

Ugrađeni predikati

Testiranje tipova termova

<code>var(X)</code>	dokazano ukoliko je X trenutno promenljiva bez dodeljene vrednosti
<code>nonvar(X)</code>	dokazano ukoliko X nije promenljiva ili je X promenljiva koja ima vrednost
<code>atom(X)</code>	tačno ako je X trenutno atom
<code>integer(X)</code>	tačno ako je X trenutno ceo broj
<code>float(X)</code>	tačno ako je X trenutno realan broj
<code>number(X)</code>	tačno ako je X trenutno broj
<code>atomic(X)</code>	tačno ako je X trenutno atom ili broj
<code>compound(X)</code>	tačno ako je X trenutno struktura

Testiranje tipova termova

```
12 ?- var(Z), Z = 2.  
Z = 2.  
  
13 ?- Z = 2, var(Z).  
false.  
  
14 ?- integer(Z), Z = 2.  
false.  
  
15 ?- Z = 2, integer(Z), nonvar(Z).  
Z = 2.  
  
16 ?- atom(3.14).  
false.
```

```
17 ?- atomic(3.14).  
true.  
  
18 ?- atom(==>).  
true.  
  
19 ?- atom(p(1)).  
false.  
  
20 ?- compound(2+X).  
true.
```

Broj pojavljivanja atoma

```
count(_, [], 0).
count(A, [A|L], N):-
    !, count(A, L, N1),
    N is N1+1.
count(A, [_|L], N):-count(A, L, N).
```

```
count(_, [], 0).
count(A, [B|L], N):-
    atom(B), A = B,
    !, count(A, L, N1),
    N is N1+1.
count(A, [_|L], N):-count(A, L, N).
```

```
27 ?- count(a, [a,b,a,a], N).
N = 3 .
```

```
28 ?- count(a, [a,b,X,Y], N).
X = Y, Y = a,
N = 3 .
```

```
29 ?- count(b, [a,b,X,Y], N).
X = Y, Y = b,
N = 3 .
```

```
30 ?- L=[a,b,X,Y], count(a, L, Na), count(b, L, Nb).
L = [a, b, a, a],
X = Y, Y = a,
Na = 3,
Nb = 1 .
```

```
32 ?- count(a, [a,b,X,Y], N).
N = 1 .
```

Izgradnja i rastavljanje termova

=..

Cilj

Term=..L

je tačan ako je L lista koja sadrži funktor terma Term i listu njegovih argumenata

```
33 ?- f(a,b)=..L.  
L = [f, a, b].
```

```
34 ?- T=..[rectangle,3,5].  
T = rectangle(3, 5).
```

```
35 ?- Z=..[p,X,f(X,Y)].  
Z = p(X, f(X, Y)).
```

Zamena podtermova

Definisati predikat

`substitute(Subterm, Term, Subterm1, Term1)`

koji svaku pojavu Subterm u Term zamenjuje sa Subterm1 u rezultujućem termu Term1

```
37 ?- substitute(sin(x),2*sin(x)*f(sin(x)),t,F).  
F = 2*t*f(t) .
```

```
38 ?- substitute(a+b,f(a,A+B),v,F).  
A = a,  
B = b,  
F = f(a, v) .
```

Zamena podtermova

```
substitute(Term,Term,Term1,Term1):-!.
substitute(_,Term,_,Term):-
    atomic(Term),!.
substitute(Sub,Term,Sub1,Term1):-
    Term=..[F|Args],
    substlist(Sub,Args,Sub1,Args1),
    Term1=..[F|Args1].

substlist(_,[],_,[]).
substlist(Sub,[Term|Terms],Sub1,[Term1|Terms1]):-
    substitute(Sub,Term,Sub1,Term1),
    substlist(Sub,Terms,Sub1,Terms1).
```

Izračunavanje vrednosti simboličkog izraza

Definisati predikat

```
eval(Exp, SymbolValues, Val)
```

koji izračunava vrednost Val simboličkog izraza Exp za vrednosti simbola datih u listi SymbolValues

```
?- eval(x*sin((x+y)/2), [x = 1, y = 2.14], V).
```

```
V = 0.999999682931...
```


call, functor, arg

call je predikat čiji je argument cilj koji treba da se dokaže

```
Goal =.. [Funktor|Arglist]  
call(Goal)
```

