ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP THỰC HÀNH 2

CƠ SỞ TRÍ TUỆ NHÂN TẠO

|ĐỀ TÀI|

TÌM HIỂU NGÔN NGỮ PROLOG

HỌC KỲ 1 / 2020 - 2021

Thành Phố Hồ Chí Minh – Năm 2020

MỤC LỤC

I. GIỚI THIỆU NGÔN NGỮ PROLOG

Prolog là ngôn ngữ được sử dụng phổ biến nhất trong dòng các ngôn ngữ lập trình lôgich (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 logic dựa trên các mệnh đề Horn (Horn logic). Một mệnh đề Horn biểu 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 sau đây là một số mệnh đề Horn :

1. Nếu một người già mà (và) khôn ngoan thì người đó hạnh phúc.
2. Jim là người hạnh phúc.
3. Nếu X là cha mẹ của Y và Y là cha mẹ của Z thì X là ông của Z.
4. Tom là ông của Sue.
5. Tất cả mọi người đều chết (hoặc Nếu ai là người thì ai đó phải chết).

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

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 lôgich bằng cách đặt câu hỏi (query/ question) truy vấn trên cơ sở dữ liệu này, chẳng hạn câu hỏi :

Một hệ thống lôgich 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.

a. CÚ PHÁP PROLOG

1. CÁC THUẬT NGỮ

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 (predicat). Một vị từ là một phát biểu nào đó về các đối tượng có giá trị chân đúng (true) hoặc sai (fail). Một vị từ có thể có các đối là các nguyên lôgich (logic atom).

Mỗi nguyên tử (nói gọn) biểu diễn một quan hệ giữa các hạng (term). Như vậy, hạng và quan hệ giữa các hạng tạo thành mệnh đề. Hạng được xem là những đối tượng "dữ liệu" trong một trình Prolog.

Hạng có thể là hạng sơ cấp (elementary term) gồm hằng (constant), biến (variable) và các hạng phức hợp (compound term).

Các hạng phức hợp biểu diễn các đối tượng phức tạp của bài toán cần giải quyết thuộc lĩnh vực đang xét. Hạng phức hợp là một hàm tử (functor) có chứa các đối (argument), có dạng:

```
Tên_hàm_tử(Đối_1, ..., Đối_n)
```

Tên hàm tử là một chuỗi chữ cái và/hoặc chũ số được bắt đầu bởi một chữ cái thường. Các đối có thể là biến, hạng sơ cấp, hoặc hạng phức hợp. Trong Prolog, hàm tử đặc biệt "." (dấu chấm) biểu diễn cấu trúc danh sách (list). Kiểu dữ liệu hàm tử tương tự kiểu bản ghi (record) và danh sách (list) tương tự kiểu mảng (array) trong các ngôn ngữ lập trình mệnh lệnh (C, Pascal...).

Ví dụ:

```
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. Prolog quy ước viết sau mỗi mệnh đề một dấu chấm để kết thúc như sau :

- Sự kiện : < ... >. (tương ứng với luật < ... > :- true.)
- Luật : < ... > :- < ... >.
- Câu hỏi ?- < ... >. (ở chế độ tương tác có dấu nhắc lệnh)

2. DỮ KIỆN

Dữ kiện là những mệnh đề Horn mà phần Body là rỗng. Kiểu mệnh đề này thường được sử dụng để mô tả các dự kiện của bài toán, ví dụ như việc khai báo "john" thích "mèo":

```
Likes(john,cat).
```

3. LUẬT

Phần còn lại của các mệnh đề trong một chương trình Prolog được gọi là luật. Nó thường thể hiện những phát biểu logic trong bài toán, ví dụ như:

```
Steal(john,X) :- Thief(john).
```

Toán tử ":-" được dịch thô là "nếu", trong logic thì nó đại diện cho toán tử suy ra "<-". Phát biểu trên được phát biểu dưới dạng văn xuôi là " nếu John là trộm thì John trộm món đồ X ". Tất nhiên, bạn có thể chưa thoả mãn với phát biểu này và bổ sung vào nó để có một phát biểu chặt chẽ hơn:

