

Lecție 11: LEGARE & EXECUȚIE

Obs: D.p.d.v. matematică, ordinea termenelor dintr-o conjuncție / disjuncție nu influențează rezultatul.

Ex: $P(A, B) \wedge Q(A, B) \Leftarrow \Rightarrow Q(A, B) \wedge P(A, B)$.

! În Prolog: nu înțelănume

Ex:

• ! - $X = Y^{(1)}$, $X = Y^{(2)}$ ^{(2) returnată} $\longrightarrow X = Y$.

De ce!:

(1) - unică \Rightarrow Prolog va considera ca X și Y sunt identice

(2) - verifică dacă x și y sunt identice,
fără a marca să le unifice.

- ? - $X == y$, $X = y$. $\xrightarrow{\text{returnată}} \text{false}$.

De ce ? :

(1) : verifică egalitatea fără a marca
să fie
 x, y sunt legate $\xrightarrow{\text{esuata}}$

(2) : nu se mai verifică dacă
 $\xrightarrow{\text{esuata}}$

PROLOG - PUTEREA GENERATIVĂ

SCOPES :

[Queries $\xrightarrow{\text{returnată}}$ true / false
* legări de variabile]

Ex:

? = întrebare / ieșire

①

append (!List1, ?List2, ?List3)

\rightarrow List3 reprezintă $(List1 ++ List2)$

- !- append ([1,2], [3,4], [1,2,3,4]).
 \rightarrow true.
- ?- append ([1,2], [3,4], L3).
 \rightarrow L3 = [1,2,3,4].
- ?- append ([1,2], L2, [1,2,5,6]).
 \rightarrow L2 = [5,6].

②

Generarea listelor cu 3 elemente

Care contin 1 și 2 ca elemente.

! - $\text{length}(L, 3)$, $\text{member}(1, L)$,
 $\text{member}(2, L)$.

→ $L = [1, 2, -]$;

$L = [1, -, 2]$;

$L = [2, 1, -]$;

$L = [2, -, 1]$;

$L = [-, 1, 2]$;

$L = [-, 2, 1]$;

False.

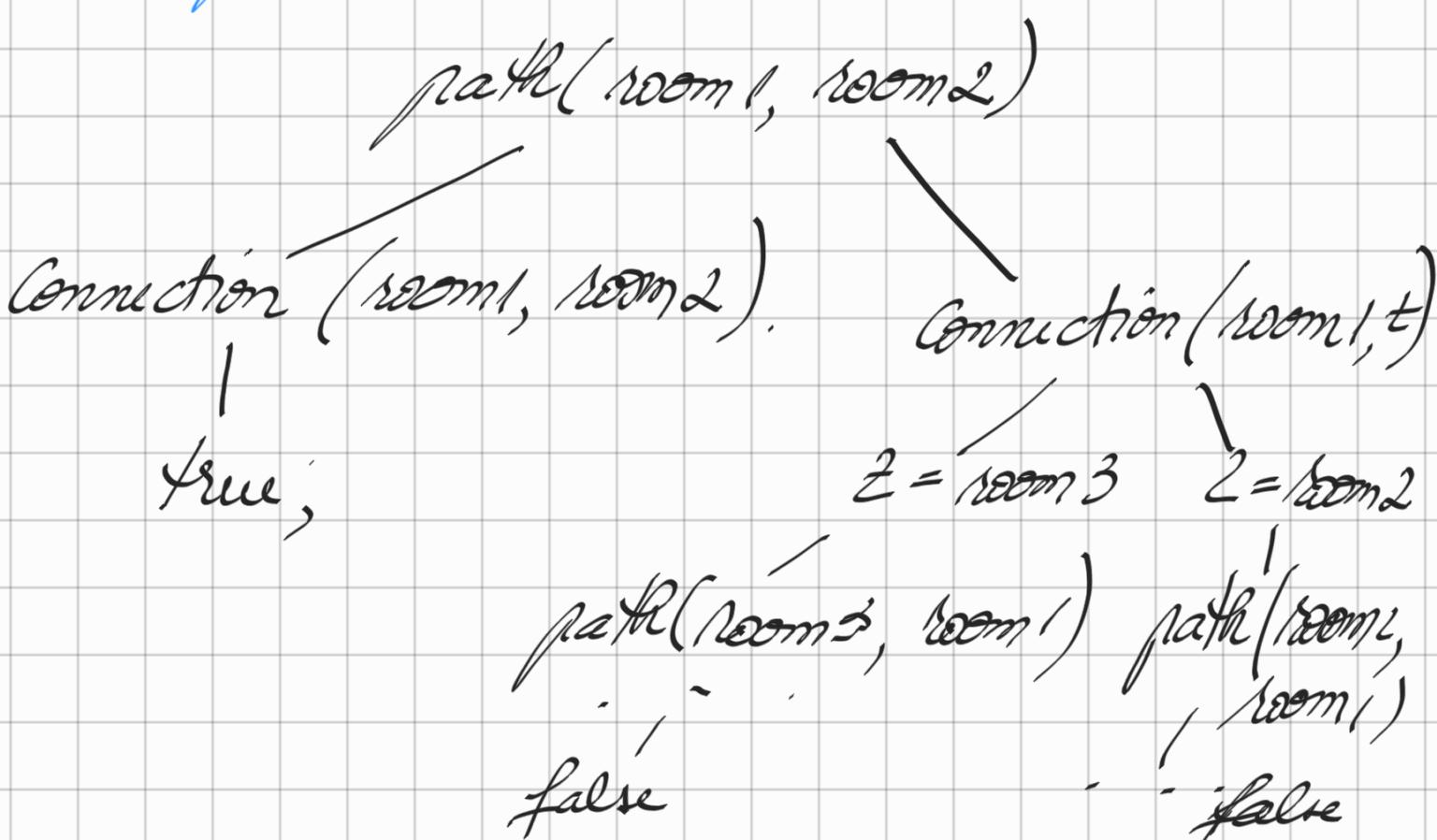
PROCESUL DE EXECUȚIE

! Prolog face Backtracking în spatiul
stăriilor.

