1. **Giới thiệu ngôn ngữ Prolog**

* Prolog là ngôn ngữ được sử dụng phổ biến nhất trong dòng các ngôn ngữ lập trình logic (Prolog có nghĩa là PROgramming in LOGic). Ngôn ngữ Prolog do giáo sư người Pháp Alain Colmerauer và nhóm nghiên cứu của ông đề xuất lần đầu tiên tại trường Đại học Marseille đầu những năm 1970. Đến năm 1980, Prolog nhanh chóng được áp dụng rộng rãi ở châu Âu, được người Nhật chọn làm ngôn ngữ phát triển dòng máy tính thế hệ 5. Prolog đã được cài đặt trên các máy vi tính Apple II, IBM-PC, Macintosh.
* Prolog còn được gọi là ngôn ngữ lập trình ký hiệu (symbolic programming) tương tự các ngôn ngữ lập trình hàm (functional programming), hay lập trình phi số (nonnumerical programming). Prolog rất thích hợp để giải quyết các bài toán liên quan đến các đối tượng (object) và mối quan hệ (relation) giữa chúng. Prolog được sử dụng phổ biến trong lĩnh vực trí tuệ nhân tạo. Nguyên lý lập trình lôgich dựa trên các mệnh đề Horn (Horn logic). Một mệnh đề Horn biễu diễn một sự kiện hay một sự việc nào đó là đúng hoặc không đúng, xảy ra hoặc không xảy ra (có hoặc không có, v.v...).
* *Ví dụ 1:* Sau đây là một số mệnh đề Horn:
  + ***Nếu*** *một người già mà (****và****) khôn ngoan* ***thì*** *người đó hạnh phúc.*
  + *Jim là người hạnh phúc.*
  + ***Nếu*** *X là cha mẹ của Y và Y là cha mẹ của Z* ***thì*** *X là ông (bà) của Z.*
  + *Tom là ông của Sue.*
  + *Tất cả mọi người đều chết (hoặc* ***nếu*** *ai là người* ***thì*** *ai đó phải chết).*
  + *Socrat là người.*
* Trong các mệnh đề Horn ở trên, các mệnh đề 1, 3, 5 được gọi là các luật (rule), các mệnh đề còn lại được gọi là các sự kiện (fact). Một chương trình lôgich có thể được xem như là một cơ sở dữ liệu gồm các mệnh đề Horn, hoặc dạng luật, hoặc dạng sự kiện, chẳng hạn như tất cả các sự kiện và luật từ 1 đến 6 ở trên. Người sử dụng (NSD) gọi chạy một chương trình logic bằng cách đặt câu hỏi (query/question) truy vấn trên cơ sở dữ liệu này, chẳng hạn câu hỏi:

*Socrat có chết không?*

(tương đương khẳng định Socrat chết đúng hay sai?).

* Một hệ thống logic sẽ thực hiện chương trình theo cách «suy luận»-tìm kiếm dựa trên vốn «hiểu biết» đã có là chương trình - cơ sở dữ liệu, để minh chứng câu hỏi là một khẳng định, là đúng (Yes) hoặc sai (No). Với câu hỏi trên, hệ thống tìm kiếm trong cơ sở dữ liệu khẳng định Socrat chết và «tìm thấy» luật 5 thoả mãn (vế **thì**). Vận dụng luật 5, hệ thống nhận được Socrat là người (vế **nếu**) chính là sự kiện 5. Từ đó, câu trả lời sẽ là:

*Yes*

có nghĩa sự kiện Socrat chết là đúng.

1. **Các yếu tố cơ bản của Turbo Prolog**

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.

1. 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.

1. 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.

1. 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”.

1. **Cấu trúc của một chương trinh 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.

1. *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.
* Ví dụ:

Domains

ten, tac\_gia, nha\_xb, dia\_chi = string

nam, thang, so\_luong = integer

dien\_tich = real

nam\_xb = nxb(thang, nam)

do\_vat = sach(tac\_gia, ten, nha\_xb, nam\_xb); xe(ten, so\_luong); nha(dia\_chi, dien\_tich)

* Trong ví dụ trên, ta đã định nghĩa các kiểu mới, trong đó các kiểu mới ten, tac\_gia, nha\_xb, dia\_chi dựa vào cùng một kiểu đã biết là string; các kiểu mới nam, thang, so\_luong dựa vào cùng một kiểu đã biết là integer; kiểu mới dien\_tich dựa vào kiểu đã biết là real; kiểu mới năm\_xb dựa vào kiểu nxb được xây dựng từ các kiểu đã biết là thang, nam; còn kiểu do\_vat lại dựa vào các kiểu sach, xe, nha mà các kiểu này lại dựa vào các kiểu đã biết.

1. *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.
* Ví dụ:

Predicates

so\_huu (ten, do\_vat)

so\_nguyen\_to(integer)

* Trong ví dụ trên ta khai báo hai vị từ. Trong đó vị từ so\_huu (ten, do\_vat) để chỉ một người có tên là ten sẽ sở hữu môt do\_vat nào đó. Còn vị từ so\_nguyen\_to(integer) để xét xem một số integer nào đó có phải là số nguyên tố hay không.