```
Steal(john,X) :- Thief(john) , Likes(X).
```

Dấu phẩy "," trong mệnh đề trên được dịch là toán tử "và"; biến trong Prolog được quy ước bắt đầu là một chữ cái hoa.

4. NGỮ NGHĨA

Một chương trình logic có ngữ nghĩa của riêng nó. Ngữ nghĩa quyết định những kết luận "đúng" nào có thể rút ra được từ một chương trình Prolog. Ví dụ một chương trình Prolog gồm một dữ kiện:

```
Likes(john,cat).
```

Khi đó, ta có thể rút ra duy nhất một dữ kiện đúng là "John thích mèo". Trong một ứng dụng Prolog, bạn có thể hỏi một trong hai câu hỏi sau để có được trả lời đúng:

```
?- Likes(john,cat).          Hoặc          ?- Likes(john,X).
yes.                                       X = dog;
                                           no.
```

Trong ví dụ trên, "no" có nghĩa là không còn câu trả lời nào nữa. Mọi câu hỏi khác đều cho trả lời là sai. Điều này có nghĩa là trong một chương trình Prolog, người ta sử dụng giả thiết thế giới đóng, mọi thứ bạn khai báo là đúng, nếu không thì nó là sai. Vì vậy trong ví dụ trên, khi bạn hỏi "Peter có thích mèo hay không", bạn sẽ nhận được câu trả lời "no".

Với một chương trình Prolog xác định, ngữ nghĩa của nó được định nghĩa là một mô hình tối thiểu của nó.

Với một chương trình Prolog bình thường, có nhiều loại ngữ nghĩa được sử dụng như ngữ nghĩa đầy đủ, ngữ nghĩa tối thiểu, ngữ nghĩa hoàn chỉnh,...

Đa số các chương trình biên dịch Prolog phổ thông (SWI-Prolog, GNU-Prolog) sử dụng ngữ nghĩa đầy đủ mà đi kèm là thủ tục suy diễn SLDNF.

b. QUAN HỆ GIỮA PROLOG VÀ LÔGICH TOÁN HỌC

Prolog có quan hệ chặt chẽ với lôgich toán học. Dựa vào lôgich toán học, người ta có thể diễn tả cú pháp và nghĩa của Prolog một cách ngắn gọn và súc tích. Tuy nhiên không vì vậy mà những người học lập trình Prolog cần phải biết một số khái niệm về lôgich toán học. Thật may mắn là những khái niệm về lôgich toán học không thực sự cần thiết để có thể hiểu và sử dụng Prolog như là một công cụ lập trình. Sau đây là một số quan hệ giữa Prolog và lôgich toán học.

Prolog có cú pháp là những công thức lôgich vị từ bậc một (first order predicate logic), được viết dưới dạng các mệnh đề (các lượng tử ∀ và ∃ không xuất hiện một cách tường minh), nhưng hạn chế chỉ đơn thuần ở dạng mệnh đề Horn, là những mệnh đề chỉ có ít nhất một trực kiện dương (positive literals). Năm 1981, Clocksin và Mellish đã đưa ra một chương trình Prolog chuyển các công thức tính vị từ bậc một thành dạng các mệnh đề.

Cách Prolog diễn giải chương trình là theo kiểu Toán học : Prolog xem các sự kiện và các luật như là các tiên đề, xem câu hỏi của NSD như là một định lý cần phỏng đoán. Prolog sẽ tìm cách chứng minh định lý này, nghĩa là chỉ ra rằng định lý có thể được suy luận một cách lôgich từ các tiên đề.

Về mặt thủ tục, Prolog sử dụng phương pháp suy diễn quay lui (back chaining) để hợp giải (resolution) bài toán, được gọi là chiến lược hợp giải SLD (Selected, Linear, Definite : Linear resolution with a Selection function for Definite sentences) do J. Herbrand và A. Robinson đề xuất năm 1995.

Có thể tóm tắt như sau : để chứng minh P(a), người ta tìm sự kiện P(t) hoặc một luật : P(t) :- L1, L2, …, Ln. Sao cho a có thể hợp nhất (unifiable) được với t nhờ so khớp. Nếu tìm được P(t) là sự kiện như vậy, việc chứng minh kết thúc. Còn nếu tìm được P(t) là luật, cần lần lượt chứng minh vế bên phải L1, L2, ..., Ln của nó.

