

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

1 Introducere

2 Inițiere în Programarea în Prolog

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

Paradigme ale programării calculatoarelor

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/compilerul 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

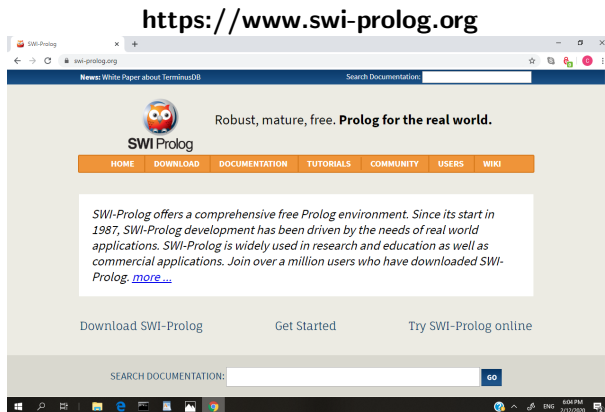
- *Prolog* (PROGRAMMING IN LOGIC): bazat pe **predicate**, pe relații între obiecte; utilizat pentru jocuri/deducții logice, în procesarea limbajului natural etc.;
- *CafeObj*, *Maude*: bazate pe **specificatii**, pe descrierea unor tipuri de structuri algebrice; destinate demonstrării automate de proprietăți 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..

1 Introducere

2 Inițiere în Programarea în Prolog



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:

The screenshot shows a web browser window with the address bar displaying `swi-prolog.org/pldoc/man?section=clpb-exprs`. The page title is "SWI-Prolog -- Manual". On the left, there is a navigation menu with the following items: Documentation, Reference manual, The SWI-Prolog library, library(clpb): CLP(B): Constraint L, Introduction, **Boolean expressions**, Interface predicates, Examples, Obtaining BDDs, Enabling monotonic CLP(B), Example: Pigeons, Example: Boolean circuit, Acknowledgments, CLP(B) predicate index, and Packages. The main content area is titled "A.8.2 Boolean expressions" and contains the text: "A Boolean expression is one of:". Below this text is a table listing various Boolean expressions and their meanings. The table is as follows:

| | |
|-------------------|------------------------------------|
| 0 | false |
| 1 | true |
| <i>variable</i> | unknown truth value |
| <i>atom</i> | universally quantified variable |
| $\sim Expr$ | logical NOT |
| $Expr + Expr$ | logical OR |
| $Expr * Expr$ | logical AND |
| $Expr \# Expr$ | exclusive OR |
| $Var \wedge Expr$ | existential quantification |
| $Expr = Expr$ | equality |
| $Expr \neq Expr$ | 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) |
| $\vee(Exprs)$ | n-fold disjunction (see below) |
| $\wedge(Exprs)$ | 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]))`.

$\vee(Exprs)$ and $\wedge(Exprs)$ 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 întotdeauna 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 *conjunția 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:

| | |
|--|---|
| <code>[]</code> | lista vidă |
| <code>[elem₁, elem₂, ..., elem_n]</code> | lista formată din elementele <i>elem₁</i> , <i>elem₂</i> , ..., <i>elem_n</i> |
| <code>[elem₁, elem₂, ..., elem_n T]</code> | lista cu primele <i>n</i> elemente <i>elem₁</i> , <i>elem₂</i> , ..., <i>elem_n</i> și coada <i>T</i> |

Exemplu

$$[1, 2, 3, 4, 5] = [1, 2, 3, 4, 5 | []] = [1 | [2, 3, 4, 5]] = [1, 2, 3 | [4, 5]] = [1, 2 | [3 | [4, 5]]]$$

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 :

$lung([], 0)$.

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

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 :

$concat([], L, L)$.

$concat([H|T], L, [H|M]) :- concat(T, L, M)$.

