



**BÁO CÁO ĐỒ ÁN 2**  
**MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO**

**Giảng viên thực hành: Dương Nguyễn Thái Bảo**

**Danh sách thành viên**

Họ tên đầy đủ	MSSV
Đặng Văn Hiến	18120363
Lê Thanh Viễn	18120647

## BẢNG PHÂN CHIA CÔNG VIỆC

Thành viên	Công việc thực hiện	Mức độ hoàn thành
<b>Đặng Văn Hiến</b>	Tìm hiểu ngôn ngữ Prolog (10%)	100%
	Tìm hiểu môi trường lập trình SWI-Prolog (10%)	100%
	Cài đặt hệ thống suy diễn logic bằng ngôn ngữ lập trình (30%)	100%
<b>Lê Thanh Viễn</b>	Giải quyết bài toán bằng công cụ SWI-Prolog (20%)	100%
	Xây dựng cơ sở tri thức với công cụ SWI-Prolog (30%)	100%

## Nội dung

<b>A) TÌM HIỂU NGÔN NGỮ PROLOG.....</b>	<b>4</b>
I) Tổng quan về ngôn ngữ Prolog .....	4
II) Cú pháp.....	4
III) Các kiểu dữ liệu của Prolog.....	5
IV) Các phép toán trong Prolog.....	7
1) Số học: .....	7
2) Các vị từ xác định:.....	8
<b>B) MÔI TRƯỜNG LẬP TRÌNH SWI-PROLOG.....</b>	<b>10</b>
I) Giới thiệu về SWI Prolog .....	10
II) Sử dụng SWI Prolog .....	10
1) Chương trình Prolog .....	10
2) Ví dụ về chương trình Prolog .....	10
3) Cú pháp trong SWI Prolog .....	12
4) Cách suy diễn trong Prolog .....	14
III) Các ví dụ minh họa.....	15
<b>C) XÂY DỰNG CƠ SỞ TRI THỨC VỚI SWI-PROLOG .....</b>	<b>18</b>
I) Xây dựng cơ sở tri thức.....	18
1) Sơ đồ quan hệ.....	18
2) Sơ đồ quan hệ chi tiết .....	19
II) Vài ví dụ về truy vấn của cơ sở tri thức.....	21
<b>D) CÀI ĐẶT HỆ THỐNG NGÔN LOGIC BẰNG NGÔN NGỮ LẬP TRÌNH.....</b>	<b>23</b>

## A) TÌM HIỂU NGÔN NGỮ PROLOG

### I) Tổng quan về ngôn ngữ Prolog

#### Khái niệm:

- Xuất hiện từ năm 1972 (do Alain Colmerauer và Robert Kowalski thiết kế).
- Là một ngôn ngữ lập trình giúp người dùng mô tả lại bài toán trên ngôn ngữ của logic, dựa trên đó, máy tính sẽ tiến hành suy diễn tự động dựa vào những cơ chế suy diễn có sẵn (hợp nhất, quay lui và tìm kiếm theo chiều sâu) để tìm câu trả lời cho người dùng.

### II) Cú pháp

Một chương trình Prolog bao gồm các luật được biểu diễn dưới dạng **mệnh đề Horn**. Một mệnh đề Horn có dạng:

Head:-Body.

Head là một vị từ logic, còn Body có thể là rỗng hoặc là một tập các vị từ logic. Ví dụ như sau:

chẵn(X):- X chia\_dư 2 = 0.

Phần lớn các bộ dịch của các chương trình Prolog đều yêu cầu vị từ logic ở phần đầu của một mệnh đề Horn là một vị từ dương (không có dấu phủ định đi kèm), còn các vị từ trong phần Body có thể có dấu phủ định đi kèm. Chương trình logic mà không có sự xuất hiện của dấu phủ định đi kèm gọi là chương trình logic xác định, còn không thì được gọi là chương trình logic thường.

#### - 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 dữ kiện của bài toán, ví dụ như việc khai báo "tôm" là một con mèo:

mèo(tôm).

Khoảng cách từ Hà Nội vào thành phố Hồ Chí Minh là khoảng 2000Km:

khoảng\_cách(Hà\_Nội,TP\_Hồ\_Chí\_Minh,2000).

