

## Lab2 – Logic Bậc Nhất

### Cơ sở Trí Tuệ Nhân tạo

Term I /2020-2021

#### Thông tin thành viên

Họ Tên	MSSV
Trần Ngọc Tịnh	18120597
Nguyễn Ngọc Năng Toàn	18120600
Trần Luật Vy	18120656

## I Làm quen với ngôn ngữ Prolog

### 1 Giới thiệu về 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. ( Theo <https://vi.wikipedia.org/wiki/Prolog> )

Prolog là ngôn ngữ lập trình logic và tính toán trên ký hiệu , được biết đến với khả năng mạnh mẽ trong việc giải quyết những vấn đề liên quan đến ký hiệu và suy diễn . Prolog được sử dụng nhiều trong các ứng dụng của trí tuệ nhân tạo và ngôn ngữ trong khoa học máy tính ( đặc biệt trong ngành xử lý ngôn ngữ tự nhiên vì đây là mục tiêu thiết kế ban đầu của nó ) . Nguyên lí lập trình logic dựa trên các mệnh đề Horn ( Horn Logic ) .

#### Ví dụ các mệnh đề Horn :

- + Nếu X là cha mẹ của Y và Y là cha mẹ của Z thì X
- + A là người
- + Tất cả mọi người đều chết ( hoặc nếu ai là Người thì người đó phải chết )
- + Trường Khoa Học Tự Nhiên có 2 cơ sở

Ở 2 ví dụ trên thì 2 mệnh đề 1, 3 được gọi là các luật ( rule ), mệnh đề 2, 4 được gọi là các sự kiện ( fact ). Một chương trình logic có thể được xem như là một cơ sở dữ liệu gồm các mệnh đề Horn, hoặc dạng luật, hoặc dạng sự kiện. Chúng ta có thể gọi chạy chương trình logic bằng cách đặt câu hỏi ( query/ question ) truy vấn trên cơ sở dữ liệu này. Ví dụ :

A có chết không ? ( Có thể hiểu là mệnh đề A có chết không là Đúng hay Sai ? ). Hệ thống logic sẽ thực hiện chương trình theo cách suy luận tìm kiếm dựa trên vốn hiểu biết đã có ( cơ sở dữ liệu ) để có thể chứng minh câu hỏi trên là đúng hay sai. A là người + Luật ở mệnh đề thứ 3 => câu trả lời là Yes. Có nghĩa là sự kiện A chết là đúng

## 2 Cú pháp Prolog

- **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 chân đúng ( True ) hoặc sai ( False ).

Mỗi nguyên tử biểu diễn một quan hệ giữa các hạng ( term ) => Hạng và quan hệ giữa các hạng tạo thành một mệnh đề.

Hạng phức hợp là một hàm tử ( functor ) có chứa các đối số ( argument ), có dạng

**Tên\_hàm\_tử ( Đối\_1,...,Đối\_n )**

Tên hàm tử là một chuỗi chữ cái và/hoặc chữ 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:**

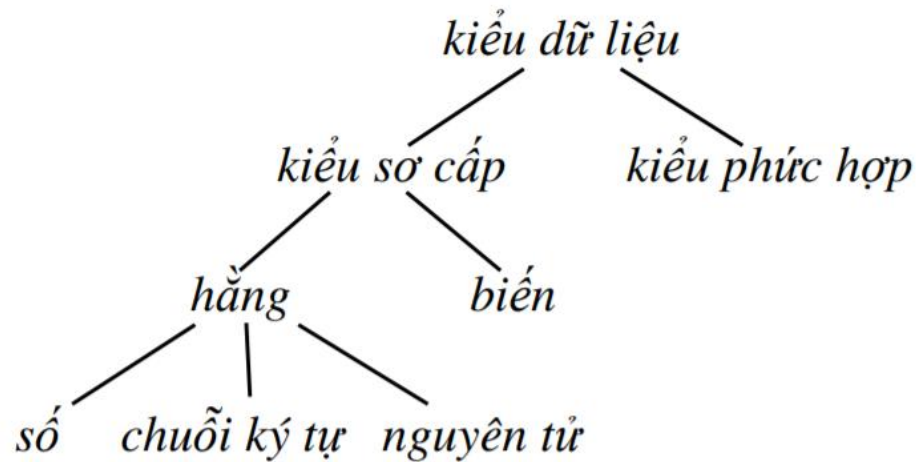
*f(5, a, b).*

*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)

- **Các kiểu dữ liệu Prolog**



**Qui ước đặt tên biến và tên hằng:**

Prolog là ngôn ngữ cho máy tính, vì vậy nó cần một qui ước rất quan trọng trong việc đặt tên biến và tên hằng, theo đó, tên một biến phải bắt đầu bằng ký tự in hoa (chẳng hạn X, Sinhvien, v.v.), còn tên hằng phải bắt đầu bằng ký tự in thường (ví dụ: an, binh, lasinhvien, v.v.). Vì Prolog là ngôn ngữ các câu Horn trong logic vị từ cấp một nên các biến chỉ xuất hiện trong các hạng thức là tham số của các vị từ.

