LOGIČKO PROGRAMIRANJE

I

PROGRAMSKI JEZIK PROLOG

Literatura

1.W.F. Clocksin, C.F. Mellish: *Programmingin PROLOG*, Springer Verlag

2. I. Bratko: *PROLOG Programming for Artificial Intelligence*, Addison-Wesley, 1990

3. D. Grookes: *Introduction to Programming in PROLOG*, Prentice-Hall, 1988.

4. M. Radovan: *Programiranje u PROLOG-u*, Informator, Zagreb, 1988.

5. D. Tošić, R. Protić: *PROLOG kroz primere*, Tehnička knjiga, Beograd, 1991.

6. B. Bajković, M.Đurišić, S.Matković: *PROLOG i logika*, Krug, Beograd, 1998.

1. Razvoj PROLOG-a - kratak istorijski pregled

1.1. Predistorija:

1879. G. Frege - Predikatski račun (sredstvo za analizu formalne strukture čistog mišljenja)

1915-1936. Gedel, Erban, Skulem, Turing,...Osnovi teorije izračunljivosti.

1960. M Foster, T. Elkok, ABSYS (Aberdeen SYStem)
Radi sa tvrđenjima (aksiomama) preko kojih se, nakon postavljanja pitanja, deduktivnim putem generiše odgovor.

1965. J.A. Robinson, Metod rezolucije.

1.2. Istorija

1971. Pod rukovodstvom A. Colmerauer-a u Marselju kreiran Q-System.

1972. Q-System preimenovan u PROLOG. (Saradnici: Ph. Roussel, R. Pasero, J. Trudel)

1974. Na kongresu IFIP-a R. Kavalski predstavio PROLOG široj javnosti.

1977. D. Warren naprvio implementaciju PROLOG-a za DEC-10 u Edinburgu. (Edinburški PROLOG).

1981. Seminari u Sirakuzi i Los Andjelesu.

1982. Prva međunarodna konferencija o PROLOG- u Marselju.

1983. Japanski projekat o razvoju računara 5. generacije.

1993. Završen Japanski projekata razvoja računara 5. generacije.

Značajna imena: A. Colmerauer, J. Robinson, R. Kavalski i D. Warren.

2. O PROLOG-u

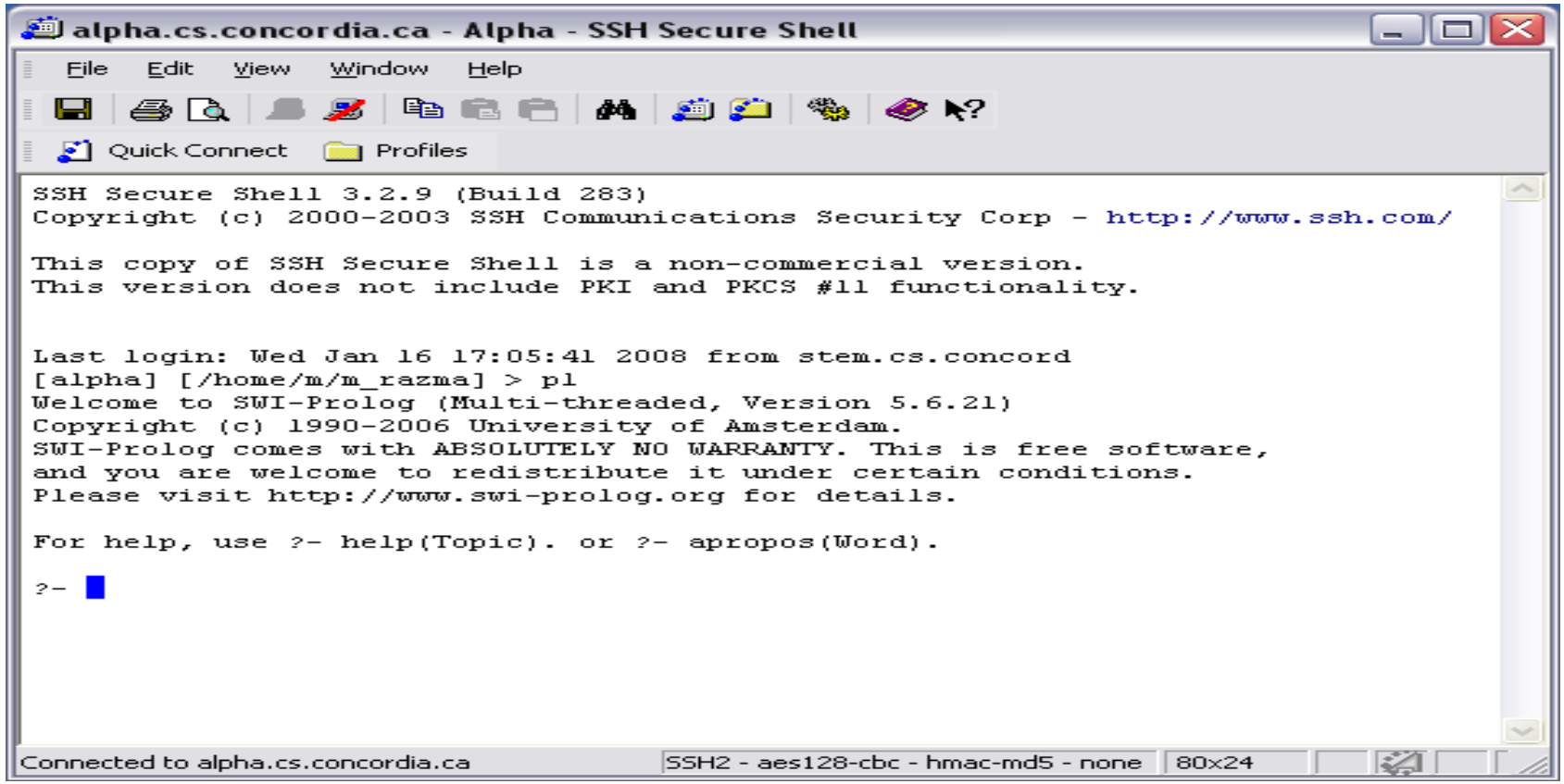
PROLOG skraćenica od PROgramming in LOGic

To je deklarativam programski jezik namenjen rešavanju zadataka simboličke prirode. Pogodan za:

- podršku relacionim bazama podataka,
- rešavanje problema matematičke logike
- rešavanje apstraktnih problema
- razumevanje prirodnih jezika,
- automatizaciju projektovanja,
- simboličko rešavanje jednačina
- razne oblasti veštačke inteligencije

PROLOG interpreter

- SWI-prolog, razvijen na Swedish Institute of Computer Science



PROLOG interpreter

- Interpreter može da učitava PROLOG datoteke ili da izvršava upite
- Izlaz iz interpretera: halt.
- PROLOG datoteke obično imaju ekstenziju .pl

U PROLOG-u se opisuju objekti i relacije među njima. Pošto je akcenat na relacijama, reč je o relacionom programskom jeziku.

Zasnovan na jednobraznoj strukturi podataka - termu. (Ne pravi se razlika između programa i podataka.)

Program na PROLOG-u je skup tvrđenja koja predstavljaju:

- činjenice o nekim objektima ili

- pravila koja pokazuju kako je rešenje povezano sa datim činjenicama, tj. kako se iz njih izvodi.

Programiranje u PROLOG-u sastoji se u:

- obezbeđivanju (proglašavanju) činjenica o objektima i odnosima među njima,

- definisanju nekih pravila o objektima i odnosima među njima,

- formiranju upita (pitanja) o objektima i odnosima među njima.

Omogućava dijalog korisnika i računara - interpretativni jezik.

Naredbe u PROLOG-u

- Postoje tri kategorije komandi u PROLOG-u:
  - Činjenice (facts): rečenice koje su uvek tačne i koje formiraju bazu znanja
  - Pravila (rules): slična funkcijama iz proceduralnih jezika; imaju if/then strukturu
  - Upiti (queries): interpreter učitava upit i pristupa bazi znanja; startovanje programa

Činjenice i pravila

- Činjenice su komande u jednoj liniji sa tačkom na kraju

  otac(marko, petar).

- Pravila se sastoje od
  - Uslovnog dela ili tela (desna strana)
  - Zaključka ili glave (leva strana)
  - Separator :- ima značenje IF
  - sestra(S,X) :- zena(S), roditelji(S,M,O), roditelji(X,M,O), not(S=X).

