Laborator 1

SICStus Prolog 3.12.2: https://mega.co.nz/#F!BMV1jQRR!cN5vJXAmO1pFiwLPhlf9iA

•Sicstus Prolog este un limbaj de programare simbolic, potrivit pentru probleme ce implica obiecte si relatii intre obiecte. Un program Prolog este o baza de cunostinte care poate fi interogata.

•Sa consideram urmatorul program Prolog (salvat intr-un fisier text cu extensia **pl):**

```
parinte(ion,maria).
parinte(ana,maria).
parinte(ana,dan).
parinte(maria,elena).
parinte(maria,radu).
parinte(elena,nicu).
parinte(radu,gigi).
parinte(radu,dragos).
```

•Pentru a interoga un program Prolog, acesta trebuie consultat.

```
consult('c:\\prolog\\pro.pl').
```

•In fereastra de interogare introduceti urmatoarele cereri:

parinte(ana,maria).
parinte(ion,radu).
parinte(X,maria).
parinte(X,Y).

Obs.

1) constantele alfanumerice (atomi) incep cu litera mica
2) variabilele incep cu litera mare
3) raspunsurile se obtin in ordinea in care sunt puse informatiile in baza de cunostinte
4) daca intrebarea contine variabile, Prolog gaseste obiectele particulare (instante) pentru care raspunsul este adevarat

•Pentru a gasi cine este bunicul lui radu, putem intreba

parinte(X,Y), parinte(Y,radu).

Reguli Prolog

•O regula are forma generala R:-C1, C2, …, CN si are semnificatia:

" X1, X2,…Xk variabile ce apar in regula, R este adevarat daca C1, C2, …, CN.

Exemplu: copil(X, Y):-parinte(Y, X).

adica “X,Y daca Y este parintele lui X atunci X este copilul lui Y”.

❑Plecand de la relatia parinte, se se scrie relatiile frate(X,Y) (X este frate cu Y) si bunic(X,Y) (X este bunicul lui Y).

•Pentru a defini o relatie predecesor, avem nevoie de o regula recursiva:

pred(X,Y):-parinte(X,Y).

pred(X,Z):-parinte(X,Y), pred(Y,Z).

Obs. Domeniul de valabilitate al unei variabile este clauza in care apare variabila respectiva.

•In Prolog putem lucra cu structuri – mai multe componente combinate intr-un singur obiect si se poate descrie in general astfel:

functor(arg1, arg2, …, argN).

Obs.

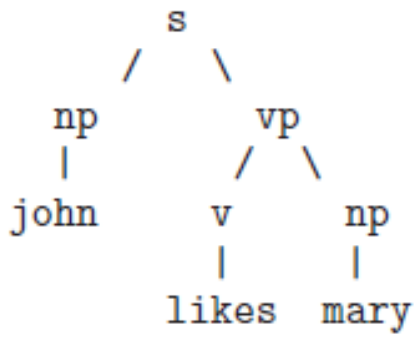
1) functorul este atom (incepe cu litera mica)
2) argumentele pot fi structuri
3) o structura este bine definita de functorul principal si de aritate (nr de argumente)

Exemple

data(1,octombrie,2005).

segment(punct(1,2),punct(6,7)).

s(np(john),vp(v(likes),np(mary)))



Unificarea (matching)
Este singura operatie permisa intre termenii Prolog.

Def Se numeste substitutie multimea
Q={(Xi,ti),i=1,n | Xi variabile, ti termeni Prolog si Xi ≠ Xj daca i ≠j}
Daca T este un termen Prolog, notam TQ termenul obtinut din T prin inlocuirea simultana a tuturor aparitiilor Xi cu ti unde (Xi,ti)ÎQ.

Def: Un termen este complet instantiat daca nu contine nici o variabila.

Def : T1 unifica cu T2 daca exista Q astfel incat T1Q=T2Q. Q se numeste unificator.

Exemple **1)** T1=f(X,a)
T2=f(b,Y)
Q={(X,a), (Y,b)}

2) T1=X
T2=f(g(Y,Z),b)
Q1={(X, f(g(Y,Z),b))}
Q2={(Y,a), (Z,c), (X,f(g(a,c),b))}

Q1 este mai generala decat **Q2**.

•Predicatul = spune daca 2 termeni Prolog unifica. In caz afirmativ, se da substitutia cea mai generala.

Obs.

1) doi atomi unifica daca sunt identici
2) o variabila poate unifica cu orice
3) doua structuri unifica daca au acelasi functor principal, acelasi numar de argumente si argumentele ce se corespund unifica

Aritmetica in Prolog

- = = egalitate de termeni

  1+2= = 2+1 no

- =:= egalitate de valori numerice a doua expresii aritmetice

  1+2=:=2+1 yes

- \= = termeni diferiti

- X is Y variabila X este instantiata cu valoarea Y.

  X is 3+2.