- **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** /\* 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** : % Chú thích %.

### 3 Các kiểu dữ liệu sơ cấp của Prolog

- **Kiểu hằng**

+ Kiểu Logic

Prolog sử dụng hai hằng Logic có giá trị là True và Fail ( False )

+ Kiểu hằng số

Prolog sử dụng cả kiểu số nguyên và số thực , điều mà ta đã khá quen ở các ngôn ngữ lập trình khác như C, C++ ...

**Ví dụ** :      10.4      9      0.5      ....

+Kiểu chuỗi ký tự

Các hằng là chuỗi (string) các ký tự được đặt giữa hai dấu nháy kép.

**Ví dụ :**

" **Tri tue nhan tao** " chuỗi có tùy ý ký tự

"" Chuỗi không có giá trị ( rỗng )

- **Biến**

Tên biến là một chuỗi ký tự gồm chữ cái, chữ số, bắt đầu bởi chữ hoa hoặc dấu gạch dưới  
dòng : **Ví dụ :** X, Y, Z

Tri\_tue

\_tritue

- **Cấu trúc**

Cấu trúc (nhóm các hạng thức lại thành cấu trúc), ví dụ như: mau[red, green, blue],  
[march, 17, 2011], v.v.

#### 4 Chương trình Prolog , các câu Horn dương

Chương trình prolog về cơ bản là dãy các câu Horn dương( định nghĩa Horn dương <https://voer.edu.vn/m/cac-phuong-phap-bieu-dien-dien-tri-thuc/a10d150a> ) Các câu này có dạng Horn dương trong prolog có dạng tổng quát như sau:

**head:- p1, p2, ..., pn.**

{nghĩa là: if (p1 and p2 and ... and pn) then head}

Ở đây *head*, *p1*, *p2*, ..., *pn* là các vị từ (có thể có các tham số); vị từ *head* gọi là phần đầu của luật, còn *p1*, *p2*, ..., *pn* gọi phần thân (phần điều kiện) của luật. Nếu  $n > 0$  thì câu Horn dương trên là câu dạng luật; còn nếu  $n = 0$  thì câu không có phần điều kiện, khi này ta có câu mô tả sự kiện và có thể viết đơn giản là:

**head.**

**Chú ý:** các câu trong chương trình prolog đều kết thúc bởi dấu chấm (“.”). Tất cả các câu đều là câu đóng, nếu có ký hiệu biến xuất hiện trong câu thì ta ngầm hiểu rằng biến đó là biến buộc, đặt dưới lượng từ  $\forall$  , trừ các biến chỉ xuất hiện trong phần điều kiện của câu thì biến đó được hiểu là đặt dưới lượng từ  $\exists$  (thực chất thì nếu chuyển dạng câu tuyên thì  $\exists$  sẽ chuyển sang  $\forall$  do chuyển về và lấy phủ định)

#### 5 Vị từ hạng thức

Chương trình prolog bao gồm hai loại câu: câu sự kiện (câu đơn) và câu luật (câu phức). Các câu này được xây dựng từ các vị từ (*head*, *p1*, *p2*, ..., *pn*), mỗi vị từ có cú pháp như sau:

**tên\_vi\_tu(hang\_thuc1, hang\_thuc2, ..., hang\_thucn)**

Trong đó tên\_vị\_từ tuân theo qui tắc đặt tên hằng; các hạng\_thứci có thể là các kiểu giá trị đã được trình bày ở phần trên.

## 6 Câu truy vấn

Câu truy vấn tổng quát có dạng tổng quát như sau:

**p1, p2, ..., pn.**

Chúng ta chia câu truy vấn thành hai loại:

+ Câu truy vấn không chứa biến: khi đó câu truy vấn có nghĩa là “biểu thức logic (**p1 and p2 and ...and pn**) có là đúng (có giá trị true) trong cơ sở tri thức (chương trình prolog) đã cho hay không?”. Chẳng hạn, câu truy vấn *chame(duong, hoa)* trong ví dụ ở Phần 1 là hỏi: “có phải *hoa* là *chame* của *duong* không?”. Trong trường hợp này, SWI Prolog sẽ trả lời là *true* hoặc *false*.