#### - 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 nếu công tắc đèn bật thì đèn sáng:

đèn\_sáng(X):- công\_tắc\_bật(X).

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 công tắc của đèn X bật thì đèn X sẽ sáng". 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:

`đèn_sáng(X):- công_tắc_bật(X), có_điện.`

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.

### III) Các kiểu dữ liệu của Prolog

**Clause<mệnh đề>:** Các mệnh đề cấu trúc nên một chương trình Prolog .

**Predicat<vị từ>:** Mỗi mệnh đề được xây dựng từ các vị từ. Một vị từ là một phát biểu về các đối tượng có giá trị là đúng (true) hoặc sai (false). Một vị từ có thể chứa có nguyên tử logic.

**Các kí hiệu** bao gồm:

`:-` (điều kiện nếu).

`,` (điều kiện và).

`;` (điều kiện hoặc).

`.` (kết thúc vị từ)

**Logic atom:** < nguyên tử logic>: biểu diễn quan hệ giữa các hạng (term) => Hạng và quan hệ của các hạng tạo thành một mệnh đề

**Term:** đối tượng dữ liệu của prolog. Hạng sơ cấp: hằng (constant), biến (variable) . Hạng phức hợp: biểu diễn các đối tượng phức tạp của bài toán. Hạng phức hợp là một hàm tử functor chứa các đối số argument

**Functor:** Tên\_hàm \_tử(Đối\_1, Đối\_2,..., Đối\_n)

Tên hàm tử là một chuỗi chữ cái và/ hoặc chữ số , bắt đầu bằng một chữ cái thường. dùng xây dựng các cấu trúc

**Luật:** <...>:-<...>

**Câu hỏi:** ?-<...>

**Chú thích:** /\* chú thích \*/ để chú thích nhiều dòng hoặc % để chú thích theo dòng

**Kiểu hằng số:** Prolog sử dụng cả số nguyên và số thực:

Ví dụ: 1 1.23 -0.005 -1000

**Kiểu hàng logic:** true và false. Được dùng như các mệnh đề và kết quả trả về.

**Kiểu hằng chuỗi ký tự:** được đặt trong dấu nháy kép

Ví dụ: “Đại học khoa học tự nhiên”: chuỗi

“”: chuỗi rỗng

“\””: chuỗi chỉ có nháy kép

**Biến:** Tên biến là một chuỗi ký tự gồm chữ cái, chữ số, bắt đầu bởi chữ hoa hoặc dấu gạch dưới dòng.

X, Y, A Result, List\_of\_members \_x23, \_X, \_, ... Phép Toán và Số học của Prolog

Nếu biến chỉ xuất hiện một lần trong mệnh đề thì không cần đặt tên chi nó, ta sử dụng biến nặc danh với tên biến là \_ dấu gạch dưới.

Ví dụ: co\_con(X):-chame(X,\_)

Tức là X có con khi là cha mẹ của một ai đó, mà ta không cần quan tâm người con đó là ai

Nếu trong mệnh đề có nhiều biến nặc danh thì các biến này là khác nhau. Nếu biến nặc danh xuất hiện trong câu hỏi thì kết quả không hiển thị giá trị tìm được của biến nặc danh đó.

Các biến chỉ có tác dụng trong một mệnh đề. Tức là ở hai mệnh đề khác nhau thì hai biến cùng tên là khác nhau. Còn hằng thì thể hiện một đối tượng duy nhất trong suốt chương trình.

**Câu hỏi:** dùng để tra cứu một điều gì đó. Ví dụ: Muốn tìm con nào là con mèo:

?-mèo(X).

Sau khi hiển thị câu trả lời đầu tiên, Prolog sẽ lần lượt tìm kiếm dữ kiện thoả mãn và lần lượt hiển thị kết quả nếu chừng nào người còn yêu cầu cho đến khi không còn kết quả lời giải nào nữa (kết thúc bởi Yes) . Ví dụ:

?:-Cat(X)

X=tôm;

X=abc;

Ở kết quả đầu tiên, để tiếp tục nhận các kết quả khác, người dùng tiếp tục yêu cầu bằng dấu hai chấm (; ). Nhấn enter hoặc dấu chấm . để kết thúc luồng trả lời.