Trong Prolog, câu hỏi luôn luôn là một dãy từ một đến nhiều đích. Prolog trả lời một câu hỏi bằng cách tìm kiếm để xoá (erase) tất cả các đích. Xoá một đích nghĩa là chứng minh rằng đích này được thoả mãn, với giả thiết rằng các quan hệ của chương trình là đúng. Nói cách khác, xoá một đích có nghĩa là đích này được suy ra một cách lôgich bởi các sự kiện và luật chứa trong chương trình.

Nếu có các biến trong câu hỏi, Prolog tìm các đối tượng để thay thế vào các biến, sao cho đích được thoả mãn. Sự ràng buộc giá trị của các biến tương ứng với việc hiển thị các đối tượng này. Nếu Prolog không thể tìm được ràng buộc cho các biến sao cho đích được suy ra từ chương trình thì nó sẽ trả lời No.

II. CÁC MỨC NGHĨA CỦA CHƯƠNG TRÌNH PROLOG

Cho đến lúc này, qua các ví dụ minh hoạ, ta mới chỉ hiểu được tính đúng đắn về kết quả của một chương trình Prolog, mà chưa hiểu được làm cách nào để hệ thống tìm được lời giải. Một chương trình Prolog có thể được hiểu theo nghĩa khai báo (declarative signification) hoặc theo nghĩa thủ tục (procedural signification). Vấn đề là cần phân biệt hai mức nghĩa của một chương trình Prolog, là nghĩa khai báo và nghĩa thủ tục. Người ta còn phân biệt mức nghĩa thứ ba của một chương trình Prolog là nghĩa lôgich (logical semantic).

Trước khi định nghĩa một cách hình thức hai mức ngữ nghĩa khai báo và thủ tục, ta cần phân biệt sự khác nhau giữa chúng. Cho mệnh đề :

P :- Q, R. với P, Q, và R là các hạng nào đó.

Theo nghĩa khai báo, ta đọc chúng theo hai cách như sau :

- P là đúng nếu cả Q và R đều đúng.
- Q và R dẫn ra P.

Theo nghĩa thủ tục, ta cũng đọc chúng theo hai cách như sau :

- Để giải bài toán P, đầu tiên, giải bài toán con Q, sau đó giải bài toán con R.
- Để xoá P, đầu tiên, xoá Q, sau đó xoá R.

Sự khác nhau giữa nghĩa khai báo và nghĩa thủ tục là ở chỗ, nghĩa thủ tục không định nghĩa các quan hệ lôgich giữa phần đầu của mệnh đề và các đích của thân, mà chỉ định nghĩa thứ tự xử lý các đích.

Trong phần này, ta chỉ tìm hiểu về ngữ nghĩa logic và ngữ nghĩa thủ tục

1. NGỮ NGHĨA LOGIC CỦA MỆNH ĐỀ

Nghĩa lôgich thể hiện mối liên hệ giữa đặc tả lôgich (logical specification) của bài toán cần giải với bản thân chương trình.

a. Các mệnh đề không chứa biến

Mệnh đề	*Nghĩa lôgich*	*Ký hiệu Toán học*
`P(a).`	`P(X)` đúng **nẽu** `X = a`	$P(X) \Leftrightarrow X = a$
`P(a).` `P(b).`	`P(X)` đúng **nẽu** `X = a` hoặc `X = b`	$P(X) \Leftrightarrow (X = a) \vee (X = b)$
`P(a) :-` `Q(c).`	`P(X)` đúng **nẽu** `X = a` và `Q(c)` đúng	$P(X) \Leftrightarrow X = a \wedge Q(c)$

`P(a) :-`
`Q(c).`
`P(b).`
	`P(X)` đúng **nẽu hoặc** (`X = a` và `Q(c)` đúng) **hoặc** `X = b`	$P(X) \Leftrightarrow (X = a \wedge Q(c)) \vee (X = b)$

Quy ước : **nẽu** = nếu và chỉ nếu.

b. Các mệnh đề chứa biến

Mệnh đề	*Nghĩa lôgich*	*Ký hiệu Toán học*
`P(X).`	Với mọi giá trị của `X`, `P(X)` đúng	$\forall X\ P(X)$
`P(X) :- Q(X).`	Với mọi giá trị của `X`, `P(X)` đúng **nếu** `Q(X)` đúng	$P(X) \Leftarrow Q(X)$
`P(X) :- Q(X, Y).`	Với mọi giá trị của `X`, `P(X)` đúng **nếu** tồn tại `Y` là một biến cục bộ sao cho `Q(X, Y)` đúng	$P(X) \Leftarrow \exists Y\ Q(X, Y)$
`P(X) :- Q(X, _).`	Với mọi giá trị của `X`, `P(X)` đúng **nếu** tồn tại một giá trị nào đó của `Y` sao cho `Q(X, Y)` đúng (không quan tâm đến giá trị của `Y` như thế nào)	$P(X) \Leftarrow \exists Y\ Q(X, Y)$
`P(X) :- Q(X, Y), R(Y).`	Với mọi giá trị của `X`, `P(X)` đúng **nếu** tồn tại `Y` sao cho `Q(X, Y)` và `R(Y)` đúng	$P(X) \Leftarrow \exists Y\ Q(X, Y) \wedge R(Y)$
`P(X) :- Q(X, Y), R(Y).` `p(a).`	Với mọi giá trị của `X`, `P(X)` đúng **nếu hoặc** tồn tại `Y` sao cho `Q(X, Y)` và `R(Y)` đúng, **hoặc** `X = a`	$P(X) \Leftarrow (\exists Y\ Q(X, Y) \wedge R(Y)) \vee (X = a)$