1. *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)

* Ví dụ:

Clauses

so\_nguyen\_to(2) :- !.

so\_nguyen\_to(N) :- N>0,

so\_nguyen\_to(M),

M < N,

N MOD M <> 0.

so\_huu(“Nguyen Van A”, sach(“Do Xuan Loi”, “Cau truc DL”, “Khoa hoc Ky thuat”, nxb(8,1985))).

* Chú ý: Nếu trong các tham số của một vị từ có biến thì biến này phải xuất hiện ít nhất 2 lần trong vị từ đó hoặc trong các vị từ dùng để suy diễn ra vị từ đó. Nếu chỉ xuất hiện một lần thì bắt buộc phải dùng biến tự do.
* Ví dụ: Để diễn tả sự kiện: Tổ hợp chập 0 của N (N bất kỳ) bằng 1, ta không thể viết Tohop(N,0,1) vì biến N chỉ xuất hiện đúng một lần trong vị từ này, do đó ta phải viết Tohop(\_,0,1).

1. *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.
* Ví dụ:

Constants

Pi = 3.141592653

1. *Ví dụ về chương trình Prolog*

Xét xem một số N có phải là số nguyên tố hay không.

domains

so\_nguyen = integer

predicates

so\_nguyen\_to(so\_nguyen)

Clauses

so\_nguyen\_to(2) :- !.

so\_nguyen\_to(N) :- N>0,

so\_nguyen\_to(M),

M < N,

N MOD M <> 0.

goal

so\_nguyen\_to(13).

1. **Các nguyên tắc của ngôn ngữ Prolog**

Việc giải quyết vấn đề trong ngôn ngữ Prolog chủ yếu dựa vào hai nguyên tắc sau: đồng nhất và quay lui.

1. *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.

1. *Quay lui*

* Giả sử hệ thống đang chứng minh goal g, trong đó g được mô tả như sau:

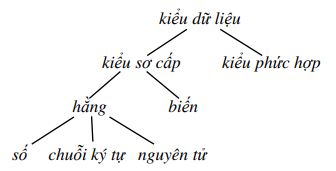
g :- g1, g2, …, gj-1, gj, …, gn.

* Khi các gi kiểm chứng từ trái sang phải, đến gj là sai thì hệ thống sẽ quay lui lại gj-1 để tìm lời giải khác.

1. **Các kiểu dữ liệu Prolog**

Hình dưới đây biểu diễn một sự phân lớp các kiểu dữ liệu trong Prolog gồm kiểu dữ liệu sơ cấp và kiểu dữ liệu có cấu trúc. Sự phân lớp này nhận biết kiểu của một đối tượng nhờ bề ngoài cú pháp.

Cú pháp của Prolog quy định mỗi kiểu đối tượng có một dạng khác nhau. Prolog không cần cung cấp một thông tin nào khác để nhận biết kiểu của một đối tượng. Trong Prolog, NSD không cần khai báo kiểu dữ liệu.



Các kiểu dữ liệu trong Prolog được xây dựng từ các ký tự ASCII:

• Các chữ cái in hoa A, B, ..., Z và chữ cái in thường a, b, ..., z

• Các chữ số 0, 1, ..., 9

• Các ký tự đặc biệt, chẳng hạn + - \* / < > = : . & \_ ~

1. **Các ví dụ**

* *Ví dụ 1:* ta có một cơ sở tri thức KB1

woman(mia).

woman(jody).

woman(yolanda).

playsGuitar(jody).

party.

KB1 đơn giản chỉ là các sự kiện. Ở đây, chúng ta có năm sự kiện cung cấp cho chúng ta một số thông tin: Mia, Jody và Yolanda là những người phụ nữ, Jody chơi guitar và có một party.

Bây giờ chúng ta sẽ thực hiện một vài truy vấn. Giả sử ta muốn hỏi Mia có phải là một người phụ nữ hay không, ta đưa ra truy vấn (chú ý dấu chấm và không cần gõ ?- vì đó là dấu nhắc của Prolog):

?- woman(mia).

Khi đó, Prolog sẽ trả lời là true vì rõ ràng đây là một trong số các sự kiện có trong KB1.

Tương tự, ta có thể đưa ra truy vấn:

?- playsGuitar(jody).

và Prolog trả lời true. Tuy nhiên nếu ta đưa ra truy vấn:

?- playsGuitar(mia).

thì Prolog sẽ trả lời false.

Điều này là do đây là sự kiện không xuất hiện trong KB1, hơn nữa trong KB1 không có thêm thông tin hữu ích nào (đó là các luật) để Prolog có thể suy luận tiếp.

Do đó Prolog kết luận sự kiện playsGuitar(mia) không thể suy ra được từ KB1.  
Sau đây là hai ví dụ quan trọng khác. Giả sử ta đặt ra truy vấn:

?- playsGuitar(vincent).

Khi đó, Prolog sẽ trả lời false.