Khi hỏi các câu hỏi một lần và cách nhau dấu , tức là tìm đáp án thoả mãn tất cả các câu hỏi đó(tức là phép và (and))

Ví dụ cần hỏi. Lan và Quang có cùng cha không?

?- cha(X, Lan),cha(X,Quang)

Tức là prolog sẽ tìm một người X thoả mãn vừa là cha của Lan và của Quang. Không có thì sẽ trả lời là no. câu trả lời “no” tức là không tìm ra câu trả lời thoả mãn câu hỏi (mà khác với các câu trả lời trước đó).

Việc trả lời câu trong prolog là đi tìm các dữ kiện thế vào các luật thoả mãn.

Ví dụ: Ta đặt câu hỏi: ?:-father(X,Y)

Lúc này prolog sẽ đi tìm định nghĩa của luật này hay dữ kiện về father. Nếu đó là một luật thì sẽ bắt đầu tìm câu trả lời cho phần thân của luật đó. Nếu nó là một dữ kiện thì X,Y là những hạng được xác định quan hệ parent trước đó. Tóm lại là để trả lời một câu hỏi thì prolog tìm và thay thế, trả lời những câu hỏi nhỏ để trả lời có tồn tại hay không.

Nếu trong các vị từ là các hạng thì kết quả là có (yes) hay không (no) các hạng đó thoả mãn phần thân của luật hay là dữ kiện được xác định trước đó.

### **Đệ qui trong prolog**

Để xác định qua hệ tổ tiên trong cây phả hệ. Ta cần xác nhiều luật theo nhiều bậc của cây phả hệ

Chẳng hạn: Bậc 1: Ông b

totien(X,Y):-chame(X,Z),chame(Z,Y).

Ông bà cố: totien(X,Y):- chame(X,Z),chame(Z,U),chame(U,Y).

Cây phả hệ càng lớn thì các định nghĩa càng để nhiều. Ta sử dụng sức mạnh của prolog là Đệ qui. Ta có định nghĩa đệ qui của quan hệ tổ tiên như sau: Tổ tiên nếu là cha mẹ hay tổ tiên của cha mẹ. Ta có quan hệ viết bằng prolog như sau

totien(X,Y):-chame(X,Y)

totien(X,Y):-totien(X,Z),chame(Z,X).

## **IV) Các phép toán trong Prolog**

### **1) Số học:**

**Cộng:** +

**Trừ:** -

**Nhân: \***

**Chia số thực: /**

**Chia lấy số nguyên: //**

**Mod:** chia lấy phần dư

Biểu thức số học được xây dựng bằng vị từ is. Đối số nằm bên trái là một đối tượng sơ cấp. Đối số bên phải là một biểu thức toán học.

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

### - Chú ý:

+ Vì Prolog hiểu đây là hàm tử + với hai đối số 1 và 2.

+ Phần biểu thức toán học là các phép tính giữa các số hay các hàm (function (sin, cos,...))

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

```
?- X is 5*2.
X = 10.
```

```
?- is(X, 5*2)
|
X = 10.
```

```
?- 1.0 is sin(pi/2)
|
true.
```

```
?- 1 is sin(pi/2)
|
false.
```

```
?- X is 1+Z.
```

```
ERROR: Arguments are not sufficiently instantiated
```

```
?- X is 1+z.
```

```
ERROR: Arithmetic: 'z' is not a function
```

```
?- X is cos(1).
```

```
X = 0.5403023058681398.
```

### Các phép so sánh

>(lớn), < (nhỏ), <= (nhỏ hơn hoặc bằng), >= (lớn hơn hoặc bằng), =\= (khác), =:= (bằng),  
between(a,b,Z) (a<Z=0), succ(Int1, Int2) (thành công khi Int2=Int1+1 và Int1>=0),  
plus(Int1,Int2,Int3)(thành công khi Int3=Int2+Int1)

### 2) Các vị từ xác định:

Nonvar(X) : X không phải là biến?

Atom(A): A là một nguyên tử?