+ Câu truy vấn có chứa tập các biến (ví dụ X,Y, ...): khác với các câu trong cơ sở tri thức (chương trình prolog) mà ở đó mặc định hiểu rằng các biến là đi với lượng từ  $\forall$ , các biến trong câu truy vấn lại ngầm định đi với lượng từ  $\exists$ , khi đó câu truy vấn có nghĩa là: “có  $\exists$  X, Y, ... sao cho biểu thức logic (**p1 and p2 and ...and pn**) có là đúng (có giá trị true) không?”. Chẳng hạn, câu truy vấn *ongba(X,nam)* trong ví dụ ở Phần 1 là hỏi: “có tồn tại X mà có *nam* là *ongba* của X không?”. Trong trường hợp này SWI Prolog sẽ tìm một giá trị X sao cho *ongba(X,nam)* có giá trị là true. Nếu chúng ta muốn SWI Prolog tìm tất cả các giá trị X thỏa mãn *ongba(X,nam)*, sau mỗi trả lời của Hệ thống, chúng ta ấn phím “;” thay vì ấn phím “enter”.

**Chú ý:** câu truy vấn *q* trong Prolog có dạng như trên sẽ tương đương  $\neg q$  là câu dạng Horn âm  $\forall x,y,\dots (\neg p1 \vee \neg p2 \vee \dots \vee \neg pn)$ . (câu tuyển không có literal dương nào).

## 7 Luật

Luật cho phép chúng ta tạo ra những phát biểu có điều kiện. Mỗi luật có nhiều biến thể, gọi là các mệnh đề.

Xét phát biểu “Ai rồi cũng sẽ chết.”, phát biểu này có thể được biểu diễn bởi luật:

**se\_chet(X) :- nguoi(X)**

Có hai thể hiện của mệnh đề này. Cách thể hiện tuyên bố là “Cho X, X sẽ chết nếu X là người.” Cách thể hiện quy trình là “Để chứng minh đích chính là X sẽ chết, phải chứng minh đích phụ là X là người.”

Tiếp tục thêm vào cơ sở tri thức sự thật “a là người.”, ta có:

**se\_chet(X) :- nguoi(X)**

**nguoi(a).**

Lúc này, nếu truy vấn: " **?- se\_chet(a).**

Prolog sẽ đưa ra câu trả lời là true.

Chúng ta cũng có thể dùng biến trong các truy vấn. Ví dụ, để tìm coi những ai sẽ chết, ta truy vấn:

**?- se\_chet(A).**

Và Prolog sẽ trả về: **A = a.**

Điều này nghĩa là Prolog đã tìm được đích đến se\_chet(A) bằng cách đồng nhất biến A với a. Quy trình làm việc này tương tự trên: ta chứng minh đích chính se\_chet(A) bằng cách chứng minh đích phụ người (A). Prolog sẽ tìm bất kì A nào là người. Nhận thấy người (A) có thể đồng nhất người (a) bằng cách gắn kết A và a. Gắn kết này truyền về đích chính và Prolog trả về như trên.

## 8 Vị từ đệ quy

Vị từ đệ quy là vị từ xuất hiện trong cả phần đầu và phần thân của luật, hay nói cách khác, vị từ gọi chính nó. Định nghĩa vị từ đệ quy bao giờ cũng có 2 phần, phần sự kiện và phần đệ quy. Ví dụ, chương trình sau định nghĩa vị từ fibonacci(N,X) để tính phần tử thứ N trong dãy fibonacci, kết quả đưa vào biến X (dãy Fibonacci là dãy có phần tử thứ nhất bằng 0, phần tử thứ hai bằng 1, phần tử thứ ba trở đi sẽ là tổng của hai phần tử liền ngay trước).

**fibonacci( 1,0). % phần tử đầu tiên là 0**

**fibonacci( 2,1). % phần tử thứ đầu tiên là 1**

**fibonacci( N,F) :- N>2, N1 is N-1, N2 is N-2, fibonacci(N1,F1),**

**fibonacci(N2,F2), F is F1+F2.**

Truy vấn chương trình logic này với các tham số N khác nhau ta sẽ được kết quả lưu trên biến F là phần tử thứ N của dãy. Ví dụ:

**1 ? - fibonacci(3,F).**

**F=1**

**2 ? - fibonacci(4,F).**

**F=2**

**3 ? - fibonacci(10,F).**

**F=34**

**Chú ý:** Vị từ fibonacci(N,F) ở trên là để định nghĩa phần tử thứ N của dãy Fibonacci và kết quả lưu trong F, vì vậy mà SWI chỉ có thể thực hiện các câu truy vấn mà ở đó tham số thứ nhất là hằng số, ví dụ câu truy vấn như fibonacci(10,F) để tìm phần tử thứ 10 của dãy; câu truy vấn như fibonacci(10,34) để kiểm tra xem phần tử thứ 10 của dãy có là 34 không.

