Laborator 11

Fundamentele Limbajelor de Programare

Laboratorul 11

TODO

- Aritmetica în Prolog
- Recursivitate în Prolog
- Liste în Prolog

Material suplimentar

- `Capitolul 2` - `Capitolul 6` din *Learn Prolog Now!*.

Aritmetica în Prolog

Aritmetica în Prolog

Exemplu

```
?- 3+5 = +(3,5).
true
?- 3+5 = +(5,3).
false
?- 3+5 = 8.
false
```

Explicații:

- 3+5 este un termen.
- Prolog trebuie anunțat explicit pentru a îl evalua ca o expresie aritmetică, folosind predicate predefinite în Prolog, cum sunt `is/2`, `=:=/2`, `>/2` etc.

Aritmetica în Prolog

Exercițiu. *Analizați următoarele exemple:*

```
?- 3+5 is 8.
false

?= X is 3+5.
X = 8

?- 8 is 3+X.
is/2: Arguments are not sufficiently instantiated

?- X=4, 8 is 3+X.
false
```

Aritmetica în Prolog

Exercițiu. *Analizați următoarele exemple:*

```
?- X is 30-4.
X = 26

?- X is 3*5.
X = 15

?- X is 9/4.
X = 2.25
```

Aritmetica în Prolog

Operatorul `is`:

- Primește două argumente
- Al doilea argument trebuie să fie o expresie aritmetică validă, cu toate variabilele inițializate
- Primul argument este fie un număr, fie o variabilă
- Dacă primul argument este un număr, atunci rezultatul este `true` dacă este egal cu evaluarea expresiei aritmetice din al doilea argument.
- Dacă primul argument este o variabilă, răspunsul este pozitiv dacă variabila poate fi unificată cu evaluarea expresiei aritmetice din al doilea argument.

Totuși, nu este recomandat să folosiți `is` pentru a compara două expresii aritmetice, ci operatorul `=:=`.

Aritmetica în Prolog

Exercițiu. *Analizați următoarele exemple:*

```
?- 8 > 3.
true

?- 8+2 > 9-2.
true

?- 8 < 3.
false
```

```
?- 8 >= 3.
true

?- 8 =:= 3.
false

?- 8 =\= 3.
true
```

Operatori aritmetici

Operatorii aritmetici predefiniți în Prolog sunt de două tipuri:

- funcții
- relații

Funcții

- Adunarea și înmulțirea sunt exemple de funcții aritmetice.
- Aceste funcții sunt scrise în mod uzual și în Prolog.

Exemplu

```
2 + (-3.2 * X - max(17,X)) / 2 ** 5
```

- 2**5 înseamnă 2^5
- Exemple de alte funcții disponibile:
  `min/2`, `abs/1` (modul), `sqrt/1` (radical), `sin/1` (sinus)
- Operatorul `//` este folosit pentru împărțire întreagă.
- Operatorul `mod` este folosit pentru restul împărțirii întregi.

Relații

- Relatiile aritmetice sunt folosite pentru a compara evaluarea expresiilor aritmetice (e.g, `X > Y`)
- Exemple de relații disponibile:
  <, >, =<, >=, =\= (diferit), =:= (aritmetic egal)
- Atenție la diferența dintre =:= și =:
  - =:= compară două expresii aritmetice
  - = caută un unificator

Exemplu

```
?- 2 ** 3 =:= 3 + 5.
true
?- 2 ** 3 = 3 + 5.
false
```

Practică

Exercițiul 1: distanța dintre două puncte

Definiți un predicat `distance/3` pentru a calcula distanța dintre două puncte într-un plan 2-dimensional. Punctele sunt date ca perechi de coordonate.

Exemple:

```
?- distance((0,0), (3,4), X).
X = 5.0

?- distance((-2.5,1), (3.5,-4), X).
X = 7.810249675906654
```

Recursivitate

Bază de conștințe

În laboratorul trecut am folosit următoarea bază de cunoștițe:

```
parent_of(rickardStark,eddardStark).
parent_of(rickardStark,lyannaStark).
parent_of(lyarraStark,eddardStark).
parent_of(lyarraStark,lyannaStark).

parent_of(aerysTargaryen,rhaegarTargaryen).
parent_of(rhaellaTargaryen,rhaegarTargaryen).

parent_of(rhaegarTargaryen,jonSnow).
parent_of(lyannaStark,jonSnow).
```

Recursivitate - strămoși

Am definit un predicat `ancestor_of(X,Y)` care este adevărat dacă X este un strămoș al lui Y.

Definiția recursivă a predicatului `ancestor_of(X,Y)` :

```
ancestor_of(X,Y) :- parent_of(X,Y).
ancestor_of(X,Y) :- parent_of(X,Z), ancestor_of(Z,Y).
```

Practică

Exercițiul 2: numerele Fibonacci

Scrieți un predicat `fib/2` pentru a calcula al n-ulea număr Fibonacci. Secvența de numere Fibonacci este definită prin:

$$\begin{aligned} F_0 &= 1 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-1} \text{ for } n \geq 2 \end{aligned}$$

Exemple:

```
?- fib(1,X).
X=1.
true

?- fib(2,X).
X=2.
true
```

```
?- fib(5,X).
X=8.
true
```

Practică

Exercițiul 2 (cont.)

Programul scris anterior vă găsește răspunsul la întrebarea de mai jos?

```
?- fib(50,X).
```

Dacă da, felicitări! Dacă nu, încercați să găsiți o soluție mai eficientă!

Hint: Încercați să construiți toate numerele Fibonacci până ajungeți la numărul căutat.

Afișări în Prolog

- Pentru a afișare se foloseste predicatul `write/1`.
- Predicatul `nl/0` conduce la afișarea unei linii goale.

