**NGÔN NGỮ PROLOG**

1. **Khái niệm cơ bản Prolog**

* Một chương trình Prolog là một cơ sở dữ liệu gồm các mệnh đề ***(clause)***
* Mỗi mệnh đề được xây dựng từ các vị từ
* Vị từ ***(predicat)*** là một phát biểu nào đó về các đối tượng có giá trị chân đúng ***(true)*** hoặc sai ***(fail)***, vị từ có thể có các đối là các nguyên logic
* Mỗi nguyên tử biểu diễn một quan hệ giữa các hạng ***(term)***, hạng và quan hệ giữa các hạng tạo thành mệnh đề.
* Hạng được xem là  **“dữ liệu”**. Hạng có thể là:
* **Hạng sơ cấp** *(elementary term)* gồm: hằng ***(constant)***, biến ***(variable)*** và các hạng phức hợp ***(compound term)***.
* **Hạng phức hợp** biểu diễn các đối tượng phức tạp. Hạng phức hợp là một hàm tử ***(functor)*** có chứa các đối số ***(argument)***

1. **Cú pháp prolog:**
   1. **Hàm (function)**



* **Tên hàm** là một chuỗi chữ cái và/hoặc chữ số được bắt đầu bởi một chữ cái thường. Các đối số có thể là biến, hạng sơ cấp hoặc hạng phức hợp.

Vd:

* f(5, a, b).
* student(robert, 1975, info, 2,).
* address(6, ‘mal juin’, ‘Caen’).
* [a, b, c]
* **Mệnh đề** có thể là một sự kiện, một luật (hay quy tắc), hay một câu hỏi. Mỗi mệnh đề kết thúc bằng dấu chấm (.)

Vd:

* Sự kiện : < … >.

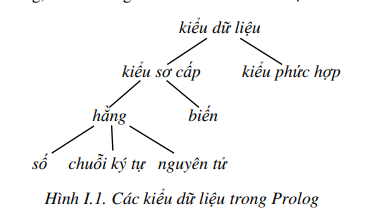
(tương ứng với luật < … > :- true. )

* Luật : < … > :- < … >.
* Câu hỏi ?- < … >.

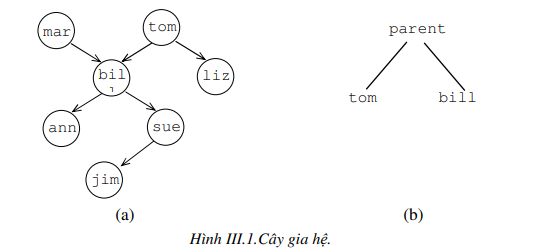
(ở chế độ tương tác có dấu nhắc lệnh)

* 1. **kiểu dữ liệu Prolog:**
* Prolog quy định mỗi kiểu đối tượng có một dạng khác nhau.

Prolog không cần cung cấp thông tin để nhận biết kiểu của một đối tượng.

Trong Prolog, không cần khai báo kiểu dữ liệu.

* 1. **Xây dựng sự kiện:**



Trong cây gia hệ (a):

* Các nút chỉ người, còn các mũi tên chỉ quan hệ cha mẹ

của (parent of).

* Sự kiện Tom là cha mẹ của Bill được viết thành một vị từ Prolog

như sau (**chú ý** ***mệnh đề được kết thúc bởi một dấu chấm***)

*parent(tom, bill).*

(**chú ý:** ***không có dấu cách trước dấu mở ngoặc***)

Trong hình b) vị từ parent có hai đối số là *tom* và *bill*. Có thể biểu diễn vị từ này bởi một cây: nút gốc là tên của vị từ, còn các nút lá là các đối số.