c. Nghĩa logic của các đích

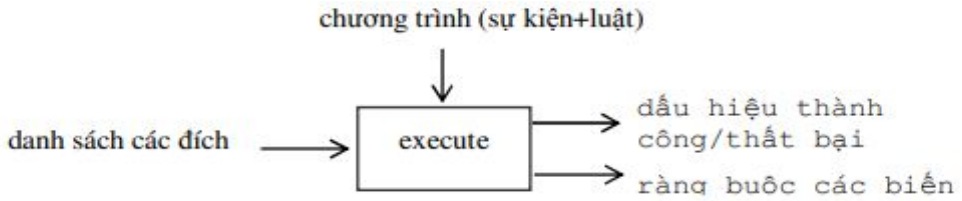
Đích	*Nghĩa lôgich*
`p(a).`	Có phải `p(a)` đúng (thoả mãn) ?
`p(a), Q(b).`	Có phải cả `p(a)` và `Q(b)` đều đúng ?
`P(X).`	Cho biết giá trị của `X` để `P(X)` là đúng ?
`P(X), Q(X, Y).`	Cho biết các giá trị của `X` và của `Y` để `P(X)` và `Q(X, Y)` đều là đúng ?

2. NGHĨA THỦ TỤC CỦA PROLOG

Nghĩa thủ tục, hay ngữ nghĩa thao tác (operational semantic), lại xác định làm cách nào để nhận được kết quả, nghĩa là làm cách nào để các quan hệ được xử lý thực sự bởi hệ thống Prolog.

Nghĩa thủ tục tương ứng với cách Prolog trả lời các câu hỏi như thế nào (how) hay lập luận trên các tri thức. Trả lời một câu hỏi có nghĩa là tìm cách xóa một danh sách. Điều này chỉ có thể thực hiện được nếu các biến xuất hiện trong các đích này được ràng buộc sao cho chúng được suy ra một cách lôgich từ chương trình (hay từ các tri thức đã ghi nhận).

Prolog có nhiệm vụ thực hiện lần lượt từng đích trong một danh sách các đích từ một chương trình đã cho. «Thực hiện một đích» có nghĩa là tìm cách thoả mãn hay xoá đích đó khỏi danh sách các đích đó.



Mô hình vào/ra của một thủ tục thực hiện một danh sách các đích

Gọi thủ tục này là execute (thực hiện), cái vào và cái ra của nó như sau :

- Cái vào : một chương trình và một danh sách các đích
- Cái ra : một dấu hiệu thành công/thất bại và một ràng buộc các biến

Nghĩa của hai cái ra như sau :

(1) Dấu hiệu thành công/thất bại là Yes nếu các đích được thoả mãn (thành công), là No nếu ngược lại (thất bại).

(2) Sự ràng buộc các biến chỉ xảy ra nếu chương trình được thực hiện.

Đích cần tìm là : ?- ancestor(tom, sue)

Ta biết rằng parent(bill, sue) là một sự kiện. Để sử dụng sự kiện này và luật 1 (về tổ tiên trực tiếp), ta có thể kết luận rằng ancestor(bill, sue). Đây là một sự kiện kéo theo : sự kiện này không có mặt trong chương trình, nhưng có thể được suy ra từ các luật và sự kiện khác. Ta có thể viết gọn sự suy diễn này như sau :

parent(bill, sue) ⇒ ancestor(bill, sue)

Nghĩa là parent(bill, sue) kéo theo ancestor(bill, sue) bởi luật 1.
Ta lại biết rằng parent(tom, bill) cũng là một sự kiện.
Mặt khác, từ sự kiện được suy diễn ancestor(bill, sue), luật 2 (về tổ tiên gián tiếp) cho phép kết luận rằng **ancestor(tom, sue).** Quá trình suy diễn hai giai đoạn này được viết :

parent(bill, sue) ⇒ ancestor(bill, sue)

parent(tom, bill) và ancestor(bill, sue) ⇒ ancestor(tom, sue)

Ta vừa chỉ ra các giai đoạn để xoá một đích, gọi là một chứng minh. Tuy nhiên, ta chưa chỉ ra làm cách nào Prolog nhận được một chứng minh như vậy.

Prolog nhận được phép chứng minh này theo thứ tự ngược lại những gì đã trình bày. Thay vì xuất phát từ các sự kiện chứa trong chương trình, Prolog bắt đầu bởi các đích và, bằng cách sử dụng các luật, nó thay thế các đích này bởi các đích mới, cho đến khi nhận được các sự kiện sơ cấp.

Để xoá đích : **?- ancestor(tom, sue).**

Prolog tìm kiếm một mệnh đề trong chương trình mà đích này được suy diễn ngay lập tức. Rõ ràng chỉ có hai mệnh đề thoả mãn yêu cầu này là luật 1 và luật 2, liên quan đến quan hệ **ancestor**. Ta nói rằng phần đầu của các luật này tương ứng với đích.

Hai mệnh đề này biểu diễn hai khả năng mà Prolog phải khai thác xử lý. Prolog bắt đầu chọn xử lý mệnh đề thứ nhất xuất hiện trong chương trình :

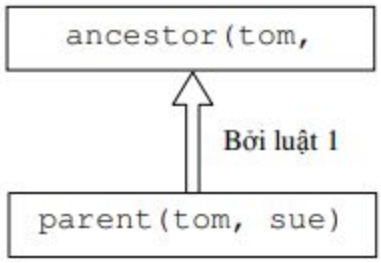
```
ancestor(X, Z) :- parent(X, Z).
```

Do đích là **ancestor(tom, sue)**, các biến phải được ràng buộc như sau :

