1. Giới Hiệu về Prolog

Prolog là một ngôn ngữ cấp cao, có đặc điểm gần với ngôn ngữ tự nhiên, từ những người mới học đến những lập trình viên chuyên nghiệp đều có thể tiếp cận một cách nhanh chóng, viết ra một chương trình ứng dụng hữu ích.

Prolog ra đời vào năm 1973 do C.Camerauer (Đại học Marseilles, Pháp) và nhóm đồng sự phát triển. Từ đó đến nay, qua nhiều lần cải tiến, đặc biệt hãng Borland cho ra đời phần mềm TURBO PROLOG với nhiều ưu điểm, thuận tiện cho người sử dụng. Để giải quyết một số vấn đề, ta nhận thấy sử dụng ngôn ngữ Prolog cho ta chương trình gọn nhẹ hơn nhiều so với các ngôn ngữ khác.

2. Đặc trưng

- Đặc điểm của ngôn ngữ là xử lý tri thức của các bài toán được mã hóa bằng ký hiệu.
- Một điểm mạnh khác của ngôn ngữ là xử lý danh sách trên cơ sở xử lý song song và đệ qui với các thuật toán tìm kiếm.
- Ngôn ngữ cho phép liên kết với các ngôn ngữ khác như C, Pascal và Assembler.Các yếu tố cơ bản:

Trong một chương trình Prolog, ta cần khai báo các yếu tố sau đây: đối tượng, quan hệ giữa các đối tượng, sự kiện và các luật.

Đối tượng:

Gồm có các hằng và biến. Hằng mang giá trị cho sẵn ở đầu chương trình hoặc trong quá trình viết ta đưa vào; Các biến có giá trị thay đổi sẽ được gán giá trị khi chạy chương trình. Tên biến là một ký tự hoa hoặc một chuỗi ký tự, bắt đầu bằng một ký tự hoa.

Có một loại biến đặc biệt gọi là biến tự do, biến này không có tên và người ta dùng ký hiệu _ (dấu gạch dưới) thay cho tên biến.

Quan hệ giữa các đối tượng:

Quan hệ giữa các đối tượng được dùng dưới hình thức vị từ.

Ví dụ: Thich(X,Y) là vị từ diễn tả câu “X thích Y” trong ngôn ngữ tự nhiên.

Blue(car) là vị từ diễn tả câu “Car is blue”.

Như vậy các vị từ sẽ bao gồm tên của vị từ và các đối số của nó. Các đối số được đặt trong ngoặc và phân cách nhau bởi dấu phẩy.

Sự kiện và luật:

Sự kiện là một vị từ diễn tả một sự thật.

Ví dụ: “2 là một số nguyên tố” là một sự kiện vì nó diễn tả sự thật 2 là một số nguyên tố.

Luật là vị từ diễn tả quy luật suy diễn mà ta công nhận đúng. Luật được trình bày dưới dạng một mệnh đề.

Ví dụ để suy diễn số nguyên N bất kỳ là một số nguyên tố ta viết:

“N là một số nguyên tố nếu N>0, M là số nguyên tố nào đó, M<N và N không chia hết cho M”.

Cấu trúc của một chương trình Prolog:

Một chương trình Prolog thường gồm có 3 hoặc 4 đoạn cơ bản: clauses, predicates, domains và goal. Phần goal có thể bỏ đi, nếu ta không thiết kế goal trong chương trình, thì khi thực hiện, hệ thống sẽ yêu cầu ta nhập goal vào.

Phần Domains:

Đây là phần định nghĩa kiểu mới dựa vào các kiểu đã biết. Các kiểu được định nghĩa ở đây sẽ được sử dụng cho các đối số trong các vị từ. Nếu các vị từ sử dụng đối số có kiểu cơ bản thì có thể không cần phải định nghĩa lại các kiểu đó. Tuy nhiên để cho chương trình sáng sủa, người ta sẽ định nghĩa lại cả các kiểu cơ bản.

Cú pháp: <**danh sách kiểu mới**> = <**kiểu đã biết**> hoặc <**danh sách kiểu mới**> = <**danh sách kiểu đã biết**>

Trong đó các kiểu mới phân cách nhau bởi dấu phẩy, còn các kiểu đã biết phân cách nhau bởi dấu chấm phẩy.

Phần Predicates:

Đây là phần bắt buộc phải có. Trong phần này chúng ta cần phải khai báo đầy đủ các vị từ sử dụng trong phần Clauses, ngoại trừ các vị từ mà Turbo Prolog đã xây dựng sẵn.

Cú pháp: <**Tên vị từ**> (<**danh sách các kiểu**>)

Các kiểu là các kiểu cơ bản hoặc là các kiểu đã được định nghĩa trong phần domains và được viết phân cách nhau bơi dấu phẩy.