Từ cây gia hệ trên đây, có thể tiếp tục viết các vị từ khác để nhận được một chương trình Prolog gồm 6 vị từ như sau :

*parent(mary, bill).*

*parent(tom, bill).*

*parent(tom, liz).*

*parent(bill, ann).*

*parent(bill, sue).*

*parent(sue, jim).*

Sau khi hệ thống Prolog nhận được chương trình này, thực chất là một cơ sở dữ liệu, người ta có thể đặt ra các câu hỏi liên quan đến quan hệ parent.

***Câu hỏi Bill có phải là cha mẹ của Sue***

*?- parent(bill, sue).*

Sau khi tìm thấy sự kiện này trong chương trình

***Prolog trả lời*** : Yes

Ta có thể tiếp tục đặt ra các câu hỏi thú vị khác. Chẳng hạn, ***ai là cha (hay mẹ) của Liz ?***

*?- parent(X, liz).*

Lần này, Prolog không trả lời Yes hoặc No, mà đưa ra một giá trị của X làm thoả mãn câu hỏi trên đây :

*X = tom*

Để biết được ***ai là con của Bill***, ta chỉ cần viết :

*?- parent(bill, X).*

Với câu hỏi này, ***Prolog sẽ có hai câu trả lời***, đầu tiên là :

*X = ann ->;*

Để biết được câu trả lời tiếp theo, trong hầu hết các cài đặt của Prolog, NSD phải ***gõ vào một dấu chấm phẩy (;) sau ->*** (Arity Prolog) :

*X = sue*

Nếu đã ***hết phương án trả lời*** mà vẫn ***tiếp tục yêu cầu (;)***, ***Prolog trả lời***

*No.*

***Ai là cha mẹ của ai?*** Nói cách khác, cần tìm X và Y sao cho X là cha mẹ của Y. Ta viết như sau :

*?- parent(X, Y).*

Prolog sẽ lần lượt tìm kiếm những cặp cha mẹ − con thoả mãn và lần lượt hiển thị kết quả

Đến khi không còn kết quả lời giải nào nữa (***kết thúc bởi Yes***)

*X = mary*

*Y = bill ->;*

*X = tom*

*Y = bill;*

*X = tom*

*Y = liz ->;*

*X = bill*

*Y = ann ->;*

*X = bill*

*Y = sue ->;*

*X = sue*

*Y = jim*

*Yes*

***Ai là ông (bà) của Jim ?*** Cần phải phân tách câu hỏi này thành hai phần sơ cấp hơn :

***1.* Ai là cha (mẹ) của Jim ? Giả sử có tên là Y.**

**2. Ai là cha (mẹ) của Y ? Giả sử có tên là X.**

Lúc này, có thể viết trong Prolog như sau :

*?- parent(Y, jim), parent(X, Y).*

***Prolog trả lời :***

*Y = sue*

*X = bill*

*Yes*

***Ai là cháu của Tom ?***

*?- parent(tom, X), parent(X, Y).*

***Prolog trả lời :***

*X = bill*

*Y = ann->;*

*X = bill*

*Y = sue ->;*

*No*

***Ann và Sue có cùng người ông không ?***

*?- parent(X, ann), parent(X, sue).*

**1. Tìm X là cha mẹ của Ann.**

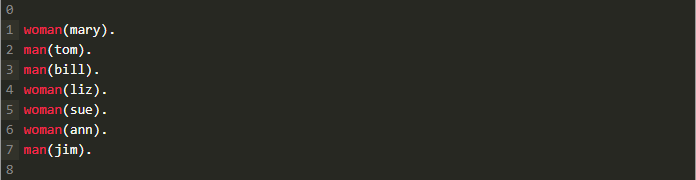
**2. X tìm thấy có cùng là cha mẹ của Sue không ?**

***Prolog trả lời :***

*X = bill*

* 1. **Xây dựng luật**
  2. **Định nghĩa luật**

Ví dụ về luật



* Quan hệ đơn ***(unary)***: là quan hệ chỉ liên quan đến một đối tượng duy nhất. Dùng để thiết lập các thuộc tính của một đối tượng.