Upiti

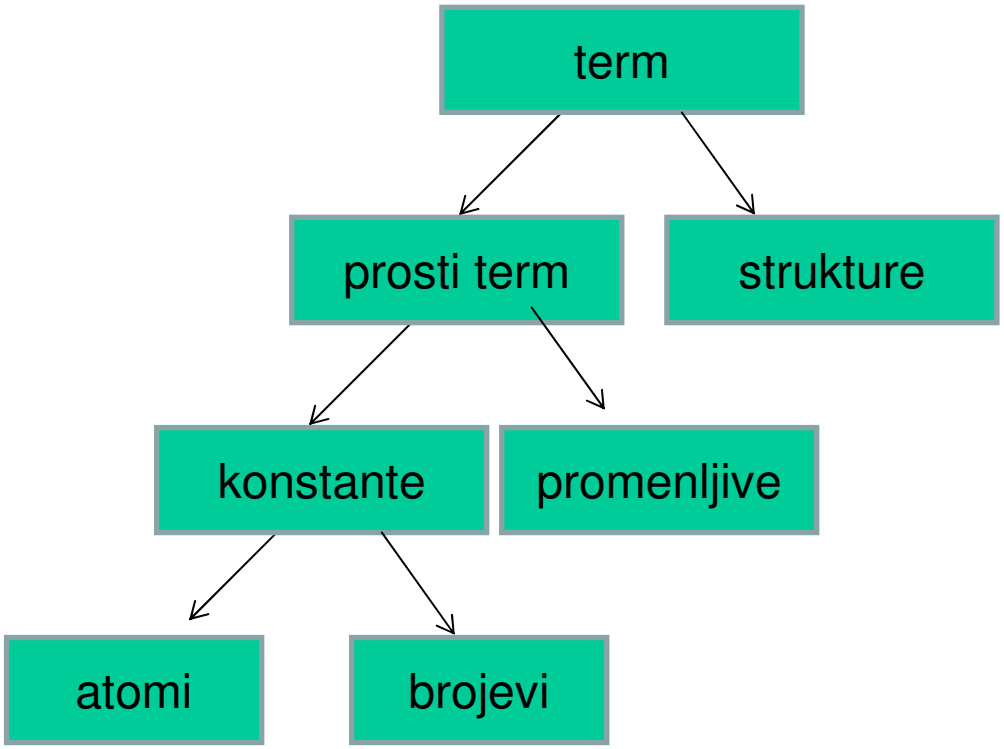
- Interpreter pokušava da izvede upit koristeći činjenice i pravila iz baze znanja
- Dve vrste odgovora
  - Yes/No: otac(toma, bojan).
  - Unifikacija/No: otac(X, bojan).
- Svi mogući odgovori se dobijaju sa ; dok ENTER prekida izvršavanje

Pokretanje programa

- Činjenice i pravila se snimaju u jednu ili više datoteka i čine bazu znanja
- Datoteke se učitavaju u interpreter
- Ako se kasnije ove datoteke menjaju, moraju se ponovo učitati
- Upiti se zadaju iza prompta ?-
- Učitavanje: [ime_datoteke].

3. Osnovni pojmovi PROLOG jezika

Sintaksno ispravni niz znakova naziva se TERM.



3. Opis sintakse i semantike PROLOG-a

Sve konstrukcije PROLOG-a sastavljene su od terma (izgrađene nad termima).

Term može biti: konstanta, promenljiva ili struktura.

Svaki term se zapisuje kao niz slova.

Slovo je element azbuke PROLOG-a.

Razlikuju se 4 kategorije slova:

Velika Slova: **A , B, C, … X, Z**

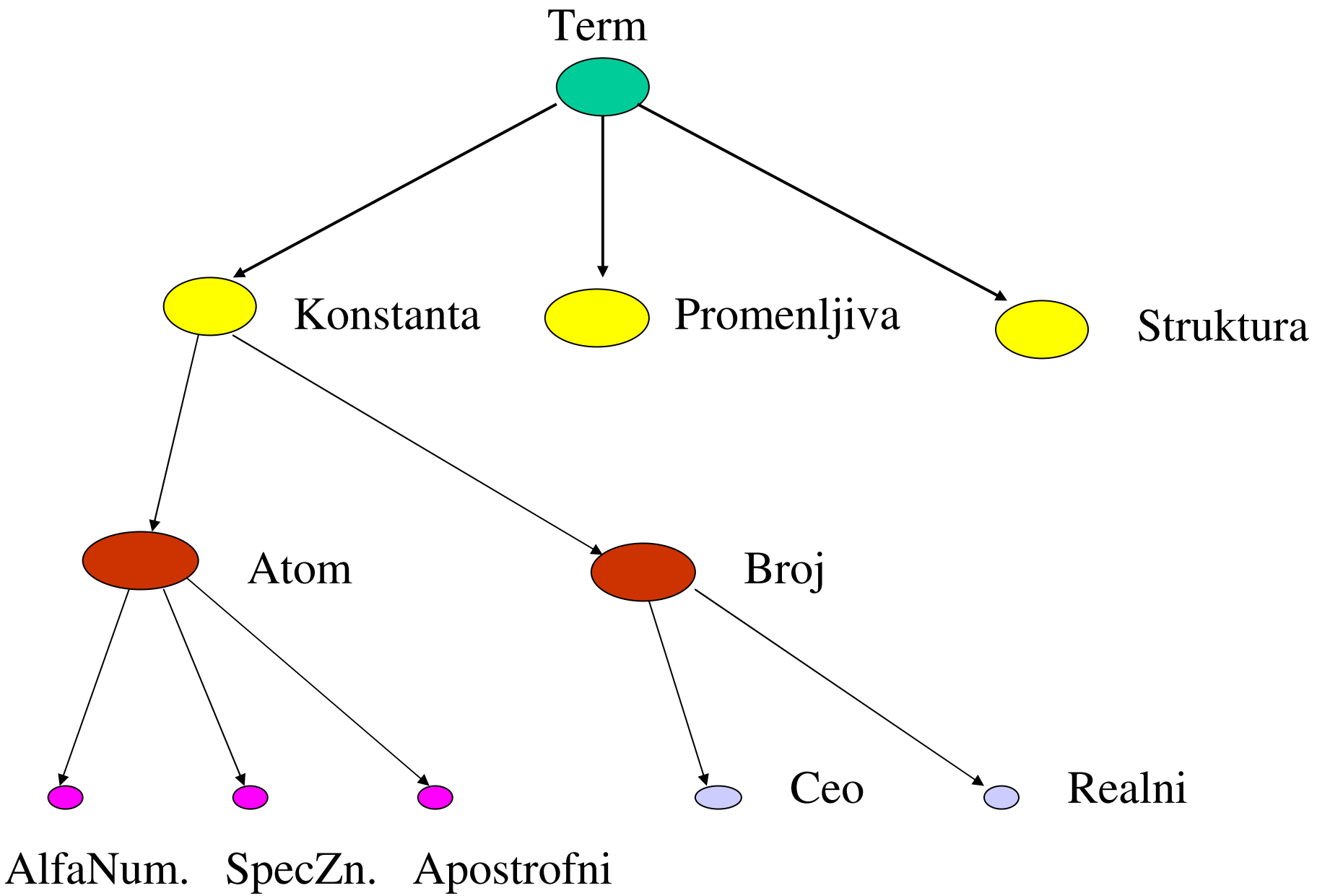
Mala slova: **a, b, c, … x, z**

cifre: **0, 1, 2, … 9**

Specijalni znaci:

štampajući: **! “ # $ % & ‘ ( ) = - ~ | \ [ ] _ @ + ; * : < > , ? /**

neštampajući : **<TAB>,<CR>, <SPACE>, ...**


Term
Konstanta
Promenljiva
Struktura
Atom
Broj
Ceo
Realni
AlfaNum.
SpecZn.
Apostrofni

3.1. Konstante

Konstante služe za označavanje konkretnih objekata ili relacija.

Konstanta može biti: atom ili broj.

Atom može biti: nad slovima, ciframa i _ (alfanumerički);

nad specijalnim znacima (specijalnoznačni);

niz slova između apostrofa (apostrofni).

Primeri alfanumeričkih atoma:

marko, x1, i_ovo_je_atom, brojJedinica, a11Z, …

Nisu alfanumerički: **234xy, novi-beograd, Promenljiva, _ne, …**

Primeri specijalnoznačnih: **<--->, ::=, :-, ===>, ?-, ...**

Primeri apostrofnih: **‘Marko Kraljevic’, ‘1Covek’, ‘Mister-Dzon’**...

Brojevi mogu biti: **celi** **(235, -45, 0, 73636 -23873, ...)**

realni **(2.35, -0.456, 3.4e-3, 0.123E+10, …)**

3.2. Promenljive

Imaju sličnu ulogu kao u matematici (drugim programskim jezicima).

To su niske koje se sastoje iz slova, cifara i _. **Ime promenljive mora početi velikim slovom ili _.**

Primeri promeljivih: **X, Y, Ko, Sta, Nepoznata, A11, I_Ovo, ..**

_takodje, _123, _ , …

Anonimna promenljiva je _

Upotrebljava se kada ime promenljive nije potrebno

?- poseduje (pavle, _).

Ako se javi više anonimnih promenljivih, one su međusobno nezavisne.

3.3. Strukture

Struktura je objekat (celina) koji se sastoji iz nekoliko komponenti. Kao komponente strukture mogu se pojaviti konstante, promenljive i druge strukture.

Služi da se kao jedan objekat tretira skup objekata. Slična slogovima u bazi podataka.