```
X = tom, Z = sue
```

Lúc này, đích ban đầu trở thành : **paren(tom,sue)**

Hình dưới đây biểu diễn giai đoạn chuyển một đích thành đích mới sử dụng một luật. Thất bại xảy ra khi không có phần đầu nào trong các mệnh đề của chương trình tương ứng với đích **parent(tom, sue)**.



Xử lý bước đầu tiên : Đích phía trên được thoã mãn nếu Prolog có thể xoá đích ở phía dưới.

Lúc này Prolog phải tiến hành quay lui (backtracking) trở lại đích ban đầu, để tiếp tục xử lý mệnh đề khác là luật thứ hai :

```
ancestor(X, Z) :- parent(X, Y), ancestor(Y, Z).
```

Tương tự bước xử lý thứ nhất, các biến X và Z được ràng buộc như sau :

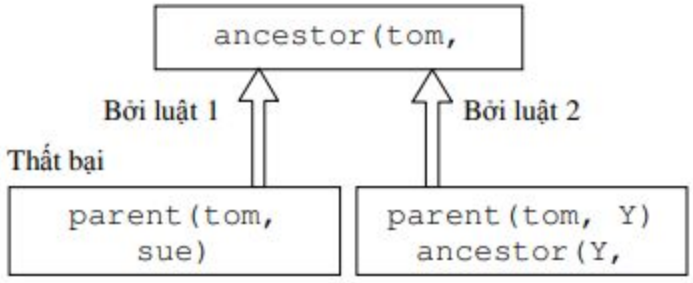
```
X = tom, Z = sue
```

Đích phía trên **ancestor(tom, sue)** được thay thế bởi hai đích là :

