat(T,L2,[b,c]).
```

Acesta unifică:

cu $\mathrm{concat}([\,],\mathrm{L},\mathrm{L})$, pentru $\mathbf{T} = [\,]$ și $\mathbf{L2} = \mathbf{L} = [\mathbf{b},\mathbf{c}]$ – așadar *a doua soluție* a interogării este: $\mathbf{L1} = [\mathbf{a}|\mathbf{T}] = [\mathbf{a}|[\,]] = [\mathbf{a}]$ și $\mathbf{L2} = [\mathbf{b},\mathbf{c}]$,

dar și:

cu $\mathrm{concat}([\mathrm{H1}|\mathrm{T1}],\mathrm{L},[\mathrm{H1}|\mathrm{M1}])$, pentru $\mathbf{H1} = \mathbf{b}$, $\mathbf{T} = [\mathbf{b}|\mathbf{T1}]$, $\mathbf{L2} = \mathbf{L}$ și $\mathbf{M1} = [\mathbf{c}]$,

așadar **următorul scop** este:

```
?- concat(T1,L2,[c]).
```

Acesta unifică:

cu $\mathrm{concat}([\,],\mathrm{L},\mathrm{L})$, pentru $\mathbf{T1} = [\,]$ și $\mathbf{L2} = \mathbf{L} = [\mathbf{c}]$ – așadar *a treia soluție* a interogării este: $\mathbf{L1} = [\mathbf{a}|\mathbf{T}] = [\mathbf{a}|[\mathbf{b}|\mathbf{T1}]] = [\mathbf{a}|[\mathbf{b}|[\,]]] = [\mathbf{a},\mathbf{b}]$ și $\mathbf{L2} = [\mathbf{c}]$,

dar și:

cu $\mathrm{concat}([\mathrm{H2}|\mathrm{T2}],\mathrm{L},[\mathrm{H2}|\mathrm{M2}])$, pentru $\mathbf{H2} = \mathbf{c}$, $\mathbf{T1} = [\mathbf{c}|\mathbf{T2}]$, $\mathbf{L2} = \mathbf{L}$ și $\mathbf{M2} = [\,]$, așadar **următorul scop** este:

```
?- concat(T2,L2,[ ]).
```

Întrucât:

[ ] nu unifică cu $[\mathrm{H3}|\mathrm{M3}]$,

avem:

$\mathrm{concat}(\mathrm{T2}, \mathrm{L2}, [\,])$ nu unifică cu $\mathrm{concat}([\mathrm{H3}|\mathrm{T3}], \mathrm{L}, [\mathrm{H3}|\mathrm{M3}])$.

Prin urmare:

$\mathrm{concat}(\mathrm{T2}, \mathrm{L2}, [\,])$ unifică numai cu $\mathrm{concat}([\,], \mathrm{L}, \mathrm{L})$, pentru $\mathbf{T2} = [\,]$ și $\mathbf{L2} = \mathbf{L} = [\,]$ – așadar *a patra* și *ultima soluție* a interogării este:
$\mathbf{L1} = [\mathbf{a}|\mathbf{T}] = [\mathbf{a}|[\mathbf{b}|\mathbf{T1}]] = [\mathbf{a}|[\mathbf{b}|[\mathbf{c}|\mathbf{T2}]]] = [\mathbf{a}|[\mathbf{b}|[\mathbf{c}|[\,]]]] = [\mathbf{a}, \mathbf{b}, \mathbf{c}]$ și $\mathbf{L2} = [\,]$.

În cazul unui scop compus, de exemplu:

```
?- concat(A,[2,3],[1,2,3]),concat(A,[4,5],C).
```

toate scopurile trebuie satisfăcute, cu **aceleași valori** pentru *variabilele comune*. Pentru exemplul de mai sus, *unica soluție*: $\mathbf{A} = [\mathbf{1}]$, $\mathbf{C} = [\mathbf{1}, \mathbf{4}, \mathbf{5}]$.

Cuantificatorii și simbolul $\exists\,!$

Cuantificatorii:

- *cuantificatorul universal:* $\forall$
- *cuantificatorul existențial:* $\exists$

Dacă x este o variabilă, iar $p(x)$ este o proprietate referitoare la x (mai precis o proprietate referitoare la elementele pe care le parcurge/le poate denumi x), atunci:

- $\text{non } [(\forall\, x)\,(p(x))] \;\Leftrightarrow\; (\exists\, x)\,(\text{non } p(x))$
- $\text{non } [(\exists\, x)\,(p(x))] \;\Leftrightarrow\; (\forall\, x)\,(\text{non } p(x))$

Notație

Alăturarea de simboluri $\exists\,!$ semnifică "există un unic", "există și este unic".

Observație

$\exists!$ nu este un cuantificator, ci este o notație prescurtată pentru enunțuri compuse: dacă x este o variabilă, iar $p(x)$ este o proprietate asupra lui x, atunci scrierea $(\exists\,!\,x)\,(p(x))$ este o abreviere pentru enunțul scris, desfășurat, astfel:

$$(\exists\, x)\,(p(x)) \text{ și } (\forall\, y)\,(\forall\, z)\,[(p(y) \text{ și } p(z)) \Rightarrow y = z],$$

unde y și z sunt variabile.

