Prolog

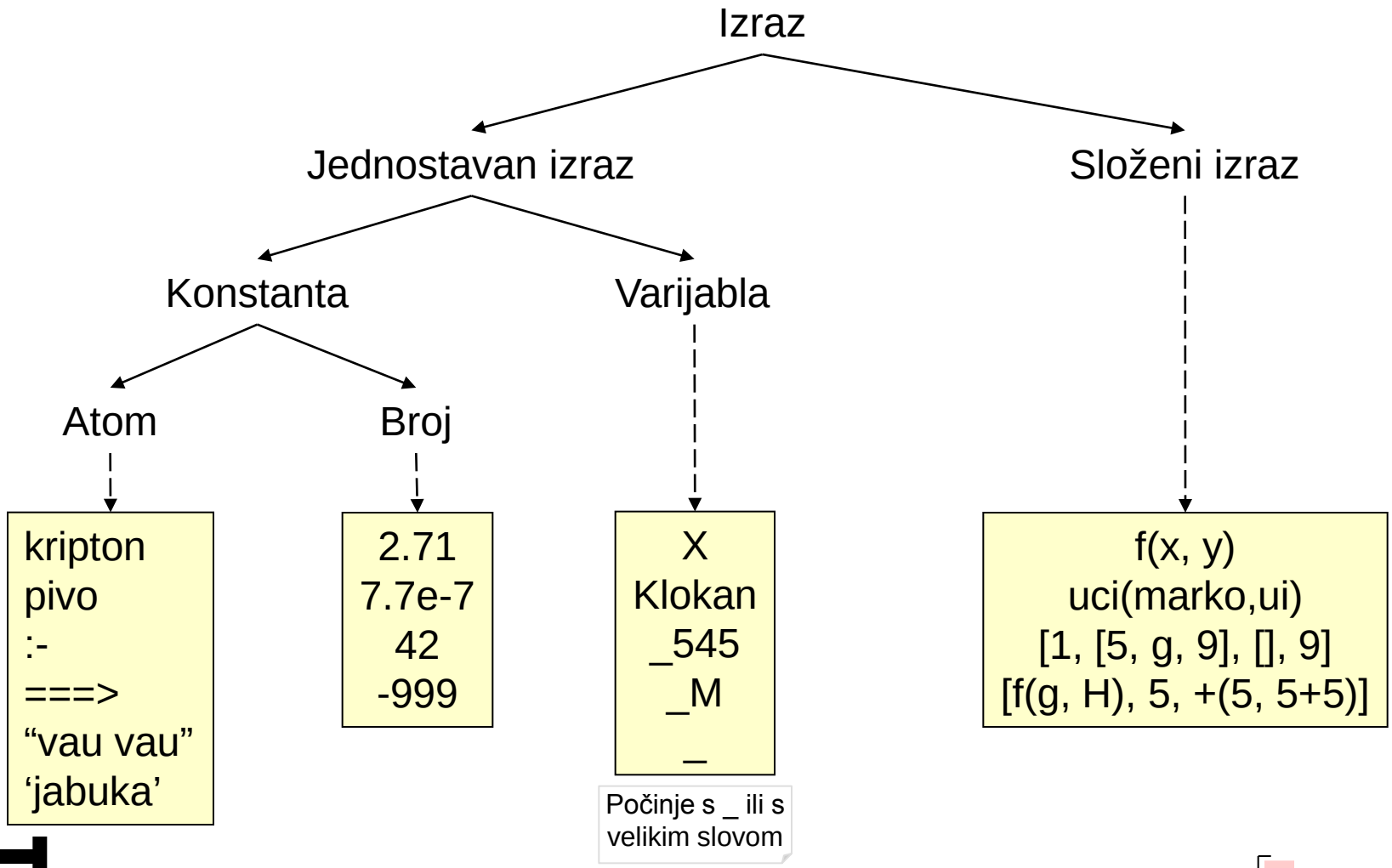
Vježbe iz umjetne inteligencije

Matko Bošnjak, 2009

Uvod

- Programiranje
  - Deklarativno ←→ Postupkovno (imperativno)
  - (Prolog, SQL, Mathematica*, R, ...) ←→ (C, Java, C++, ...)
- Prolog
  - Programski jezik za simboličku nenumeričku obradu
  - Temelji se na FOL, koristi ograničenu verziju klauzalnog oblika
    - Hornov klauzalni oblik
  - Pogodan za rješavanje problema koji se mogu opisati objektima i relacijama među njima
    - Opiši problem
    - Pitaj što točno želiš doznati (Prolog sam zaključuje nove činjenice)
- VJEŽBE: SWI-Prolog (implementacija)