Svaka struktura sastavljena je iz funktora iza kojeg sledi mala otvorena zagrada i spisak komponenti razdvojenih zapetama. Struktura se završava zatvorenom malom zgradom. Funktor je alfanumerički atom.

Primeri: **datum(1, mart, 2001),**

kniga(Naslov, ivo_andric, 256, God)

pasos(Ime,datum_rodjena(3,maj, 1986), Drzava, 23345)

trougao(tacka(2,3), tacka(5,6), tacka(8,9)).

dozvola(vlasnik(pera, dujric), Datum, Broj).

Svaku strukturu karakteriše: ime (funktor) i

arnost (broj argumenata)

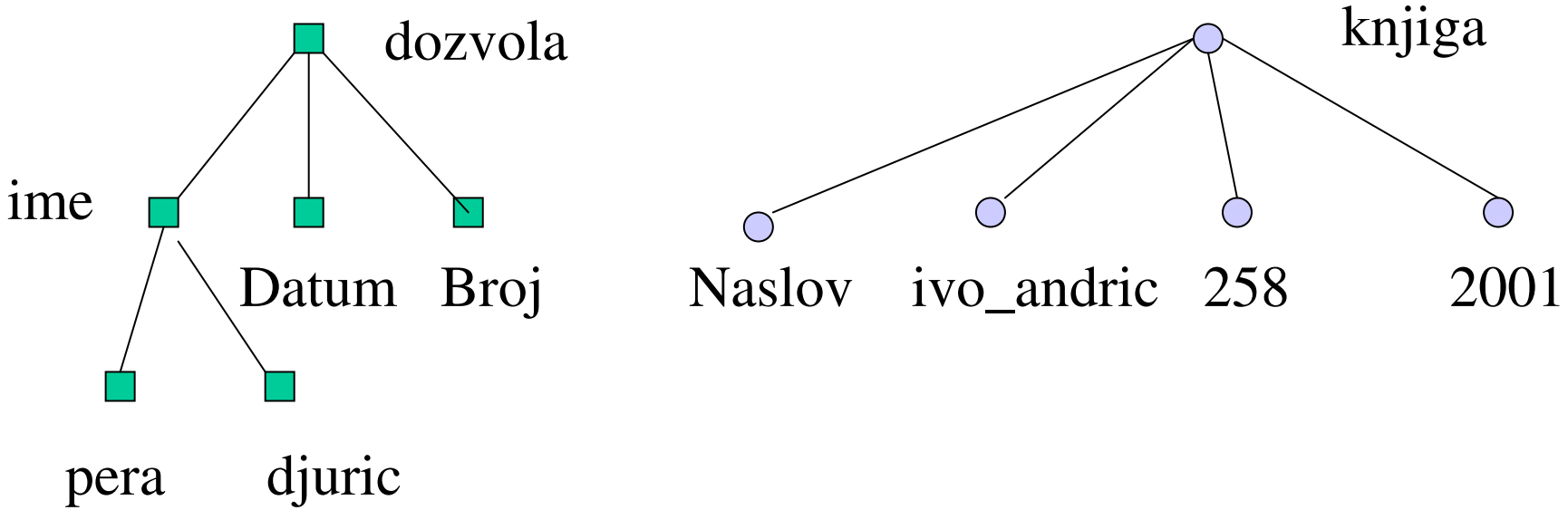
Strukture se javljaju u činjenicama, pravilama i upitima.

poseduje (milan, koala).

voli(ana,X) :- voli(pavle,X).

?- knjiga(X, ivo_andic, Y, 2000).

Svaka struktura se može predstaviti pomoću drvoidne strukture.
Na primer:



3.4. Komentari

U PROLOG-programima se mogu koristiti komentari.
Komentar je tekst između graničnika: /* i */ .

Primeri:

/* Ovo je moj prvi Prolog-program */

/* budite oprezni pri koriscenju promenljive _ */

Komentari se koriste kao i u drugim programskim jezicima.

Primeri programa:

rodbinske_relacije

suncev_sistem

3.5.Unifikacija

Unifikacija (ujednačavanje) je jedna od najvažnijih operacija nad termima. Simbol za ovu operaciju je **=**.

Unifikacija nad termima T i S se vrši na sledeći način:

(a) Ako su T i S konstante, unifikuju se ako predstavljaju isti objekat (konstantu).

(b) Ako je S promenljiva, a T proizvoljan objekat, unifikacija se vrši tako što se S-u dodeli T. Obrnuto, ako je S proizvoljan objekat, a T promenljiva, onda T primi vrednost S-a.

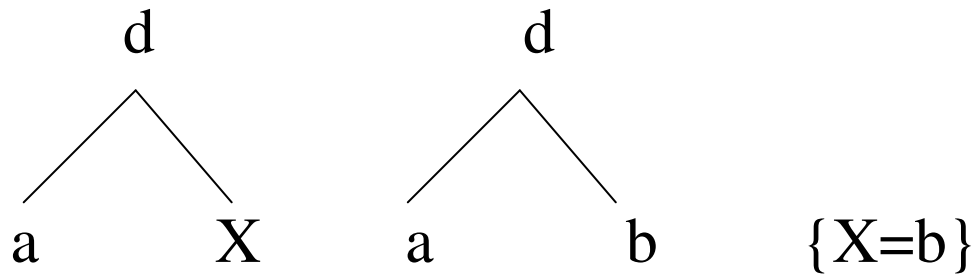
(b)Ako su S i T strukture, unifikacija se može izvršiti ako

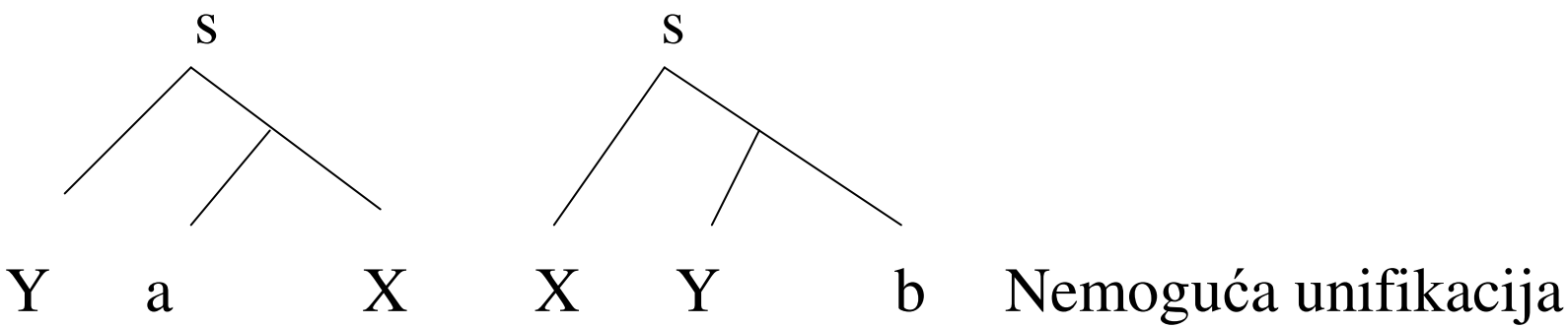
- imaju istu arnost i jednake funktore i
- sve odgovarajuće komponente se mogu unifikovati.

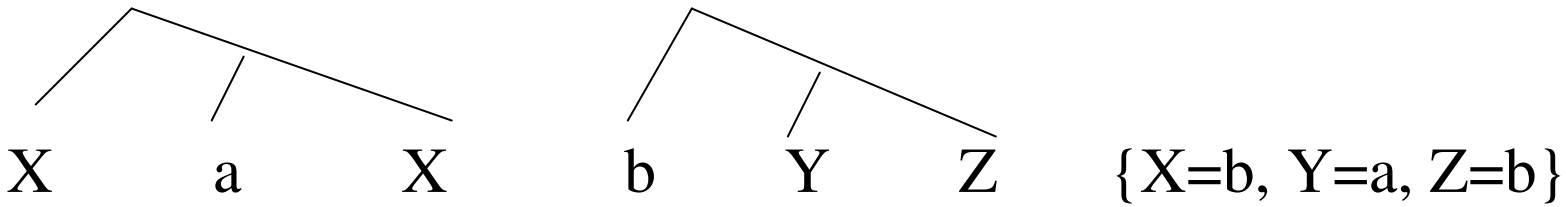
- **Unifikacija se u Prologu se zadaje operatorom =.**

?-datum(D,M,2010)=datum(10, mart,G).
D=10
M=mart
G=2010
yes
?-datum(D,M,2010)=datum(mart,G).
No
?-X=Y, X=1.
X=1
Y=1
yes
?-X=3, X=1.
no

Primeri unifikacije







3.5.1. Strožija definicija unifikacije

Unifikaciju možemo strožije definisati ako uvedemo pojam supstitucije.

Def 1. Supstitucija je preslikavanje između promenljivih i terma. Supstitucije se sastoji iz konačnog broja pravila preslikavanja oblika x→ a ili y → f(a).