Cuantificatorii și simbolul $\exists\,!$

Cuantificatorii:

- *cuantificatorul universal:* $\forall$
- *cuantificatorul existențial:* $\exists$

Dacă x este o variabilă, iar $p(x)$ este o proprietate referitoare la x (mai precis o proprietate referitoare la elementele pe care le parcurge/le poate denumi x), atunci:

- $\text{non } [(\forall\, x)\,(p(x))] \;\Leftrightarrow\; (\exists\, x)\,(\text{non } p(x))$
- $\text{non } [(\exists\, x)\,(p(x))] \;\Leftrightarrow\; (\forall\, x)\,(\text{non } p(x))$

Notație

Alăturarea de simboluri $\exists\,!$ semnifică "există un unic", "există și este unic".

Observație

$\exists!$ nu este un cuantificator, ci este o notație prescurtată pentru enunțuri compuse: dacă x este o variabilă, iar $p(x)$ este o proprietate asupra lui x, atunci scrierea $(\exists\,!\,x)\,(p(x))$ este o abreviere pentru enunțul scris, desfășurat, astfel:

$$(\exists\, x)\,(p(x)) \text{ și } (\forall\, y)\,(\forall\, z)\,[(p(y) \text{ și } p(z)) \Rightarrow y = z],$$

unde y și z sunt variabile.

Negarea enunțurilor cuantificate

Cum se neagă un enunț cu mai mulți cuantificatori? Aplicând proprietățile de mai sus, și iterând acest procedeu:

Exemplu

Fie x, y, z, t, u variabile, iar $p(x, y, z, t, u)$ o proprietate depinzând de x, y, z, t, u. Atunci:

$$\begin{array}{l}
\text{non } [(\forall\, x)\,(\forall\, y)\,(\exists\, z)\,(\forall\, t)\,(\exists\, u)\,(p(x, y, z, t, u))] \;\Leftrightarrow \\
(\exists\, x)\,[\text{non } [(\forall\, y)\,(\exists\, z)\,(\forall\, t)\,(\exists\, u)\,(p(x, y, z, t, u))]] \;\Leftrightarrow \\
(\exists\, x)\,(\exists\, y)\,[\text{non } [(\exists\, z)\,(\forall\, t)\,(\exists\, u)\,(p(x, y, z, t, u))]] \;\Leftrightarrow \\
(\exists\, x)\,(\exists\, y)\,(\forall\, z)\,[\text{non } [(\forall\, t)\,(\exists\, u)\,(p(x, y, z, t, u))]] \;\Leftrightarrow \\
(\exists\, x)\,(\exists\, y)\,(\forall\, z)\,(\exists\, t)\,[\text{non } [(\exists\, u)\,(p(x, y, z, t, u))]] \;\Leftrightarrow \\
(\exists\, x)\,(\exists\, y)\,(\forall\, z)\,(\exists\, t)\,(\forall\, u)\,(\text{non } p(x, y, z, t, u))
\end{array}$$

Nu vom mai aplica acest procedeu pas cu pas. Reținem că procedeul constă în transformarea fiecărui cuantificator universal într–unul existențial și invers, și negarea proprietății de sub acești cuantificatori.

Cuantificatori aplicați fixând un domeniu al valorilor

Fie M o mulțime, x o variabilă, iar $p(x)$ o proprietate referitoare la elementele lui M. Atunci următoarele scrieri sunt abrevieri pentru scrierile fără domeniu al valorilor lângă cuantificatori:

- $(\forall\, x \in M)\,(p(x)) \stackrel{\text{not.}}{\Leftrightarrow} (\forall\, x)\,(x \in M \Rightarrow p(x))$
- $(\exists\, x \in M)\,(p(x)) \stackrel{\text{not.}}{\Leftrightarrow} (\exists\, x)\,(x \in M \text{ și } p(x))$

Toate proprietățile logice pentru enunțuri cuantificate, inclusiv negarea enunțurilor cuantificate de mai sus, se scriu la fel și sunt valabile și pentru cuantificatori urmați de un domeniu al valorilor pentru variabila cuantificată.

Cuantificatorii de același fel comută, cei diferiți nu

Fie x și y variabile, iar $p(x,y)$ o proprietate asupra lui x și y. Atunci:

- $(\forall\, x)\,(\forall\, y)\,(p(x,y)) \;\Leftrightarrow\; (\forall\, y)\,(\forall\, x)\,(p(x,y))$
- $(\exists\, x)\,(\exists\, y)\,(p(x,y)) \;\Leftrightarrow\; (\exists\, y)\,(\exists\, x)\,(p(x,y))$
- $(\forall\, x)\,(\exists\, y)\,(p(x,y)) \;\overset{\not\Rightarrow}{\Leftarrow}\; (\exists\, y)\,(\forall\, x)\,(p(x,y))$ (pentru fiecare valoare a lui x, valoarea lui y pentru care e satisfăcut enunțul din stânga depinde de valoarea lui x)

Analog, dacă A și B sunt mulțimi, avem:

- $(\forall\, x \in A)\,(\forall\, y \in B)\,(p(x,y)) \;\Leftrightarrow\; (\forall\, y \in B)\,(\forall\, x \in A)\,(p(x,y))$
- $(\exists\, x \in A)\,(\exists\, y \in B)\,(p(x,y)) \;\Leftrightarrow\; (\exists\, y \in B)\,(\exists\, x \in A)\,(p(x,y))$
- $(\forall\, x \in A)\,(\exists\, y \in B)\,(p(x,y)) \;\overset{\not\Rightarrow}{\Leftarrow}\; (\exists\, y \in B)\,(\forall\, x \in A)\,(p(x,y))$

Desigur, la fel pentru cazul în care doar unul dintre cuantificatori este aplicat cu un domeniu al valorilor pentru variabila cuantificată:
$(\forall\, x)\,(\forall\, y \in B)\,(p(x,y)) \;\Leftrightarrow\; (\forall\, y \in B)\,(\forall\, x)\,(p(x,y))$ etc..

