Laborator 5

2017-2018

Programare Logică

Laboratorul 5

TODO

- Puzzle-uri în Prolog.

Practică

Exercițiul 1: Zebra puzzle

Zebra puzzle este unul dintre cele mai cunoscute puzzle-uri logice.

Citiți mai jos mai multe detalii despre acest puzzle:
`https://en.wikipedia.org/wiki/Zebra_Puzzle`

Vom rezolva următoarea versiune publicată în 1962 în care fiecare personaj

- locuiește într-o casă care are o anumită culoare,
- are o naționalitate,
- are un anumit animal de companie,
- are o băutură preferată,
- fumează un anumit tip de țigări.

Practică

Exercițiul 1 (cont.)

Vrem să aflăm cine are o zebră având următoarele informații:

1. Sunt cinci case.
2. Englezul locuiește în casa roșie.
3. Spaniolul are un câine.
4. În casa verde se bea cafea.
5. Ucraineanul bea ceai.
6. Casa verde este imediat în dreapta casei bej.
7. Fumătorul de "Old Gold" are melci.
8. În casa galbenă se fumează "Kools".

Practică

Exercițiul 1 (cont.)

9. În casa din mjloc se bea lapte.
10. Norvegianul locuiește în prima casă.
11. Fumătorul de "Chesterfields" locuiește lângă cel care are o vulpe.
12. "Kools" sunt fumate în casa de lângă cea în care se ține calul.
13. Fumătorul de "Lucky Strike" bea suc de portocale.
14. Japonezul fumează "Parliaments".
15. Norvegianul locuiește lângă casa albastră.

Practică

Exercițiul 1 (cont.)

Vom rezolva acest puzzle în Prolog.

1) Definiți un predicat `la_dreapta(X,Y)` care este adevărat când numerele X și Y sunt consecutive, X fiind cel mai mare dintre ele.

2) Definiți un predicat `la_stanga(X,Y)` care este adevărat când numerele X și Y sunt consecutive, Y fiind cel mai mare dintre ele.

3) Definiți un predicat `langa(X,Y)` care este adevărat când numerele X și Y sunt consecutive.

Practică

Exercițiul 1 (cont.)

În continuare definim un predicat `solutie(Strada,PosesorZebra)` în care includem toate informațiile pe care le deținem:

```
solutie(Strada,PosesorZebra) :-
  Strada = [
      casa(1,_,_,_,_,_,_),
      casa(2,_,_,_,_,_,_),
      casa(3,_,_,_,_,_,_),
      casa(4,_,_,_,_,_,_),
      casa(5,_,_,_,_,_,_)],
    member(casa(_,englez,rosie,_,_,_), Strada),
    member(casa(_,spaniol,_,caine,_,_), Strada),
    member(casa(_,_,verde,_,cafea,_), Strada),
    ...,
    member(casa(_,PosesorZebra,_,zebra,_,_), Strada),
```

Completați toate informațiile pentru a afla soluția ținând cont de codarea:

```
casa(Numar,Nationalitate,Culoare,AnimalCompanie,Bautura,Tigari)
```

Practică

Exercițiul 2: Cel mai lung cuvânt

Acest exemplu este din
Ulle Endriss, *Lecture Notes – An Introduction to Prolog Programming.*

Countdown este un joc de televiziune popular în Marea Britanie în care jucătorii trebuie să găsească un cuvânt cât mai lung cu literele dintr-o mulțime dată de nouă litere.

Să încercăm să rezolvăm acest joc cu Prolog!

Practică

Exercițiul 2: Cel mai lung cuvânt

Concret, vom încerca să găsim o soluție optimă pentru următorul joc:

Primind nouă litere din alfabet (nu neapărat unice),
trebuie să construim cel mai lung cuvânt format din literele date
(pot rămâne litere nefolosite).

Vom rezolva jocul pentru cuvinte din limba engleză.

Scorul obținut este lungimea cuvântului găsit.

Practică

Exercițiul 2 (cont.)

Scopul final este de a construi un predicat în Prolog `topsolution/3`:

dându-se o listă de nouă litere în primul său argument, trebuie să returneze o soluție cât mai bună, adică un cuvânt din limba engleză ce poate fi format cu aceste litere în al doilea argument și lungimea cuvântului în al treilea argument.

```
?- topsolution([[r,d,i,o,m,t,a,p,v]],Word, Score).
Word = dioptra,
Score = 7.
```

Practică

Exercițiul 2 (cont.)

Începeți prin a descărca fișierul `words.pl` de la adresa

http://tinyurl.com/prolog-words

Salvați acest fișier în același director cu fișierul programului vostru.

Acest fișier conține o listă cu peste 350.000 de cuvinte din limba engleză, de la `a` la `zyzzyva`, sub formă de fapte.

Includeți linia `:- include('words.pl').` în programul vostru pentru a putea folosi aceste fapte.

Practică

Exercițiul 2 (cont.)