Điều này là do truy vấn này đề cập đến một người (là Vincent) không có thông tin nào trong KB1. Nên Prolog kết luận playsGuitar(vincent) không thể suy diễn từ KB1.

Hãy thử đặt ra truy vấn sau:

?- tatooed(jody).

Khi đó, Prolog sẽ báo lỗi vì thuộc tính tatooed chưa được định nghĩa trong KB1.

Chúng ta cũng có thể đưa ra các truy vấn khác, chẳng hạn như:

?- party.

và Prolog sẽ trả lời true.

Nếu ta đặt ra truy vấn:

?- rockConcert.

thì Prolog sẽ báo lỗi vì mệnh đề này chưa được định nghĩa trong KB1.

* *Ví dụ 2:*

Bây giờ chúng ta sẽ đi đến một ví dụ về cơ sở tri thức có chứa các luật. Ta gọi cơ sở tri thức này là KB2:

happy(yolanda).

listens2Music(mia).

listens2Music(yolanda) :- happy(yolanda).

playsGuitar(mia) :- listens2Music(mia).

playsGuitar(yolanda) :- listens2Music(yolanda).

Trong KB2 có hai sự kiện và ba luật. Kí hiệu :- đọc là "nếu", ví dụ luật listen2Music(yolanda) :- happy(yolanda) có nghĩa là "Yolanda nghe nhạc nếu như Yolanda vui". Phần bên trái của :- được gọi là head (phần đầu) và phần bên phải được gọi body (phần thân) của luật. Như vậy một luật có dạng head :- body, có nghĩa là nếu body đúng thì head cũng đúng.

Khi một cơ sở tri thức có chứa một luật head :- body, và trong cơ sở tri thức có chứa body hoặc body có thể được suy diễn từ cơ sở tri thức, khi đó Prolog suy ra được head.

Bây giờ hãy thử đặt ra một vài truy vấn. Giả sử ta muốn hỏi Mia có chơi guitar hay không:

?- playsGuitar(mia).

Khi đó Prolog sẽ trả lời true. Ta thấy mặc dù sự kiện playsGuitar(mia) không xuất hiện trong KB2 nhưng Prolog tìm thấy một luật có chứa sự kiện này là playsGuitar(mia) :- listen2Music(mia) và listen2Music(mia) là một sự kiện xuất hiện trong KB2.

Một ví dụ khác, xem xét truy vấn sau:

?- playsGuitar(yolanda).

Prolog cũng trả lời true. Bằng cách sử dụng sự kiện happy(yolanda) và luật listen2Music(yolanda) :- happy(yolanda), Prolog suy ra được sự kiện listen2Music(yolanda). Sử dụng sự kiện này và luật playsGuitar(yolanda) :- listen2Music(yolanda), Prolog suy luận ra sự kiện playsGuitar(yolanda).  
Như vậy, một sự kiện được sinh ra bằng một phép suy diễn từ một luật nào đó trong cơ sở tri thức có thể được dùng như là sự kiện đầu vào cho các luật khác.

Các sự kiện và các luật nằm trong cơ sở tri thức còn được gọi chung là các **mệnh đề** (clause). Do đó ta nói KB2 có chứa năm mệnh đề, trong đó có hai sự kiện và ba luật, hay KB1 cũng có năm mệnh đề nhưng trong đó có bốn sự kiện.

Ta cũng nói rằng, KB2 có chứa ba **vị từ** (predicate) hay **thủ tục** (procedure), đó là: listen2Music, happy, playsGuitar.  
Vị từ happy được định nghĩa bằng cách dùng mệnh đề đơn (sự kiện).  
Vị từ listen2Music và playsGuitar được định nghĩa bằng cách dùng hai mệnh đề: listen2Music được định nghĩa bằng cách dùng một sự kiện và một luật, còn playsGuitar được định nghĩa bằng hai luật.

* *Ví dụ 3:*

Ví dụ này sẽ giới thiệu sơ lược qua dùng các biến trong cơ sở tri thức. Cơ sở tri thức KB3 của chúng ta như sau:

loves(vincent,mia).

loves(marsellus,mia).

loves(pumpkin,honey\_bunny).

loves(honey\_bunny,pumpkin).

jealous(X,Y):- loves(X,Z), loves(Y,Z).

Chú ý đến cách viết luật của chúng ta trong ví dụ này, đó là có chứa ba biến X,Y và Z. Luật này nói rằng "X ghen với Y nếu tồn tại Z nào đó sao cho X yêu Z và Y cũng yêu Z". Không giống như các cơ sở tri thức ở các ví dụ trước, luật trong KB5 có tính tổng quát. Bây giờ hãy thử đặt truy vấn sau:

?- jealous(marsellus,W).

Truy vấn trên có nghĩa là "Marsellus có thể ghen với ai ?". Ta thấy Vincent là một trong số đó vì cả hai cùng yêu Mia, nên Prolog sẽ trả lời:

W = vincent

Nếu thử nhấn ; thì chúng ta sẽ thấy thêm một kết quả khá vô lý:

W = vincent ;

W = marsellus.

Tuy nhiên nó vẫn đúng luật như đã nêu trong KB3.