Exemplu

Enunțul $(\forall\, x \in \mathbb{N})\,(\exists\, y \in \mathbb{Z})\,(x + y = 0)$ este adevărat.
Enunțul $(\exists\, y \in \mathbb{Z})\,(\forall\, x \in \mathbb{N})\,(x + y = 0)$ este fals.

Cuantificatori, disjuncții și conjuncții logice

Să observăm și următoarele proprietăți logice: dacă x este o variabilă, iar $p(x)$ și $q(x)$ sunt enunțuri referitoare la x, atunci:

- $(\forall\, x)\,(p(x) \text{ și } q(x)) \;\Leftrightarrow\; (\forall\, x)\,(p(x)) \text{ și } (\forall\, x)\,(q(x))$
- $(\exists\, x)\,(p(x) \text{ sau } q(x)) \;\Leftrightarrow\; (\exists\, x)\,(p(x)) \text{ sau } (\exists\, x)\,(q(x))$
- $(\forall\, x)\,(p(x) \text{ sau } q(x)) \;\overset{\nRightarrow}{\Leftarrow}\; (\forall\, x)\,(p(x)) \text{ sau } (\forall\, x)\,(q(x))$
- $(\exists\, x)\,(p(x) \text{ și } q(x)) \;\underset{\nLeftarrow}{\Rightarrow}\; (\exists\, x)\,(p(x)) \text{ și } (\exists\, x)\,(q(x))$

Exemplu

Enunțul $(\forall\, x \in \mathbb{N})\,(2 \mid x \text{ sau } 2 \nmid x)$ este adevărat.
Enunțul $(\forall\, x \in \mathbb{N})\,(2 \mid x)$ este fals. Enunțul $(\forall\, x \in \mathbb{N})\,(2 \nmid x)$ este tot fals. Prin urmare, enunțul $[(\forall\, x \in \mathbb{N})\,(2 \mid x) \text{ sau } (\forall\, x \in \mathbb{N})\,(2 \nmid x)]$ este fals.

Exemplu

Enunțul $(\exists\, x \in \mathbb{R})\,(x < 0 \text{ și } x \geq 10)$ este fals.
Enunțul $(\exists\, x \in \mathbb{R})\,(x < 0)$ este adevărat. Enunțul $(\exists\, x \in \mathbb{R})\,(x \geq 10)$ este tot adevărat. Prin urmare, enunțul $[(\exists\, x \in \mathbb{R})\,(x < 0) \text{ și } (\exists\, x \in \mathbb{R})\,(x \geq 10)]$ este adevărat.

Scoaterea de sub un cuantificator a unui enunț care nu depinde de variabila cuantificată

Dacă, în enunțurile compuse cuantificate de mai sus, în locul lui $q(x)$ avem un enunț q care nu depinde de x, atunci:

$$(\forall\, x)\, q \;\Leftrightarrow\; q \;\Leftrightarrow\; (\exists\, x)\, q,$$

așadar, în acest caz, din proprietățile anterioare obținem:

- $(\forall\, x)\,(p(x)$ și $q)$ $\Leftrightarrow$ $(\forall\, x)\,(p(x))$ și q
- $(\exists\, x)\,(p(x)$ sau $q)$ $\Leftrightarrow$ $(\exists\, x)\,(p(x))$ sau q
- $(\forall\, x)\,(p(x)$ sau $q)$ $\overset{\not\Rightarrow}{\Leftarrow}$ $(\forall\, x)\,(p(x))$ sau q
- $(\exists\, x)\,(p(x)$ și $q)$ $\overset{\Rightarrow}{\not\Leftarrow}$ $(\exists\, x)\,(p(x))$ și q

Mnemonic despre proprietățile cuantificatorilor

Scrieri echivalente ale enunțurilor din exemplele anterioare, fără domeniu al valorilor după cuantificatori:

$$\begin{aligned}
&(\forall\, x \in \mathbb{N})\,(2 \mid x \text{ sau } 2 \nmid x) \;\Leftrightarrow\\
&(\forall\, x)\,[x \in \mathbb{N} \;\Rightarrow\; (2 \mid x \text{ sau } 2 \nmid x)] \;\Leftrightarrow\\
&(\forall\, x)\,[x \in \mathbb{N} \;\Rightarrow\; (2 \mid x \text{ sau } 2 \nmid x)] \;\Leftrightarrow\\
&(\forall\, x)\,[(x \in \mathbb{N} \;\Rightarrow\; 2 \mid x) \text{ sau } (x \in \mathbb{N} \;\Rightarrow\; 2 \nmid x)];
\end{aligned}$$

$$\begin{aligned}
&[(\forall\, x \in \mathbb{N})\,(2 \mid x) \text{ sau } (\forall\, x \in \mathbb{N})\,(2 \nmid x)] \;\Leftrightarrow\\
&[(\forall\, x)\,(x \in \mathbb{N} \Rightarrow 2 \mid x) \text{ sau } (\forall\, x)\,(x \in \mathbb{N} \Rightarrow 2 \nmid x)];
\end{aligned}$$

$$\begin{aligned}
&(\exists\, x \in \mathbb{R})\,(x < 0 \text{ și } x \geq 10) \;\Leftrightarrow\\
&(\exists\, x)\,(x \in \mathbb{R} \text{ și } x < 0 \text{ și } x \geq 10) \;\Leftrightarrow\\
&(\exists\, x)\,[(x \in \mathbb{R} \text{ și } x < 0) \text{ și } (x \in \mathbb{R} \text{ și } x \geq 10)];
\end{aligned}$$

$$\begin{aligned}
&[(\exists\, x \in \mathbb{R})\,(x < 0) \text{ și } (\exists\, x \in \mathbb{R})\,(x \geq 10)] \;\Leftrightarrow\\
&[(\exists\, x)\,(x \in \mathbb{R} \text{ și } x < 0) \text{ și } (\exists\, x)\,(x \in \mathbb{R} \text{ și } x \geq 10)].