```
parent(tom, Y), ancestor(Y, sue).
```

Nhưng lúc này, Y chưa có giá trị. Lúc này cần xoá hai đích. Prolog sẽ tiến hành xoá theo thứ tự xuất hiện của chúng trong chương trình. Đối với đích thứ nhất, việc xoá rất dễ dàng vì đó là một trong các sự kiện của chương trình. Sự tương ứng sự kiện dẫn đến Y được ràng buộc bởi giá trị bill.

Các giai đoạn thực hiện được mô tả bởi cây hợp giải sau đây :



Các giai đoạn thực hiện xử lý xoá đích.

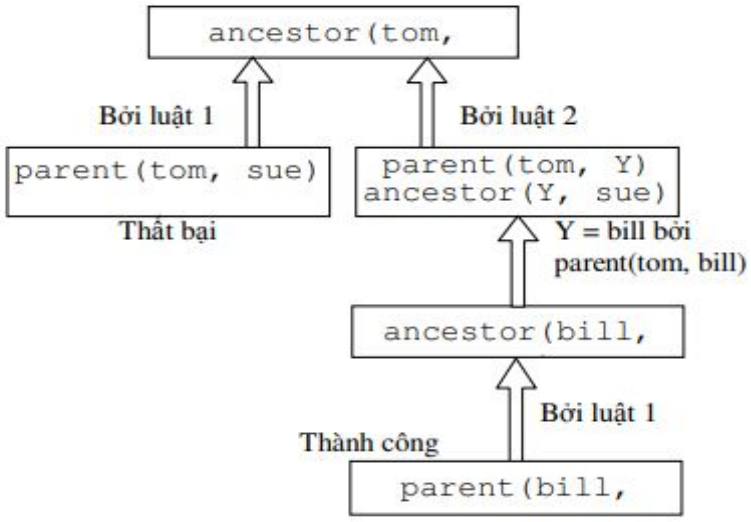
Sau khi đích thứ nhất **parent(tom, bill)** thoả mãn, còn lại đích thứ hai : **ancestor(bill, sue)** cũng phải được thoả mãn.

Một lần nữa, luật 1 được sử dụng. Chú ý rằng việc áp dụng lần thứ hai cùng luật này không liên quan gì đến lần áp dụng thứ nhất. Prolog sử dụng các biến mới mỗi lần luật được gọi đến. Luật 1 bây giờ có thể được đặt tên lại như sau :

```
ancestor(X', Z') :- parent(X', Z').
```

Phần đầu phải tương ứng với đích thứ nhất, **ancestor(bill, sue)**, tức là :