Integer(I): I là một số nguyên?



float(R) R là một số thực (dấu chấm động) ?

number(N) N là một số (nguyên hoặc thực) ?

atomic(A) A là một nguyên tử hoặc một số ?

compound(X) X là một hạng có cấu trúc ?

ground(X) X là một hạng đã hoàn toàn ràng buộc ?

functor(T, F, N) T là một hạng với F là hạng tử và có N đối (arity) T =..L Chuyển đổi hạng T thành danh sách L  
clause(Head, Term) Head :- Term là một luật trong chương trình ?  
arg(N, Term, X) Thế biến X cho tham đối thứ N của hạng Term  
name(A, L) Chuyển nguyên tử A thành danh sách L gồm các mã ASCII.

Ví dụ:

?- 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

## **B) MÔI TRƯỜNG LẬP TRÌNH SWI-PROLOG**

### **I) Giới thiệu về SWI Prolog**

Prolog là một ngôn ngữ lập trình. Tên gọi Prolog được xuất phát từ cụm từ tiếng Pháp Programmation en logique, nghĩa là lập trình logic. Prolog được sử dụng nhiều trong các ứng dụng trí tuệ nhân tạo và ngôn ngữ học trong khoa học máy tính.

SWI Prolog là một môi trường lập trình Prolog rất phổ biến, có các phiên bản chạy trên các hệ điều hành khác nhau như Windows, MacOS, Linux. SWI Prolog là một môi trường mã nguồn mở và thường được sử dụng trong dạy học.

SWI Prolog còn được sử dụng như là một ngôn ngữ nhúng và dùng như là các cơ sở dữ liệu suy diễn

### **II) Sử dụng SWI Prolog**

#### **1) Chương trình Prolog**

Mỗi chương trình Prolog là một cơ sở dữ liệu gồm các mệnh đề. Các mệnh đề trong chương trình có thể được sắp xếp theo bất kỳ trật tự nào. Các mệnh đề của một vị từ được sử dụng theo đúng trật tự của chúng trong chương trình.