Sintaksa prologa



Opis problema

Stavak

- Temeljna građevna jedinica programa (činjenica, pravilo)

```
radi(petra).
studira(petra).

zaposlen(X):-studira(X); radi(X).
prezaposlen(X):-studira(X), radi(X).
```

Činjenice

- opisuju pojedinačne elemente relacije

Pravila

- definiraju nove relacije na temelju postojećih

glava

tijelo

Točka na kraju svakog stavka!

Prolog: prezaposlen(X) :- radi(X) , studira(X).

FOL: prezaposlen(X) ← radi(X) ∧ studira(X)

(Hornova klauzula – klauzula s najviše jednim pozitivnim literalom)

```
kupujeKod(X, Y, Z).
```

funktor argumenti → 3-mjesni (arity) predikat: kupujeKod/3

Upiti

```
studira(ana).
studira(petra).
radi(petra).

zaposlen(X):-studira(X); radi(X).
prezaposlen(X):-studira(X), radi(X).
```

Doseg varijable
-jedan stavak

```
?- radi(petra).

?- studira(X),radi(X).

?- prezaposlen(ana).

?- zaposlen(X).
```

```
Prolog:  yes

Prolog:  X = petra
         yes
Prolog:  no

Prolog:  X = ana ?
         yes
```

Upiti

Poseban predikat za ispis klauzula u bazi znanja:

```
?- listing.
```

```
Prolog:
% file: F:/UI/Prolog_vj/kb1.pro

studira(ana).
studira(petra).

radi(petra).

zaposlen(A) :-
        (   studira(A)
        ;   radi(A)
        ).

prezaposlen(A) :-
        studira(A),
        radi(A).

(16 ms) yes
```

Unifikacija

Prolog instancira (pretražuje prostor potrebnih supstitucija) tako da se postigne identičnost dva izraza.

Detaljnija definicija unifikacije:

1. Ako su I1 i I2 konstante, onda se one mogu unificirati ako predstavljaju isti atom ili broj
2. Ako je I1 varijabla, a I2 bilo koji tip izraza, onda je I1 i I2 moguće unificirati i I1 se instancira u I2.
3. Ako su I1 i I2 kompleksni izrazi onda se mogu unificirati ako:
   a) predstavljaju isti predikat ili funkciju i imaju isti broj argumenata (arity)
   b) korespondirajući argumenti predikata ili funkcija se mogu unificirati
   c) instanciranje varijabli je kompatibilno

Unifikacija

```
oprezan(pero).
oprezan(ivo).
oprezan(stef).
trijezan(pero).
trijezan(stef).
koncentriran(ivo).
koncentriran(stef).
pijan(ivo).
dobar_vozac(X) :- oprezan(X), trijezan(X), koncentriran(X).
```

```
?- dobar_vozac(Z).
```

```
Prolog: Z = stef

        yes
```

Unifikacija

trace mod, prikazuje svaki korak zaključivanja (stabla pretraživanja)

```
?- trace.
                    Prolog:  The debugger will first creep – showing
                             everything (trace)
?- notrace.         Prolog:  The debugger is switched off
```

```
?- notrace.
?- dobar_vozac(Z).          Prolog: Z = stef
                                    yes
```