## II Tìm hiểu SWI-Prolog

### 1 Tổng quan

Trang chủ của SWI-Prolog nằm ở: <http://www.swi-prolog.org/>

Đường dẫn để tải chương trình SWI-Prolog phiên bản ổn định

<http://www.swiprolog.org/download/stable>. Trong đó có cả hướng dẫn tải cho các hệ điều

hành khác nhau (Windows, Ubuntu, Mac,...).

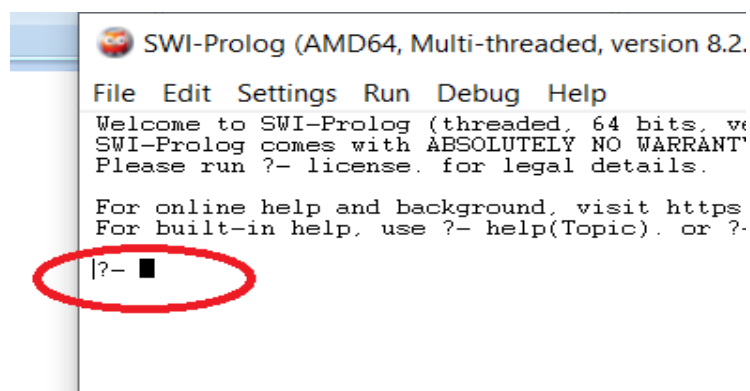
Bản quyền sử dụng phần mềm SWI-Prolog là miễn phí cho mục đích học tập và nghiên cứu

## Giao diện của SWI-Prolog ( version 8.2.2 )



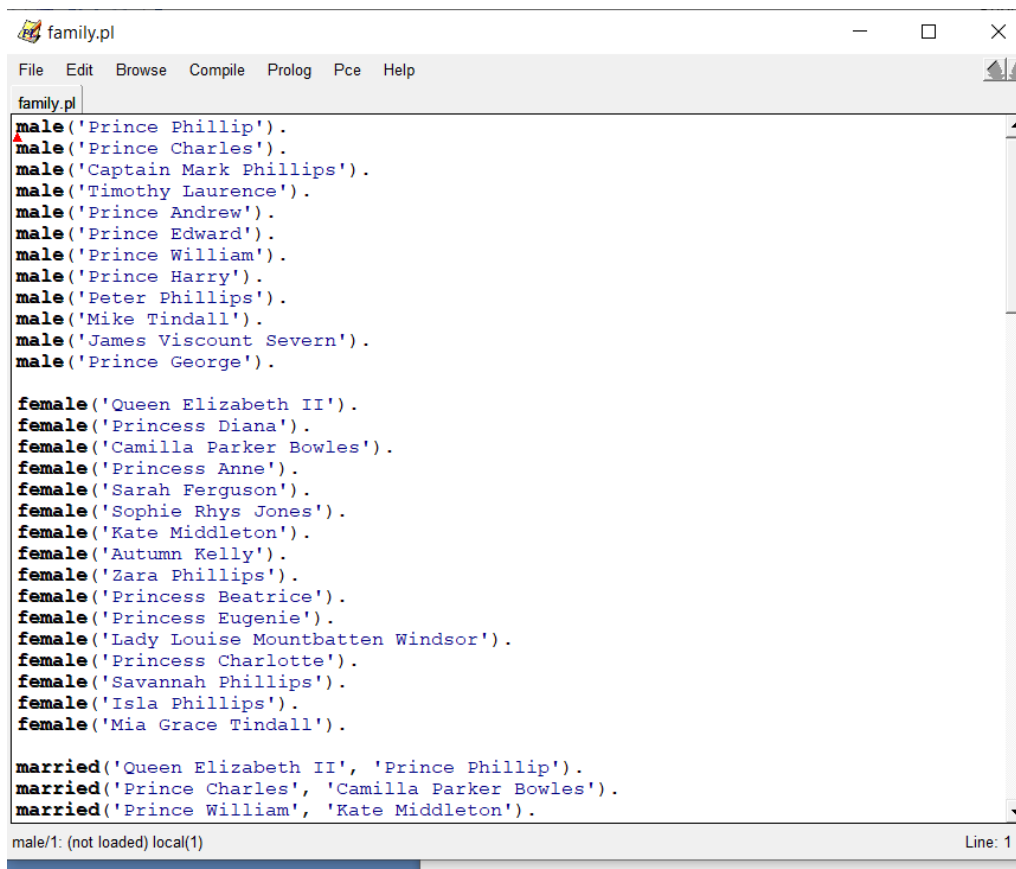
## 2 Cách triển khai ngôn ngữ Prolog trên môi trường SWI-Prolog