#### **2) Ví dụ về chương trình Prolog**

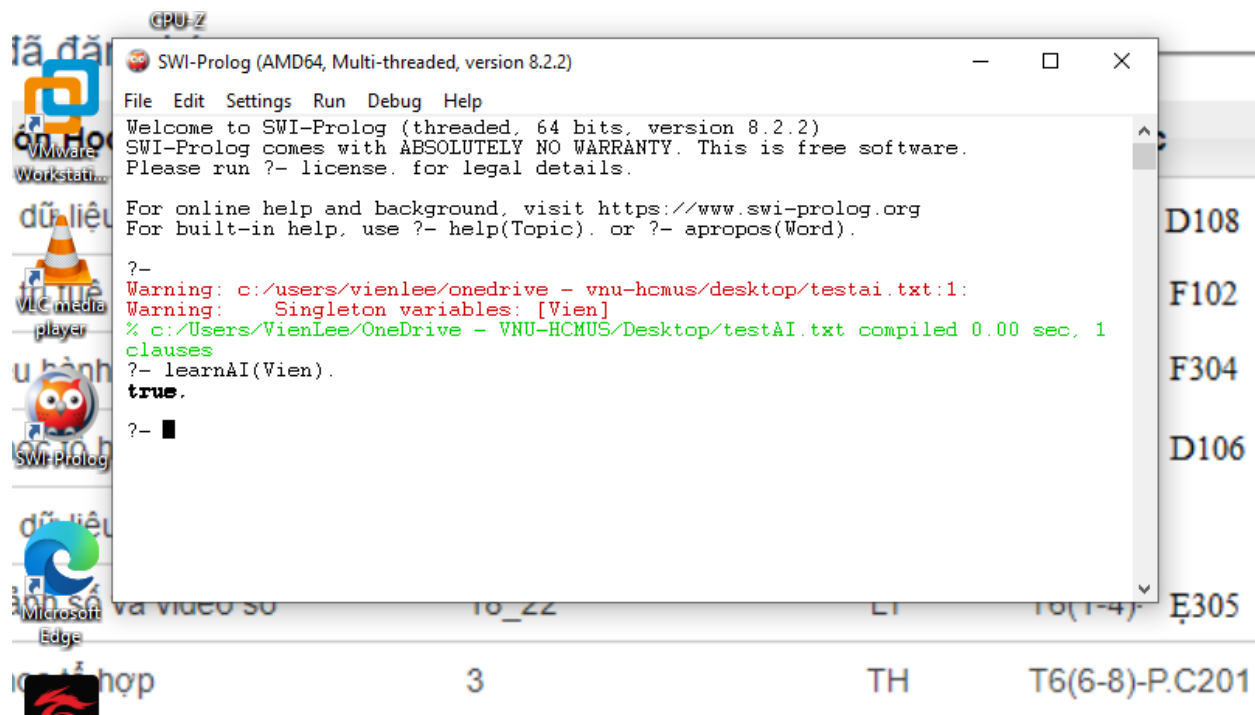
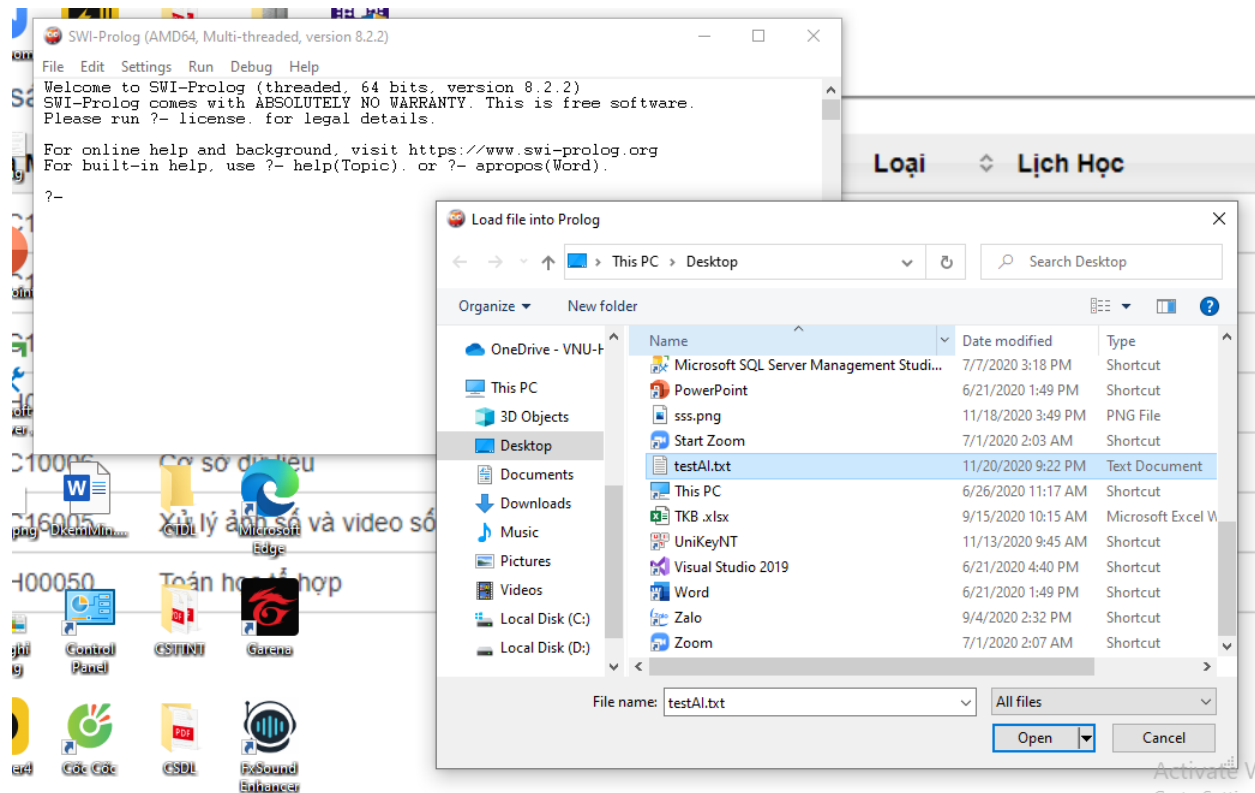
Sử dụng một chương trình soạn thảo để tạo ra cơ sở tri thức. Ghi lại chương trình trong một tập tin định dạng văn bản sử dụng đuôi của tập tin là .pl

Để chạy chương trình Prolog, ta mở phần mềm SWI Prolog và consult tới tập tin của chương trình. Sau đó đưa ra câu hỏi mong muốn.