Încărcăm acest program în versiunea online a SWI-Prolog

The screenshot displays the SWISH web interface for running Prolog code. The browser address bar shows 'swish.swi-prolog.org'. The SWISH application has a menu bar with 'File', 'Edit', 'Examples', and 'Help'. A search bar and a '359 users online' indicator are also present.

The main editor area contains a Prolog program with the following code:

```
1 % Aici: fapte si reguli.
2
3 lung([],0).                % fapt: intotdeauna adevarat
4 lung([_|T],N) :- lung(T,K), N is K+1. % regula: are structura:
5                               % are loc acest fapt :- daca au loc aceste fapte.
6
7 /* Daca denumim capul listei [_|T] in loc sa-l dam ca
8 * variabila nedenumita: _, atunci primim avertismentul:
9 * variabila singleton. */
10
11 concat([],L,L).            % fapt
12 concat([H|T],L,[H|M]) :- concat(T,L,M). % regula
13
14
15
16
17
18
19
20
21
22 % Aici: intrebari: formate din scopuri. ----->
23
```

The right-hand pane shows the execution results of several queries:

- `lung([a,b,c],3).` returns `true`.
- `lung([a,b,c],5).` returns `false`.
- `lung([a,b,c,d,e],Cat).` returns `Cat = 5`.
- `concat([1,2],[3,4,5],[1,2,3,4,5]).` returns `true`.
- `concat([1,2],[],[1,2,3,4,5]).` returns `false`.
- `concat([a,b,c],[1,2,3,4,5],CeLista).` returns `CeLista = [a, b, c, 1, 2, 3, 4, 5]`.

The bottom of the interface includes buttons for 'Examples', 'History', 'Solutions', and 'Run!', along with a checkbox for 'table results'.

Interogări cu scopuri sau variabile multiple

SWISH -- SWI-Prolog for SHaring

← → ↻ 🔒 swish.swi-prolog.org

SWISH

File Edit Examples Help

347 users online

Search 🔍

🔔 25

Program

```
1 % Aici: fapte si reguli.
2
3 lung([],0). % fapt: intotdeauna adevarat
4 lung([_:T],N) :- lung(T,K), N is K+1. % regula: are structura:
5 % are loc acest fapt :- daca au loc aceste fapte.
6
7 /* Daca denumim capul listei [_:T] in loc sa-l dam ca
8 * variabila nedenumita: _, atunci primim avertismentul:
9 * variabila singleton. */
10
11 concat([],L,L). % fapt
12 concat([H:T],L,[H:M]) :- concat(T,L,M). % regula
13
14
15 % Ati observat sintaxa pentru comentarii:
16 % pe un rand
17 /* pe mai
18 * multe randuri */
19
20
21
22 % Sa punem si intrebari formate din mai multe scopuri.
23
24 % Sa vedem si intrebari care au mai multe raspunsuri.
25 % Sa cerem cu "Next" mai multe rezultate.
26 % In loc de "Next", in versiunea desktop a SWI-Prolog, se foloseste ";".
27 % La final, se afiseaza "false", semnificand ca nu mai exista alte solutii.
28
```

concat([a,b,c],[1,2,3],Ce),concat(Ce,[10,-1],Alta),lung(Alta,Cat).

Alta = [a, b, c, 1, 2, 3, 10, -1].
Cat = 8.
Ce = [a, b, c, 1, 2, 3]

concat(CeLista,[1,2,3],[-1,0,1,2,3]).

CeLista = [-1, 0]
false

concat([1,2,3],CuCeLista,[1,2,3,4,5]).

CuCeLista = [4, 5]

concat(CeLista,CuCeLista,[1,2,3]).

CeLista = [].
CuCeLista = [1, 2, 3]
CeLista = [1].
CuCeLista = [2, 3]
CeLista = [1, 2].
CuCeLista = [3]

Next 10 100 1,000 Stop

?- concat(CeLista,CuCeLista,[1,2,3]).

