

Laborator 6

2017-2018

Programare Logică

TODO

- Operatori în Prolog.
- Exerciții
 - definirea limbajului calculului propozițional clasic
 - prelucrarea formulelor

Scopul acestor exerciții este de a face prelucrări în vederea implementării unui SAT solver bazat pe rezoluție.

Operatori în Prolog

Operatori în Prolog

- Ați întâlnit până acum mai mulți operatori în Prolog: +, *, is, ...
- Fiecare operator are o precedență și o regulă pentru asociativitate.

Exemplu

```
?- X is 2+3+4.
```

```
X = 9.
```

```
true
```

```
?- 2+3+4=2+(3+4).
```

```
false.
```

```
?- 2+3+4=(2+3)+4.
```

```
true.
```

Operatori în Prolog

- Putem afla informații despre un operator folosind predicatul `current_op`

Exemplu

```
?- current_op(Precedence, Associativity, is).  
Precedence = 700,  
Associativity = xfx.
```

```
?-current_op(Precedence, Associativity, +).  
Precedence = 200,  
Associativity = fy  
Precedence = 500,  
Associativity = yfx.
```

- Observăm că operațiile cu precedență mai mică se efectuează primele.
- Ce înseamnă xfx și fy?

Operatori în Prolog

- Asociativitatea operatorilor este desemnată prin:

xf , yf , xfx , xfy , yfx , fy or fx

- f este functorul
- y este un termen cu precedența mai mică sau egală cu a functorului
- x este un termen cu precedența strict mai mică decât a functorului

Exemplu

Operatorul – binar are precedența 500 și asociativitatea yfy . Verificați aceasta folosind `current_op` și înțelegeți exemplele de mai jos.

```
?- current_op(500, yfx, -).  
true.  
?- 10-5-2 = 10-(5-2).  
false.  
?- 10-5-2 = (10-5)-2.  
true.
```

Operatori în Prolog

| Pattern | Associativity | | Examples |
|---------|---|-------------------|-----------------------------------|
| yfx | infix | left-associative | +, -, * |
| xfy | infix | right-associative | , (for subgoals) |
| xfx | infix | non-associative | =, is, < (i.e., no nesting) |
| yfy | makes no sense, structuring would be impossible | | |
| fy | prefix | associative | - (i.e., - - 5 allowed) |
| fx | prefix | non-associative | :- (i.e., :- :- goal not allowed) |
| yf | postfix | associative | |
| xf | postfix | non-associative | |

sursa tabelului

Operatori în Prolog

- În Prolog putem defini operatori noi astfel

`:- op(Precedence, Type, Name).`

Atenție! Definirea unui operator este sintactică, nu spune nimic despre semnificația sa, care trebuie definită separat.

Exemplu

```
:- op(500, xf, is_dead).
```

```
kill(marsellus,zed).
```

```
is_dead(X) :- kill(_,X).
```

Citiți mai multe despre operatori:

[SWI-Prolog](#)

[Learn Prolog Now!](#)

Exercitji

Exercițiul 1: definiți limbajul logicii propoziționale clasice în Prolog.

Începeți prin a defini:

- variabilele: `is_var(a).` `is_var(b).`
- operatorii: `nu`, `si`, `sau`, `imp`
 `:- op(620, xfy, si).`
 `:- op(610, fy, nu).`

Exemplu:

```
?- X = a si nu b.
```

```
X = a si nu b.
```

Exercițiul 2: scrieți un predicat care să întoarcă true dacă argumentul este o formulă corectă.

Exemplu:

```
?- formula(nu nu a si b sau c).  
true.
```

Atenție! dacă formula nu este sintactic corectă se poate primi răspunsul false sau mesaj de eroare:

```
?- formula(a si sau).  
false.  
?- formula(a si sau a).  
ERROR: Syntax error: Operator expected
```

Exercițiul 2 (cont): scrieți un predicat care să întoarcă true dacă argumentul este o formulă corectă.

Pentru a evita mesajele de eroare, putem defini:

```
test :- catch(read(X), Error, false) , X.
```

```
?- test.
```

```
|: formula(a imp a).
```

```
true.
```

```
?- test.
```

```
|: formula(a si sau a).
```

```
false.
```

Evaluarea unei formule

- Fie φ o formulă în calculul propozițional clasic și $Var(\varphi)$ mulțimea variabilelor lui φ . Spunem că φ este *tautologie* dacă $e^+(\varphi) = 1$ oricare ar fi $e : Var(\varphi) \rightarrow \{0, 1\}$ o evaluare.
- Reamintim că $e^+ : Form \rightarrow \{0, 1\}$ este unica funcție care satisface următoarele proprietăți:
 - $e^+(p) = e(p)$ pentru orice variabila propozițională p ,
 - $e^+(\neg\psi) = \neg e^+(\psi)$,
 - $e^+(\psi_1 \vee \psi_2) = e^+(\psi_1) \vee e^+(\psi_2)$,
 - $e^+(\psi_1 \wedge \psi_2) = e^+(\psi_1) \wedge e^+(\psi_2)$,
 - $e^+(\psi_1 \rightarrow \psi_2) = e^+(\psi_1) \rightarrow e^+(\psi_2)$,oricare ar fi ψ, ψ_1, ψ_2 formule.