Def 2. (Primena supstitucije) Ako je σ oznaka za supstituciju, a t oznaka za term, onda se primena supstitucije σ na term t definiše na sledeći način:

$$t\sigma = \begin{cases} u & \text{ako je } t = x \text{ i } x \to u \in \sigma \\ f(t_1\sigma,\ldots,t_n\sigma) & \text{ako } t = f(t_1,\ldots,t_n) \\ t & \text{inace} \end{cases}$$

Prethodna operacija se ostvaruje tako što prođemo kroz argumente funkcije tražeći promenljive koje imaju odgovarajuća pravila zamene u supstituciji i zamenjujući ih supstitucionim termom. To znači, ako je: $\sigma = \{ x \rightarrow h(a) , y \rightarrow b\}$ i $t=f(x,g(y),z)$, tada je $t\sigma = f(h(a), g(b), z)$.

Supstitucija može biti komponovana od primene jedne supstitucije na neku drugu. Primena supstitucije τ na θ označava se sa $\theta\tau$.

Def 3. (Unifikacija) Dva terma ***t*** i ***u*** se mogu unifikovati ako i samo ako postoji supstitucija σ tako da važi $t\sigma = u\sigma$ Rezultujući term je $t \cup u$. Supstitucija σ naziva se unifajer za terme t i u.

Def4. (Uopšten unifajer) Unifajer σ je uopšteni unifajer terma ***t*** i ***u*** ako ako za neki drugi unifajer θ postoji supstitucija τ tako da važi: $\sigma\theta = \tau$

Izračunavanje odgovora u Prolog-u

p(X,Y):--q(X),r(X,Y).

p(d,4).

q(a).

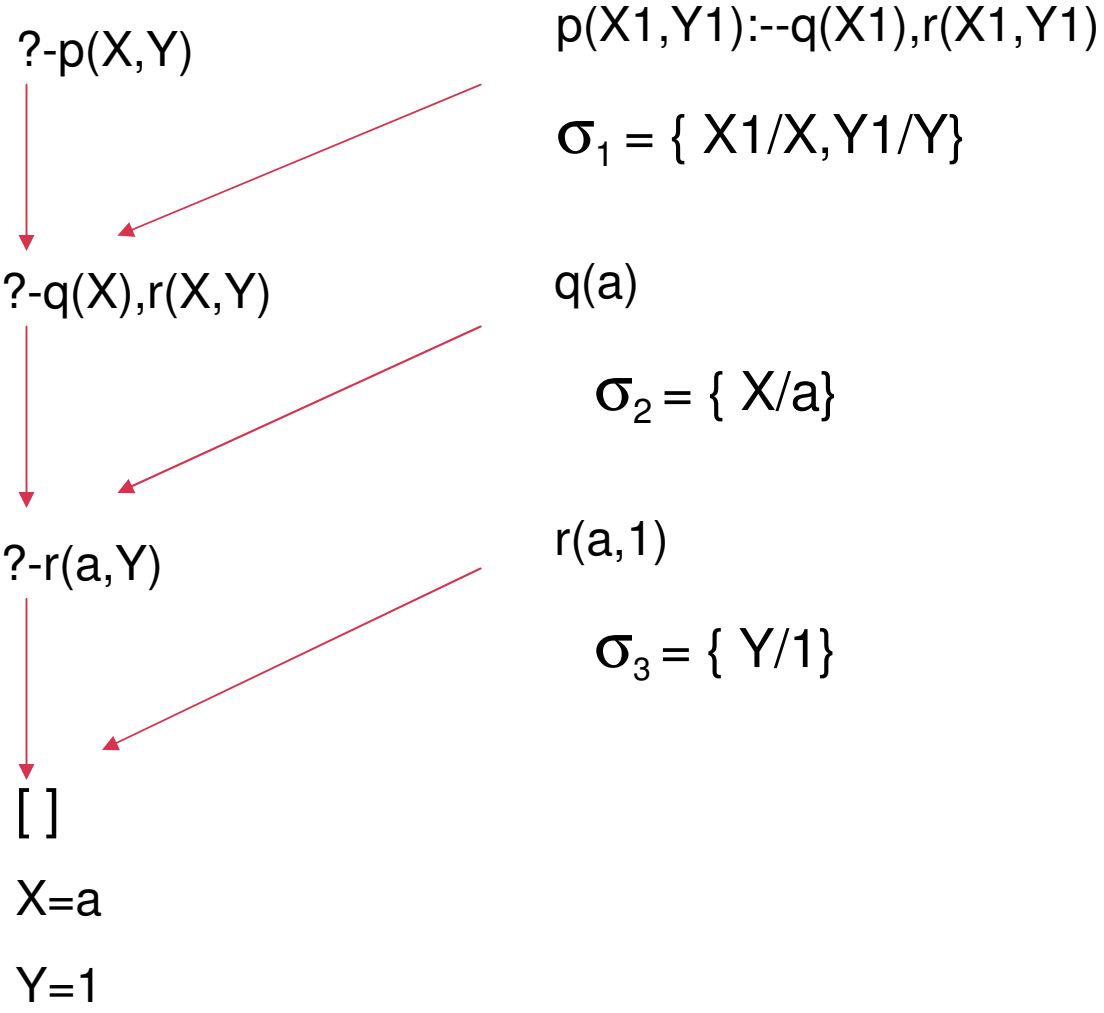
q(b).

q(c).

r(a,1).

r(a,2).

r(c,3).



Primer : Izračunavanje odgovora u Prolog-u

veliko(medved).

veliko(slon).

malo(macka).

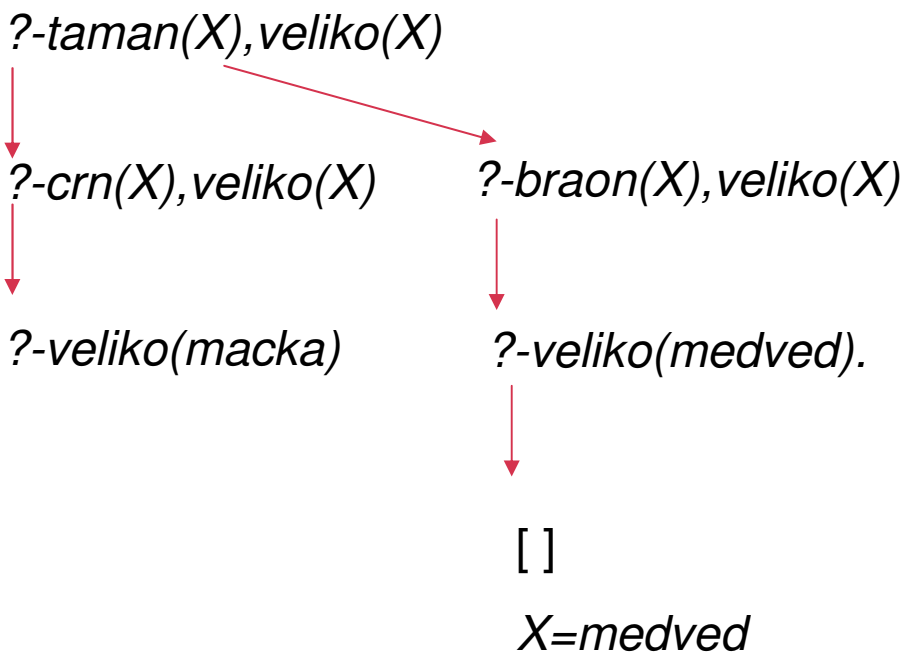
braon(medved).

crn(macka).

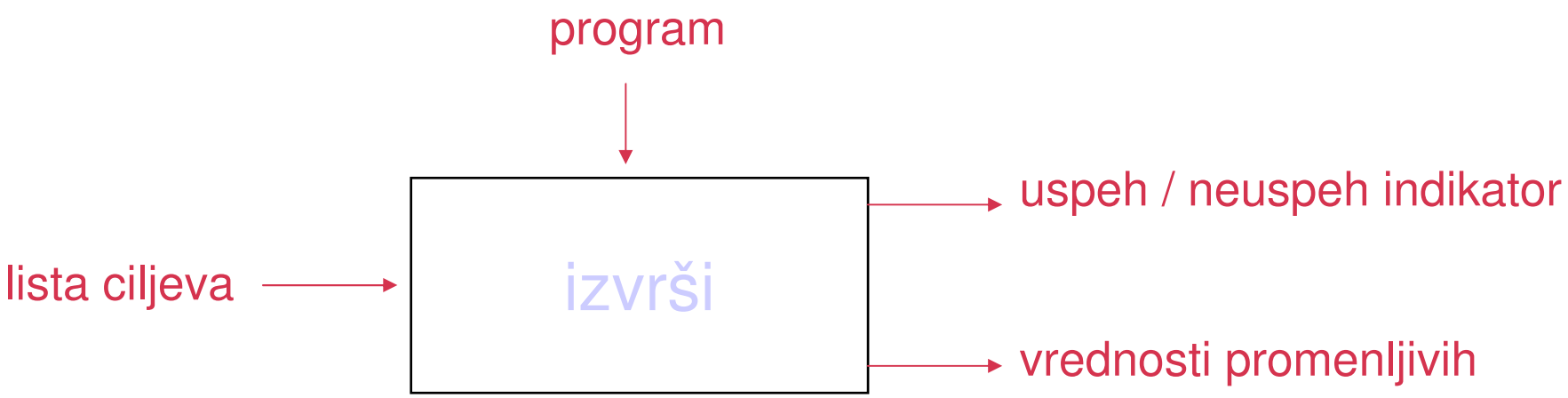
siv(slon).

taman(Z):-crn(Z).

taman(Z):-braon(Z).