\end{aligned}$$

Definiția unei funcții

Definiție

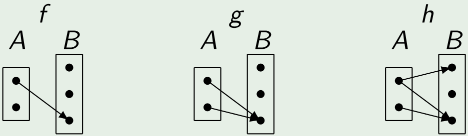
Fie A și B mulțimi oarecare. Se numește *funcție* de la A la B un triplet $f := (A, G, B)$, unde $G \subseteq A \times B$, a. î., pentru orice $a \in A$, există un unic $b \in B$, cu proprietatea că $(a, b) \in G$.
Formal: $(\forall\, a \in A)\,(\exists\,!\, b \in B)\,((a, b) \in G)$. Scris desfășurat:

$$(\forall\, a)\,[a \in A \Rightarrow [(\exists\, b)\,(b \in B \text{ și } (a, b) \in G) \text{ și}$$
$$(\forall\, c)\,(\forall\, d)\,[(c \in B \text{ și } d \in B \text{ și } (a, c) \in G \text{ și } (a, d) \in G) \Rightarrow c = d]]].$$

Faptul că f este o funcție de la A la B se notează cu $f : A \to B$ sau $A \xrightarrow{f} B$.
Mulțimea A se numește *domeniul* funcției f, B se numește *codomeniul* sau *domeniul valorilor* lui f, iar G se numește *graficul* lui f.
Pentru fiecare $a \in A$, unicul $b \in B$ cu proprietatea că $(a, b) \in G$ se notează cu $f(a)$ și se numește *valoarea funcției f în punctul a*.

Exemplu

Care dintre următoarele corespondențe este o funcție de la A la B?



Egalitatea a două funcții

Remarcă ($(a, b) \in G \;\Leftrightarrow\; f(a) = b$)

Dacă $f = (A, G, B)$ este o funcție ($f : A \to B$), atunci graficul G al lui f este mulțimea de perechi: $G = \{(a, f(a)) \mid a \in A\} \subseteq A \times B$.

Definiție

Fie $f = (A, F, B)$ și $g = (C, G, D)$ două funcții ($f : A \to B$, iar $g : C \to D$).

- Egalitatea $f = g$ semnifică egalitatea de triplete $(A, F, B) = (C, G, D)$, i. e. spunem că $f = g$ ddacă:

  $A = C$ (are loc egalitatea domeniilor),
  $B = D$ (are loc egalitatea codomeniilor) și
  $F = G$ (are loc egalitatea graficelor celor două funcții, ceea ce, conform scrierii acestor grafice din remarca anterioară, se transcrie în egalitate punctuală, adică egalitate în fiecare punct: pentru orice $a \in A = C$, $f(a) = g(a)$).

- Dacă X este o mulțime a. î. $X \subseteq A$ și $X \subseteq C$, atunci spunem că *f și g coincid pe X* ddacă f și g au aceleași valori în elementele lui X, adică: oricare ar fi $x \in X$, $f(x) = g(x) \in B \cap D$.

Există o unică funcție de la $\emptyset$ la o mulțime arbitrară

Notație (putere de mulțimi)

Pentru orice mulțimi A și B, se notează cu B^A mulțimea funcțiilor de la A la B:

$$B^A = \{f \mid f : A \to B\}$$

Remarcă (pentru orice B, $B^\emptyset$ are exact un element)

Fie B o mulțime oarecare (**poate fi vidă și poate fi nevidă**). Atunci există o unică funcție $f : \emptyset \to B$.