Ví dụ: Ta có các suy diễn logic sau

- Viễn học IT
- Mọi người học IT đều học AI
- Hỏi : Viễn có học AI không?

Ta consult tới tập tin đã được tạo để chứa cơ sở tri thức



Ta thực hiện câu truy vấn `learnAI(vien)`. (phải có dấu chấm). Ta thấy giá trị trả về là **true** có nghĩa là chứng minh được.

Còn nếu không phải là `learnAI(vien)`. Thì giá trị trả về sẽ là **false**.

```
?- learnAI(viebn).  
false.  
?-
```

### 3) Cú pháp trong SWI Prolog

- Một cấu trúc gồm tên và không, một hoặc nhiều tham số. Nếu không có tham số thì bỏ đi dấu ngoặc.

- Một cấu trúc chính là một mệnh đề cơ sở. Một mệnh đề cơ sở biểu diễn một sự kiện (fact)

Ví dụ:

- `hien.`
- `vien.`
- `vi.`
- `love(hien,vi).`
- `love(vien,vi).`

```
SWI-Prolog (AMD64, Multi-threaded, version 8.2.2)
File Edit Settings Run Debug Help
?- learnAI(Vien).
Vien = vien.
?- learnAI(vien).
true.
?- learnAI(viebn).
false.
?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, 4 clauses
?- love(X,vi).
X = vien ;
X = hien.
?-
```

```
testAI.txt - Notepad
File Edit Format View Help
hien.
vien.
vi.
love(vien,vi).
love(hien,vi).
```

- Một luật được biểu diễn bao gồm:
  - + Một cấu trúc biểu diễn mệnh đề kết luận của luật
  - + Ký hiệu :-
  - + Một danh sách các cấu trúc biểu diễn mệnh đề giả thiết của luật ngăn cách bởi dấu ‘,’. Dấu ‘,’ giữa các cấu trúc có nghĩa như toán tử logic AND

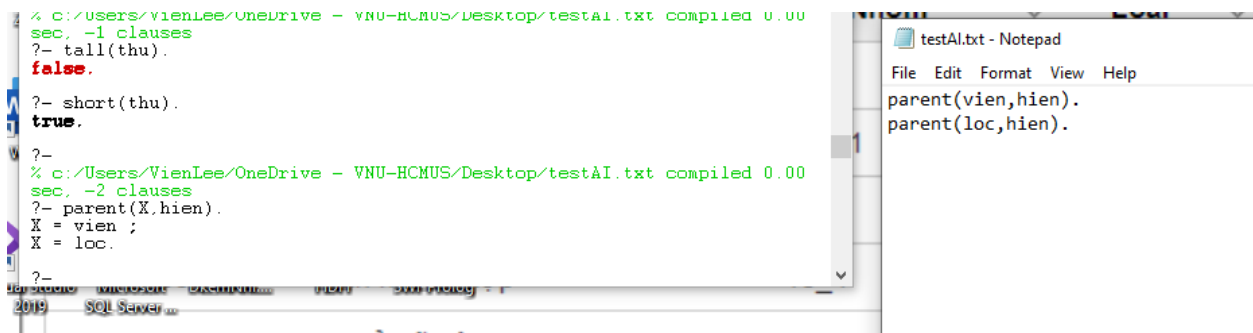
```
SWI-Prolog (AMD64, Multi-threaded, version 8.2.2)
File Edit Settings Run Debug Help
X = vien ;
X = hien.
?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, -1 clauses
?- tall(thu).
false.
?- short(thu).
true.
?-
```

```
testAI.txt - Notepad
File Edit Format View Help
man(vien).
woman(thu).
tall(X):-man(X).
short(Y):-woman(Y).
```

- Một vị từ là tập hợp các mệnh đề với cùng tên và một số tham số

Ví dụ: Các mệnh đề biểu diễn vị từ parent:

