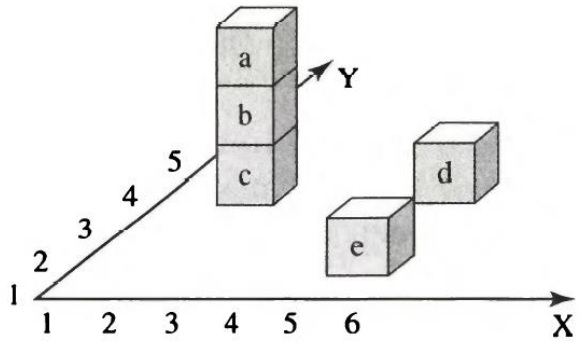
# Zaključivanje u Prologu

# Robot

Robot u skladištu ima kameru kojom može da detektuje objekte

- Objekti su oblika kocke jedinične ivice
- Robot detektuje oznake na objektima i njihove pozicije
- Koordinate u skladištu se celi brojevi




Za objekte na slici robot može da detektuje objekte a, d i e i vodi evidenciju na čemu koji objekat stoji

```
see(a,2,5).
see(d,5,5).
see(e,5,2).
```

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

# Robot

```
see(a,2,5).
see(d,5,5).
see(e,5,2).
```

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

```
2 ?- on(Block,_).
Block = a ;
Block = b ;
Block = c ;
Block = d ;
Block = e.
```

```
3 ?- see(B1,_,Y),see(B2,_,Y).
B1 = B2, B2 = a,
Y = 5 ;
B1 = a,
Y = 5,
B2 = d ;
B1 = d,
Y = 5.
```

```
4 ?- see(B1,_,Y),see(B2,_,Y),B1\=B2.
B1 = a,
Y = 5,
B2 = d ;
B1 = d,
Y = 5,
B2 = a ;
false.
```

```
5 ?- on(B,_),on(_,B).
B = b ;
B = c ;
false.
```

# Robot

```
see(a,2,5).
see(d,5,5).
see(e,5,2).
```

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

```
z(B,0):-on(B,table).
z(B,Z):-on(B,B0),z(B0,Z0),Z is Z0+1.
```

```
zz(B,0):-on(B,table).
zz(B,Z0+1):-on(B,B0),zz(B0,Z0).
```

```
16 ?- z(a,Z).
Z = 2 .

17 ?- z(a,Z).
Z = 2 ;
false.

17 ?- zz(a,Z).
Z = 0+1+1 .

18 ?- zz(a,ZFormula),Z is ZFormula.
ZFormula = 0+1+1,
Z = 2 .
```

# Unifikacija

Dva terma se mogu unifikovati

- Ako su identični
- Ako se varijablama koje se pojavljuju u termima mogu se pridružiti objekti, tako da zamenom vrednosti tih varijabli, termi postanu identični

# Unifikacija

date(D,M,2021) i date(D1,may,Y1) se mogu unifikovati

- D -> D1
- M -> may
- Y1 -> 2021

```
19 ?- date(D,M,2021)=date(D1,may,Y1).
D = D1,
M = may,
Y1 = 2021.
```

Ovi termi će postati identični i za sledeće vrednosti promenljivih

- D =1
- D1 = 1
- M = may
- Y1 = 2021

D = third
D1 = third
M = may
Y1 = 2021

```
20 ?- date(D,M,2021)=date(D1,may,Y1),
      date(D,M,2021)=date(15,M,Y).
D = D1, D1 = 15,
M = may,
Y1 = Y, Y = 2021.
```

# Algoritam unifikacije

Ako su S i T konstante, tada se S i T mogu unifikovati samo ako su isti objekat

Ako je S promenljiva i T ima ma koju vrednost, tada se mogu unifikovati ako se S dodeli vrednost T.