Procedura izvrši



1. Indikator uspeha je **yes** ako se cilj može zadovoljiti, a inače je **no.**
2. Vrednost promenljivih se dobijaju samo u slučaju uspešnog zadovoljenja cilja koji sadrži promenljive.

Opis procedure izvrši

Neka je zadat niz ciljeva $G_1,G_2,...,G_m$.

Procedura izvrši: Ako ne postoji cilj koji treba zadovoljiti procedura se završava uspehom, vrednosti promenljivih su određene kompozicijom korišćenih supstitucija. U suprotnom prelazi se na proceduru pretraži.

Procedura pretraži: Traži klauzulu u programu odozgo na dole dok ne nađe prvu klauzulu C takvu da se glava te klauzule može unifikovati sa prvim podciljem G_1. Ako nema takve klauzule, procedura izvrši se prekida sa neuspehom. Ako postoji takva klauzula C oblika: $H:-B_1,B_2,....,B_n$
preimenuju se promenljive u C i dobija se varijanta C' od C, ali takva da C' i niz $G_1,G_2,...,G_m$ nemaju zajedničke promenljive. Neka je C' oblika
$H':-B_1',B_2', ... B_n'$
Neka je σ unifikator za G_1 i H. U ciljni niz $?- G_1,G_2,...,G_m$
zamenjuje se G_1 nizom $B_1',B_2', ... B_n'$ i novi ciljni niz je

$?-(B_1',B_2', ... B_n',G_2,...,G_m)\ \sigma$.

Sada procedura Pretraži poziva proceduru Izvrši za novi ciljni niz. Ako se procedura Izvrši za novi ciljni niz završi uspehom, tada se i polazni ciljni niz završava uspehom. Ako se izvrašavanje novog ciljnog niza završava neuspehom, onda se prelazi na proceduru Pretraži sa prethodnim ciljnim nizom. Pretraživanje se nastavlja traženjem klauzule koja neposredno sledi iza C.

Ako sistem uspešno završi zadatak, korisnik može zahtevati od sistema da backtracking-om nađe drugo rešenje. U ovom slučaju se sistem vraća na proceduru Pretraži kao da zadatak nije uspešno završen sa poslednjom klauzulom C.

Stablo pretraživanja

Neka je P program i neka je G polazni cilj. Stablo pretraživanja se definiše koracima 1-3:

(1) Koren stabla je polazni cilj G.

(2) Neka je cilj

?- $G_i,G_{i+1},...,G_m$

čvor stabla pretraživanja. Tada čvor ima po jednog potomka za svaku klauzulu

$H:-B_1,B_2,....,B_n$ (*)

iz programa P za koju se struktura Gi i H mogu unifikovati. Neka je

$H':-B'_1,B'_2,....,B'_n$

varijanta (*) i neka je σ najopštitji unifikator za G_i i H'. Tada je potomak čvora

?-$(B'_1,B'_2,....,B'_{n,}\ G_{i+1i},...,G_m)\ \sigma$.

Klauzule, čije glave se unifikuju sa ciljem Gi traže se u programu odozgo na dole, a generisani potomci čvora postavljaju se s leva nadesno.

(3) Potomak koji nema podcilj u telu naziva se praznim ciljem i označava **[ ]**

Neki važni pojmovi

Čvorom uspeha nazivamo čvor koji sadrži prazan cilj. Granu stabla koja zavrašava čvorom uspeha nazivamo grana uspeha. Čvorom neuspeha nazivamo čvor koji sadrži neprazan cilj, a nema potomaka. Grana neuspeha je grana koja se se zavrašava čvorom neuspeha. Konačne grane su grane uspeha i grane neuspeha. Beskonačne grane su grane koje nisu konačne. Stablo pretraživanja nazivamo konačnim ako su sve grane stabla konačne. Konačno stablo koje nema nijednu granu uspeha nazivamo konačno stablo neuspeha.

Stablo pretraživanja

(k_1) p(X,Y):--q(X),
r(X,Y).

(k_2) p(d,4).

(k_3) q(a).

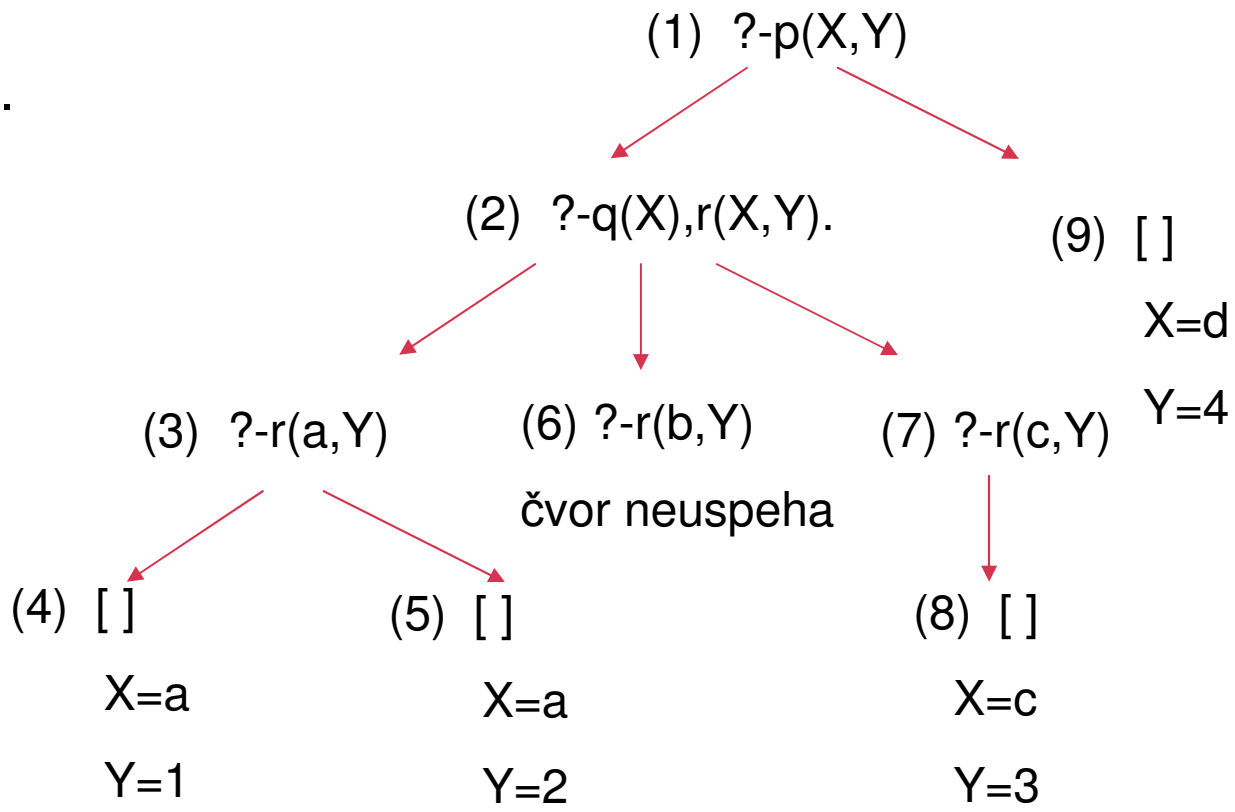
(k_4) q(b).

(k_5) q(c).

