6. Sistemski (ugrađeni) predikati

To su unapred definisani predikati i ugrađeni u sam PROLOG.

Već smo upoznali neke od njih: !, fail, not, +, *, ...

Odgovaraju rezervisanim rečima u drugim programskim jezicima.

Većina njih se teško može realizovati u samom PROLOG-u.

Bez sistemskih predikata PROLOG je čist logički (deklarativni) jezik.

Dodavanjem sistemskih predikata PROLOG postaje programski jezik

Izazivaju bočne efekte i doprinose proceduralnim karakteristikama PROLOG-a.

Najveći broj njih (zavisno od prirode sistemskog predikata) zahteva konkretne argumente (konkretizovane promenljive). Na primer,

X < Y, zahteva konkretizovane promenjjive X i Y.

Svaka verzija Prologa ima neke specifične sistemske predikate.

6.1. Sistemski predikati za rad sa datotekama PROLOG-programa

Uobičajen način implementiranje PROLOG-sistema.

1. **user**-datoteka

2. **consult(X).**

Argument X mora biti konkretizovan na ime postojeće datoteke.

Klauzule iz datoteke X dodaje klauzulama radne datoteke.

3. **reconsult(X).**

Za uspeh ovog predikata važe iste pretpostavke kao i za prethodni. Efekat: brišu se tvrđenja u radnoj datoteci koja imaju istu glavu i istu arnost kao tvrđenja u datoteci X. Tvrđenja iz datoteke X dodaju se radnoj datoteci. Pogodan za korekcije sadržaja radne datoteke.

4. **[fajl1, fajl2, fajl3, ...]** -uzastopna primena consult-predikata na datoteke: fajl1, fajl2, fajl3, …

5. **[-fajl1, -fajl2, -faj3, …]** - uzastopna primena predikata reconsult na fajl1, fajl2, fajl3, ...

Prethodni zapisi mogu da se kombinuju. Ima smisla:

?-[-prva, druga, -treca, cetvrta].

6. file_list(X). Sadržaj radne datoteke upisuje u datoteku X na neku spoljašnju memoriju. (X može predstavljati putanju).

7. delete(X). X predstavlja ime datoteke. Briše datoteku sa zadatim imenom.

Primeri (primena na raznim datotekama).

6.2. Predikati true i fail

1. **true** - predikat koji uvek uspeva.

2. **fail -** predikat koji uvek propada.

Primeri:

```
razliciti(X,X) :- fail, !.
razliciti(X,Y).          ekv.   razliciti(X,Y):-X=Y,!,fail; true.
not(P) :- P,!, fail; true.
```

6.3. Predikati za klasifikaciju terma

1. **var(X).** Uspeva ako je X nekonkretizovana promenljiva.
   Primeri: ?- var(X). ?-var(55). ?- X=Y, Y=3, var(X).
   yes no no
2. **nonvar(X).** Inverzan prethodnom. Može se realizovati ovako:
   nonvar(X) :- var(X),!,fail.
   nonvar(_).
3. **atom(X).** Saglasan sa bazom ako je tekuća vrednost X-a atom.
   Primeri: ?-atom(55). ?- atom("niz"). ?-atom(pera).
   Brojac.pl no no yes
4. **integer(X).** Saglasan sa bazom ako je vrednost X-a ceo broj.
   Primeri: ?-integer(123). ?-integer(X). ?- integer('124').
   yes no no
5. **atomic(X).** Saglasan sa bazom ako je vrednost X-a atom ili ceo broj
   Može se realizovati ovako: atomic(X) :- atom(X).
   atomic(X):- integer(X).

6.4. Predikati za rad sa klauzulama kao sa termima

Klauzule u PROLOG-u mogu da se tretiraju kao strukture i sa njima se može raditi kao sa termima. Činjenica:

voli(marko, vino).

može se razmatrati kao struktura sa funktorom voli.

Pravilo:

voli(marko, X) :- voli(X, vino).

može se tretirati kao struktura sa funktorom :- i dva argumenta:

:-(voli(marko, X), voli(X, vino)).

Svaka činjenica može da se tretira kao pravilo.

voli(marko, vino). ekv. voli(marko, vino) :- true.

Na taj način je moguće jednobrazno obrađivati činjenice i pravila.

1. **listing**. Prikazuje na standardnom izlazu tvrđenja radne datoteke

2. **asserta(X).** Ubacuje klauzulu X na početak radne datoteke.

3. **assertz(X).** Ubacuje tvrđenje X na kraj radne datoteke.

4**. retract(X).** Briše tvrđenje X iz radne datoteke. (U Ariti-PROLOG-u uklanja prvo pojavljivanje klauzule X u bazi.)

5**. clause(X,Y).** Argument X unifikuje sa glavom tvrđenja iz radne datoteke, a Y sa telom tog tvrđenja. X ne može biti promenljiva (mora biti struktura), a Y je obično promenljiva.