Ex :

```
connection(room1, room3).  
connection(room1, room2).  
connection(room3, room4).  
connection(room2, room4).  
  
path(X, Y) :- connection(X, Y).  
path(X, Y) :- connection(X, Z), path(Z, Y).
```

? - path(room1, room2).



? - path (room 1, room 4).

→ true ;

true ;

false .

Solutie : op de controll  cut (!)
~~false~~

CUT

→ Are rolul de a elmina punetele de lărgire din execuția predicatului curent.

ce e în stanga : nu mai poate să rezolvă
în dreapta . da + se poate aplica Backtracking .

Ex:

Pentru a opri procesul de Backtracking după ce se găsește o soluție pentru care $\text{path}(\text{room1}, \text{room4})$ este true, se folosește cut:

```
connection(room1, room3).  
connection(room1, room2).  
connection(room3, room4).  
connection(room2, room4).
```

```
path(X, Y) :- connection(X, Y), !. (1)  
path(X, Y) :- connection(X, Z), path(Z, Y), !. (2)
```

(1): Dacă se găsește o conexiune directă între camerele X și Y , nu se mai verifică și regula 2

(2): Dacă s-a găsit o cameră 2 astfel încât să existe un $\text{path } X - 2 - Y$, nu se mai verifică și altă valoare pentru 2.

* GREEN CUT

→ dacă se elimină (!), execuția rămâne corectă.

Ex :

1) FĂRĂ CUT

```
f(X, first_interval) :- X < 3.          % (-inf, 3)
f(X, second_interval) :- 3 =  
= X, X < 6. % [3, 6)
f(X, third_interval) :- 6 =  
= X.          % [6, +inf)
```

? - $\not f(1, I).$

$I = \text{first_interval}$

false.

2) FOLOSIND CUT

```
f1(X, first_interval) :- X < 3, !.
f1(X, second_interval) :- 3 =  
= X, X < 6, !.
f1(X, third_interval) :- 6 =  
= X.
```

? - $\not f1(1, I).$

$I = \text{first_interval}.$

Obs: • Am obtinut legarea corectă

pentru I în ambele variante doară
regulele sunt excludoare mutuale.

(dacă un număr X e în first-interval,
nu poate fi și în second-interval)

• În ceea ce privind varianta 2) se evită

Backtrackingul inutil.

* RED CUT

→ Fără (!) nu se mai obține
soluția corectă.

Ex :

1) **FĂRĂ CUT**

```
g(X, first_interval) :- X < 3.  
g(X, second_interval) :- X < 6.  
g(_, third_interval).
```

? - g(1, I).

I = first_interval;

I = second_interval;

I = third_interval.

2) **FOLOSIND CUT**

```
g1(X, first_interval) :- X < 3, !.  
g1(X, second_interval) :- X < 6, !.  
g1(_, third_interval).
```

? - g1(1, I).

I = first_interval

Obs : dacă clasele nu mai sunt
mutual exclusive, da se obțin
rezultate greșite fără cuf.

METAPREDICATE

① FINDALL (+ Template, + Goal, - Bag)
→ pentru fiecare satisfacere a goal,
se crează o instanță a template-ului
care se adaugă în Bag.

Ex :

*) Extrage elementele par sănătoase dintr-o listă

even(L, Evens) :- findall(X,
(member(X, L), X mod 2 == 0),
Evens).

x)

Generaza' liste paralele formate din elemente distincte din lista

pairs(L, Pairs) :- findall((X,Y),
(member(X,L), member(Y,L), X \= Y),
Pairs).

② FOR ALL (+ Cond, + Action)
→ True / False

→ Verifica dacă se pot implementa
condițiile din acțiunii pentru orice
legare din cond.

Ex :

*) Verifica dacă o listă conține
doar elemente pozitive

onlyPositives(L) :- forall(number(X,L),
 $X > 0$).

*) + : Lista de liste

- : Să se verifice dacă toate

listele din lista data contin elemente
mai mari ca 5.

$p(\text{All}) :- \text{forall}(\text{number}(L, \text{All}),$
 $(\text{forall}(X, L), X > 5)).$

*) Verifică dacă $L_1 \subset L_2$
este

$p(L_1, L_2) :- \text{forall}(\text{member}(X, L_1),$
 $\text{member}(X, L_2))$

③ ~~BAG OF~~ (+ Template, + goal, - Bag)

→ Asemănătoare cu findAll, dar face cără și instanță a lui Bag pentru fiecare întâiere a variabilelor libere (care nu apar în template)

Ex :

*) Pt. fiecare x din L se face și lista cu valoare din lista $V > x$.

largerValues(L , Res) :- bagof(y ,
(member(x , L), member(y , L),
 $x < y$),
Res)