Într–adevăr, o funcție $f : \emptyset \to B$ trebuie să fie un triplet $f = (\emptyset, G, B)$, cu $G \subseteq \emptyset \times B = \emptyset$, deci $G = \emptyset$. Așadar, există cel mult o funcție $f : \emptyset \to B$, anume $f = (\emptyset, \emptyset, B)$ este unica posibilitate. Să arătăm că acest triplet satisface definiția funcției:

$$(\forall\, a \in \emptyset)\,(\exists\,!\, b \in B)\,((a, b) \in \emptyset), \quad \text{i. e.:}$$

$$(\forall\, a)\,[a \in \emptyset \Rightarrow (\exists\,!\, b)\,(b \in B \text{ și } (a, b) \in \emptyset)].$$

Pentru orice element a, proprietatea $a \in \emptyset$ este falsă, așadar, pentru orice element a, implicația $[a \in \emptyset \Rightarrow \ldots]$ este adevărată. Iar acest lucru înseamnă exact faptul că întreaga proprietate $(\forall a)[a \in \emptyset \Rightarrow \ldots]$ este adevărată, deci f este funcție. Prin urmare, există o unică funcție $f : \emptyset \to B$, anume $f = (\emptyset, \emptyset, B)$.

Nu există nicio funcție de la o mulțime nevidă la $\emptyset$

Remarcă (pentru orice $A \neq \emptyset$, $\emptyset^A = \emptyset$)

Fie A o mulțime **nevidă** (i. e. $A \neq \emptyset$). Atunci nu există nicio funcție $f : A \to \emptyset$. Într–adevăr, o funcție $f : A \to \emptyset$ trebuie să fie un triplet $f = (A, G, \emptyset)$, cu $G \subseteq A \times \emptyset = \emptyset$, deci $G = \emptyset$. Așadar, dacă ar exista o funcție $f : A \to \emptyset$, atunci am avea neapărat $f = (A, \emptyset, \emptyset)$. Să vedem dacă acest triplet verifică definiția funcției:

$$(\forall\, a \in A)\, (\exists\, !\, b \in \emptyset)\, ((a, b) \in \emptyset), \quad \text{i. e.:}$$

$$(\forall\, a)\, [a \in A \Rightarrow [(\exists\, b)\, (b \in \emptyset \text{ și } (a, b) \in \emptyset) \text{ și}$$

$$(\forall\, c)\, (\forall\, d)\, [(c \in \emptyset \text{ și } d \in \emptyset \text{ și } (a, c) \in \emptyset \text{ și } (a, d) \in \emptyset) \Rightarrow c = d]]].$$

Oricare ar fi elementul b, proprietatea $b \in \emptyset$ este falsă, deci, oricare ar fi elementele a și b, conjuncția ($b \in \emptyset$ și $(a, b) \in \emptyset$) este falsă, deci, oricare ar fi elementul a, proprietatea $(\exists\, b)\, (b \in \emptyset$ și $(a, b) \in \emptyset)$ este falsă, așadar, oricare ar fi elementul a, conjuncția care succede mai sus implicației având ca antecedent pe $a \in A$ este falsă. În schimb, întrucât A este nevidă, rezultă că proprietatea $a \in A$ este adevărată pentru măcar un element a. Prin urmare, implicația $[a \in A \Rightarrow \ldots]$ de mai sus este falsă pentru cel puțin un element a, ceea ce înseamnă că întreaga proprietate $(\forall\, a)\, [a \in A \Rightarrow \ldots]$ este falsă, și deci f nu este funcție.

Imaginea și preimaginea printr–o funcție

Definiție

Pentru orice mulțimi A și B, orice funcție $f : A \to B$ și orice submulțimi $X \subseteq A$ și $Y \subseteq B$, se definesc:

- *imaginea lui X prin f* sau *imaginea directă a lui X prin f*, notată $f(X)$, este submulțimea lui B:

$$f(X) = \{f(x) \mid x \in X\} \subseteq B$$

- $f(A)$ se mai notează cu $Im(f)$ și se numește *imaginea lui f*:

$$Im(f) = f(A) = \{f(a) \mid a \in A\} \subseteq B$$

- *preimaginea lui Y prin f* sau *imaginea inversă a lui Y prin f*, notată $f^{-1}(Y)$ ($f^*(Y)$ în unele cărți, pentru a o deosebi de imaginea lui Y prin inversa f^{-1} a lui f, care există numai atunci când f este inversabilă, deci numai atunci când f este bijectivă – a se vedea în cele ce urmează –, pe când preimaginea unei submulțimi a codomeniului poate fi definită pentru orice funcție), este submulțimea lui A:

$$f^{-1}(Y) = \{x \in A \mid f(x) \in Y\} \subseteq A$$

Funcții injective, surjective, bijective

Definiție

Fie A și B mulțimi și $f : A \to B$ o funcție. f se zice:

- *injectivă* ddacă are loc oricare dintre următoarele condiții echivalente:
  - pentru orice $b \in B$, există cel mult un $a \in A$, astfel încât $f(a) = b$
  - pentru orice $a_1, a_2 \in A$, dacă $a_1 \neq a_2$, atunci $f(a_1) \neq f(a_2)$
  - pentru orice $a_1, a_2 \in A$, dacă $f(a_1) = f(a_2)$, atunci $a_1 = a_2$
- *surjectivă* ddacă are loc oricare dintre următoarele condiții echivalente:
  - pentru orice $b \in B$, există (cel puțin un) $a \in A$, astfel încât $f(a) = b$ (formal: $(\forall\, b \in B)\,(\exists\, a \in A)\,(f(a) = b)$)
  - $f(A) = B$
- *bijectivă* ddacă are loc oricare dintre următoarele condiții echivalente:
  - f este simultan injectivă și surjectivă
  - pentru orice $b \in B$, există exact un $a \in A$, astfel încât $f(a) = b$ (formal: $(\forall\, b \in B)\,(\exists\,!\, a \in A)\,(f(a) = b)$)

Funcțiile injective, surjective, respectiv bijective se mai numesc *injecții*, *surjecții*, respectiv *bijecții*.