Exemplu

```
?- write('Hello World!'), nl.
Hello World!
true

?- X = hello, write(X), nl.
hello
X = hello
```

Practică

Exercițiul 3: afișarea unui pătrat de caractere

Scrieți un program în Prolog pentru a afișa un pătrat de $n \times n$ caractere pe ecran.

Denumiți predicatul `square/2`. Primul argument este un număr natural diferit de 0, iar al doilea un caracter (i.e, orice termen în Prolog) care trebuie afișat.

Exemplu:

```
?- square(5, '* ').
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Liste

Liste

- Listele în Prolog sunt un tip special de date (termeni speciali).
- Listele se scriu între paranteze drepte, cu elementele despărțite prin virgulă.
- `[]` este lista vidă.

Exemplu

- `[elephant, horse, donkey, dog]`
- `[elephant, [], X, parent(X, tom), [a, b, c], f(22)]`

Head & Tail

- Primul element al unei liste se numește *head*, iar restul listei *tail*.
- Evident, o listă vidă nu are un prim element.
- În Prolog există o notație utilă pentru liste cu separatorul `|`, evidențiind primul element și restul listei.

Exemplu

```
?- [1, 2, 3, 4, 5] = [Head | Tail].
Head = 1
Tail = [2, 3, 4, 5]
```

Cu această notație putem să returnăm ușor, de exemplu, al doilea element dintr-o listă.

```
?- [quod, licet, jovi, non, licet, bovi] = [_, X | _].
X = licet
```

Lucrul cu liste

Exemplu (`elements_of/2`)

- un predicat care verifică dacă o listă conține un anumit termen
- `element_of(X,Y)` trebuie să fie adevărat dacă `X` este un element al lui `Y`.

```
/* Dacă primult element al listei este termenul
pe care îl căutăm, atunci am terminat. */

element_of(X,[X|_]).

% Altfel, verificăm dacă termenul se află în restul listei.

element_of(X,[_|Tail]) :- element_of(X,Tail).

?- element_of(a,[a,b,c]).
?- element_of(X,[a,b,c]).
```

Lucrul cu liste

Exemplu (`concat_lists/3`)

- un predicat care este poate fi folosit pentru a concatena două liste
- al treilea argument este concatenarea listelor date ca prime două argumente

```
concat_lists([], List, List).
concat_lists([Elem | List1], List2, [Elem | List3]) :-
      concat_lists(List1, List2, List3).


?- concat_lists([1, 2, 3], [d, e, f, g], X).
?- concat_lists(X, Y, [a, b, c, d]).
```

Lucrul cu liste

În Prolog există niște predicate predefinite pentru lucrul cu liste. De exemplu:

- `length/2`: al doilea argument întoarce lungimea listei date ca prin argument
- `member/2`: este adevărat dacă primul argument se află în lista dată ca al doilea argument
- `append/3`: identic cu predicatul anterior `concat_lists/3`
- `last/2`: este adevărat dacă al doilea argument este identic cu ultimul element al listei date ca prim argument
- `reverse/2`: lista din al doilea argument este lista data ca prim elemenent în oglindă.

Practică

Exercițiul 4

A) Definiți un predicat `all_a/1` care primește ca argument o listă și care verifică dacă argumentul său este format doar din a-uri.

```
?- all_a([a,a,a,a]).
?- all_a([a,a,A,a]).
```

B) Scrieti un predicat `trans_a_b/2` care "traduce" o listă de a-uri într-o listă de b-uri. `trans_a_b(X,Y)` trebuie să fie adevărat dacă "intrarea" `X` este o listă de a-uri și "ieșirea" `Y` este o listă de b-uri, iar cele două liste au lungimi egale.

```
?- trans_a_b([a,a,a],L).
?- trans_a_b([a,a,a],[b]).
?- trans_a_b(L,[b,b]).
```

Practică

Exercițiul 5: Operații cu vectori

A) Scrieți un predicat `scalarMult/3` al cărui prim argument este un întreg, al doilea argument este o listă de întregi, iar al treilea argument este rezultatul înmulțirii cu scalari al celui de-al doilea argument cu primul.

De exemplu, la intrebarea

```
?-scalarMult(3,[2,7,4],Result).
```

ar trebui să obțineți `Result = [6,21,12]`.

Practică

Exercițiul 5 (cont.)

B) Scrieți un predicat `dot/3` al cărui prim argument este o listă de întregi, al doilea argument este o listă de întregi de lungimea primeia, iar al treilea argument este produsul scalar dintre primele două argumente.

De exemplu, la întrebarea

```
?-dot([2,5,6],[3,4,1],Result).
```

ar trebui să obțineți `Result = 32`.

Practică

Exercițiul 5(cont.)

C) Scrieți un predicat `max/2` care caută elementul maxim într-o listă de numere naturale.

De exemplu, la întrebarea

```
?-max([4,2,6,8,1],Result).
```

ar trebui să obțineți `Result = 8.`

Practică

Exercițiul 6

Definiți un predicat `palindome/1` care este adevărat dacă lista primită ca argument este palindrom (lista citită de la stânga la dreapta este identică cu lista citită de la dreapta la stânga).

De exemplu, la întrebarea

```
?-palindrome([r,e,d,i,v,i,d,e,r]).
```

ar trebui să obțineți `true`.

Nu folosiți predicatul predefinit `reverse`, ci propria implementare a acestui predicat.

Practică

Exercițiul 7

Definiți un predicat `remove_duplicates/2` care șterge toate duplicatele din lista dată ca prim argument și întoarce rezultatul în al doilea argument.

De exemplu, la întrebarea

```
?- remove_duplicates([a, b, a, c, d, d], List).
```

ar trebui să obțineți `List = [b, a, c, d]`.

Pe săptămâna viitoare!