Unifikacija

```
?- trace.
?- dobar_vozac(Z).
```

Prolog: koraci zaključivanja

Prolog: finalni izlaz

```
Z = stef

yes
{trace}
```

```
1  1  Call: dobar_vozac(_23) ?
2  2  Call: oprezan(_23) ?
2  2  Exit: oprezan(pero) ?
3  2  Call: trijezan(pero) ?
3  2  Exit: trijezan(pero) ?
4  2  Call: koncentriran(pero) ?
4  2  Fail: koncentriran(pero) ?
2  2  Redo: oprezan(pero) ?
2  2  Exit: oprezan(ivo) ?
3  2  Call: trijezan(ivo) ?
3  2  Fail: trijezan(ivo) ?
2  2  Redo: oprezan(ivo) ?
2  2  Exit: oprezan(stef) ?
3  2  Call: trijezan(stef) ?
3  2  Exit: trijezan(stef) ?
4  2  Call: koncentriran(stef) ?
4  2  Exit: koncentriran(stef) ?
1  1  Exit: dobar_vozac(stef) ?
```

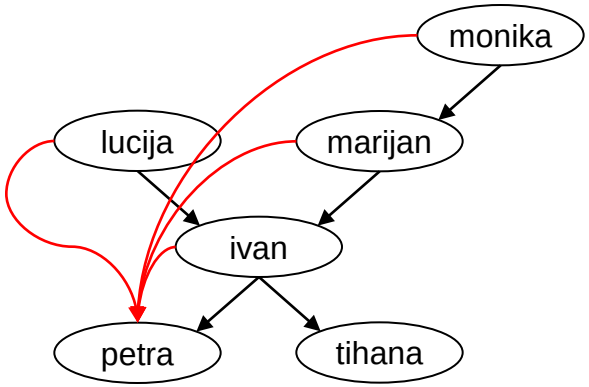
Rekurzija

Naredbe pridruživanja, programske petlje? → Rekurzija!

```
roditelj(ivan, petra).
roditelj(ivan, tihana).
roditelj(lucija, ivan).
roditelj(marijan, ivan).
roditelj(monika, marijan).

predak(X,Y):-roditelj(X,Y).
predak(X,Y):-roditelj(X,Z),predak(Z,Y).
```

```
predak(X,Y):-roditelj(X,Z),...
```




```
?- predak(ivan, petra).

?- predak(X,petra).

?- predak(lucija,X).
```

```
Prolog:   true ?
          yes
Prolog: na drugom i trećem upitu
          X = ivan ?
          yes
```

...uključite trace mod, pogledajte stablo pretraživanja

Rekurzija

```
predak1(X,Y):-roditelj(X,Y).
predak1(X,Y):-roditelj(X,Z),predak1(Z,Y).
```

```
predak2(X,Y):-roditelj(X,Y).
predak2(X,Y):-predak2(Z,Y),roditelj(X,Z).
```

```
predak3(X,Y):-predak3(X,Z),roditelj(Z,Y).
predak3(X,Y):-roditelj(X,Y).
```

```
predak4(X,Y):-roditelj(Z,Y),predak4(X,Z).
predak4(X,Y):-roditelj(X,Y).
```

```
?- predak1(X, petra).

?- predak2(X, petra).

?- predak3(X, petra).

?- predak4(X, petra).
```

Prolog ima ugrađeni redoslijed obavljanja operacija

- pretraga baze znanja odozgo prema dolje
- procesiranje klauzula s lijeva na desno
- backtracking, povratak iz neuspjelih unifikacija

Nešto od ovoga ne valja...

...zašto?

Nije čisto deklarativan jer:

$A \wedge B \rightarrow C \neq B \wedge A \rightarrow C$

Rekurzija

```
?- predak1(X, petra).
Prolog:  X = ivan ?
         yes

?- predak2(X, petra).
Prolog:  X = ivan ?
         yes

?- predak3(X, petra).
Prolog: raspad gprolog has stopped working

?- predak4(X, petra).
Prolog:  X = monika ?
         yes
```

Liste

```
?- X = [kvasac, klokan, krmenadl].
```
Elementi liste odvojeni zarezom

```
?- X = [Y, 2, 3+3, 2, mama(ana)].
```
Elementi liste svi prolog objekti ponavljanja dozvoljena

```
?- X = [].
```
Prazna lista

```
?- X = [[], [2, [x, Y, [3, z]], 7]].
```
Liste mogu sadržavati druge liste