Sau khi cài đặt và chạy chương trình SWI- Prolog . Hệ thống hiển thị dấu nhắc yêu cầu nhập vào truy vấn như hình ở phía dưới :



Trước khi nhập câu truy vấn , chúng ta phải cho hệ thống biết chúng ta sẽ truy vấn trên cơ sở truy thức nào . Một cơ sở truy thức là một khai báo các sự kiện và các luật về một lĩnh vực nào đó , và được lưu trong một File . Để Load một file cơ sở truy thức , ta sử dụng menu File =>Consult => Chọn File . Các tệp tin chương trình Prolog có phần mở rộng là “.pl” hoặc “.pro” khi có xung đột về tên mở rộng tệp tin giữa các phần mềm . Các mô tả các sự kiện và luật ( các câu Horn ) trong các file cơ sở tri thức được gọi là chương trình Prolog . Nhiệm vụ của người lập trình logic là viết các chương trình Prolog này và các câu truy vấn .

**Ví dụ** về file chương trình Prolog ( cơ sở tri thức ) có tên file là family.pl ( có thể sử dụng bất cứ bộ soạn thảo văn bản nào , SWI cũng cung cấp một bộ soạn thảo bằng cách sử dụng *menu* => *File* => *New/Edit* => *Nhập tên File* ).



```
family.pl
File Edit Browse Compile Prolog Pce Help
family.pl
male('Prince Phillip').
male('Prince Charles').
male('Captain Mark Phillips').
male('Timothy Laurence').
male('Prince Andrew').
male('Prince Edward').
male('Prince William').
male('Prince Harry').
male('Peter Phillips').
male('Mike Tindall').
male('James Viscount Severn').
male('Prince George').

female('Queen Elizabeth II').
female('Princess Diana').
female('Camilla Parker Bowles').
female('Princess Anne').
female('Sarah Ferguson').
female('Sophie Rhys Jones').
female('Kate Middleton').
female('Autumn Kelly').
female('Zara Phillips').
female('Princess Beatrice').
female('Princess Eugenie').
female('Lady Louise Mountbatten Windsor').
female('Princess Charlotte').
female('Savannah Phillips').
female('Isla Phillips').
female('Mia Grace Tindall').

married('Queen Elizabeth II', 'Prince Phillip').
married('Prince Charles', 'Camilla Parker Bowles').
married('Prince William', 'Kate Middleton').

male/1: (not loaded) local(1) Line: 1
```

Như ta thấy thì trong file family.pl dữ liệu gồm nhiều dòng ( là các mệnh đề được đánh dấu và kết thúc bằng dấu chấm ) . Mỗi mệnh đề là một sự thật hay một luật ( phân cách giữa phần hệ quả và phần tiền đề bằng dấu :- ) .

Tiếp theo chúng ta nhập các câu truy vấn từ dấu nhắc của SWI Prolog .



- Truy vấn liệt kê

#### Ví dụ

+ Ai là cháu gái của Princess Anne?

```
?- niece(X, 'Princess Anne').
```

SWI Prolog sẽ hiện theo thứ tự lần lượt liệt kê nếu bạn ấn dấu “chấm phẩy ( ; )”, hoặc ấn dấu chấm ( . ) nếu muốn dừng.

```
?- niece(X, 'Princess Anne').
X = 'Zara Phillips' ;
X = 'Zara Phillips' ;
X = 'Princess Beatrice' ;
X = 'Princess Beatrice' ;
X = 'Princess Eugenie' ;
X = 'Princess Eugenie' ;
X = 'Lady Louise Mountbatten Windsor' ;
X = 'Lady Louise Mountbatten Windsor' ;
?-!
```

+ Ai là cháu trai của Prince Harry?

```
?- nephew(X, 'Prince Harry').
X = 'Prince George' ;
X = 'Prince George' ;
?-!
```

- Truy vấn True / False

#### Ví dụ

+ Zara Phillips có phải là chị/em gái với Peter Phillips không?