Trong ví dụ trên: women, men chỉ liên quan đến một đối tượng:

woman(mary) 🡪 Mary là nữ

* Quan hệ nhị phân : liên quan đến một cặp đối tượng.

Ta có thể sử dụng quan hệ nhị phân để định nghĩa giới tính:

*sex(mary, female).*

*sex(tom, male).*

*sex(bill, male).*

Đưa vào một quan hệ mới child, đối ngược với parent như sau :

*child(liz, tom).*

Từ đó, định nghĩa luật mới như sau :

*child(Y, X) :- parent(X, Y).*

Luật trên được hiểu là :

***Với mọi X và Y,***

***Y là con của X nếu X là cha (hay mẹ) của Y.***

* 1. **Biểu thức số học**

**Biểu thức số học** (***arithmetic expressions***) được xây dựng nhờ vị từ **is**. Vị từ này là một phép toán tiền tố (infix operator) có dạng :

**Number** is **Expr**

Vì phép toán is sẽ khởi động việc tính toán, cho nên khi thực hiện đích này, tất cả các biến cần phải được ràng buộc với các giá trị số.

Prolog so khớp thành công nếu Number khớp được với Expr.

Nếu Expr là kiểu thực (float) thì được xem như một só nguyên.

*?- X is 3\*4.*

*X = 12 Yes*

*?- is(X, 40+50).*

*X = 90*

*Yes*

*?- 1.0 is sin(pi/2).*

*No*

% sai do sin(pi/2) được làm tròn thành 1

*?- 1.0 is float(sin(pi/2)).*

*Yes*

Cộng hai số 1 và 2 mà lại viết như sau :

*?- X = 1 + 2*

Prolog sẽ trả lời theo kiểu của Prolog :

*X = 1 + 2* **(không phải là X = 3)**

Để Prolog tiến hành tính toán trên các phép toán số học, **sử dụng phép toán** **is** như sau :

*?- X is 1 + 2.*

*X = 3*

Ví dụ phép toán:

*?- X = 1 + 1 + 1, Y is X.*

*X = 1 + 1 + 1, Y = 3.*

*?- X is 1 + 1 + a.*

**ERROR**: Arithmetic: `a/0' is not a function (sai do a không phải là hàm số)

*?- Z = 2, X is 1 + 1 + Z.*

*Z = 2*

*X = 4*

Các phép so sánh giá trị số học trong Prolog được thực hiện theo nghĩa Toán học thông thường.

Chẳng hạn, ta cần **so sánh tích của 277 với 37 có lớn hơn 10000** không?

*?- 277 \* 37 > 10000.*

*Yes*

Prolog có sẵn các hàm số học như : sin, cos, tan, atan, sqrt, pi, e, exp, log, ...

*?- X is exp(10).*

*X = 22026.5*

*Yes*

*?- X is sqrt(9).*

*X = 3*

*Yes*

*?- X is abs(1.99).*

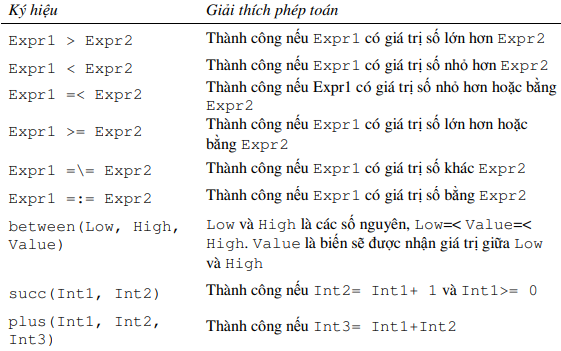
*X = 1.99*

*Yes*

*?- X is pi.*

*X = 3.14159*

*Yes*



?- X = Y.

X = \_G997

Y = \_G997

Yes

?- 1 + 2 =:= 2 + 1.

Yes.

?- 1 + 2 = 2 + 1.

No.

?- 1 + 2 = 1 + 2.

Yes.

