3.4 Poza predykatów zbudowanych

(i) "Sprawdzanie" typu

Z is X + Y.

integer(X), integer(Y), Z is X + Y.

var(X), nonvar(X),
integer(X), real(X),
atom(X), jeżeli X = anna, ...
X = ==>, ...
X = 'Tom', ...

atomic(X), jeżeli X atom lub
X jest integer lub real

Przykład:

? var(Z), Z = 2.
Z = 2

? Z = 2, var(Z).
no

? - integer(Z), Z = 2.
no

? - var(Z), Z = 2, integer(Z), nonvar(Z).
Z = 2

```
?- atom(3).
   no
?- atomic(3).
   yes
?- atomic(f(a)).
   no
```

Przykład: Ile razy wystąpi A w liście L?

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

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

```
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).
```

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

ale

```
?- count(X, [a, b, X, Y], N).
   N = 0.
```

(ii) Dekompozycja termów

```
Term =.. L
```

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

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

Przykład: Figury geometryczne

square(Side)
triangle(Side1, Side2, Side3)
circle(R)

enlarge(square(A), F, square(A1)) :- A1 is F*A.
enlarge(circle(R), F, circle(R1)) :- R1 is F*R.
enlarge(rectangle(A,B), F, rectangle(A1,B1))
:- A1 is F*A, B1 is F*B.

figure(G, [A1, ... An]) ?

enlarge(figure(G,[]), F, figure(G,[])).
enlarge(figure(G,[X1|L1]), F, figure(G,[X2|L2]))
:- X2 is F*X1,
enlarge(figure(G,L1), F, figure(G,L2)).

lepiej:

enlarge(Type(Par1), F, Type(Par2)) :-
Par2 is F*Par1.

ale teraz Type jest zmienną ↯
bo Prolog jest językiem pierwszego rzędu.

```
enlarge(Fig1, F, Fig2) :-
    Fig1 =.. [Type | Parameters1],
    multiply_list(Parameters1, Parameters2),
    Fig2 =.. [Type | Parameters2].
```

Prezy Vlad:

(i) `?- fpresent(father, [mother(a,b), father(c,b)]).`

yes

`?- fpresent(father, [mother(a,b)]).`

no

```
fpresent(A, [X|_]) :- X =.. [A|_], !.
fpresent(A, [_|L]) :- fpresent(A, L).
```

(ii)

```
animate(fred).
animate(alice).
male(fred).
canine(alice).
feline(fred).
```

`?- checkprops(fred, [animate, male]).`

yes

`?- checkprops(alice, [canine, feline]).`

no

```
checkprops(_,[]) :- !.
checkprops(A,[X,L]) :-
    T =.. [X,A],
    call(T),
    checkprops(A,L).
```

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

```
?- functor(t(f(X),X,a),F,N).
   F = t, N = 3.
? arg(2,f(X,t(a),t(b)),Y).
   Y = t(a).
```

```
? functor(D,date,3),
   arg(1,D,29),
   arg(2,D,april),
   arg(3,D,2015).
D = date(29,april,2015).
```

(iii) Manipulacja predykatów

assert(C)

retract(C)

Przykład:

```
fast(ann).
slow(tom).
slow(pat).

?- assert((fastes(X,Y):-fast(X), slow(Y))).
yes
?- fastes(A,B).
    A = ann, B = tom;
    A = ann, B = pat
?- retract(slow(X)).
    X = tom;
    X = pat;
    no
?- fastes(ann,A).
no
```

```
assert a(C).
assert z(C).
```

Przykład:

```
product(X,Y,Z) :- Z is X*Y.

maketable :-
    L = [0,1,2,3,4,5,6],
    member(X,L), member(Y,L),
    Z is X*Y,
    assert(product(X,Y,Z)),
    fail.

?- maketable.
no
```

(iv) kolekcjonowanie rozwiązań

bagof(X,P,L).

lista elementów X, które spełniają P.

Przykład:

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

?- age(Child,5).
    X = ann;
    X = tom;
    no
```

```
?- bagof(Child, age(Child,5), L).
   L = [ann,tom].

?- bagof(Child, age(Child,Age), L).
   Age = 7, L = [pete] ;
   Age = 5, L = [ann,tom]

?- bagof(Child, Age^age(Child,Age), L).
   L = [pete,ann,pat,tom].
```

```
findall(X,Goal,List) :-
    call(Goal), asserta(q(X)), fail.
findall(X,Goal,List) :-
    asserta(q(bottom)), collect(List).

collect(List) :-
   retract(q(X)), !,
   ((X == bottom, !, L = [])
    ;
    (L = [X|R], collect(R))).
```