Ako su S i T strukture, one se mogu unifikovati samo ako

- S i T imaju isti predikat
- I svi njihovi argumenti se mogu unifikovati

triangle(point(1,1), A, point(2,3)) = triangle(X, point(4,Y), point(2,Z))

- triangle = triangle
- X = point(1,1)
- A = point(4,Y)
- point(2,3) = point(2,Z)
  - Z = 3

# Deklarativano vs. proceduralno značenje Prološkog programa

P:-Q,R. - P, Q, R imaju sintaksu termova

Deklarativno tumačenja klauze:

- P je tačno ako su Q i R tačni
- Iz Q i R sledi P

Proceduralno tumačenja klauze:

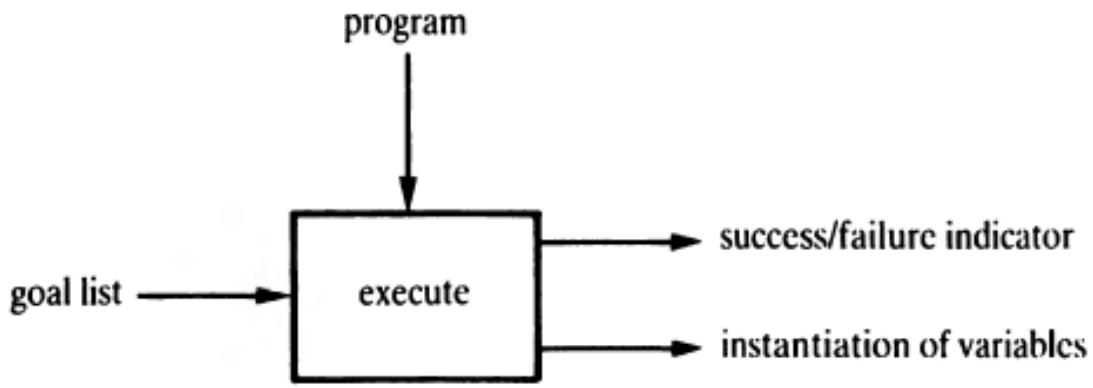
- Da bi se rešio problem P, najpre mora da se reši potproblem Q, a potom potproblem R.

U proceduralnom tumačenju ne određuju se samo ciljevi, već i redosled realizacije cilja.

# Proceduralno značenje

Izlazni podaci:

- Indikator uspeha – success/failure
- Generisane vrednosti u slučaju uspešnog izvršavanja



# Koraci izvršavanja

```
big(bear).
big(elephant).
small(cat).
brown(bear).
black(cat).
gray(elephant).
dark(Z):-black(Z).
dark(Z):-brown(Z).
```

```
22 ?- dark(X),big(X).
X = bear.
```

Inicijalna lista ciljeva: dark(X), big(X)

Scan (top to bottom) – dark(X)
- dark(Z):-black(Z).
- Nova lista ciljeva black(X),big(X)

Scan – black(X)
- black(cat) -> X = cat
- Nova lista ciljeva big(cat) -> fail
- Lista ciljeva black(cat),big(cat) -> fail
- Lista ciljeva black(X) -> fail
- Lista ciljeva dark(X), big(X)

Scan – dark(X)
- dark(Z):-brown(Z).
- Nova lista ciljeva brown(X),big(X)

Scan – brown(X)
- brown(bear) -> X = bear
- Nova lista ciljeva: big(bear)

# Beskonačna petlja

p:-p.

- p je tačno, ako je p tačno

?- p.

Program je deklarativno ispravan, a proceduralno neispravan

U pojedinim slučajevima promenom redosleda klauza i ciljeva može se izbeći beskonačna petlja.

