Prolog

- **Prolog reprezintă o prescurtare de la "Programming în logic".**
- **Este un limbaj conceput pentru programarea logică (deci un limbaj declarativ).**
- **A fost elaborat în cursul anilor 1970 în Europa (mai ales în Franţa şi Scoţia).**
- **Primul compilator de Prolog a fost creat în 1972 de către Philippe Roussel, la Universitatea din Marsilia.**
- **Prologul este un limbaj compilat, care utilizează, în locul relaţiilor matematice, relaţii logice între mulţimi de date.**

- **Exista limbaje *algoritmice, orientate obiect* si *logice*.**

- ***Limbajele programării logice*** **(Prolog) sunt *limbaje declarative*. Un limbaj de programare declarativ scuteşte programatorul de a mai menţiona procedura exactă pe care trebuie să o execute calculatorul pentru a realiza o funcţie. Programatorii folosesc limbajul pentru *a descrie* o mulţime de *fapte* şi de *relaţii* astfel încât utilizatorul să poată *interoga* apoi sistemul pentru a obţine un anumit rezultat.**

- Programele Prolog opereaza cu obiecte si cu relatii intre obiecte (*fapte* si/sau *clauze*).

- Negatia joaca un rol foarte important in programarea in acest limbaj, deoarece orice proprietate care nu se poate deduce din entitatile programului este considerata falsa, deci negatia ei este adevarata.

- Prolog foloseste un model de rationament minimal numit *ipoteza lumii inchise*. Conform acestui model, tot ceea ce nu este stiut de program, deci afirmat explicit ca fiind adevarat in program, este considerat fals.

- Prologul modeleaza negatia ca esec al satisfacerii unui scop (*negatia ca insucces*), aceasta fiind de fapt o particularizare a ipotezei lumii inchise.

- **O *clauza* Prolog sau *regula* este de forma:**

H :- B1, B2, … , Bn.

unde: H este *capul regulii*, iar membrul drept constituie *corpul regulii* (care este o conjunctie sau o disjunctie de *scopuri*).

- **Sensul clauzei Prolog anterioare este:**

If B1 & B2 & … & Bn then H

- **In Prolog *constantele* se scriu cu litera mica, iar *variabilele* cu litera mare.**
- **Simbol……..Sens**

:-	**daca**
,	**conjunctie**
;	**disjunctie**
_	**variabila universala (semnifica orice)**

- **O *fapta Prolog* este o clauza fara corp, adica de forma H. Ea reprezinta o clauza care este tot timpul adevarata, deoarece nu este conditionata.**

- **Structura unui program Prolog:**

➢ **definitii constante**

➢ **tipuri de obiecte definite de utilizator**

➢ **declaratiile predicatelor**

➢ **reguli**

- **Un program Prolog contine urmatoarele *entitati*:**
  - ***fapte*** **despre obiecte si relatiile existente intre aceste obiecte;**
  - ***reguli*** **despre obiecte si relatiile dintre ele, care permit deducerea (inferarea) de noi fapte pe baza celor cunoscute;**
  - ***intrebari*****, numite si *scopuri*, despre obiecte si relatiile dintre ele, la care programul raspunde pe baza faptelor si regulilor existente.**

- **Baza de cunostinte Prolog:**

  Multimea *faptelor* unui program Prolog formeaza baza de cunostinte Prolog. In baza de cunostinte a unui program Prolog sunt incluse si *regulile* Prolog.

- **Scopuri:**

  Scopurile sunt *predicate* pentru care se doreste aflarea valorii de adevar, in contextul faptelor existente in baza de cunostinte.

- **Rezultatul unui program Prolog:**

  Obtinerea *consecintelor* sau a *rezultatului* unui program Prolog se face prin fixarea unor scopuri care pot fi adevarate sau false, in functie de continutul bazei de cunostinte. Cum scopurile pot fi vazute ca intrebari, rezultatul unui program Prolog este raspunsul la o intrebare sau la o conjunctie de intrebari.

- **O *regula* Prolog exprima un fapt care depinde de alte fapte si este de forma**

S :- S1, S2, … , Sn.

cu S reprezentand *capul* sau *antetul* regulii, iar membrul drept reprezentand *corpul* regulii.

- **Constantele se mai numesc *atomi simbolici* si denumirile lor incep cu litera mica.**

- **Numele de *variabile* incep cu litera mare; o variabila poate fi *instantiata sau legata* daca exista un obiect asociat acestei variabile sau *neinstantiata, respectiv nelegata, respectiv libera* daca nu se stie inca ce obiect va desemna variabila.**

La fixarea unui scop Prolog care contine variabile, acestea sunt neinstantiate, iar sistemul incearca satisfacerea acestui scop cautand printre faptele din baza de cunostinte un fapt care poate identifica cu scopul, printr-o instantiere adecvata a variabilelor din scopul dat. Este vorba de fapt de un *proces de unificare* a predicatului scop cu unul din predicatele fapte existente in baza de cunostinte. La incercarea de satisfacere a scopului, cautarea se face intotdeauna pornind de la inceputul bazei de cunostinte. Exista atatea solutii cate unificari diferite exista. Obtinerea primei solutii este numita *satisfacerea scopului*, iar obtinerea altor solutii *resatisfacerea scopului*. La satisfacerea unui scop cautarea se face intotdeauna de la inceputul bazei de cunostinte. La resatisfacerea unui scop cautarea se face incepand de la marcajul stabilit de satisfacerea anterioara a acelui scop.