Phần Clauses:

Đây là phần bắt buộc phải có dùng để mô tả các sự kiện và các luật, sử dụng các vị từ đã khai báo trong phần predicates.

Cú pháp:

<**Tên vị từ**>(<**danh sách các tham số**>) <**kí hiệu**>

<**Tên vị từ 1**>(<**danh sách các tham số 1**>) <**kí hiệu**>

… … …

<**Tên vị từ N**>(<**danh sách các tham số N**>) <**kí hiệu**>

Trong đó: *Tên vị từ* phải là các *tên vị từ* đã được khai báo trong phần predicates. Các *tham số* có thể là các hằng hoặc biến có kiểu tương thích với các kiểu tương ứng đã được khai báo trong các vị từ ở trong phần predicates; các tham số được viết cách nhau bởi dấu phẩy. Các *kí hiệu* bao gồm:

:- (điều kiện nếu).

, (điều kiện và).

; (điều kiện hoặc).

. (kết thúc vị từ)

Phần Goal

Bao gồm các mục tiêu mà ta yêu cầu Turbo Prolog xác định và tìm kết quả. Đây là phần không bắt buộc phải có. Nếu ta viết sẵn trong chương trình thì đó gọi là goal nội; Nếu không, khi chạy chương trình Turbo Prolog sẽ yêu cầu ta nhập goal vào, lúc này gọi là goal ngoại.

Cú pháp phần goal giống như cú pháp phần clauses. Tức là ta đưa vào một hoặc một số các vị từ.

Nếu tất cả các tham số của vị từ là hằng thì kết quả nhận được là Yes (đúng) hoặc No (sai). Nếu trong các tham số của vị từ có biến thì kết quả trả về sẽ là các giá trị của biến.

Ngoài các phần chủ yếu nói trên, ta có thể đưa vào các phần liên quan đến khai báo hằng, các tập tin liên quan hoặc chỉ thị dịch.

Nguyên Tắc

Có 2 nguyên tắc: đồng nhất và quay lui

- ❖ Đồng nhất:
  - Một quan hệ có thể đồng nhất với một quan hệ nào đó cùng tên, cùng số lượng tham số, các đại lượng con cũng đồng nhất theo từng cặp
  - Một hằng có thể đồng nhất với một hằng
  - Một biến có thể đồng nhất với một hằng nào đó và có thể nhận luôn giá trị hằng đó
- ❖ Nguyên tắc quay lui (backtract, backtracting)
  - Cần chứng minh Goal : :- g1 , g2 , …, gj-1 , gj , …, gn
  - Kiểm chứng từ trái sang phải, đến gi là sai, hệ thống cần phải qui lui lại gi-1

Bộ ký tự và từ khóa

- ❖ Prolog dùng bộ ký tự sau:
  - ➢ Các chữ cái và chữ số (A – Z, a – z, 0 – 9);
  - ➢ Các toán tử (+, -, *, /,)
  - ➢ Các ký hiệu đặc biệt
- ❖ Một vài từ khóa
  - ➢ Trace: Khi có từ khoá này ở đầu chương trình, thì chương trình được thực hiện từng bước để theo dõi
  - ➢ Fail: Khi ta dùng goal nội, để nhận về tất cả các kết quả khi chạy goal nội, ta dùng toán tử Fail
  - ➢ ! hay còn gọi là nhát cắt, nhận chỉ một kết quả từ goal ngoại, ta dùng ký hiệu !

Kiểu dữ liệu chuẩn

- ❖ Kiểu do prolog định nghĩa sẵn: char,integer, real string và symbol
  - ➢ char: ký tự, hằng phải nằm trong dấu nháy: „a‟, „#‟
  - ➢ integer: -32768 đến 32767
  - ➢ real: số thực
  - ➢ string: chuỗi ký tự, hằng chuỗi ký tự nằm trong dấu nháy kép; ”prolog”
- ❖ Kiểu mẩu tin
  - ➢ Cú pháp = tên mẩu tin (danh sách các kiểu phần tử)

Các hàm xuất nhập chuẩn

- ❖ Xuất ra màn hình

- write(Arg1, Arg2, … ,Argn) in ra màn hình giá trị của các đối số.
- writef(đinh_dang, Arg1, Arg2, … ,Argn) in ra màn hình giá trị của các đối số theo định_dạng

❖ Các định_dạng
- "%d": In số thập phân bình thường; đối số phải là char hoặc integer
- "%c": Đối số là một số integer, in ký tự có mã Ascci là đối số đó, chẳng hạn writef("%c",65) được A
- "%e": In số thực dưới dạng lũy thừa của 10
- "%x": In số Hexa; đối số phải là char hoặc integer
- "%s": In một chuỗi hoặc một symbol