Cilj

```
functor(Term, F, N)
```

je tačan ako je F funktor od Term i N arnost od F

Cilj

```
arg(N, Term, A)
```

je tačan ako je A N-ti argument Terma, pod pretpostavkom da je redni broj prvog argumenta 1

```
41 ?- functor(t(f(X),X,t),Fun,Arity).  
Fun = t,  
Arity = 3.  
  
42 ?- arg(2,f(X,t(a),t(b)),Y).  
Y = t(a).  
  
43 ?- functor(D,date,3),arg(1,D,29),  
arg(2,D,june),arg(3,D,1982).  
D = date(29, june, 1982).
```

Dodavanje i brisanje kaluza

Prološki program se može posmatrati kao relacionalna baza podataka

Ugrađenim predikatima baza se može ažurirati tokom izvršavanja

- Dodavanje novih klauza – assert, asserta, assertz
- Brisanje klauza – retract

Cilj oblika assert(C) je uvek tačan, a prateći efekat je dodavanje klauze C

Cilj oblika retract(C) je tačan ukoliko postoji klauza C i pri „dokazivanju“ klauza C se briše

- Ukoliko klauza C ne postoji, cilj ne može da se dokaže

Korišćenje ovih predikata je zgodno u situacijama gde se interakcijom menja status „sveta“

Dodavanje i brisanje kaluza

```
:-dynamic on/2.
```

```
on(a,b).  
on(b,table).  
on(c,table).
```

```
move(X,Y,Z):-  
    retract(on(X,Y)),  
    assert(on(X,Z)).
```

```
2 ?- move(a,b,c).  
true.  
  
3 ?- on(X,Y).  
X = b,  
Y = (table) ;  
X = c,  
Y = (table) ;  
X = a,  
Y = c.  
  
4 ?- retract(on(X,Y)),fail.  
false.  
  
5 ?- on(X,Y).  
false.
```

Dodavanje i brisanje kaluza

```
6 ?- assert(member(X,[X|_])),
    assert(member(X,[_|L):-member(X,L)).
true.

7 ?- member(2,[1,2,3]).
true .
```

asserta dodaje kaluzu na početak baze

assertz dodaje klauzu na kraj baze

assert u većini Prolog okruženja se ponaša kao assertz

asserta može da se koristi za čuvanje prethodno rešenih instanci problema

Dodavanje i brisanje kaluza

```
8 ?- assert(p(b)), assertz(p(c)), assert(p(d)), asserta(p(a)).
true.
```

```
10 ?- p(X).
X = a ;
X = b ;
X = c ;
X = d.
```

```
:-dynamic product/3.
```

```
maketable:-
```

```
    L=[0,1,2,3,4,5,6,7,8,9],
    member(X,L),
    member(Y,L),
    Z is X*Y,
    assert(product(X,Y,Z)),
    fail.
```

```
11 ?- maketable.
false.
```

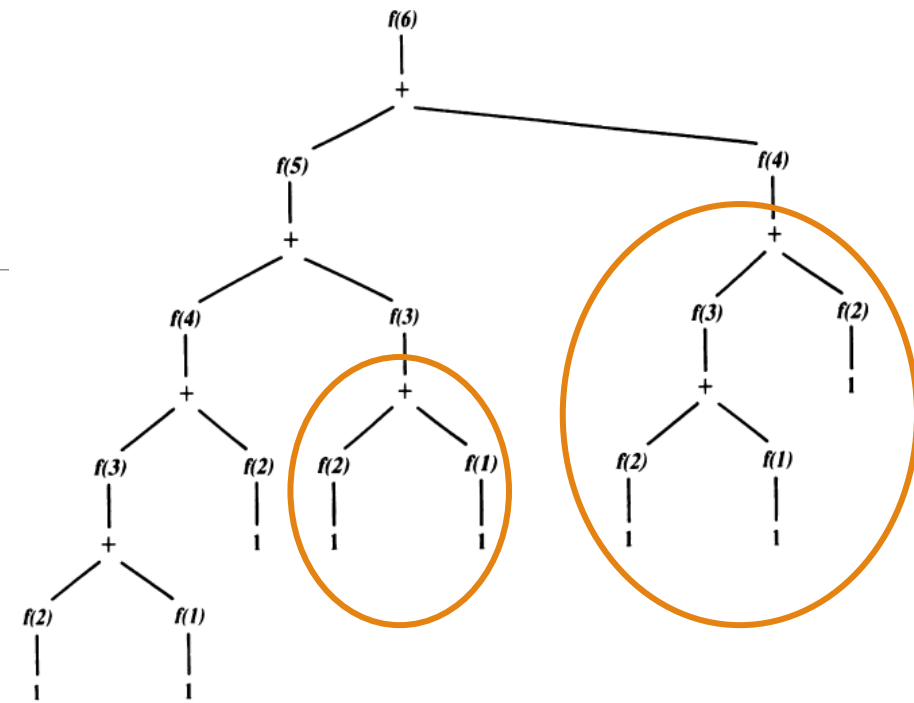
```
12 ?- product(X,Y,8).
X = 1,
Y = 8 ;
X = 2,
Y = 4 ;
X = 4,
Y = 2 ;
X = 8,
Y = 1.
```