Examples History Solutions

table results Run

Întrebări cu mai multe răspunsuri posibile

The screenshot displays the SWISH web interface, a tool for running Prolog code. The browser window shows the URL `swish.swi-prolog.org`. The SWISH application has a menu bar with `File`, `Edit`, `Examples`, and `Help`. A search bar and a status indicator for 340 users online are also present.

The main editor area contains a Prolog program with the following code:

```
1 % Aici: fapte si reguli.
2
3 lung([],0). % fapt: intotdeauna adevarat
4 lung([_:T],N) :- lung(T,K), N is K+1. % regula: are structura:
5 % are loc acest fapt :- daca au loc aceste fapte.
6
7 /* Daca denumim capul listei [_:T] in loc sa-l dam ca
8 * variabila nedenumita: _, atunci primim avertismentul:
9 * variabila singleton. */
10
11 concat([],L,L). % fapt
12 concat([H:T],L,[H:M]) :- concat(T,L,M). % regula
13
14
15 % Ati observat sintaxa pentru comentarii:
16 % pe un rand
17 /* pe mai
18 * multe randuri */
19
20
21
22 % Sa punem si intrebari formate din mai multe scopuri.
23
24 % Sa vedem si intrebari care au mai multe raspunsuri.
25 % Sa cerem cu "Next" mai multe rezultate.
26 % In loc de "Next", in versiunea desktop a SWI-Prolog, se foloseste ";".
27 % La final, se afiseaza "false", semnificand ca nu mai exista alte solutii.
28
```

The right-hand pane shows the execution results for three queries:

```
concat(CeLista,CuCeLista,[1,2,3]).
CeLista = [],
CuCeLista = [1, 2, 3]
CeLista = [1],
CuCeLista = [2, 3]
CeLista = [1, 2],
CuCeLista = [3]
CeLista = [1, 2, 3],
CuCeLista = []
false

concat([1,2,X;Y],[5,Z,7;T],[1,2,3,4,5,6,7]).
false

concat([1,2,X;Y],[5,Z,7],[1,2,3,4,5,6,7]).
X = 3,
Y = 4,
Z = 6
```

At the bottom of the interface, there are buttons for `Examples`, `History`, `Solutions`, and `Run!`, along with a checkbox for `table results`.

Vom vedea cum găsește Prologul aceste răspunsuri

The screenshot shows the SWISH web interface for SWI-Prolog. The browser address bar shows `swish.swi-prolog.org`. The SWISH logo and navigation menu (File, Edit, Examples, Help) are at the top. The main editor displays a Prolog program with the following code:

```
1 lung([],0).
2 lung([_|T],N) :- lung(T,K), N is K+1.
3
4 concat([],L,L).
5 concat([H|T],L,[H|M]) :- concat(T,L,M).
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22 % Sa vedem si intrebari care au mai multe raspunsuri.
23 % Sa cerem cu "Next" mai multe rezultate.
24 % In loc de "Next", in versiunea desktop a SWI-Prolog, se foloseste ";".
25 % La final, se afiseaza "false", semnificand ca nu mai exista alte solutii.
26
```

On the right, the execution results are shown for two queries:

Query 1: `concat([1,2,X,Y],L,[1,2,3,4,5,6]).`
Results:
`L = [5, 6],`
`X = 3,`
`Y = 4`

Query 2: `concat([1,2,X,Y|T],L,[1,2,3,4,5,6]).`
Results:
`L = [5, 6],`
`T = [],`
`X = 3,`
`Y = 4`
`L = [6],`
`T = [5],`
`X = 3,`
`Y = 4`
`L = [],`
`T = [5, 6],`
`X = 3,`
`Y = 4`
false

At the bottom, there is a query input field with `?- concat([1,2,X,Y|T],L,[1,2,3,4,5,6]).` and buttons for Examples, History, Solutions, table results, and a Run button.