```
X' = bill, Z' = sue
```



Quá trình thực hiện xoá đích ancestor(tom, sue).

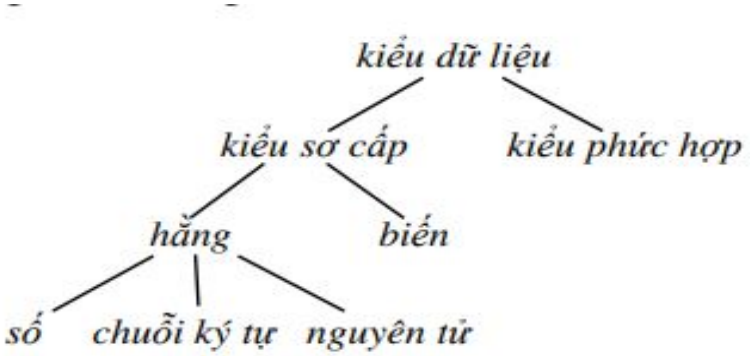
Từ đó đích (trong phần thân) phải thay thế bởi : **parent(bill, sue)**

Đích này được thoả mãn ngay lập tức, vì chính là một sự kiện trong chương trình.

III. CÁC KIỂU DỮ LIỆU PROLOG

Hình bên dưới 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 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 + - * / < > = : . & _ ~.

Chú thích

Trong một chương trình Prolog, chú thích (comment) được đặt giữa hai cặp ký hiệu /* và */ (tương tự ngôn ngữ C). Ví dụ :

```
/************************/
/*** Đây là một chú thích ***/
/************************/
```

Trong trường hợp muốn đặt một chú thích ngắn sau mỗi phần khai báo Prolog cho đến hết dòng, có thể đặt trước một ký hiệu %.

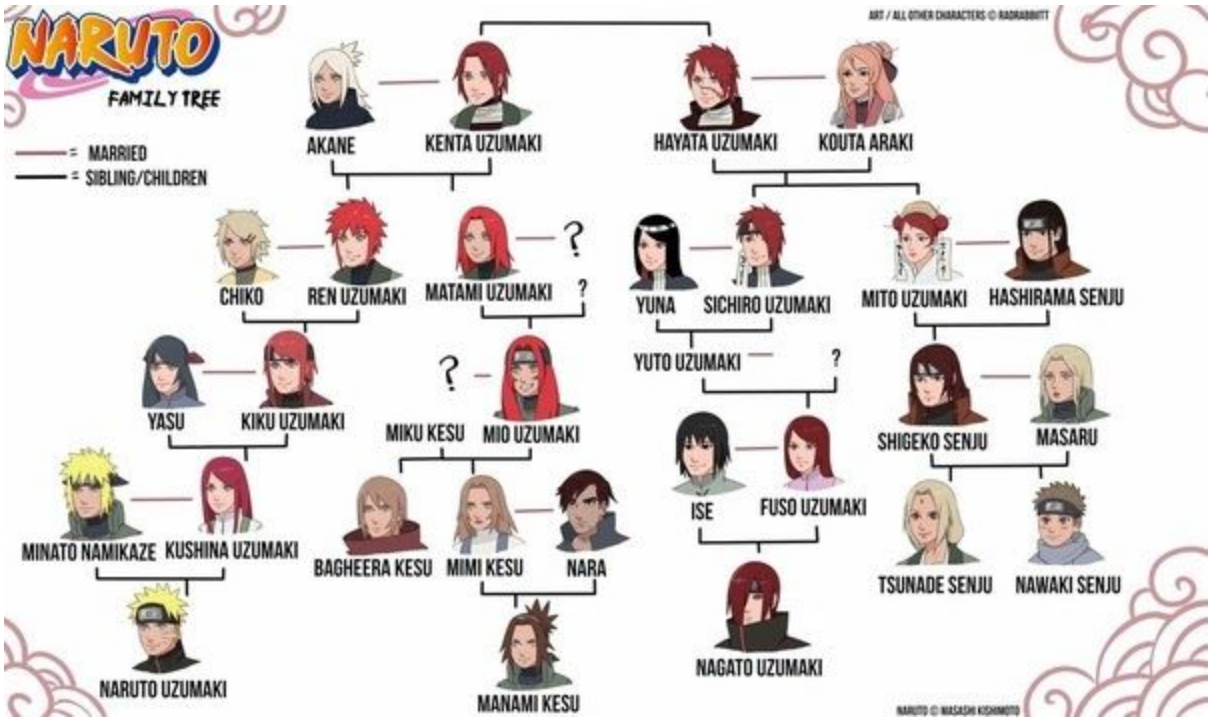
Ví dụ :

Prolog sẽ bỏ qua tất cả các phần chú thích trong thủ tục.