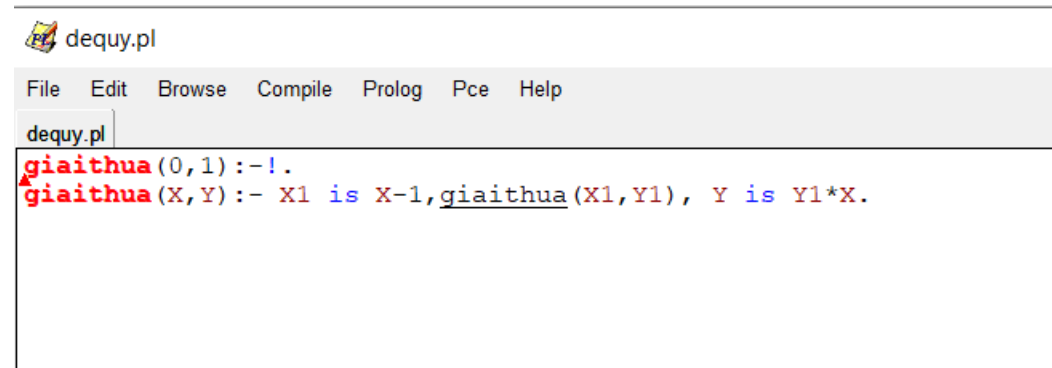
```
?- sister('Zara Phillips', 'Peter Phillips').
true ■
```

+ Zara Phillips có phải là mẹ của Mia Grace Tindal?

```
?- mother('Zara Phillips', 'Mia Grace Tindal').
false.

?-
```

- Cấu trúc đệ quy



Sau khi nạp vào hệ thống, chúng ta sẽ đặt cho hệ thống câu hỏi dạng nghi vấn như sau:

### Giaithua(2,5):

Hiểu theo nghĩa ngôn ngữ tự nhiên sẽ là có phải giai thừa của 2 có phải là 5 hay không. Câu trả lời được miêu tả cho 2 ví dụ ở hình dưới :

```
?- giaithua(2,5).  
false.  
?- giaithua(2,2).  
true.  
?-
```

Tiếp theo chúng ta sẽ đặt câu hỏi tìm lời giải cho SWI-Prolog `giaithua(3,X)`. Có nghĩa là giai thừa của 3 là bao nhiêu => và câu trả lời chương trình đưa ra cho chúng ta sẽ là 6

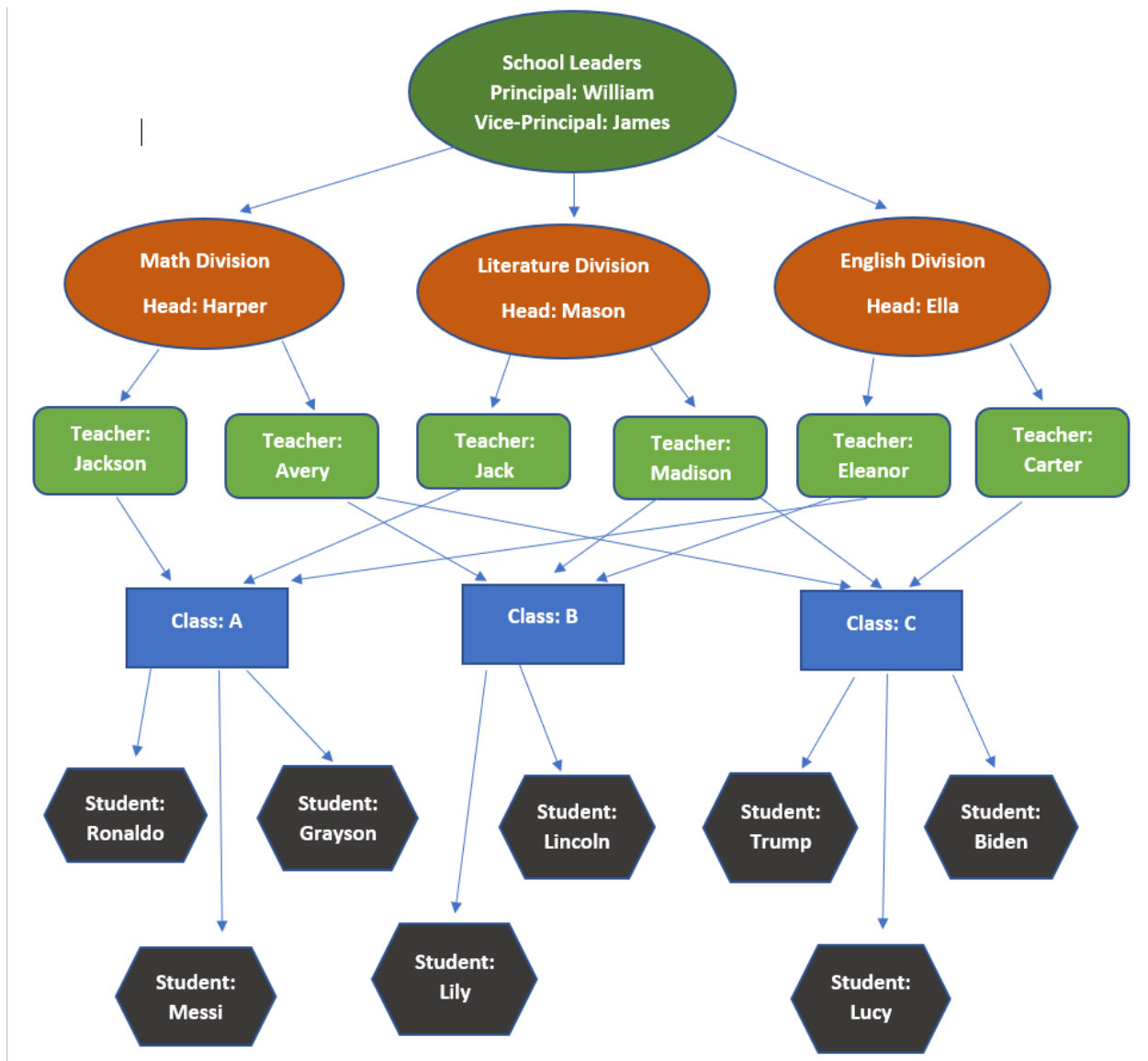
```
?- giaithua(2,X).  
X = 2.  
?- giaithua(5,X).  
X = 120.  
?-
```