Familii arbitrare de elemente, familii arbitrare de mulțimi

- Ce este un șir de numere reale indexat de $\mathbb{N}$? Un *șir* $(x_n)_{n\in\mathbb{N}} \subset \mathbb{R}$ este o funcție $f : \mathbb{N} \to \mathbb{R}$. Pentru orice $n \in \mathbb{N}$, se notează $x_n := f(n) \in \mathbb{R}$.
- Ce este o familie arbitrară de numere reale? Fie I o mulțime arbitrară. Ce este o familie de numere reale indexată de I? O *familie* $(x_i)_{i\in I} \subseteq \mathbb{R}$ este o funcție $f : I \to \mathbb{R}$. Pentru orice $i \in I$, se notează $x_i := f(i) \in \mathbb{R}$. Elementele mulțimii I se numesc *indicii* familiei $(x_i)_{i\in I}$.

Dată o mulțime arbitrară A:

- ce este un șir de elemente ale lui A indexat de $\mathbb{N}$?
- ce este o familie arbitrară de elemente ale lui A?

Înlocuind mai sus pe $\mathbb{R}$ cu A, se obțin definițiile acestor noțiuni. Să reținem:

Definiție (familie de elemente ale lui A indexată de I: $(x_i)_{i\in I} \subseteq A$)

Fie I și A mulțimi arbitrare.
O *familie de elemente ale lui A indexată de I* este o funcție $f : I \to A$. Pentru orice $i \in I$, se notează $x_i := f(i) \in A$, astfel că familia f se mai notează sub forma $(x_i)_{i\in I} \subseteq A$.
Elementele mulțimii I se numesc *indicii* familiei $(x_i)_{i\in I}$.

- Ce este un șir de mulțimi indexat de $\mathbb{N}$?
- Ce este o familie arbitrară de mulțimi?

Egalitatea între familii de elemente

- Există o singură familie vidă de elemente ale unei mulțimi arbitrare A, pentru că există o singură funcție de la $\emptyset$ la A (a se vedea și mai jos).
- Familia vidă nu este egală cu nicio familie nevidă, ci este egală doar cu ea însăși.

Fie I și A două mulțimi nevide, iar $(a_i)_{i \in I}$ și $(b_i)_{i \in I}$ două familii de elemente din A indexate de I:

- $I \neq \emptyset$, $A \neq \emptyset$
- $(a_i)_{i \in I} \subseteq A$, i. e., pentru orice $i \in I$, $a_i \in A$
- $(b_i)_{i \in I} \subseteq A$, i. e., pentru orice $i \in I$, $b_i \in A$

Conform celor menționate anterior, familiile de elemente din A indexate de I sunt, prin definiție, funcții de la I la A, iar notația lor ca mai sus se adoptă pentru comoditate:

- $(a_i)_{i \in I} = f : I \to A$, unde, pentru orice $i \in I$, $f(i) = a_i$
- $(b_i)_{i \in I} = g : I \to A$, unde, pentru orice $i \in I$, $g(i) = b_i$

Egalitatea între familii de elemente

Egalitatea de funcții semnifică egalitatea domeniilor și a codomeniilor (valabilă pentru f și g, pentru că au ambele domeniul I și codomeniul A) și *egalitatea punctuală*, adică egalitatea în fiecare punct: două funcții cu același domeniu și același codomeniu sunt egale ddacă sunt egale în fiecare punct al domeniului lor comun:

$$f = g \quad \text{ddacă}, \quad \text{pentru orice } i \in I, \ f(i) = g(i).$$

Deci ce semnifică egalitatea a două familii de elemente din aceeași mulțime indexate de aceeași mulțime? *Egalitatea pe componente:* cele două familii sunt egale ddacă sunt egale pe componente:

$$(a_i)_{i \in I} = (b_i)_{i \in I} \quad \text{ddacă}, \quad \text{pentru orice } i \in I, \ a_i = b_i.$$

- **Caz particular:** cazul finit nevid: $I = \overline{1, n}$, cu n natural nenul:

$$(a_1, a_2, \ldots, a_n) = (b_1, b_2, \ldots, b_n) \quad \text{ddacă} \quad \begin{cases} a_1 = b_1, \\ a_2 = b_2, \\ \quad \vdots \\ a_n = b_n. \end{cases}$$

Familii arbitrare de mulțimi

Definiție

Fie T o mulțime arbitrară. Se numește *șir de submulțimi ale lui T indexat de $\mathbb{N}$* o funcție $f : \mathbb{N} \rightarrow \mathcal{P}(T)$. Pentru fiecare $n \in \mathbb{N}$, se notează $A_n := f(n) \in \mathcal{P}(T)$, iar șirul de submulțimi ale lui T se notează cu $(A_n)_{n\in\mathbb{N}}$. Scriem $(A_n)_{n\in\mathbb{N}} \subseteq \mathcal{P}(T)$ cu semnificația că, pentru fiecare $n \in \mathbb{N}$, $A_n \in \mathcal{P}(T)$.

