Czibula, G., Pop, H.F., *Elemente avansate de programare în Lisp şi Prolog. Aplicaţii în Inteligenţa Artificială.*, Ed. Albastră, Cluj-Napoca, 2012

Capitolul 11

Elemente fundamentale ale limbajului Prolog

11.1. Scurt istoric al limbajului Prolog

Limbajul Prolog (PROgrammation en LOGique) a fost elaborat la Universitatea din Marsilia în jurul anului 1970, ca instrument pentru programarea şi rezolvarea problemelor ce implicau reprezentări simbolice de obiecte şi relaţii dintre acestea.

Primul interpretor de Prolog a fost scris în limbajul Fortran. Prolog a fost inclus de Japonia ca limbaj de bază în planul de dezvoltare a sistemelor de generaţia a cincea.

Ca şi Lisp, Prolog este un limbaj cu fundament matematic. Dar, spre deosebire de Lisp, specializat în definirea de funcţii, Prolog permite definirea şi prelucrarea relaţiilor.

Limbajul Prolog are un pronunţat caracter descriptiv: un program Prolog este o colecţie de definiţii ce descriu relaţii sau funcţii de calculat. Execuţia programului constă din utilizarea definiţiilor pentru a găsi o relaţie sau un obiect corespunzător unor specificaţii date. Soluţia problemelor nu se mai vede ca o execuţie pas cu pas a unei secvenţe de instrucţiuni.

Prolog are un câmp de aplicaţii foarte larg: baze de date relaţionale, inteligenţă artificială, logică matematică, demonstrarea de teoreme, sisteme expert, rezolvarea de probleme abstracte sau ecuaţii simbolice, etc.

În această parte vom studia implementarea Turbo Prolog 2.0 a firmei Borland International.

11.2. De la limbajul natural la programele Prolog

In Prolog, soluţiile se obţin prin deducţii logice plecând de la lucruri deja cunoscute. Un program Prolog nu este o înşiruire de acţiuni de îndeplinit, ci o colecţie de fapte împreuna cu regulile după care se pot trage concluzii din aceste fapte. Prolog se bazează pe un subset al logicii predicatelor. Oferă o sintaxă foarte apropiată limbajului natural. O facilitate importantă a limbajului Prolog este că, în plus faţă de găsirea logică a raspunsurilor la întrebări, poate trata alternativele şi să găsească toate soluţiile, nu numai una dintre ele.

Iată câteva exemple de afirmaţii din limbajul natural şi 'traducerile' respective în logica predicatelor:

- O maşină rapidă este distractivă. — distractiv(maşină_rapidă).
- O maşină mare este utilă. — util(maşină_mare).

- Lui Bill îi place o maşină dacă maşina este distractivă. — place(bill, Maşină) if distractiv(Maşină).

11.2.1. Propoziţii: fapte şi reguli

Un programator Prolog va defini obiecte şi relaţii, apoi va defini reguli despre când aceste relaţii vor fi adevarate.

11.2.1.1. Fapte: ceea ce se cunoaşte

În Prolog, o relaţie între obiecte se numeşte fapt. În limbajul natural o relaţie este simbolizată de o propoziţie. În Prolog, o relaţie este sumarizată printr-o frază simplă ce constă din numele relaţiei urmat între paranteze de obiectele participante la relaţie. Ca şi în limbajul natural, un fapt se termină cu punct. Exemplul următor prezintă mai multe fapte exprimând noţiunea de plăcere din limbajul natural:

- Vasile o place pe Ana.
- Ana îl place pe Vasile.
- Lui Vasile îi plac câinii.

Iată aceste fapte scrise în Prolog:

- place(vasile, ana).
- place(ana, vasile).
- place(vasile, câini).

Faptele pot exprima şi proprietăţi. De exemplu, afirmaţia “Renault este o maşină” se scrie în Prolog astfel:

- maşină(renault).

11.2.1.2. Reguli: ce anume se poate deduce din faptele date

Regulile permit deducerea de fapte din alte fapte. Altfel spus, o regulă este o concluzie care se ştie că este adevărată atunci când alte concluzii sau fapte sunt adevarate. Iată câteva reguli privind relatie de plăcere de mai sus:

- Anei îi place tot ceea ce îi place lui Vasile.
- Mariei îi place orice maşină.

Fiind date aceste reguli, se poate deduce din faptele cunoscute deja, că:

- Anei îi place de Ana.
- Mariei îi place Renault-ul.

Iată cum se scriu aceste reguli în Prolog:

- place(ana, Ceva) if place(vasile, Ceva).
- place(maria, Ceva) if maşină(Ceva).

11.2.2. Întrebări

Odată ca am scris o serie de fapte, putem începe să punem întrebări relativ la aceste fapte. În limbajul natural am întreba:

- Ii place lui Vasile de Ana?

În sintaxă Prolog, vom scrie

- place(vasile, ana).

La această întrebare, Prolog va răspunde

- Yes

Întrebarea

- Ce îi place lui Vasile?

va fi redactată în Prolog astfel:

- place(vasile, Ce).