Neka je u bazi znanja sledeća činjenica:

voli (pera, med).

?- clause(voli(X,Y), Z).

X=pera, Y=med, Z=true.

yes

6. **halt**. Izlaz iz interpretatora.

7. **cls.** Briše ekran.

8. **shell**. Privremeni izlaz u shell. (Povratak u Prolog se realizuje preko komande: exit.)

9. **listing(X).** Lista sve klauzule sa imenom X.

10. **listing(X/atnost).** Lista sve klauzule sa glavom X zadate arnosti.

11. **assert(X).** Ubacuje klauzulu X u radnu datoteku. (Obično ima isti efekat kao assertz(X).)

12. **abolish(X/arnost).** Uklanja iz baze podataka sve klauzule sa glavom X zadate arnosti.

Demonstracija opisanih sistemskih predikata na konkretnim primerima.

6.5. Predikati za rad sa strukturama

1. **functor(S,F,N).** Uspeva ako je F funktor strukture S, a N broj njenih argumenata. Ima dvostruku ulogu:

- nalazi funktor strukture i broj argumenta i

- formira novu strukturu za zadati funktor i broj argumenata.

Primeri:

?-functor(mis(a, b), F, N).

F=mis, N=2.

?-functor([a,b,c],a,3).

no

?- functor(jabuka, F, N).

F= jabuka, N=0.

?-functor([a,b,c],F,N).

F= . , N=2.

?-functor(X,a,3).

S= a(_123, _124, _125).

?-functor(a+b, F, N).

F= +, N=2.

2. **arg(N,S,A).** Uspeva ako se N-ti argument strukture S unifikuje sa A. Prvi i drugi argument moraju biti konkretizovani.

?- arg(2, roditelji(marko, marija, milan), X).

X=marija.

?-arg(2, [a,b,c,d], X).

X=[b,c,d].

3. **S =.. L (=.. Juniv-predikat)** Strukturu S prevodi u listu L, i obrnuto, listu L prevodi u strukturu S. Funktor strukture odgovara glavi liste, a argumenti strukture odgavaraju komponentama repa liste.

?-proizvod(a,b,c) =.. L.	?-X=..[a,b,c,d].	?- (a+b) =.. L
L=[proizvod,a,b,c].	X=a(b,c,d).	L=[+, a,b]
yes	yes	yes

4.**name(A,L).** Prevodi atom A u listu L ASCII-kodova slova iz kojih se sastoji atom A i obrnuto (prevodi listu ASCII-kodova u atom).

Primeri:

?-name(alp, X). ?- name(X,[109,97,114,107,111]).

X=[97,108,112] X= marko.

Slucajan.ari

Stabilna.ari

Organizacija.ari

6.6. Predikati repeat i call

repeat - obezbeđuje dodatne mogućnosti za nalaženje skupa rešenja.

repeat.

repat :- repeat.

call(X) - uspeva ako uspeva X.

6.7. Operatori jednakosti i nejedankosti

Ovde je reč o sistemskim predikatima koji su realizovani kao operatori. Neke od njih smo već opisali.

1. **X = Y** - opeartor unifikacije

2. **X \ = Y** - inverzan opertoru unifikacije

3. **X is Y** - operator dodeljivanja vredosti aritmetičkog izraza X-u.

4. **X =:= Y** - uspeva ako su jednake vrednosti aritmetičkih izraza X i Y.

5. **X =/=Y** - suprotan prethodnom. Uspeva ako su vrednosti aritmetičkih izraza X i Y različite.

6. **X = = Y** - uspeva ako su termi X i Y identični, tj. imaju jednaku strukturu pri čemu su odgovarajuće komponente poklapaju.

7. **X \ = = Y** - uspeva ako su X i Y različiti (neidentični) termi.

Primeri:

?- 2+X = = 2+3.
no

?- 2+X =:= 2+3.
no (Error…)

?- 2+X = 2+3.
X=3. (yes)

?- 2+3 = = 7- 2.
no

?- 2+3 =:= 7-2.
yes

?- 2+3 = 7 - 2.
no

?- s(X,p) = = s(X, p).
X= _0123 (yes)