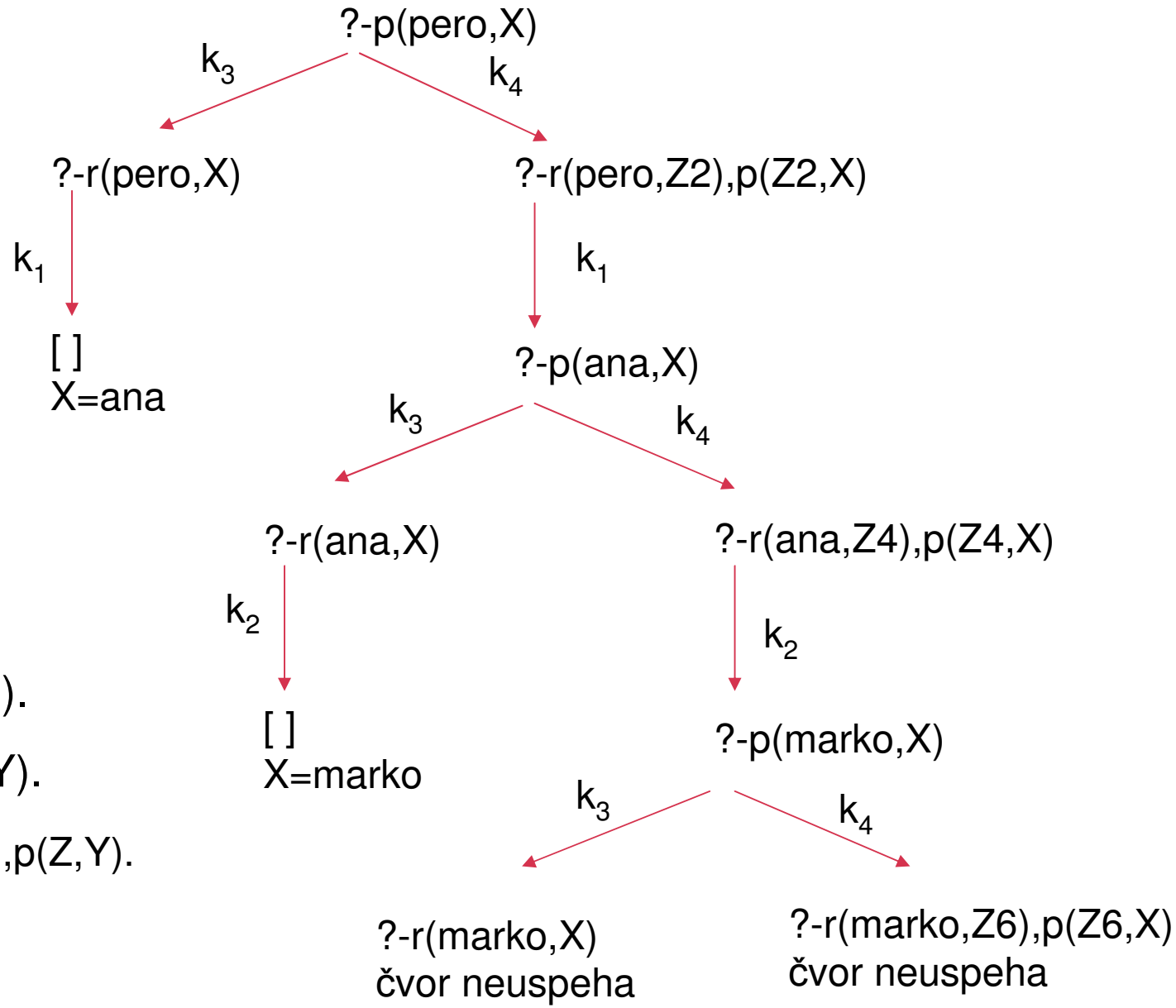
(k_6) r(a,1).

(k_7) r(a,2).

(k_8) r(c,3).



SIMULACIJA PROCEDURE IZVRŠI



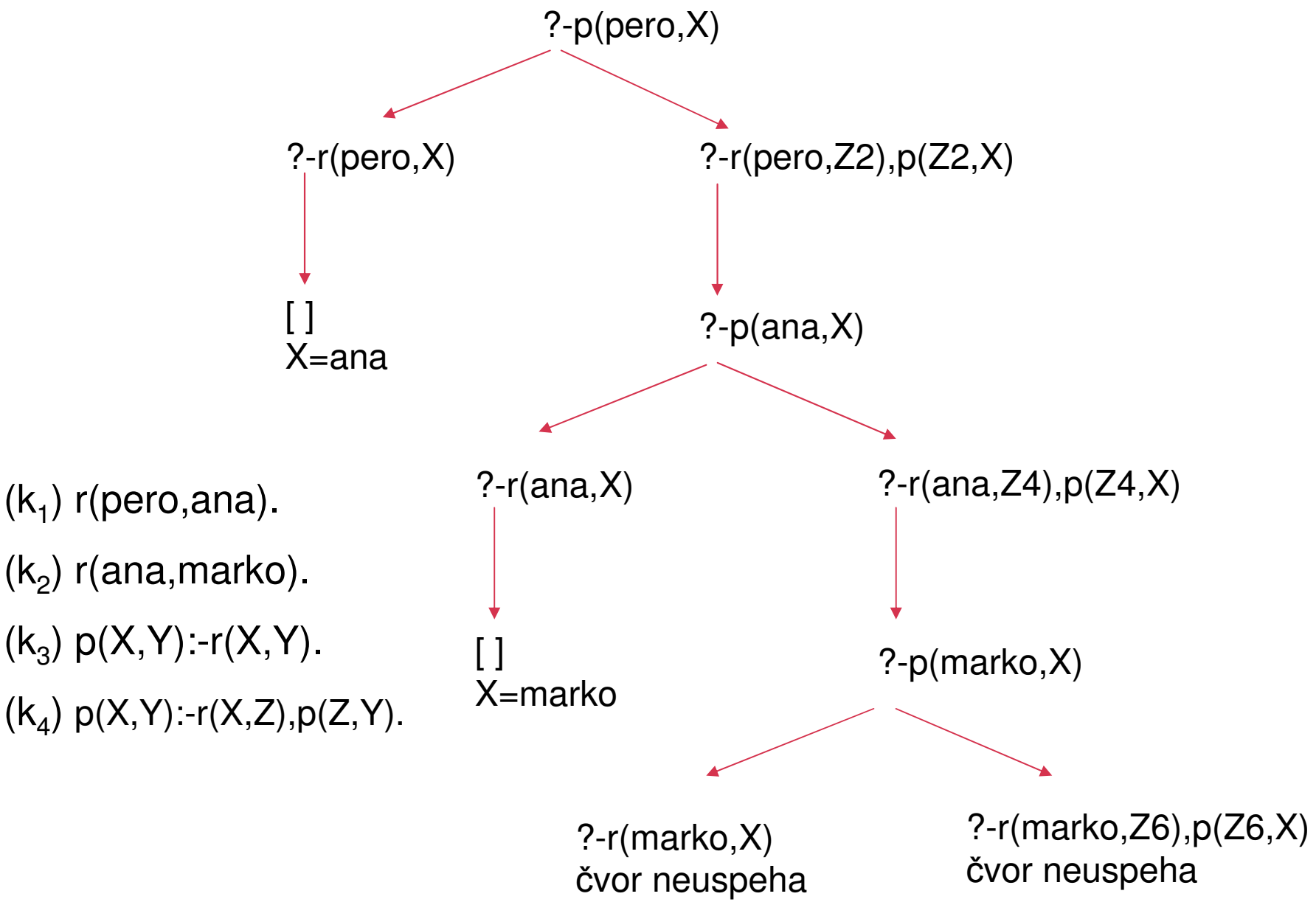
(k_1) r(pero,ana).

(k_2) r(ana,marko).

(k_3) p(X,Y):-r(X,Y).

(k_4) p(X,Y):-r(X,Z),p(Z,Y).

STABLO PRETRAŽIVANJA



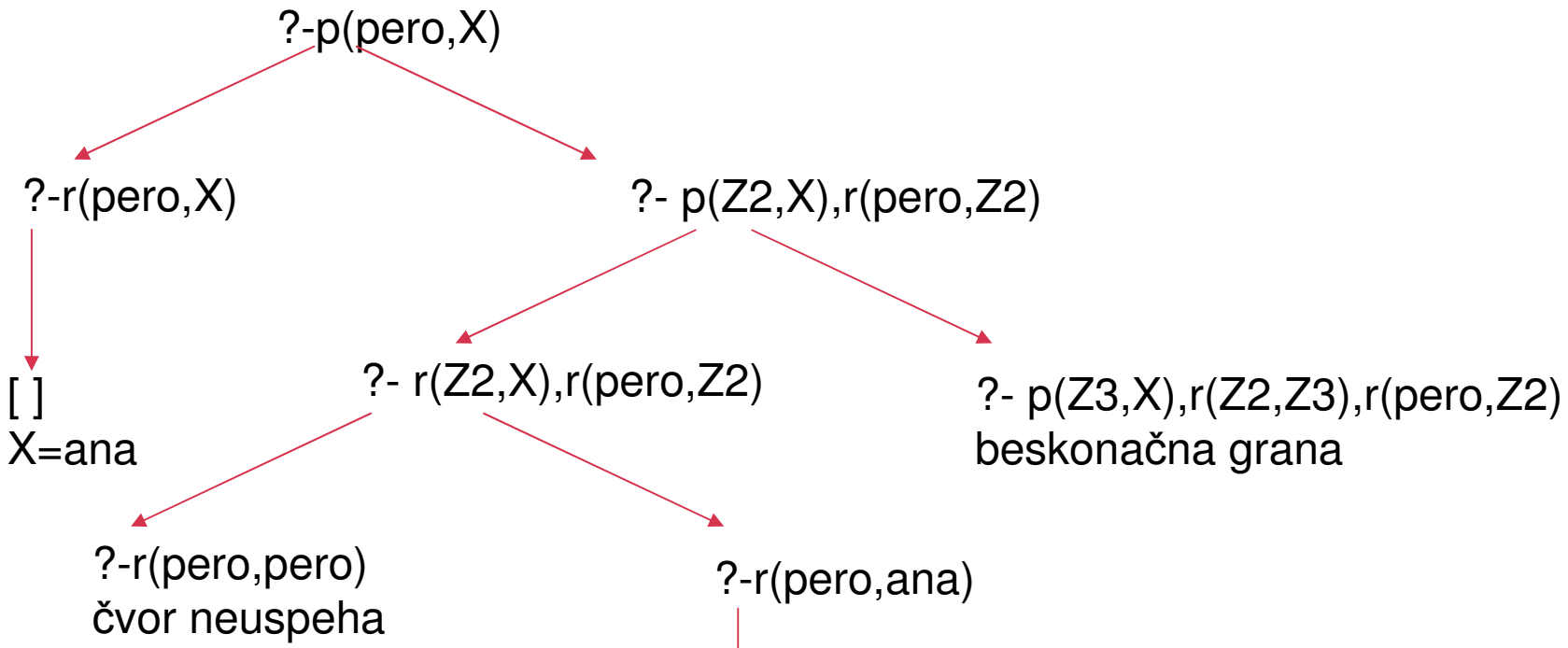
(k_1) r(pero,ana).