Definiție

Fie T și I două mulțimi arbitrare. Se numește *familie de submulțimi ale lui T indexată de I* o funcție $f : I \rightarrow \mathcal{P}(T)$. Pentru fiecare $i \in I$, se notează $A_i := f(i) \in \mathcal{P}(T)$, iar familia de submulțimi ale lui T se notează cu $(A_i)_{i\in I}$. Scriem $(A_i)_{i\in I} \subseteq \mathcal{P}(T)$ cu semnificația că, pentru fiecare $i \in I$, $A_i \in \mathcal{P}(T)$. Elementele mulțimii I se numesc *indicii* familiei $(A_i)_{i\in I}$.

Putem generaliza definițiile anterioare la șiruri de mulțimi oarecare și familii de mulțimi oarecare, nu neapărat părți ale unei mulțimi precizate, dar vom avea nevoie de acea definiție mai cuprinzătoare a noțiunii de funcție, care permite unei funcții f definite pe $\mathbb{N}$, respectiv pe I, să aibă drept codomeniu o clasă (nu neapărat o mulțime), anume clasa tuturor mulțimilor în acest caz.

Operații cu familii arbitrare de mulțimi

Definiție

Fie I o mulțime arbitrară și $(A_i)_{i\in I}$ o familie de mulțimi (părți ale unei mulțimi T sau mulțimi arbitrare) indexată de I.
Se definesc următoarele operații:

- *reuniunea familiei* $(A_i)_{i\in I}$ este mulțimea notată $\bigcup_{i\in I} A_i$ și definită prin:

$$\bigcup_{i\in I} A_i = \{x \mid (\exists\, i \in I)\,(x \in A_i)\} = \{x \mid (\exists\, i)\,(i \in I \text{ și } x \in A_i)\}$$

- *intersecția familiei* $(A_i)_{i\in I}$ este mulțimea notată $\bigcap_{i\in I} A_i$ și definită prin:

$$\bigcap_{i\in I} A_i = \{x \mid (\forall\, i \in I)\,(x \in A_i)\} = \{x \mid (\forall\, i)\,(i \in I \Rightarrow x \in A_i)\}$$

Operații cu familii arbitrare de mulțimi

Definiție (continuare)

- *produsul cartezian al familiei* $(A_i)_{i \in I}$ (numit și *produsul direct al familiei* $(A_i)_{i \in I}$) este mulțimea notată $\prod_{i \in I} A_i$ și definită prin:

$$\prod_{i \in I} A_i = \{(a_i)_{i \in I} \subseteq \bigcup_{i \in I} A_i \mid (\forall\, i \in I)\,(a_i \in A_i)\} =$$

$$= \{(a_i)_{i \in I} \subseteq \bigcup_{i \in I} A_i \mid (\forall\, i)\,(i \in I \Rightarrow a_i \in A_i)\},$$

sau, altfel scris (cu definiția unei familii de elemente exemplificate mai sus pe familii de numere reale):

$$\prod_{i \in I} A_i = \{f \mid f : I \to \bigcup_{i \in I} A_i, (\forall\, i \in I)\,(f(i) \in A_i)\} =$$

$$= \{f \mid f : I \to \bigcup_{i \in I} A_i, (\forall\, i)\,(i \in I \Rightarrow f(i) \in A_i)\}.$$

Notație

Fie $n \in \mathbb{N}^*$ și mulțimile $A_1, A_2, \ldots, A_n$. Produsul direct $\prod_{i \in \overline{1,n}} A_i$ se mai notează cu $\prod_{i=1}^{n} A_i$, iar un element $(a_i)_{i \in \overline{1,n}}$ al acestui produs direct se mai notează cu $(a_1, a_2, \ldots, a_n)$. În cazul particular în care $A_1 = A_2 = \ldots = A_n = A$, $\prod_{i \in \overline{1,n}} A \overset{\text{notație}}{=} \prod_{i=1}^{n} A \overset{\text{notație}}{=} A^n$.

Remarcă (puterile unei mulțimi: caz particular al produsului direct)

În definiția anterioară, dacă $I \neq \emptyset$ și, pentru orice $i \in I$, $A_i = A$, atunci $\bigcup_{i \in I} A_i = \bigcup_{i \in I} A = A$, așadar $\prod_{i \in I} A = \{f \mid f : I \to A, (\forall\, i \in I)\,(f(i) \in A)\} = \{f \mid f : I \to A\} = A^I$. În particular, pentru orice $n \in \mathbb{N}^*$, dacă $|I| = n$, avem: $A^n = \{(a_1, \ldots, a_n) \mid (\forall\, i \in \overline{1,n})\,(a_i \in A)\} = \{f \mid f : \overline{1,n} \to A\} = A^{\overline{1,n}} = A^I$.

Pentru orice mulțimi A, B, notăm cu $A \cong B$ faptul că există o bijecție între A și B.

Remarcă

Pentru orice mulțimi A, I și J, dacă $I \cong J$, atunci $A^I \cong A^J$.