Functia $f(x) = \begin{cases} 0, x \leq 3 \\ 2, x \in (3,6] \\ 4, x > 6 \end{cases}$ poate fi implementata in Prolog prin urmatoarele reguli:

```
f(X,0):-X=<3.
f(X,2):-3<X, X=<6.
f(X,4):-6<X.
```

Daca punem intrebarea

```
f(1,Y),2<Y.
```

cautarea solutiei se face pe toate cele trei reguli desi logic ar fi trebuit sa se opreasca dupa prima regula.
Putem preveni backtracking-ul cu ajutorul predicatului ! (cut). Programul devine astfel:

```
f(X,0):-X=<3,!.
f(X,2):-3<X, X=<6,!.
f(X,4):-6<X.
```

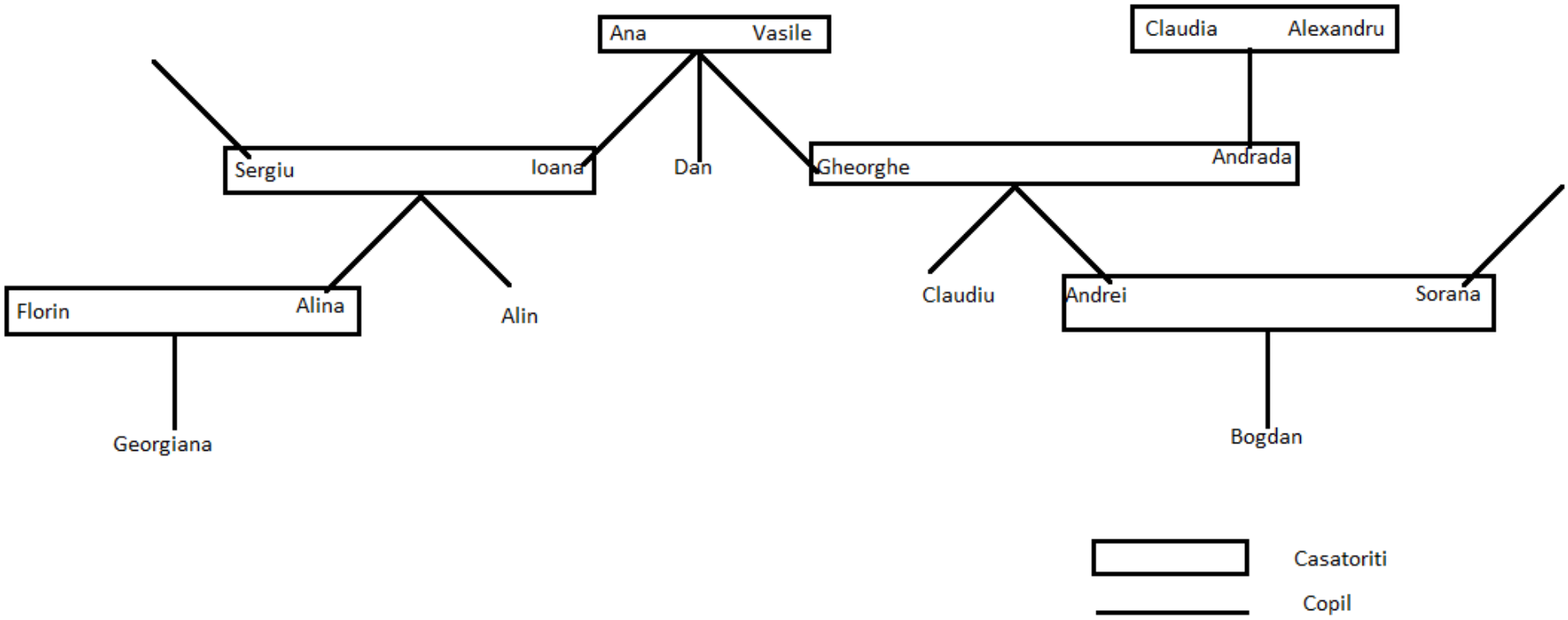
Daca X=<3 e adevarat nu se cauta mai departe (se taie backtracking-ul).
In regula 2 testul 3<X este redundant deoarece se ajunge aici doar daca nu s-a atins cut-ul din prima regula.

Deci programul poate fi scris:

```
f(X,0):-X=<3,!.
f(X,2):-X=<6,!.
f(X,4).
```

- Sa se scrie predicatul max care calculeaza maximul dintre 2 valori.

- Implementati un program prolog care sa defineasca gradele de rudenie (frate / sora, parinte, bunic, strabunic, unchi / matusa , verisori) dintre numele din figura de mai jos.



- Sa se implementeze urmatoarele operatii aritmetice in Prolog:

  - 1 + 4 ;
  - 4 – 3;
  - 4 – 5;
  - 4 / 2;
  - 2 * 4;
  - 7 / 2;
  - Restul impartirii: 7/2;
  - 3 + 5 * 2 – 1;
  - 2^3;

- Sa se implementeze urmatoarea functie in Prolog:

$$f(x)\begin{cases} x^2, & x \leq 5; \\ x/2, & x \in (5,10]; \\ x * 2, & x > 10; \end{cases}$$