În continuare vom scrie un program care, dată fiind o formulă:

- determina toate evaluările posibile (tabelul de adevăr asociat),
- determină dacă formula este tautologie.

Practică

Exercițiul 3: scrieți un predicat `find_vars` care determină mulțimea variabilelor unei formule.

Exemplu:

```
?- find_vars(a imp (b imp (c sau a)), [], Vars).  
Vars = [c, b, a].
```

Indicație: predicatul are trei argumente

`find_vars(X,V,Vfin)`

cu următoarea semnificație:

- ☐ X este formula,
- ☐ V este o listă care conține variabilele găsite până în acel moment,
- ☐ Vfin este lista care se obține adăugând la V variabilele lui X.

Atenție! V și Vfin sunt mulțimi.

Exercițiul 4: scrieți un predicat `all_assigns` care, pentru un număr natural n dat, construiește lista tuturor listelor de lungime n cu elemente 0 și 1.

Exemplu:

```
?- all_assigns(0,LA).
```

```
LA = [[]].
```

```
?- all_assigns(2,LA).
```

```
LA = [[0, 0], [0, 1], [1, 0], [1, 1]].
```

Exercițiul 5: definiți tabelul de adevăr pentru operatorii nu, si, sau, imp.

Exemplu:

```
table_nu(0,1). table_nu(1,0).  
table_si(0,0,0). table_si(0,1,0). ...  
...
```


Exercițiul 6: scrieți un predicat `truth_value` care determină valoarea de adevăr a unei formule pentru o evaluare dată.

Indicație: Predicatul are patru argumente

`truth_value(X, Var, A, Val)`

cu următoarea semnificație:

- ☐ `X` este formula,
- ☐ `Var` este mulțimea variabilelor formulei,
- ☐ `A` este o lista de lungime `n` cu elemente 0 și 1, unde `n` este numărul de variabile din formulă; atribuirea este definită astfel:
variabila din poziția `i` din `Var` are valoarea din poziția `i` din `A`,
- ☐ `Val` este obținută prin evaluarea formulei.

Exercițiul 6 (cont): scrieți un predicat `truth_value` care determină valoarea de adevăr a unei formule pentru o evaluare dată.

Exemplu:

```
?- truth_value(a imp (b sau nu c), [a,b,c],[1,0,1],Val).  
Val = 0
```

În exemplul de mai sus:

- formula este $\varphi = a \rightarrow (b \vee \neg c)$
- evaluarea este: $e(a) = 1, e(b) = 0, e(c) = 1$
- valoarea formulei este $e^+(\varphi) = 1 \rightarrow (0 \vee \neg 1) = 0$

Practică

Exercițiul 7: scrieți un predicat `all_values` care determină valorile de adevăr ale unei formule pentru o listă de evaluări. Acest predicat generalizează predicatul `truth_value` definit anterior.

Indicație: Predicatul are patru argumente

`all_values(X, Var, LA, LVal)`

cu următoarea semnificație:

- ☐ `X` este formula,
- ☐ `Var` este mulțimea variabilelor formulei,
- ☐ `LA` este o lista de liste de lungime `n` cu elemente 0 și 1, unde `n` este numărul de variabile din formulă; fiecare listă definește o atribuire ca în exercițiul precedent,
- ☐ `LVal` este lista evaluărilor.

Practică

Exercițiul 7 (cont): scrieți un predicat `all_values` care determină valorile de adevăr a ale unei formule pentru o listă de evaluari. Acest predicat generalizează predicatul `truth_value` definit anterior.

Exemplu:

```
?- all_values(a imp b, [a,b], [[0,1],[1,1]], LVal).  
LVal = [1, 1]
```

În exemplul de mai sus:

- formula este $\varphi = a \rightarrow b$
- evaluarile sunt definite astfel:
 - `[0,1]` definește evaluarea $e_1(a) = 0, e_1(b) = 1$
 - `[1,1]` definește evaluarea $e_2(a) = 1, e_2(b) = 1$
- $LVal = [e_1^+(\varphi), e_2^+(\varphi)] = [1, 1]$

Practică

Exercițiul 8: scrieți predicatele:

- `values_all_assigns(X,LVal)` care, pentru o formulă `X` determină lista tuturor evaluărilor,
- `is_taut(X)` care scrie 'este tautologie' dacă `X` este tautologie și 'nu este tautologie' în caz contrar.

Exemplu:

```
?- values_all_assigns(a imp b, LVal).
```

```
Lval = [1, 0, 1, 1]
```

```
?- is_taut(a imp b).
```

```
nu este tautologie
```

```
true
```

```
?- values_all_assigns(a imp a, LVal).
```

```
Lval = [1, 1]
```

```
?- is_taut(a imp a).
```

```
este tautologie
```

```
true
```