❖ Nhập vào từ bàn phím
- Readln(X): Nhập một chuỗi ký tự vào biến X
- ReadInt(X): Nhập một số nguyên vào biến X
- ReadReal(X): Nhập một số thực vào biến X
- ReadChar(X): Nhập vào một ký tự vào biến X

3. Ví dụ

QuickSort

```
split(H, [A|X], [A|Y], Z):-
  order(A, H), split(H, X, Y, Z).
split(H, [A|X], Y, [A|Z]):-
  not(order(A, H)), split(H, X, Y, Z).
split(_, [], [], []).
quicksort([], X, X).
quicksort([H|T], S, X):-
  split(H, T, A, B),
  quicksort(A, S, [H|Y]),
  quicksort(B, Y, X).
```

Tháp Hà Nội

```
hanoi(N):- move(N, left, centre, right).
move(0, _, _, _):-!.
move(N, A, B, C):-
  M is N-1,
  move(M, A, C, B), inform(A, B), move(M, C, B, A).
inform(X, Y):-
  write([move, a, disc, from, the, X, pole, to, the, Y, pole]),
  nl.
```

Đại số

```
/* Tính đạo hàm */
d(X,X,1):-!.                          /* d x     dx = 1                  */
d(C,X,0):- atomic(C).                 /* d c     dx = 0                  */
d(-U,X,-A):- d(U,X,A).                /* d -u    dx = - d u dx           */
d(U+V,X,A+B):- d(U,X,A), d(V,X,B).    /* d u+v   dx = d u dx + d v dx    */
d(U-V,X,A-B):- d(U,X,A), d(V,X,B).    /* d u-v   dx = d u dx - d v dx    */
d(C*U,X,C*A):- atomic(C), C \= X, d(U,X,A),!.   /* d c*u   dx = c*d u dx   */
d(U*V,X,B*U+A*V):- d(U,X,A), d(V,X,B).          /* d u*v   dx = u*d v dx + v*d u dx */
d(U/V,X,A):- d(U*V^(-1),X,A).         /* d u/v   dx = d (u*v)^-1 dx      */
d(U^C,X,C*U^(C-1)*W):- atomic(C), C \= X, d(U,X,W).  /* d u^c   dx = c*u^(c-1)*d u dx */
d(log(U),X,A*U^(-1)):- d(U,X,A).      /* d ln(u) dx = u^-1 * d u dx      */
/* Tính tích phân */
i(0,X,0):-!.                          /* Int 0   dx = 0                  */
i(X,X,(X*X)/2):-!.                    /* Int X   dx = (X^2)/2            */
i(C,X,C*X):- atomic(C).               /* Int c   dx = c*x                */
i(-U,X,-A):- i(U,X,A).                /* Int -U  dx = - Int U dx         */
i(U+V,X,A+B):- i(U,X,A), i(V,X,B).    /* Int U+V dx = Int U dx + Int V dx */
i(U-V,X,A-B):- i(U,X,A), i(V,X,B).    /* Int U-V dx = Int U dx - Int V dx */
i(C*U,X,C*A):- atomic(C), C \= X, i(U,X,A),!.   /* Int cU  dx = c (Int U dx) */
i(X^C,X,(X^(C+1))/(C+1)):- atomic(C),!.         /* Int x^c dx = x^(c+1)/(c+1) */
i(U,V,U*V-A):- d(V,U,A),!.            /* Int u   dv = u*v - Int v du     */
/* Một số luật đơn giản */
s(+,X,0,X).                           /* x + 0 = x                       */
s(+,0,X,X).                           /* 0 + x = x                       */
s(+,X,Y,X+Y).                         /* x + y = x + y                   */
s(+,X,Y,Z):- integer(X), integer(Y), Z is X+Y.  /* x + y = z     <- Calculate */
s(*,_,0,0).                           /* anything * 0 = 0                */
s(*,0,_,0).                           /* 0 * anything = 0                */
s(*,1,X,X).                           /* 1 * x = x                       */
s(*,X,1,X).                           /* x * 1 = x                       */
s(*,X,Y,X*Y).                         /* x * y = x * y                   */
s(*,X*Y,W,X*Z):- integer(Y), integer(W), Z is Y*W.
s(*,X,Y,Z):- integer(X), integer(Y), Z is X*Y.  /* x * y = z     <- Calculate */
/* Đơn giản hoá */
simp(E,E):- atomic(E),!.
simp(E,F):- E =.. [Op, La, Ra], simp(La,X), simp(Ra,Y), s(Op,X,Y,F).
```