Notaţi că sintaxa Prolog nu se schimbă atunci când puneţi o întrebare. Este important de notat că al doilea obiect (*Ce*) începe cu majusculă, în timp ce primul (*vasile*) începe cu minusculă. Aceasta deoarece *vasile* este o constantă (o valoare cunoscută) iar *Ce* este o variabilă. În general, constantele încep cu literă mică, iar variabilele cu literă mare.

La întrebarea de mai sus Prolog va raspunde

- Ce = câini
  Ce = ana
  2 Solutions

Ca un alt exemplu, la întrebarea

- place(ana, Ce).

Prolog va răspunde

- Ce = caini
  Ce = ana
  Ce = vasile
  3 Solutions

11.2.3. Asamblarea faptelor, regulilor şi întrebărilor

Să analizăm următorul program Prolog:

```
place(ellen, tenis).
place(john, fotbal).
place(tom, basket).
place(eric, inot).
place(mark, tenis).
place(bill, Activitate) :-
        place(tom, Activitate).
```

La întrebarea

?place(bill, basket).

Prolog va răspunde cu

true

11.2.4. Variabile: propoziţii generale

În Prolog, variabilele ne ajută să scriem fapte generale şi reguli, şi să cerem răspuns la întrebări. Necesitatea variabilelor apare mai ales în cazul întrebărilor compuse. Afirmaţii generale tipice ar putea fi:

- Lui Bill îi plac aceleaşi lucruri ca şi lui Kim.
- Temperatura curentă este caldă.

Iată aceste lucruri scrise în Prolog:

- place(bill, Lucruri) and place(kim, Lucruri).
- temperatura(TempCrt) and cald(TempCrt).

11.3. Elemente ale limbajului Prolog

11.3.1. Clauze (fapte şi reguli)

Principial, în Prolog exista doar două tipuri de fraze: fapte şi reguli. Aceste fraze sunt cunoscute sub numele de clauze.

11.3.1.1. Fapte

Un fapt reprezintă o singură apariţie fie a unei proprietăţi a unui obiect, fie a unei relaţii între obiecte. Un fapt este de sine stătător; Prolog nu are nevoie să caute altceva pentru confirmarea unui fapt, şi faptul poate fi folosit ca bază pentru inferenţe.

11.3.1.2. Reguli

Constructorul din Prolog care descrie ceea ce se poate deduce din alte informaţii este regula. O regulă este o proprietate sau o relaţie care se ştie că este adevarată atunci când alte relaţii sunt adevărate.

De exemplu:

1. diana_poate_mânca(Mâncare_de_pe_meniu) if
   vegetal(Mâncare_de_pe_meniu) and
   pe_lista_doctorului(Mâncare_de_pe_meniu).

2. părinte(Persoana1, Persoana2) if
   tata(Persoana1, Persoana2).
   părinte(Persoana1, Persoana2) if
   mama(Persoana1, Persoana2).

Se observă că ‘sau’ din limbajul natural se traduce în Prolog prin două reguli separate:

- Persoana1 este părintele lui Persoana2 dacă Persoana1 este tatăl lui Persoana2.
- Persoana1 este părintele lui Persoana2 dacă Persoana1 este mama lui Persoana2.

3. poate_cumpăra(Nume, Model) if
   persoană(Nume) and
   maşină(Model) and
   place(Nume, Model) and
   de_vânzare(Model).

11.3.2. Predicate (relaţii)

Numele simbolic al unei relaţii se numeşte numele predicatului. Obiectele la care se referă se numesc argumentele sale.

Iată câteva exemple de predicate Prolog cu zero sau mai multe argumente:

- pred(integer, symbol)
- person(last, first, gender)
- run
- insert_mode
- birthday(firstName, lastName, date)

11.3.3. Variabile (clauze generale)

Într-o întrebare simplă, puteţi utiliza variabile pentru a cere sistemului Prolog să găsească informaţii. De exemplu, întrebarea

place(X, tenis).

foloseşte litera X ca variabilă, pentru a indica o persoană necunoscută. Numele de variabile trebuie să înceapă cu literă mare, după care pot urma litere, cifre şi caracterul underline (_).

11.3.4. Cum îşi primesc valori variabilele?

Aţi observat că Prolog nu are instrucţiuni de atribuire. Variabilele în Prolog îşi primesc valorile prin potrivire cu constante din fapte sau reguli.

Până când o variabilă primeşte o valoare, ea este **liberă** (free); când variabila primeşte o valoare, ea este **legată** (bound). Dar ea stă legată atâta timp cât este necesar pentru a obţine o soluţie a problemei. Apoi, Prolog o dezleagă, face backtracking şi caută soluţii alternative.

Observaţie. Este important de reţinut că nu se pot stoca informaţii prin atribuire de valori unor variabile. Variabilele sunt folosite ca parte a unui proces de potrivire, nu ca un tip de stocare de informaţii.