Fibonačijev niz

```
fib(1,1).  
fib(2,1).  
fib(N,F):-N>2, N1 is N-1, fib(N1,F1),  
          N2 is N-2, fib(N2,F2), F is F1+F2.
```

```
:-dynamic fib2/2.
```

```
fib2(1,1).  
fib2(2,1).  
fib2(N,F):-N>2, N1 is N-1, fib2(N1,F1),  
          N2 is N-2, fib2(N2,F2), F is F1+F2,  
          asserta(fib2(N,F)).
```



```
?- fib(6,F).  
F = 8 .
```

(P -> Q ; R)

if-then-else konstrukcija -- if P then Q else R

Maksimum dva broja -- (X>=Y -> M=X ; M=Y)

```
2 ?- X=10, member(Y,[5,15]), (X>=Y -> M=X; M=Y).  
X = M, M = 10,  
Y = 5 ;  
X = 10,  
Y = M, M = 15.
```

(P -> Q ; R) :- P, !, Q.

(P -> Q ; R) :- R.

repeat

Cilj koji može uvek da se dokaže

```
repeat.
```

```
repeat:-repeat.
```

```
dosquares:-  
    repeat,  
    read(X),  
    ( X = stop,!  
      ;  
      Y is X*X,write(Y),nl,  
        fail  
    ).
```

```
4 ?- dosquares.  
|: 3.  
9  
|: 15.  
225  
|: stop.  
  
true.
```


bagof, setof,
findall

Sva rešenja koja zadovoljavaju neki cilj možemo da generišemo bektrekingom jedno po jedno

- Svaki put generiše se novo rešenje, a prethodno gubi

Cilj je prikupiti sva rešenja u listu

bagof, setof, findall

bagof(X, P, L) – generiše listu L svih objekata X koji zadovoljavaju cilj P

```
age(peter,7).  
age(ann,5).  
age(pat,8).  
age(tom,5).
```

```
6 ?- bagof(Child,age(Child,5),List).  
List = [ann, tom].  
8 ?- bagof(Child,age(Child,Age),List).  
Age = 5,  
List = [ann, tom] ;  
Age = 7,  
List = [peter] ;  
Age = 8,  
List = [pat].  
8 ?- bagof(Child,Age^age(Child,Age),List).  
List = [peter, ann, pat, tom].
```

Ukoliko ne postoji rešenje za P – dokazivanje nije uspešno

Ukoliko se isti objekat X pronađe više puta – lista L **ima duplikate**

bagof, setof, findall

setof(X, P, L) – generiše listu L svih objekata X koji zadovoljavaju cilj P

- Lista L je uređena
- Duplikati uklonjeni

Ne postoje ograničenja za vrstu objekata koji se prikupljaju

```
age(peter,7).  
age(ann,5).  
age(pat,8).  
age(tom,5).
```

```
10 ?- setof(Child, Age^age(Child, Age), ChildList),  
      setof(Age, Child^age(Child, Age), AgeList).  
ChildList = [ann, pat, peter, tom],  
AgeList = [5, 7, 8].
```

```
11 ?- setof(Age/Child, age(Child, Age), List).  
List = [5/ann, 5/tom, 7/peter, 8/pat].
```

bagof, setof, findall

`findall(X, P, L)` – generiše listu L svih objekata X koji zadovoljavaju cilj P

- Lista L sadrži sva rešenja X, čak i ako postoje razlike u vrednostima promenljivih u P, koje nisu deljene sa X
- Ukoliko ne postoji X koje zadovoljava P, tada je `L = []`

```
age(peter,7).  
age(ann,5).  
age(pat,8).  
age(tom,5).
```

```
12 ?- findall(Child,age(Child,Age),List).  
List = [peter, ann, pat, tom].
```