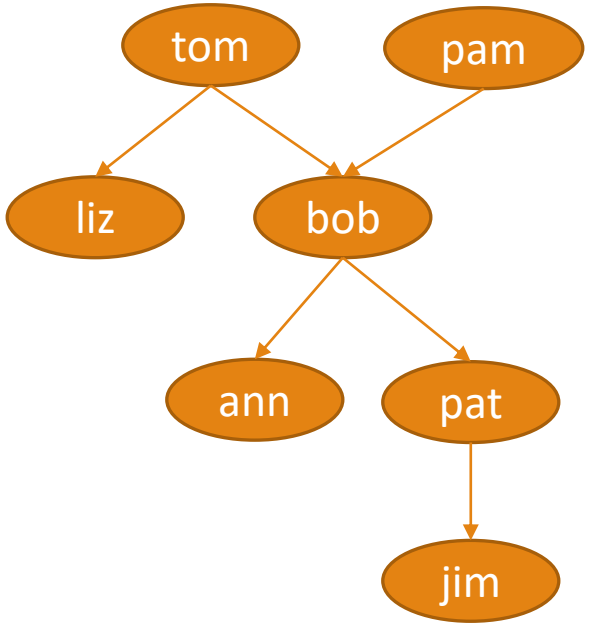
# Izbegavanje beskonačne petlje

```
anc1(X,Z):-parent(X,Z).
anc1(X,Z):-parent(X,Y),
           anc1(Y,Z).
```

```
anc2(X,Z):-parent(X,Y),
           anc2(Y,Z).
anc2(X,Z):-parent(X,Z).
```

```
anc3(X,Z):-parent(X,Z).
anc3(X,Z):-anc3(X,Y),
           parent(Y,Z).
```

```
anc4(X,Z):-anc4(X,Y),
           parent(Y,Z).
anc4(X,Z):-parent(X,Z).
```




```
28 ?- anc1(tom,pat).
true .

29 ?- anc2(tom,pat).
true .

30 ?- anc3(tom,pat).
true .

31 ?- anc4(tom,pat).
ERROR: Stack limit (1.0Gb
ERROR:   Stack sizes: loc
```

# Liste

# Reprezentacija listi

[ann, tennis, tom, skiing]

- ann – glava liste
- [tennis, tom,skiing] – rep liste

[Head|Tail]

```
36 ?- [a,b,c] = [a|[b|[c|[]]]].
true.

36 ?- List1 = [a,b,c], List2 = [a|[b|[c|[]]]].
List1 = List2, List2 = [a, b, c].

37 ?- Hobbies1 = [tennis|[music|[]]],
      Hobbies2 = [skiing,food],
      L = [ann,Hobies1,tom,Hobies2].
Hobbies1 = [tennis, music],
Hobbies2 = [skiing, food],
L = [ann, Hobies1, tom, Hobies2].
```

L = [ann,[tennis,music],tom,[skiing,food]]

# Element liste

X je element liste L ako je

- X glava liste L ili je
- X element repa liste L

```
member(X,[X|Tail]).
member(X,[Head|Tail]):-
                member(X,Tail).
```

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

```
41 ?- member(b,[a,b,c]).
true .

42 ?- member(X,[a,b,c]).
X = a ;
X = b ;
X = c ;
false.

43 ?- member(a,L).
L = [a|_370] ;
L = [_1028, a|_1036] ;
L = [_1028, _1694, a|_1702] ;
L = [_1028, _1694, _2360, a|_2368] .

44 ?- member(a,L),member(b,L),member(c,L).
L = [a, b, c|_4632] ;
L = [a, b, _5290, c|_5298] ;
L = [a, b, _5290, _5956, c|_5964] ;
L = [a, b, _5290, _5956, _6622, c|_6630] ;
L = [a, b, _5290, _5956, _6622, _7288, c|_7296] .

45 ?- L=[_,_,_],member(a,L),member(b,L),member(c,L).
L = [a, b, c] ;
L = [a, c, b] ;
L = [b, a, c] ;
L = [c, a, b] ;
L = [b, c, a] ;
L = [c, b, a] ;
false.
```