?- 1 + X = 1 + 2. X = 2

?- 1 + A = B + 2.

A = 2 B = 1

?- 1 + 2 =:= 2 + 1.

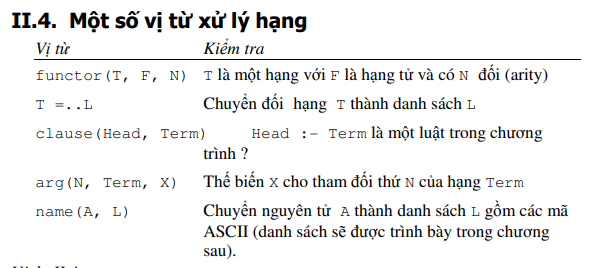
Yes.

?- 1 + X =:= 1 + 2.

ERROR: Arguments are not sufficiently instantiated (sai do a không phải là số)

**Các phép so sánh hạng**





*?- functor(t(a, b, c), F, N).*

*F = t*

*N = 3*

*Yes*

*?- functor(father(jean, isa), F, N).*

*F = father, N = 2.*

*Yes*

*?- functor(T, father, 2).*

*T = father(\_G346, \_G347).*

*% \_G346 và \_G347 là hai biến của Prolog*

*?- t(a, b, c) =..L. L = [t, a, b, c]*

*Yes*

*?- T =..[t, a, b, c, d, e].*

*T = t(a, b, c, d, e)*

*Yes*

*?- arg(1, father(jean, isa), X).*

*X = jean*

*?- name(toto, L).*

*L = [116, 111, 116, 111].*

*Yes*

*?- name(A, [116, 111, 116, 111]).*

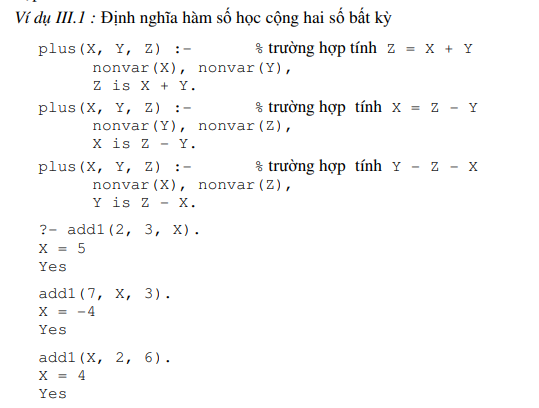
*A = toto.*

*Yes*

**Định nghĩa hàm**

Prolog không có kiểu hàm, hàm phải được định nghĩa như một quan hệ trên các đối tượng.

Các tham đối của hàm và giá trị trả về của hàm phải là các đối tượng của quan hệ đó. Điều này có nghĩa là không thể xây dựng được các hàm tổ hợp từ các hàm khác.



Ví dụ: ***Tìm ước số chung lớn nhất*** (GCD: Greatest Common Divisor)

Cho trước hai số nguyên X và Y, ta cần tính ước số D và USCLN dựa trên ba quy tắc như sau :

**1. Nếu X = Y, thì D bằng X.**

**2. Nếu X < Y, thì D bằng USCLN của X và của Y - X.**

**3. Nếu X > Y, thì thực hiện tương tự bước 2, bằng cách hoán vị vai trò X và Y.**

Với X =20 và Y =25, thì ta nhận được D =5 sau một dãy các phép trừ.

Chương trình Prolog được xây dựng như sau :

*gcd( X, X, X ).*

*gcd( X, Y, D ) :-*

*X < Y,*

*Y1 is Y – X,*

*gcd( X, Y1, D ).*

*gcd( X, Y, D ) :-*

*X > Y,*

*gcd( Y, X, D ).*

Đích cuối cùng trong mệnh đề thứ ba trên đây có thể được thay thế bởi :

Kết quả chạy Prolog như sau :

*?- gcd( 20, 55, D ).*

*D = 5*