Predicatul predefinit în Prolog `atom_chars(Atom,CharList)` descompune un atom într-o listă de caractere.

Folosiți acest predicat pentru a defini un predicat `word_letters/2` care transformă un cuvânt (i.e, un atom în Prolog) într-o listă de litere.
De exemplu:

```
?- word_letters(hello,X).
X = [h,e,l,l,o]
```

Ca o paranteza, observați că puteți folosi acest predicat pentru a găsi cuvinte în engleză de 45 de litere:

```
?- word(Word), word_letters(Word,Letters),
   length(Letters,45).
```

Practică

Exercițiul 2 (cont.)

Mai departe, scrieți un predicat `cover/2` care, primind două liste, verifică dacă a doua listă "acoperă" prima listă (i.e., verifică dacă fiecare element care apare de k ori în prima listă apare de cel puțin k ori în a doua listă).

De exemplu

```
?- cover([a,e,i,o], [m,o,n,k,e,y,b,r,a,i,n]).
true

?- cover([e,e,l], [h,e,l,l,o]).
false
```

Practică

Exercițiul 2 (cont.)

Scrieți un predicat `solution/3` care primind o listă de litere ca prim argument și un scor dorit ca al treilea argument, returnează prin al doilea argument un cuvânt cu lungimea egală cu scorul dorit, "acoperit" de lista respectivă de litere.

De exemplu

```
?- solution([g,i,g,c,n,o,a,s,t], Word, 3).
Word = act
```

Practică

Exercițiul 2 (cont.)

Acum implementați predicatul `topsolution/3`.

Testați, de exemplu, predicatul definit pe mulțimea de litere:

```
[y,c,a,l,b,e,o,s,x]
```

Aceasta este una listele de litere folosite în ediția de *Countdown* din 18 Decembrie 2002 din Marea Britanie în care Julian Fell a obținut cel mai mare scor din istoria concursului. Pentru lista de mai sus, el a găsit cuvântul *cables*, câștigând astfel 6 puncte.

Poate programul vostru să bată acest scor?

Practică

Exercițiul 3: numere romane

Acest exemplu este din
Ulle Endriss, *Lecture Notes – An Introduction to Prolog Programming.*

În acest exercițiu vom scrie un predicat în Prolog numit `roman2arabic/2` care traduce numere romane în numere arabe.

De exemplu,

```
?- roman2arabic('XXI', Number).
Number = 21

?- roman2arabic('MCMXCIX', Number).
Number = 1999
```

Înainte de a rezolva acest exercițiu, amintiți-vă cum funcționează numerele romane folosind resurse online (de exemplu, Wikipedia).

Practică

Exercițiul 3 (cont.)

Ideea de bază pe care o vom folosi în rezolvarea acestei probleme este de a folosi o reprezentare intermediară ce constă în lista numerelor arabe corespunzătoare simbolurilor folosite în numerele romane.

De exemplu, vom reprezenta numărul XXI ca lista `[10,10,1]`: fiecare număr arab corespunde unui simbol, și suma numerelor este 21, adică echivalentul lui XXI.

Totuși, problema se complică puțin deoarece numerele romane folosesc și notația descrescătoare (de exemplu, avem IX și nu VIIII pentru 9). În consecință, câteodată vom reprezenta în lista intermediară două simboluri romane printr-un singur număr. Ca exemplu, în loc să reprezentăm XIX ca lista `[10,1,10]` a cărei suma este 21, vom reprezenta XIX ca `[10,9]`.

Urmați pașii de mai jos pentru a implementa `roman2arabic/2`:

Exercițiul 3 (cont.)

a) Scrieți un predicat `symbol/2` care pentru fiecare simbol folosit în sistemul roman întoarce în al doilea argument echivalentul din sistemul arab.

De exemplu,

```
?- symbol('X', Value).
Value = 10
```

Practică

Exercițiul 3 (cont.)

b) Scrieți un predicat `symbols2numbers/2` care traduce o listă de simboluri în lista corespunzătoare de numere arabe.

De exemplu,

```
?- symbols2numbers(['M', 'M', 'X', 'V'], Values).
Values = [1000, 1000, 10, 5]

?- symbols2numbers(['X', 'L', 'I', 'I'], Values).
Values = [40, 1, 1]
```

Practică

Exercițiul 3 (cont.)

c) Scrieți un predicat `sum/2` care adună numerele din lista dată ca prim argument.

De exemplu,

```
?- sum([1, 2, 3, 4, 5], Sum).
Sum = 15
```

Practică

Exercițiul 3 (cont.)

d) Acum puteți să implementați predicatul `roman2arabic/2`. Observați că trebuie să descompuneți un atom ca 'XXI' într-o listă de caractere.

```
?- roman2arabic('MMXVIII',Result).
Result = 2018

?- roman2arabic('MCMXC',Result).
Result = 1990

?- roman2arabic('MCMLIV',Result).
Result = 1954

?- roman2arabic('MDCCLXXVI',Result).
Result = 1776
```

Pe data viitoare!