# Nadovezivanje lista

```
conc([],L,L).
conc([X|L1],L2,[X|L3]):-
                    conc(L1,L2,L3).
```

```
?- conc(L1,L2,[a,b,c]).
L1 = [],
L2 = [a, b, c] ;
L1 = [a],
L2 = [b, c] ;
L1 = [a, b],
L2 = [c] ;
L1 = [a, b, c],
L2 = [] ;
false.
```

```
?- conc([a,b],[c,d],X).
X = [a, b, c, d].
```

```
?- conc([a,b,c],[1,2,3],L).
L = [a, b, c, 1, 2, 3].
```

```
?- conc([a,[b,c],d],[a,[],b],L).
L = [a, [b, c], d, a, [], b].
```

```
?- conc(Before,[may|After],
|    [jan,feb,mar,apr,may,jun,jul,avg,sep,oct,nov,dec]).
Before = [jan, feb, mar, apr],
After = [jun, jul, avg, sep, oct, nov, dec] .
```

```
?- L1=[a,b,z,z,c,z,z,z,d,e],conc(L2,[z,z,z|_],L1).
L1 = [a, b, z, z, c, z, z, z, d|...],
L2 = [a, b, z, z, c] .
```

```
member1(X,L):-conc(_,[X|_],L).
```

# Operacije sa listama

Poslednji element liste

Prvi element u listi iza X

Brisanje svih pojava elementa X u listi

Zamena elementa X sa A

Podlista liste

Brisanje duplikata liste

Permutacije liste

# Aritmetika

# Aritmetika

Operacije

- +, -, *, /,
- ** (stepenovanje),
- // (celobrojno deljenje),
- mod

Relacije

- >, <, >=, =<
- =:=
- =\=

```
47 ?- X = 1+2.
X = 1+2.

48 ?- X is 1+2.
X = 3.
```

```
49 ?- 1+2 =:= 2+1.
true.

50 ?- 1+2=2+1.
false.

51 ?- 1+A=B+2.
A = 2,
B = 1.
```

```
14 ?- X is 2, Y is X+2,Y\=X.
X = 2,
Y = 4.

15 ?- X is 2, Y is X+2,Y=\=X.
X = 2,
Y = 4.

16 ?- X = a(1,2), Y = a(2,3), X\=Y.
X = a(1, 2),
Y = a(2, 3).

17 ?- X = a(1,2), Y = a(2,3), X=\=Y.
ERROR: Arithmetic: `a/2' is not a funct
ERROR: In:
ERROR:   [11] a(1,2)=\=a(2,3)
```

# Rebusi

```
  DONALD
+ GERALD
---------
  ROBERT
```

```
donald([D,O,N,A,L,G,E,R,B,T]):-
    assign([0,1,2,3,4,5,6,7,8,9],[D,O,N,A,L,G,E,R,B,T]),
    100000*D + 10000*O + 1000*N + 100*A + 10*L + D +
    100000*G + 10000*E + 1000*R + 100*A + 10*L + D =:=
    100000*R + 10000*O + 1000*B + 100*E + 10*R + T.
```

```
assign(_,[]).
assign(Digs,[D|Vars]):-
    del(D,Digs,Digs1),
    assign(Digs1,Vars).
```

```
del(X,[X|Tail],Tail).
del(X,[Y|Tail],[Y|Tail1]):-
    del(X,Tail,Tail1).
```

# Rebusi

```
   DONALD
+  GERALD
---------
   ROBERT
```

```
2 ?- donald([D,O,N,A,L,G,E,R,B,T]),
     L1 = [D,O,N,A,L,D], L2 = [G,E,R,A,L,D], L3 = [R,O,B,E,R,T].
D = 5,
O = 2,
N = 6,
A = 4,
L = 8,
G = 1,
E = 9,
R = 7,
B = 3,
T = 0,
L1 = [5, 2, 6, 4, 8, 5],
L2 = [1, 9, 7, 4, 8, 5],
L3 = [7, 2, 3, 9, 7, 0] .
```

Napraviti opštiji predikat koji rešava ma koji rebus

```
  SEND
+ MORE
-------
 MONEY
```

Problem 8 (ili n) dama na šahovsku tablu 8x8 (ili nxn)