Input/Output

Tokom izvršavanja Prolog programa, mogu biti aktivna samo 2 fajla

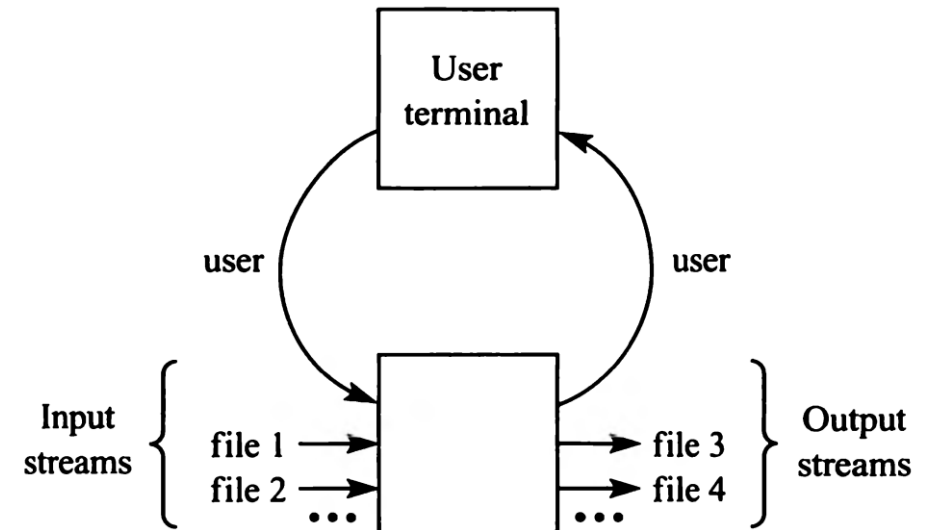
- jedan za *input stream*, i jedan za *output stream*

Promena input stream-a `-- see(FileName)`

Zatvaranje trenutnog input fajla `-- seen`

Promena output stream-a `-- tell(FileName)`

Zatvaranje trenutnog output fajla `-- told`



read, write

`read(X)` -- čita term iz trenutnog input stream-a

- `end_of_file` -- atom koji označava kraj ulaznog fajla

`write(X)` -- ispisuje term na trenutni output stream

`see(file1), read(X),`

`p(X, Y),`

`tell(file2), write(Y), write(.), ...`

`tab(N)` -- ispisuje N belina

`n1` -- ispisuje *new line*

Interakcija sa korisnikom

```
cube(N,C):- C is N*N*N.
```

```
cube:-  
    write('Next item, please '),  
    read(X),  
    process(X).  
process(stop):-!.  
process(N):-  
    C is N*N*N,  
    write('Cube of '),write(N),write(' is '),  
    write(C),nl,  
    cube.
```

```
15 ?- cube(5,Y).  
Y = 125.
```

```
16 ?- cube(12,Y).  
Y = 1728.
```

```
24 ?- cube.  
Next item, please 5.  
Cube of 5 is 125  
Next item, please |: 12.  
Cube of 12 is 1728  
Next item, please |: stop.  
true.
```

Interakcija sa korisnikom

```
cube1:-
    write('Next item, please '),
    ( read(stop),!
      ;
      read(N),
      C is N*N*N,
      write('Cube of '),write(N),write(' is '),
      write(C),nl,
      cube1).
```

```
26 ?- cube1.
Next item, please 5.
 |: 5.
Cube of 5 is 125
Next item, please |: 12.
 |: 12.
Cube of 12 is 1728
Next item, please |: stop.
true.
```


Učitavanje/ispis karaktera

- `put(C)` -- na trenutni output stream ispisuje karakter čija je ASCII vrednost C
- `get0(C)` -- učitava karakter sa input stream-a i C dobija ASCII vrednost učitanoog karaktera
- `get(C)` -- učitava ne-blanko karaktere

```
37 ?- put(65),put(66),put(67).  
ABC  
true.  
  
38 ?- get0(A),get(B).  
|: a  
  
A = 32,  
B = 97.
```

Rešavanje slagalice

Seminarski Ljubomira Matovića