Chúng ta cũng có thể đặt câu hỏi sau `giaithua(X,Y)`. Cả hai thông số trên là biến, như vậy câu hỏi có thể hiểu là : số nào ( X ) giai thừa thì thành một số khác ( Y ). Câu hỏi gần như vô nghĩa và những câu trả lời của hệ thống cũng chẳng mang một ý nghĩa thực sự nào.

```
?- giaithua(X,Y).  
X = 0,  
Y = 1.  
?- giaithua(X,Y).  
X = 0,  
Y = 1.  
?-
```

## II Xây dựng cơ sở tri thức với công cụ Prolog

**Chủ đề:** quan hệ giữa các đối tượng trong nhà trường.



**Mô tả nội dung sơ đồ:**

School Leaders: là lãnh đạo của trường gồm 1 hiệu trưởng (principal) và 1 hiệu phó (vice-principal).

Trường có 3 khoa: toán, ngữ văn và anh văn. Mỗi khoa sẽ có 1 trưởng khoa (Head).

Mỗi khoa sẽ có những giáo viên (teacher) giảng dạy các bộ môn của khoa đó.

Mỗi lớp học (class) sẽ có các giáo viên giảng dạy 3 bộ môn nêu trên.

Học sinh (student) là thành viên thuộc các lớp nêu trên.

## Các mối quan hệ:

Male(Person): người thuộc giới tính nam.

Female(Person): người thuộc giới tính nữ.

Principal(Person): người làm hiệu trưởng.

Viceprincipal(Person): người làm phó hiệu trưởng.

Mathhead(Person): trưởng bộ môn toán.

Mathteacher(Person): giáo viên dạy toán.

Literaturehead(Person): trưởng bộ môn ngữ văn.

Literatureteacher(Person): giáo viên dạy ngữ văn.

Englishhead(Person): trưởng bộ môn tiếng anh.

Englishteacher(Person): giáo viên dạy anh văn.

Class(name): tên của các lớp.

Student(Person): người làm học sinh.

Schoolleader(Person): người làm lãnh đạo trường gồm: hiệu trưởng, hiệu phó.

Divisionhead(Person): người làm trưởng khoa.

Teacher(Person): người làm giáo viên.

Boss(Higher, Lower): school leaders sẽ là xếp của trưởng khoa và giáo viên, trưởng khoa sẽ là xếp của giáo viên.

Teachin(teacher, class): giáo viên được phân công dạy ở lớp nào đó.

Classteacher(class, teacher): lớp học gồm có các giáo viên nào.

Studentinclass(student, class): học sinh sẽ thuộc một lớp học.

Classhavestudent(class, student): lớp học có học sinh nào.

Classmate(student1, student2, nameclass): bạn cùng lớp.

Schoolmate(student1, student2): bạn cùng trường.

## II Cài đặt hệ thống suy diễn logic bằng ngôn ngữ lập trình

## Code bằng Python

- Sơ lược quy trình:
- Xử lý file:

- Lấy thông tin từ file. Nếu luật có dạng A:-B;C,D. thì tách thành A:-B và A:-C, D. và xóa luật cũ.

1. Nhập câu hỏi x nào đó

2. Kiểm tra xem x có chứa biến không

2.1. Nếu không thì kiểm tra xem x là luật hay sự kiện

- Nếu là sự kiện thì in ra kết quả

- Nếu là luật tiến hành kiểm tra xem nằm ở luật thứ mấy và có bao nhiêu luật trùng tên.

+ Kiểm tra có bao nhiêu luật trùng với tên câu hỏi, thử từng luật.