(k_2) r(ana,marko).

(k_3) p(X,Y):-r(X,Y).

(k_4) p(X,Y):-r(X,Z),p(Z,Y).

Promena redosleda terma u telu pravila



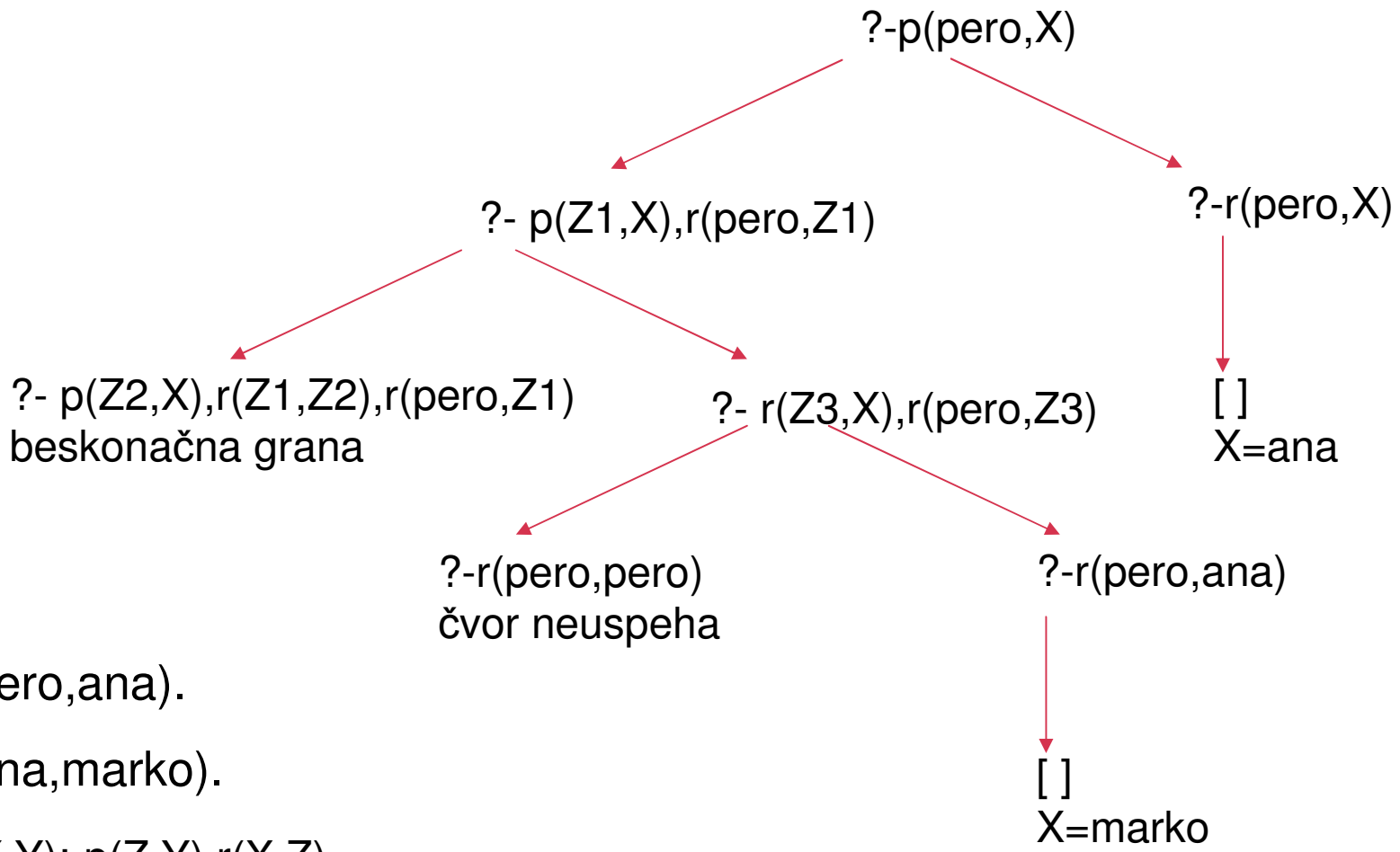
(k_1) r(pero,ana).

(k_2) r(ana,marko).

(k_3) p(X,Y):-r(X,Y).

(k_4) p(X,Y):-p(Z,Y),r(X,Z).

Promena redosleda klauzula u programu



(k_1) r(pero,ana).

(k_2) r(ana,marko).

(k_3) p(X,Y):-p(Z,Y),r(X,Z).

(k_4) p(X,Y):-r(X,Y).

3.6. Operatori

Operatori su specijalni funktori struktura čiji argumenti su uglavnom brojevi.

Umesto **saberi(2,3),** pisemo **+(2, 3),** odnosno **2+3.**

2+3 je izraz i predstavlja strukturu zapisanu u operativnoj formi.

Operativna forma struktura je često preglednija i prihvatljivija za čoveka.

Svaki izraz se može predstaviti u obliku strukture i obrnuto. Na primer:

x+y*z -----> +(x, *(y, z))

Za svaki operator bitne su 3 karakteristike:

pozicija

prioritet

asocijativnost.

Ove karakteistike su bitne pri izračunavanju vrednosti izraza.

Pozicija: Operatori mogu biti:

infiksni (x+y, X*Y, a/b, …)

prefiksni (-Y, dx, ++x …)

postifiksni (x!, n--, x^2, …)

Prioritet: Određuje koji se operator u izrazu prvi izvršava. Prioritet u Prologu je ceo broj i može da karakteriše celu klasu operatora. Pravilo je da veći broj određuje niži prioritet.

	A + B	A*B	
prioritet:	k +m	k	(m>0)

Prioritet se sprečava malim zagradama: a*(x+b).

Asosijativnost: Određuje redosled primene operatora kada su istog prioriteta. Mogu biti levoasocijativni i desnoasocijativni.

8/2/2 ----> 8/(2/2) ??? -----> (8/2)/2 ???

I ovde se problem može rešiti pomoću zgrada.

Svi aritmetički operatori su levoasocijativni.

3.7. Neki sistemski (ugrađeni) opearatori

U PROLOG-u se mogu razlikovati sistemski i uvedeni operatori. Sistemski su unapred definisani i stoje na raspolaganje korisniku pri stratovanju PROLOG-a. Uvedene operatore korisnik sam definiše na poseban način.

3.7.1. Operator unifikacije (=)

To je infiksni operator:

?- X = Y.

Cilj: učiniti da X i Y budu jednaki! Prolog nastoji da izjednači X i Y. Uvek važi: X=X. Primenjuje se u skladu sa ranije opisanim procesom unifikacije. Primeri:

?- knjiga = bukvar. /* ne uspeva */

?- pasos('Mila Katic', X, Y,12345)= pasos(Z, datum(1,2,2000), beograd,V). /* uspeva */

Ako se izjednače dve nekonkretizovane promenljive one postaju povezane. Čim se konkretizuje jedna od njih, konkretizuje se i druga. Dakle, razlikujemo: nekonkretizovane, konkretizovane i vezane promenljive.

3.7.2. Relacijski operatori

To su infiksni operatori koji se koji se primenjuju na konkretizovane (brojčane) promenljive.

X \= Y, X < Y, X > Y, X =< Y, X >= Y.

Ovom skupu se može pridružiti i operator unifikacije, ali on se primenjuje i na nekonkretizovane promenljive.

Testiranje jednakosti dva izraza A1 i A2:
A1 =:= A2 testira da li su vrednosi izraza jednake
A1 \== A2 testira da li su izrazi doslovce nejednaki **A1 \= A2 testira da li su vrednosti izraza različite**

Vojnici.ari

3.7.3. Aritmetički operatori

To su infiksni operatori:

X + Y - operator sabiranja

X -Y - operator oduzimanja

X*Y - operator množenja

X/Y - operator deljenja

X mod Y - operator izračunavanja ostatka pri deljenju

X is Y - za doljivanje vrednosti izraza Y promenljivoj X.