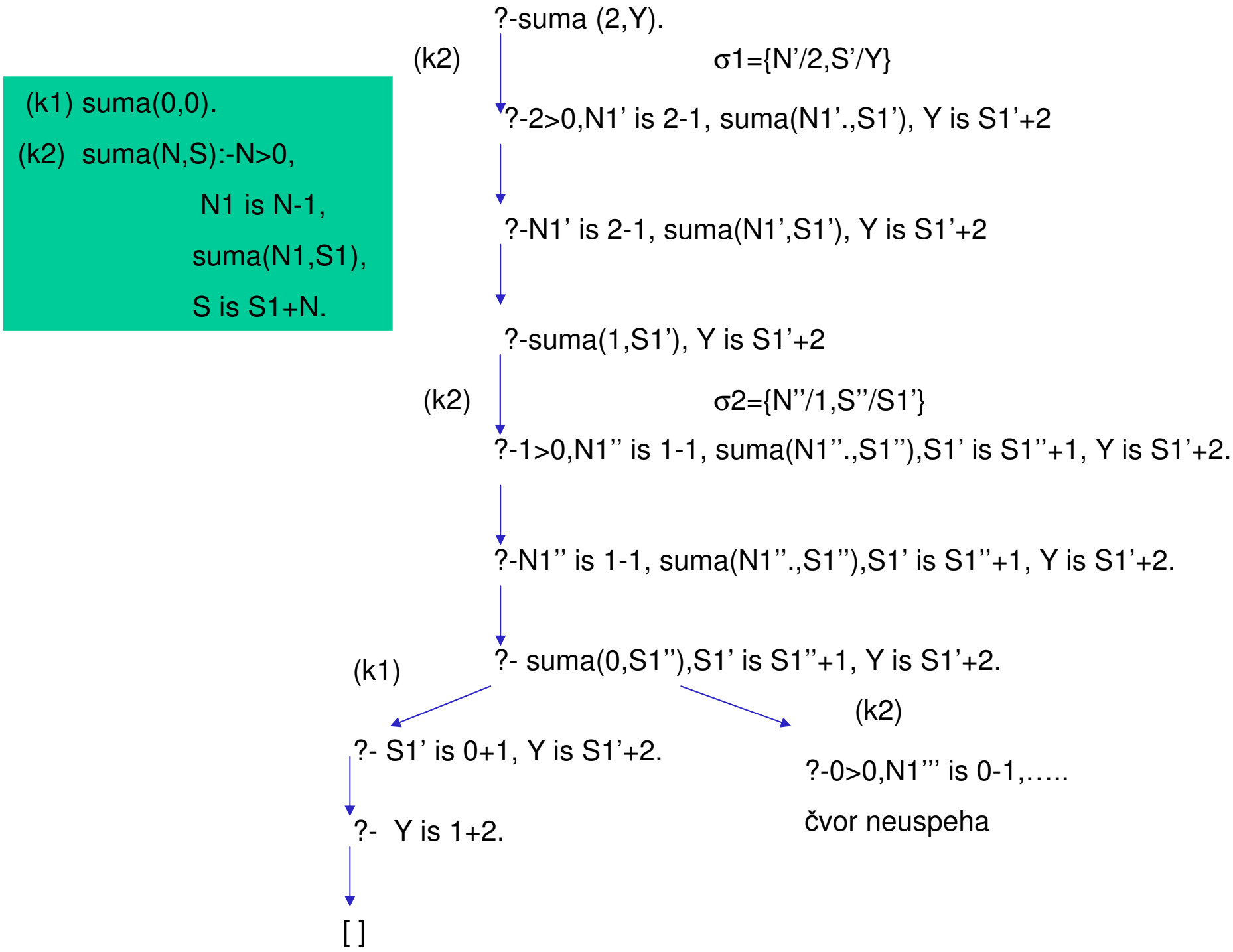
?- s(X,p) =:=s(X,p).
no

?- s(X,p)=s(X,p).
X=_0123 (yes)

?- X=Y, X = = Y.
X=_0084, Y=_0084.
yes

?-X=Y, X=:=Y.
no

?- X=Y.
X=_0084, Y=_0084.
yes


(k1) suma(0,0).
(k2) suma(N,S):-N>0,
N1 is N-1,
suma(N1,S1),
S is S1+N.
?-suma (2,Y).
(k2)
σ1={N'/2,S'/Y}
?-2>0,N1' is 2-1, suma(N1'.,S1'), Y is S1'+2
?-N1' is 2-1, suma(N1',S1'), Y is S1'+2
?-suma(1,S1'), Y is S1'+2
(k2)
σ2={N''/1,S''/S1'}
?-1>0,N1'' is 1-1, suma(N1''.,S1''),S1' is S1''+1, Y is S1'+2.
?-N1'' is 1-1, suma(N1''.,S1''),S1' is S1''+1, Y is S1'+2.
?- suma(0,S1''),S1' is S1''+1, Y is S1'+2.
(k1)
(k2)
?- S1' is 0+1, Y is S1'+2.
?- Y is 1+2.
[ ]
?-0>0,N1''' is 0-1,.....
čvor neuspeha

```
(k1) suma(N,S):-suma(N,S,0).
(k2) suma(0,S,S).
(k3) suma(N,S,T):-N>0,
                  T1 is T+N,
                  N1 is N-1,
                  sum(N1,S,T1).

suma(0,S,T):-S is T
```