+ Tách các hàm tử cấu tạo nên luật x

+ Định quy xét nếu hàm tử con của luật x là luật

+ Trả về từng tập các hàm tử khác nhau ứng với từng luật nếu hàm tử con trùng với nhiều luật.

+ Thêm vào tập kết quả, sử dụng dictionary.

+ Trong quá trình xử lý, để tránh bị trùng lặp thì key của tập kết quả sẽ là 1 chuỗi cộng dồn các vị trí của các luật trùng tên. Vị trí xét trong tập luật thu được từ tập tin

+ Nếu hàm tử con là sự kiện tiến hành bổ sung các giá trị vào các biến của hàm tử con theo tất cả hàm tử con.

+ Tiếp tục bổ sung các giá trị vào các biến

+ Trả lời câu hỏi True False.

- Nếu không in ra "ERROR!!"

2.2. Nếu x có chứa biến

- Tạo 2 tập result để chứa kết quả và tập visited để chứa giá trị đã xét qua.

- Xét trong tập fact các giá trị của đối số. Ví dụ là giá trị a.

- Thay biến trong chuỗi câu hỏi x bằng a. Nếu với a thì x đúng, ta tiến hành thêm a vào tập result.

- In ra tập biến.

- Hàm:

- is\_type(line): Kiểm tra xem chuỗi đó là luật hay là sự kiện.

- take\_rules\_laws(rule, fact, fileName): Lấy thông tin từ file và sắp xếp các chuỗi nhận được vào 2 tập luật hay sự kiện.

- take\_input(): Nhập câu hỏi.
- \_arg(ques): Tách các đối số của một hàm tử
- \_functor(ques): Tách tên hàm tử
- answer\_fact(fact, question): Trả lời các câu hỏi sự kiện
- parsed\_rule(rule): Tách các hàm tử con của một luật
- is\_rule(functor, rule): Kiểm tra câu hỏi có phải là luật, trả về vị trí của luật trong tập luật
- is\_fact(functor, fact): Kiểm tra câu hỏi xem có phải là sự kiện.
- is\_var(arg): Kiểm tra xem đối số có phải là biến.
- \_var\_fact(question,fact): in ra tập giá trị của một biến.
- fill\_rule(rule,ques\_arg): Điền đối số của câu hỏi vào các đối số của luật.
- fill\_var(predicats,idx,fact,ques\_arg): Điền các giá trị của biến.
- processrule(line,fact,rule):Xử lý câu hỏi.
- answer\_question(rule,fact,question): Trả lời câu hỏi.
- replace\_var(question,x,line,idx): thay biến trong câu hỏi bằng 1 giá trị.
- print\_var\_result(question,rule,fact,x,idx): in ra tập biến.
- valueof(clauses, fact): Trả về True False của 1 tập hàm tử.

- Ví dụ:

- Cây Hoàng Gia Anh:
  - +       ?- brother(X,'Prince Edward').
  - X = 'Prince Charles'
  - X = 'Prince Andrew'
  - +       ?- granddaughter('Isla Phillips','Princess Anne').
  - True
  - +       ?- daughter(X, 'Prince Edward').
  - X = 'Lady Louise Mountbatten Windsor'
  - +       ?- daughterr(X, 'Prince Edward').
  - ERROR!!!
  - +       ?- father(X, 'James Viscount Severn').
  - X = 'Prince Edward'
  - +       ?- niece(X,'Princess Anne').

$X = \text{'Princess Beatrice'}$   
 $X = \text{'Princess Eugenie'}$   
 $X = \text{'Lady Louise Mountbatten Windsor'}$   
 +  $\text{?- grandmother}(X, \text{'Savannah Phillips'})$ .  
 $X = \text{'Princess Anne'}$   
 +  $\text{?- aunt}(\text{'Zara Phillips'}, \text{'Isla Phillips'})$ .  
 True  
 +  $\text{?- husband}(X, \text{'Sophie Rhys Jones'})$ .  
 $X = \text{'Prince Edward'}$

Hoàn thành: In giá trị đúng sai và in được giá trị của biến.

Tài liệu tham khảo : <https://huuvinhfit.files.wordpress.com/2015/01/chuong-7-prolog.pdf>  
<https://masterwed.files.wordpress.com/2010/07/ltprolog123.pdf>

+Sách Cơ sở trí tuệ nhân tạo – Lê Hoài Bắc & Tô Hoài Việt

+Giáo trình trí tuệ nhân tạo –Khoa Công Nghệ Thông Tin Đại học sư phạm Hà Nội

+Logic vị từ - Vũ Quốc Hoàng ( Đại học Khoa học tự nhiên )