De exemplu, fie programul

```
place(ellen, citit).
place(john, calculatoare).
place(john, badminton).
place(leonard, badminton).
place(eric, înot).
place(eric, citit).
```

Fie intrebarea

```
place(Persoana, citit), place(Persoana, înot).
```

Prolog va rezolva cele două părţi ale întrebării separat, căutând în clauzele programului, de sus în jos. In prima parte,

```
place(Persoana, citit)
```

variabila Perosana este liberă. Prolog caută un fapt care să se potrivească cu această primă parte:

```
place(ellen, citit)
```

Deci, Prolog leagă variabila liberă Persoana de constanta ellen. Acum, Prolog verifică a doua parte a întrebării. Deoarece Persoana este legată de ellen, trebuie verificat că

```
place(ellen, inot)
```

Prolog caută o potrivire a acestui fapt, dar nu găseşte. Deci, a doua parte a întrebarii nu este adevărată când Persoana este ellen.

Acum Prolog dezleagă Persoana, şi încearcă să găsească o altă soluţie a primei părţi a întrebării, cu Persoana ca variabilă liberă. Mergând în continuare cu verificările, Prolog va lega Persoana de constanta eric. De această dată şi partea a doua va fi satisfacută, deci Prolog va returna

```
Persoana=eric
1 Solution
```

Observaţie. Mecanismul prin care Prolog încearcă să ‘potrivească’ partea din întrebare pe care doreste să o rezolve cu un anumit predicat se numeste unificare. Ca rezultat al unificării, Prolog a legat variabila Persoana într-o primă fază de constanta ellen, şi mai apoi de constanta eric.

Observaţie. Această întrebare este o întrebare compusă. O întrebare compusă poate să conţină şi operatorul ‘or’. În acest caz, ‘and’ este mai prioritar decat ‘or’, iar părţile de verificat vor fi identificate în acord cu acest lucru.

11.3.5. Variabile anonime

Dacă prin punerea unei întrebări doriţi să obţineţi doar informaţii parţiale, puteţi să utilizaţi variabilele anonime pentru a neglija acele părti care nu vă interesează. În Prolog, variabila anonimă este reprezentată de caracterul underline (_).

De exemplu, dacă dorim să ştim dacă îi place cuiva să înoate, dar nu vrem să ştim cui anume, vom întreba

place(_, inot).

De asemenea, variabilele anonime pot fi folosite în reprezentarea faptelor:

are(_, pantofi).

reprezintă

Toata lumea are pantofi.

11.3.6. Ce este o potrivire?

Iată câteva reguli ce vor explica termenul 'potrivire':

1. Structuri identice se potrivesc una cu alta
   - parinte(joe, tammy) se potriveste cu parinte(joe, tammy)
2. De obicei o potrivire implică variabile libere. Dacă X e liberă,
   - parinte(joe, X) se potriveste cu parinte(joe, tammy)
   - X este legat la tammy.
3. Dacă X este legat, se comportă ca o constantă. Cu X legat la tammy,
   - parinte(joe, X) se potriveste cu parinte(joe, tammy)
   - parinte(joe, X) NU se potriveste cu parinte(joe, millie)
4. Două variabile libere se potrivesc una cu alta.
   - parinte(joe, X) se potriveste cu parinte(joe, Y)

11.3.7. Modele de flux

În Prolog, legările de variabile se fac în două moduri: la intrarea în clauză sau la ieşirea din clauză. Direcţia în care se leagă o valoare se numeşte ‘model de flux’. Când o variabilă este dată la intrarea într-o clauză, aceasta este un parametru de intrare (i), iar când o variabilă este

dată la ieşirea dintr-o clauză, aceasta este un parametru de ieşire (o). O anumită clauză poate să aibă mai multe modele de flux. De exemplu clauza

factorial (n,f)

(care va calcula factorialul lui n) poate avea următoarele modele de flux:

- (i,i) - verifică dacă n! = f;
- (i,o) - atribuie f := n!;
- (o,i) - găseşte acel n pentru care n! = f.

Observaţie. Proprietatea unui predicat de a funcţiona cu mai multe modele de flux depinde de abilitatea programatorului de a programa predicatul în mod corespunzător.

Observaţie. În programele Prolog, cuvântul 'if' se poate schimba cu **':-'** (două-puncte liniuţă), cuvântul 'and' se poate schimba cu **','** (virgulă) iar cuvântul 'or' se poate schimba cu **';'** (punct-virgulă).

De exemplu, următoarele două porţiuni de cod sunt echivalente:

```
e_mai_bătrân (P1, P2) if
        vârsta(P1, V1) and
        vârsta(P2, V2) and
        V1 > V2.
```

```
e_mai_bătrân(P1, P2) :-
        vârsta(P1, V1),
        vârsta(P2, V2),
        V1 > V2.
```

11.3.8. Comentarii

Comentariile se includ între secvenţele '/*' şi '*/'. Alternativ, tot ceea ce apare pe o linie după caracterul '%' este comentariu:

```
/* comentariu */

% comentariu

/*****************/
/* comentariu */
/*****************/
```