Svaka neprazna lista se sastoji od glave i repa

```
?- [G|R] = [martin, matej, mihajlo].
```
Glava – element
Rep – lista

| iznimno važan operator!

```
?- [X1,X2|R] = [1, 2, 3, 4, 5].
```

```
?- [X1,X2,X3|R] = [1, 2, 3, 4, 5].
```

```
?- [_,X2,_|R] = [1, 2, 3, 4, 5].
```

_ anonimna (don't care) varijabla

Liste

```
?- [G|R] = [martin, matej, mihajlo].
```

```
Prolog: G = martin
        R = [matej,mihajlo]
        yes
```

```
?- [X1,X2|R] = [1, 2, 3, 4, 5].
```

```
Prolog: R = [3,4,5]
        X1 = 1
        X2 = 2
        yes
```

```
?- [X1,X2,X3|R] = [1, 2, 3, 4, 5].
```

```
Prolog: R = [4,5]
        X1 = 1
        X2 = 2
        X3 = 3
        yes
```

```
?- [_,X2,_|R] = [1, 2, 3, 4, 5].
```

```
Prolog: R = [4,5]
        X2 = 2
        yes
```

Liste

Da li je element član liste

```
clan(X,[X|_]).
clan(X,[_|R]):-clan(X,R).
```

`?- clan(5,[1,2,3,4,5]).`	`Prolog: true ?` `(16 ms) yes`
`?- clan(X,[1,2,3,4,5]).`	`Prolog: X = 1 ?` `yes`
`?- clan(7,[1,2,3,4,5]).`	`Prolog: no`

Liste

Spajanje dvije liste u jednu

```
spoji([],L,L).
spoji([G|R1],L,[G|R2]):-spoji(R1,L,R2).
```

```
?- spoji([1,2,3],[4,5,6],[1,2,3,4,5,6]).
   Prolog: yes

?- spoji([1,2,3],[4,5,6],X).
   Prolog: X = [1,2,3,4,5,6]
           yes

?- spoji([1,2,3],X,[1,2,3,4,5]).
   Prolog: X = [4,5]
           yes
```

Problem?

```
?- spoji([1,2,3],4,L).
   Prolog: L = [1,2,3|4]
           yes
```

Aritmetika

```
?- X = 2 + 2.    Prolog: X = 2+2
                         yes
```

(implicitno X = +(2,2))
- infiksna notacija je samo “sintaksni šećer”

```
?- X is 2 + 2.   Prolog: X = 4
                         yes
```

predikat is vrši evaluaciju

```
?- 2 + 2 = X.    Prolog: X = 2+2
                         yes
```

Varijable s desne strane moraju biti instancirane u trenutku izračuna

```
?- X is Y + 2.   Prolog: uncaught exception:
                         error(instantiation_error,(is)/2)
```

```
?- Y=5, Z=8, X is Z*Y+2.   Prolog:  X = 42
                                    Y = 5
                                    Z = 8
                                    yes
```

Aritmetika

Operatori (predikati) koji ne vrše izračun +, -, *, /, *mod(X, Y)*

Operatori (predikati) koji vrše izračun *is*, <, =<, =:=, =\=, >=, >

```
?- 4 = 2 + 2.
Prolog: no
```

```
?- 4 =:= 2 + 2.
Prolog: yes
```

```
?- 2 + 2 is 4.
Prolog: no
```

Problem?

Aritmetika

Neka vesela funkcija

```
f(X,Y,Rez):-Rez is X*X + Y*Y + 7.
```

```
?- f(2,3,X).
Prolog: X = 20
        yes
```

Faktorijel

```
fakt(0,1).
fakt(N,Rez):-
        N>0,
        X is N-1,
        fakt(X,R),
        Rez is R*N.
```

```
?- fakt(5,X).
Prolog: X = 120 ?
        yes

?- fakt(25,X).
Prolog: X = 11700624993288192O ?
        yes
```

Aritmetika

Duljina liste

```
len([],0).
len([_|R],N):-
        len(R,N1),
        N is N1+1.
```

```
?- len([1,2,3,4,5,6,7,8,9],M).
Prolog:  M = 9
         yes

?- Y = [1,2,3,4,5,6,7,8,9].
?- len(Y,M).
Prolog:  M = 0
         Y = [] ?
         yes
```