Vedeți și celelalte opțiuni ale SWI-Prolog online

The screenshot displays the SWI-Prolog online environment. The browser address bar shows `swish.swi-prolog.org`. The interface includes a menu bar with **File**, **Edit**, **Examples**, and **Help**. A search bar is located in the top right corner. The main editor area on the left contains the following Prolog code:

```
1 lung([],0).
2 lung([_|T],N) :- lung(T,K), N is K+1.
3
4 concat([],L,L).
5 concat([H|T],L,[H|M]) :- concat(T,L,M).
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20 % Am cerut 10 rezultate deodata. ----->
```

The right-hand pane shows the execution results, displaying the state of variables `L` and `M` for each step of the `concat` predicate. The results are as follows:

| Step | L | M |
|------|-----------------------|-----------------|
| 1 | [1, 2, 3] | [] |
| 2 | [1, 2, 3, 4, 5] | [] |
| 3 | [] | [1, 2, 3, 4, 5] |
| 4 | [1] | [2, 3, 4, 5] |
| 5 | [1, 2] | [3, 4, 5] |
| 6 | [1, 2, 3] | [4, 5] |
| 7 | [1, 2, 3, 4] | [5] |
| 8 | [1, 2, 3, 4, 5] | [] |
| 9 | [1, 2, 3, 4, 5, 6, 7] | [] |

Below the results, there is a control bar with buttons for `Next`, `10`, `100`, `1,000`, and `Stop`. The bottom of the interface features a footer with `Examples`, `History`, and `Solutions` tabs, a checkbox for `table results`, and a `Run!` button.

Inclusiv opțiuni pentru editare rapidă

The screenshot displays the SWISH web interface for Prolog programming. The browser address bar shows `swish.swi-prolog.org`. The interface includes a search bar, a user count of 334 users online, and a navigation menu with options like File, Edit, Examples, and Help.

The main editor area contains a Prolog program with the following code:

```
1 lung([],0).
2 lung([_|T],N) :- lung(T,K), N is K+1.
3
4 concat([],L,L).
5 concat([H|T],L,[H|M]) :- concat(T,L,M).
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20 % Am cerut 10 rezultate deodata. ----->
```

The execution results are displayed on the right side of the interface, showing the state of variables `L` and `M` after each step of the `concat` predicate. The results are as follows:

- Initial state: `L = [1, 2, 3]`, `M = []`
- Step 1: `concat(L,M,[1,2,3,4,5])`
- Step 2: `L = []`, `M = [1, 2, 3, 4, 5]`
- Step 3: `L = [1]`, `M = [2, 3, 4, 5]`
- Step 4: `L = [1, 2]`, `M = [3, 4, 5]`
- Step 5: `L = [1, 2, 3]`, `M = [4, 5]`
- Step 6: `L = [1, 2, 3, 4]`, `M = [5]`
- Step 7: `L = [1, 2, 3, 4, 5]`, `M = []`
- Step 8: `concat(L,M,[1,2,3,4,5,6,7])`
- Step 9: `L = []`, `M = [1, 2, 3, 4, 5, 6, 7]`

The interface also includes a "Run" button and a "table results" checkbox.

Și acum în versiunea desktop a SWI-Prolog

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

SWI-Prolog (Multi-threaded, version 8.0.3)

File Edit Settings Run Debug Help

For online help and background, visit <http://www.swi-prolog.org>

For built-in help, use `?- help(Topic).` or `?- apropos(Word).`

`?- ['d:/work/cursurilemele/pl_2019_2020/laborator_pl_2019_2020/exspliste.pl'].`

true.

`?- concat(L,M,[a,b,c]).`

```
L = [],  
M = [a, b, c] ;  
L = [a],  
M = [b, c] ;  
L = [a, b],  
M = [c] ;  
L = [a, b, c].  
false.
```

`?- trace.`

true.