Ima smisla napisati: X is Y*23-Z/A.

3.7.4. Operator negacije

Upotrebljava se kao prefiksni operator: not X ili not(X).

Cilj ?-not(X). uspeva, ako ne uspeva X.

Prosek.ari.

BrojDanaUMesecu.ari

Negacija u Prolog-u

Kada PROLOG sistem naiđe na cilj oblika *not (C')*, *on pokušava da zadovolji cilj C'* .

Ako uspe da zadovolji cilj C' onda cilj *not(C')* pada.

Ako u pokušaju da zadovolji cilj C' imamo konačno stablo neuspeha, onda cilj *not (C')* *uspeva.*

DOPUNA STABLA PRETRAŽIVANJA TAČKOM 2’

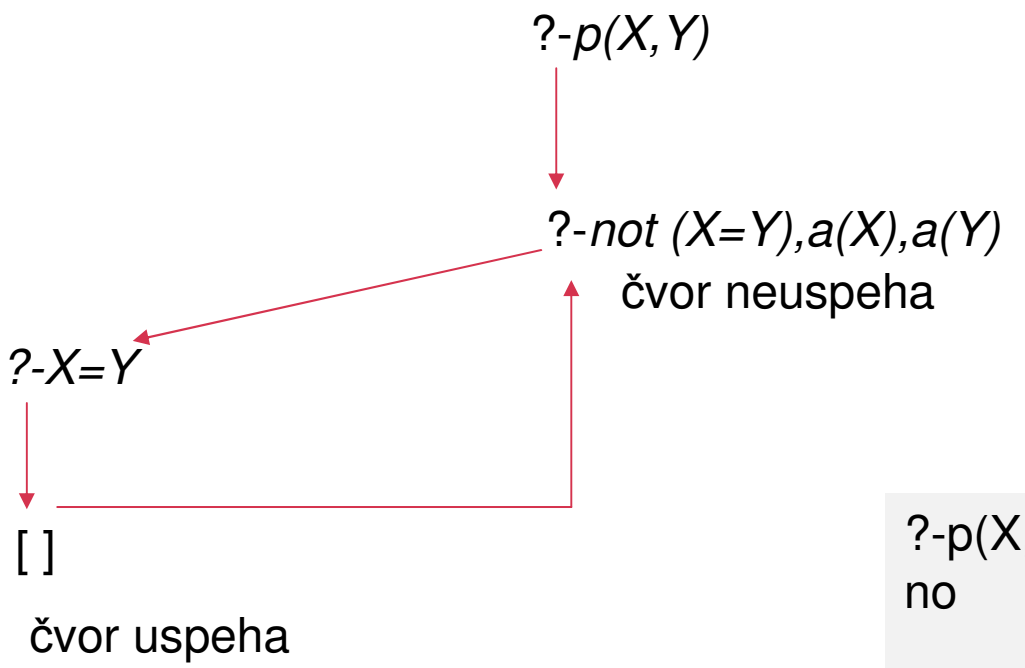
(2’) Neka je cilj

?-*not* (G_i), G_{i+1}, …,G_n

čvor stabla pretraživanja. Ako cilj G_i ima konačno stablo pretraživanja sa granom uspeha onda posmatrani čvor nema potomka. ako cilj G_i ima konačno stablo neuspeha onda je potomak datog čvora

?- G_{i+1}, …,G_n

Primer za negaciju




```
a(1).
a(2).
p(X,Y):-not (X=Y),a(X),a(Y).
p1(X,Y):-a(X),a(Y), not (X=Y).
```

```
?-p(X, Y).
no


?-p1(X, Y).
X=1
Y=2;
X=2
Y=1;
no
```

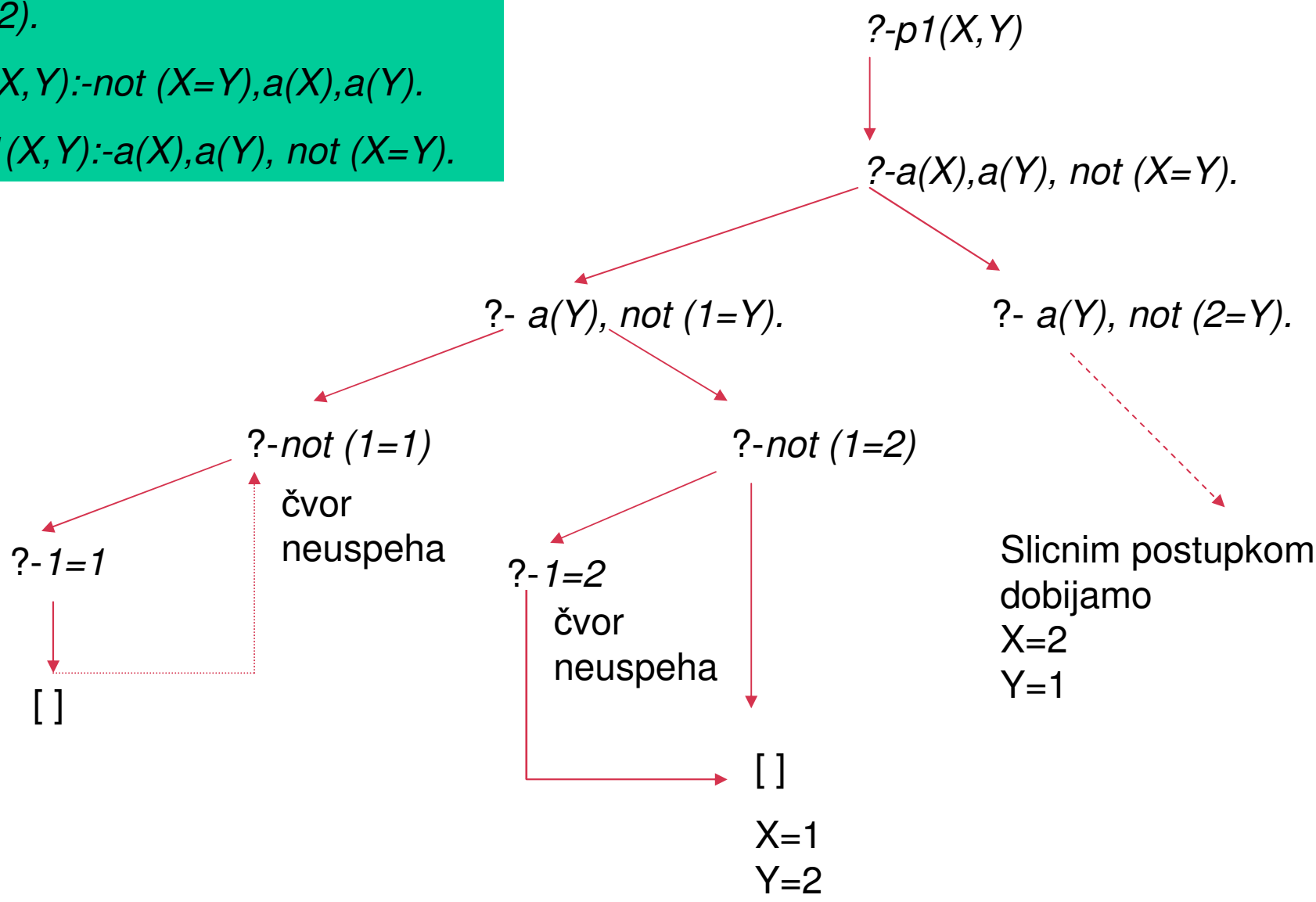
a(1).

a(2).

p(X,Y):-not (X=Y),a(X),a(Y).

p1(X,Y):-a(X),a(Y), not (X=Y).

nastavak primera



3.8. Deklarativna i proceduralna interpretacija

Prolog-konstrukcije se mogu interpretirati deklarativno i proceduralno.

Za pravilo (P,Q i R su termi):

P :- Q, R.

Deklarativna semantika:

- P je istinito ako je Q i R istinito ili
- Iz Q i R sledi P.

Proceduralna semantika:

Da bi se rešio zadatak P, prvo rešiti podzadatak Q, a zatim podzadatak R ili

Da bi se dostigao (cilj) P, prvo dostići Q, a zatim R.

U proceduralnom tumačenju se odredjuje i redosled ispunjavanja cilja.

Činjenica: p(a).

Deklarativno: p(a) je istinito.

Proceduralno: zadatak p(a) je izvršen.

Ako imamao: p(a,X).

Deklarativno: Za svako X istinito je p(a,X)

Proceduralno: Za svako X, zadatak p(a,X) je izvršen.

Ako imamo: p(X) :- q(X), r(X).

Deklarativno: Za svako X, p(X) je istinito ako je istinito q(X) i r(X).

Proceduralno: Da bi se izvršio zadatak p(X), izvrši zadatke q(X) i r(X).

Upit: ? - p(X).

Deklarativno: Da li postoji vrednost promenljive X za koji važi p(X).

Proceduralno: Naći (izračunaj) vrednost promenljive X za koju važi svojstvo p.