```
%%%%%%%%%% Đây cũng là một chú thích %%%%%%%%%%
```

IX. Xây dựng cơ sở tri thức và diễn giải logic với công cụ Prolog:

Xây dựng cơ sở tri thức với công cụ Prolog với cây phả hệ gia tộc Uzumaki trong loạt phim hoạt hình "Naruto" và "Naruto shippuden". Gia tộc Uzumaki là gia tộc lớn với số lượng thành viên trải dài qua các thế hệ. Nhóm đã xây dựng số lượng rất lớn các vị từ với cây phả hệ đồ sộ trải dài năm thế hệ.



Một số vị từ được suy diễn trong cơ sở tri thức:

1. Tìm ông hoặc bà cố: *greatgrandparent*(GGP, GGC) :- parent(GGP, GP), grandparent(GP, GGC).
2. Tìm ông là anh hoặc em trai của ông hoặc bà ruột(GU): *greatuncle*(GU, Per) :- grandfather(GF, Per), brother(GU, GF).
3. Tìm mẹ dâu hoặc mẹ rể : *motherinlaw*(M, Per) :- mother(M, Mate), spouse(Mate, Per).
4. Tìm cháu chắt là con gái: *greatgranddaughter*(GGD, GGP) :- greatgrandparent(GGP, GGD), female(GGD).
5. Tìm bà là chị hoặc em của ông hoặc bà ruột: *greataunt*(GA, Per) :- grandfather(GF, Per), sister(GA, GF).
6. Tìm cháu gái là con của anh chị em ruột : *niece*(Per, AU) :- female(Per), parent(P, Per), sibling(P, AU).

Bộ câu hỏi truy vấn dữ liệu được biên soạn gồm 23 câu hỏi nằm trong file **input_uzumaki_family_prolog.txt** như sau:

1. Chồng của Akane là ai?

2. Anh/em trai của Uzumaki Matami là ai?

3. Ông của Senju Tsunade ai?

4. Vợ của Senju Hashirama là ai?

5. Chị/em của Senju Nawaki là ai?

6. Bố/mẹ của Nara là ai?

7. Con của Chiko là ai?

8. Anh/chị em ruột của Uzumaki Sichiro là ai?

9. Anh rể/em rể của Uzumaki Kiku là ai?

10. Cháu trai của Uzumaki Mito (cậu) là ai?

11. Cháu gái của Uzumaki Ren (ông) là ai?

12. Con trai của Chiko là ai?

13. Cậu của Senju Shigeko là ai?

14. Cha của Uzumaki Naruto là ai?

15. Cháu trai của Uzumaki Hayata (ông) là ai?

16. Bà (chị/em của bà ruột) của Uzumaki Kushina là ai?

17. Ông cố của Uzumaki Naruto là ai?

18. Uzumaki Kushina và Nara có phải là chị em họ?

19. Dì của Uzumaki Kiku có phải là Uzumaki Matami không?

20. Con gái của Senju Tsunade có phải là Masaru không?

21. Senju Nawaki và Uzumaki Fuso có phải anh em họ không?

22. Bà của Senju Shigeko có phải là Araki Kouta không?

23. Chiko có phải là con dâu của Yasu không?

Kết quả được lưu vào file **output_uzumaki_family_prolog.txt** trùng với kết quả được mong đợi.

V. Danh mục tham khảo

[1] P. P. H. KHÁNH, Lập Trình Logic Trong Prolog, NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI, 2004.

[2] “Wikipedia,” [Trực tuyến]. Available: https://vi.wikipedia.org/wiki/Prolog.