Akumulator

Akumulator – analogija varijable koja čuva privremeni rezultat

Duljina liste (2)

```
lenAkk(L,N):-lenAkk(L,0,N).

lenAkk([],Ak,Ak).
lenAkk([_|R],Ak,N):-
        Ak1 is Ak+1,
        lenAkk(R,Ak1,N).
```

Wrapper predikat

Unifikacija akumulatora i rezultata i kriterij zaustavljanja

Ponašanje programa isto kao kod programa len

Tail recursive – rezultat je izračunat kada smo došli do dna rekurzije i jednostavno ga treba prenijeti dalje

U prethodnom slučaju se rezultat gradi izlaskom iz rekurzije

Što dalje...?

- Rez
- Negacija
- NLP
- Operatori
- ... (da barem imamo vremena☺)

Literatura

- S. Šegvić: “Uvod u programski jezik Prolog”, http://www.zemris.fer.hr/~ssegvic/pubs/prolog.pdf
- P. Blackburn, J. Bos, K. Striegnitz: “Learn Prolog Now!”, http://www.learnprolognow.org/

Zadaci

- Zadatak 1. Što će Prolog vratiti kao rezultat na sljedeće upite?
  - `[] = [_|_].`
  - `fun(X, [b, c, d]) = notfun(a, [Y|R]).`
  - `[1, 7, X, 9] = [X|R].`
  - `[[X, 2], 6, m, 9, [1, 45, 7]] = [Y, _, _|R], R = [_, Z|R2].`
- Zadatak 2. Napišite predikat dodaj/3 koji dodaje element na kraj liste.
- Zadatak 3. Napišite predikat usporedi/2 koji vraća *true* ukoliko su njegovi argumenti dvije liste jednakih duljina.
- Zadatak 4. Napišite predikat pot/3 koji računa potenciju dvaju brojeva
- Zadatak 5. Napišite predikat zbroji/2 koji računa zbroj vrijednosti elemenata liste.
- Zadatak 6. Napišite predikat okreni/2 koji vraća *true* ukoliko su njegovi argumenti dvije zrcalne liste. Po mogućnosti koristite akumulator

Rješenja☺

1

```
a) false.
b) false.
c) X = 1, R = [7, 1, 9].
d) Y = [X, 2], R = [9, [1, 45, 7]], Z = [1, 45, 7], R2 = [].
```

2

```
dodaj([],X,[X]).
dodaj([G|L],X,[G|L2]):-dodaj(L,X,L2).
```

3

```
usporedi([],[]).
usporedi([_|L1],[_|L2]):-usporedi(L1,L2).
```

4

```
pot(_,0,1).
pot(X,Y,Rez):-Y>0,Temp is Y-1,pot(X,Temp,R),Rez is R*X.
```

5

```
zbroji([],0).
zbroji([H|T],Rez):-
        zbroji(T,R),
        Rez is R + H.
```

6

```
okreni2(X,Y):-okreni2(X,[],Y).
okreni2([G|R],L,Y):-okreni2(R,[G|L],Y).
okreni2([],X,X).
```

Dodatni zadaci

- Zadatak 1. Što će Prolog vratiti kao rezultat na sljedeće upite?
  - `+(+(3,3),2) =:= +(3,+(2,3)).`
  - `[a(X),42,'brom',n,5] = [Y,_,X|R].`
  - `[m(X),[],Y,k,1,3]=[Y,_,X+1|R].`
  - `X=_veselo(mama(X),tata(Y)).`
  - `'Slon'=Slon.`

- Zadatak 2. Realizirati predikat fibonacci/2 koji računa n-ti član fibonaccijevog niza definiranog s:

$$F_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F_{n-1} + F_{n-2} & \dot{n} > 1 \end{cases}$$

- Zadatak 3. Realizirati predikat kompresiraj/2 koji uklanja jednake uzastopne elemente u listi npr.:

```
?- kompresiraj([1,1,1,1,2,2,2,3,3,4,4,4,4,4,5,5,6],X).
```