Operații de diferite arități

Aritatea unei operații a unei structuri algebrice (cu o singură *mulțime suport*, o singură mulțime de elemente) este **numărul argumentelor (operanzilor, variabilelor) acelei operații**. Dacă structura algebrică are mulțimea suport A, iar f este o operație n–ară pe A, cu $n \in \mathbb{N}$, atunci $f : \underbrace{A \times A \times \ldots \times A}_{n \text{ de } A} \to A$ (f este o funcție cu n argumente din A, cu valori tot în A).

Exemplu

Într–un grup $(G, \circ, ^{-1}, e)$, avem trei operații:

- operația *binară* $\circ$ (**compunerea elementelor grupului, două câte două**) (operație de aritate 2, operație cu două argumente): $\circ : G \times G \to G$
- operația *unară* $^{-1}$ (**inversarea fiecărui element din grup**) (operație de aritate 1, operație cu un singur argument): $^{-1} : G \to G$
- operația *zeroară* (sau *nulară*) e (**elementul neutru al grupului**) (operație de aritate 0, operație fără argumente, i. e. **constantă** din G): $e \in G$

Operațiile zeroare sunt elemente distinse, i. e. constante

De ce operațiile zeroare sunt același lucru cu **elemente distinse, constante** din mulțimea suport a structurii algebrice?

Urmând regula de mai sus, elementul neutru e al grupului G trebuie să fie o funcție de la produsul direct al familiei vide la G.

Produsul direct al familiei vide nu este $\emptyset$, cum s–ar putea crede, ci este un *singleton*, adică o mulțime cu un singur element.

Într–adevăr, dacă recitim de mai sus definiția produsului direct al unei familii arbitrare de mulțimi, observăm că produsul direct al familiei vide este mulțimea funcțiilor de la mulțimea vidă la reuniunea familiei vide, care satisfac o proprietate întotdeauna adevărată (anume $(\forall\, i)\,(i \in \emptyset \Rightarrow \ldots)$, iar $i \in \emptyset$ este fals pentru orice i, deci implicația anterioară este adevărată pentru orice i, deci această proprietate cu variabila i cuantificată universal este adevărată), adică produsul direct al familiei vide este mulțimea funcțiilor de la mulțimea vidă la reuniunea familiei vide, care are un singur element, pentru că de la $\emptyset$ la orice mulțime există o unică funcție, așa cum am văzut mai sus.

Remarcă (reuniunea familiei vide este vidă)

Dacă recitim și definiția reuniunii unei familii arbitrare, observăm că reuniunea familiei vide este mulțimea elementelor pentru care există un $i \in \emptyset$ cu o anumită proprietate, condiție care este întotdeauna falsă, deci reuniunea familiei vide este $\emptyset$.

Operațiile zeroare sunt elemente distinse, i. e. constante

Remarcă (produsul direct al familiei vide este un singleton)

Cele de mai sus arată că produsul direct al familiei vide este mulțimea cu unicul element dat de unica funcție de la $\emptyset$ la $\emptyset$, anume $(\emptyset, \emptyset, \emptyset)$, adică produsul direct al familiei vide este singletonul $\{(\emptyset, \emptyset, \emptyset)\}$.

Prin urmare, elementul neutru al grupului G este o funcție φ de la singletonul $G^{\emptyset} = G^0 = \{(\emptyset, \emptyset, \emptyset)\}$ la G:

$$\varphi : G^{\emptyset} = G^0 = \{(\emptyset, \emptyset, \emptyset)\} \to G,$$

iar o funcție definită pe un singleton are o singură valoare ($\varphi((\emptyset, \emptyset, \emptyset)) \in G$), i.e. are imaginea tot un singleton:

$$\varphi(G^{\emptyset}) = \varphi(G^0) = \{\varphi((\emptyset, \emptyset, \emptyset))\} \subseteq G,$$

conținând acea unică valoare, deci poate fi identificată cu această unică valoare a ei, care este un element distins, o constantă din G:

$$\varphi((\emptyset, \emptyset, \emptyset)) = e \in G,$$

și identificăm:

$$\varphi = e.$$

Exemplu (structură algebrică având mai multe mulțimi suport, i. e. mai multe mulțimi (tipuri) de elemente)

Ca o paranteză, o **structură algebrică cu două mulțimi suport** este **spațiul vectorial,** care are ca mulțimi suport **mulțimea scalarilor** (formând un corp, K), și **mulțimea vectorilor** (formând un grup abelian, V). Nu este același lucru cu o structură algebrică având ca unică mulțime suport pe $K \times V$, pentru că nu avem operații pe $K \times V$ (i. e. operații de la $K \times V \times K \times V \times \ldots \times K \times V$ la $K \times V$), ci avem operații de grup pe V, operații de corp pe K și operația de compunere (înmulțire) a scalarilor cu vectorii, de la $K \times V$ la V.

Observație (cu ce unifică o constantă, i.e. o operație zeroară (operație fără argumente, fără operanzi))

Dacă revedeți discuția informală de mai sus despre **unificare**, puteți observa că o constantă **nu unifică**:

- nici cu o altă constantă,
- nici cu o expresie având ca operator dominant o funcție (i. e. operație) cu argumente (i.e. operanzi), i.e. de aritate nenulă.

O constantă **unifică** numai cu **ea însăși** sau cu o **variabilă**.