`[trace] ?- concat(L,M,[a,b,c]).`

Call: (8) concat(_4518, _4520, [a, b, c]) ? creep

Exit: (8) concat([], [a, b, c], [a, b, c]) ? creep

L = [],

M = [a, b, c] ;

Redo: (8) concat(_4518, _4520, [a, b, c]) ? creep

Call: (9) concat(_4792, _4520, [b, c]) ? creep

Exit: (9) concat([], [b, c], [b, c]) ? creep

Exit: (8) concat([a], [b, c], [a, b, c]) ? creep

L = [a],

M = [b, c] ;

Redo: (9) concat(_4792, _4520, [b, c]) ? creep

Call: (10) concat(_4798, _4520, [c]) ? creep

Exit: (10) concat([], [c], [c]) ? creep

Exit: (9) concat([b], [c], [b, c]) ? creep

Exit: (8) concat([a, b], [c], [a, b, c]) ? creep

L = [a, b],

M = [c] ;

Redo: (10) concat(_4798, _4520, [c]) ? creep

Call: (11) concat(_4804, _4520, []) ? creep

Exit: (11) concat([], [], []) ? creep

Exit: (10) concat([c], [], [c]) ? creep

Exit: (9) concat([b, c], [], [b, c]) ? creep

Exit: (8) concat([a, b, c], [], [a, b, c]) ? creep

L = [a, b, c],

M = [] ;

Redo: (11) concat(_4804, _4520, []) ? creep

Fail: (11) concat(_4804, _4520, []) ? creep

Fail: (10) concat(_4798, _4520, [c]) ? creep

Fail: (9) concat(_4792, _4520, [b, c]) ? creep

Fail: (8) concat(_4518, _4520, [a, b, c]) ? creep

false.

`[trace] ?- notrace.`

true.

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

The screenshot displays the SWISH (SWI-Prolog for SHaring) web interface. The browser address bar shows `swish.swi-prolog.org`. The SWISH logo and navigation menu (File, Edit, Examples, Help) are at the top. The main editor on the left contains the following Prolog code:

```
1 lung([],0).
2 lung([_|T],N) :- lung(T,K), N is K+1.
3
4 concat([],L,L).
5 concat([H|T],L,[H|M]) :- concat(T,L,M).
6
```

The right-hand pane shows the execution trace for the query `trace,concat(L,M,[sgl])`. The trace details the recursive calls and exits of the `concat` predicate, showing the state of variables `L` and `M` at each step. The final result is `false`.

Below the first trace, a second trace is visible for the query `trace,concat(L,M,[1,2,3])`, showing the initial call and exit with the list `[1, 2, 3]`.

At the bottom of the right pane, there are tabs for "Examples", "History", and "Solutions", along with a "Run!" button and a checkbox for "table results".

Unificare (orientativ) – detalii într-un curs următor

Două funcții **unifică** ddacă:

- acele funcții coincid
și
- au aceleași argumente:

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

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

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

- nici cu lung([], 0),
- nici cu lung([H|T], N).

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

cu concat([], L, L), pentru **L1** = [] și **L2** = **L** = [a, b, c] – aceasta este *prima soluție* a interogării,

dar și:

cu concat([H|T], L, [H|M]), pentru **H** = **a**, **L1** = [a|T], **L2** = **L** și **M** = [b, c],

așadar **următorul scop** este:

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

Acesta unifică:

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

așadar **următorul scop** este:

?- $\text{concat}(\mathbf{T1}, \mathbf{L2}, [\mathbf{c}])$.

Acesta unifică:

cu $\text{concat}([], L, L)$, 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 $\text{concat}([\mathbf{H2}|\mathbf{T2}], L, [\mathbf{H2}|\mathbf{M2}])$, 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 [H3|M3],

avem:

concat(T2,L2,[]) nu unifică cu concat([H3|T3],L,[H3|M3]).

Prin urmare:

concat(T2,L2,[]) unifică numai cu concat([],L,L), pentru **T2** = [] și **L2** = **L** = [] – așadar *a patra* și *ultima soluție* a interogării este:
L1 = [a|T] = [a|[b|T1]] = [a|[b|[c|T2]]] = [a|[b|[c|[]]]] = [a,b,c] și **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*: **A** = [1], **C** = [1,4,5].