Obtinerea solutiilor atunci cand baza de cunostinte Prolog contine si reguli:

In acest caz, *unificarea scopului* se incearca atat cu *fapte* din baza de cunostinte, cat si cu *antetul* regulilor din baza.

La unificarea unui scop cu *antetul* unei reguli, pentru a putea satisface acest scop trebuie satisfacuta regula. Aceasta revine la a satisface toate faptele din corpul regulii, deci conjunctia de scopuri. Scopurile din corpul regulii devin *subscopuri* a caror satisfacere se va incerca printr-un mecanism similar cu cel al satisfacerii scopului initial.

Comportarea sistemului Prolog in care se incearca in mod repetat satisfacerea si resatisfacerea scopurilor din conjunctia de scopuri se numeste *backtracking*.

<u>Un exemplu de program Prolog</u> (definirea unor relatii de familie):

- **faptul ca Tom este parinte al lui Bob se poate scrie in Prolog astfel:**

parinte(tom, bob).

Aici *parinte* este numele relatiei, iar *tom* si *bob* sunt argumentele sale.

- **Program Prolog care defineste relatii de familie:**

parinte(pam, bob).
parinte(tom, bob).
parinte(tom, liz).
parinte(bob, ann).
parinte(bob, pat).
parinte(pat, jim).

Programul consta din *6 clauze*. Fiecare dintre aceste clauze declara un fapt despre relatia *parinte*. Spre exemplu, *parinte(tom, bob)* este o instantiere particulara a relatiei *parinte*. In general, o relatie se defineste ca fiind multimea tuturor instantierilor sale.

Obs.: Acest program simplu nu contine (inca) regului (ci doar fapte).

Interogarea Prologului:

Ex.: Vrem sa aflam daca Bob este parinte al lui Pat. Aceasta intrebare este comunicata sistemului Prolog tastandu-se la terminal:

?- parinte(bob, pat).

Raspunsul Prologului va fi

yes

Alte exemple de interogari:

?-parinte(liz, pat).

no

?-parinte (tom, ben).

no

Cine este parintele lui Liz?

?-parinte(X, liz).

X = tom

- **Cine sunt copiii lui Bob?**

?-parinte (bob, X).

X = ann;

X= pat;

no

- **Cine este parinte al cui?**

?-parinte(X, Y).

X = pam

Y = bob;

X = tom

Y = bob;

X = tom

Y = liz;

…

- **Cine este un *bunic* al lui Jim? i.e.**

(1) Cine este parinte al lui Jim? Presupunem ca acesta este un Y.

(2) Cine este parinte al lui Y? Presupunem ca acesta este un X.

(1) si (2) necesita urmatoarea interogare compusa:

?- parinte(Y, jim), parinte(X, Y).

Raspunsul Prologului este:

X = bob

Y = pat

Obs.: Interogarea

?- parinte(X, Y), parinte(Y, jim).

va produce acelasi rezultat.

- **Cine sunt nepotii lui Tom?**

?- parinte(tom, X), parinte (X, Y).

X = bob

Y = ann;

X = bob

Y = pat

- **Ann si Pat au un parinte comun?**

?-parinte(X, ann), parinte(X, pat).

X = bob

<u>Extinderea programului cu reguli</u>:

- **Introducem, mai intai, urmatoarele fapte noi:**

feminin(pam).

masculin(tom).

masculin(bob).

feminin(liz).

feminin(pat).

feminin(ann).

masculin(jim).

si

sex(pam, feminin).

sex(tom, masculin).

sex(bob, masculin).

…

si

urmas(liz, tom).

…

Obs.: Relatia *urmas* poate fi definita intr-un mod mult mai elegant prin folosirea relatiei deja definite *parinte*.

Pentru toti X si toti Y,

Y este urmas al lui X daca

X este parinte al lui Y.

Aceasta interpretare genereaza o clauza Prolog reprezentata de urmatoarea *regula:*

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

- **Interogarea Prologului cand baza de cunostinte contine si aceasta regula:**

  ?- urmas(liz, tom).

  Cum lucreaza Prologul acum: Intrucat nu exista fapte referitoare la urmasi, trebuie folosita regula, astfel: variabilele X si Y vor fi instantiate in felul urmator

  X = tom si Y = liz

<u>Continuare</u>:

- **Dupa instantiere se obtine un caz particular al regulii generale. Acest caz particular este:**

  urmas(liz, tom) :- parinte(tom, liz).

- **Corpul regulii reprezinta partea de conditie, iar antetul ei partea de concluzie. Prolog incearca sa determine daca partea de conditie este adevarata i.e. daca parinte(tom, liz) este adevarat. In acest moment, *scopul initial*, urmas(liz,tom), a fost inlocuit cu *subscopul* parinte(tom, liz).**
- **Noul scop este imediat satisfacut intrucat el reprezinta o fapta Prolog (din baza de cunostinte).**
- **Aceasta inseamna ca partea de concluzie este, de asemenea, adevarata, iar Prolog va raspunde la intrebare cu *yes*.**