- + parent(vien,hien).
- + parent(loc,hien).



```
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, -1 clauses
?- tall(thu).
false.

?- short(thu).
true.

?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, -2 clauses
?- parent(X,hien).
X = vien ;
X = loc.

?-
2019 SQL Server ...
```

```
testAI.txt - Notepad
File Edit Format View Help
parent(vien,hien).
parent(loc,hien).
```

### - Biến và hằng:

+ Các biến bắt đầu bằng chữ cái in hoa hoặc ký tự đặc biệt

Ví dụ: X, Socrates, \_result

+ Các hằng không bắt đầu bằng các chữ cái in hoa hoặc các ký tự đặc biệt

Ví dụ: x, socrates

### - Kiểu atom

+ Một atom có thể là:

+ Một chuỗi các ký tự được tạo thành từ các chữ hoa, chữ thường, chữ số và dấu gạch dưới và phải bắt đầu bằng một chữ thường

+ Một chuỗi các ký tự đặc biệt. Ví dụ @=, @==>, :-, ...

### - Kiểu số

+ Các số nguyên được sử dụng nhiều trong Prolog

+ Cú pháp của nó rất đơn giản: 365, -1000, 0, 1, ...

Các chú ý khi sử dụng SWI Prolog:

- Phân biệt chữ hoa và chữ thường là rất quan trọng

- Không được có khoảng trắng giữa tên và danh sách tham số của một cấu trúc

- Kết thúc một mệnh đề phải có dấu chấm ‘.’

## 4) Cách suy diễn trong Prolog

Prolog thực hiện suy diễn lùi

Giả sử chúng ta có cơ sở tri thức:

- love(vien,X):-female(X),rich(X).

- female(lan).

- female(thu).

- rich(thu).

Giả sử người dùng đặt câu hỏi:

- love(vien,X).

Quá trình suy diễn lùi của Prolog:

1. female(X) = female(lan), X = lan.

- rich(lan). (False)

2. female(X) = female(thu), X = thu.

- rich(thu). (True)

### III) Các ví dụ minh họa

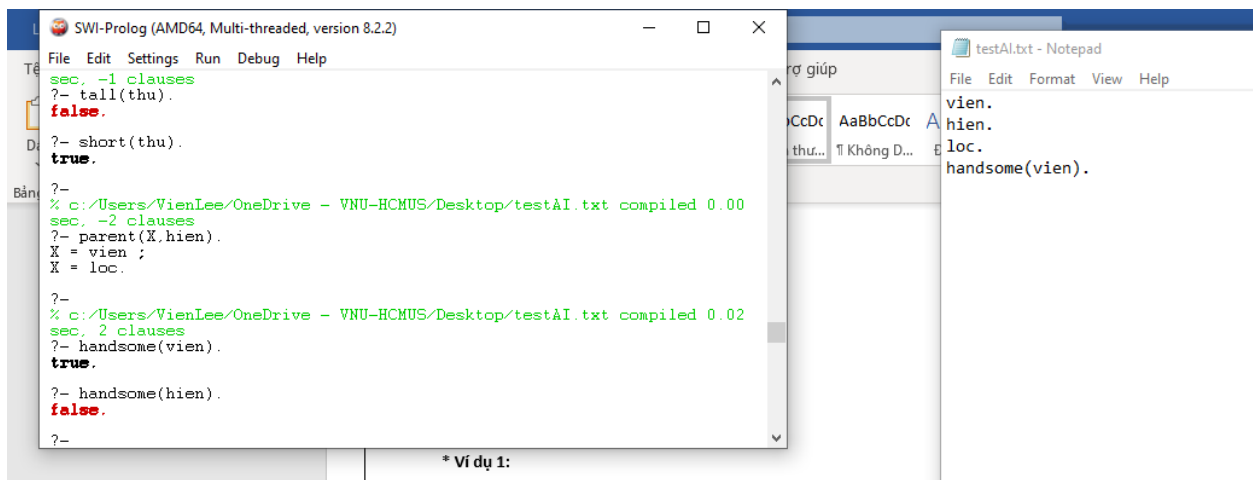
#### \* Ví dụ 1:

vien.

hien.

loc.

handsome(vien).



The screenshot shows two windows. The left window is titled 'SWI-Prolog (AMD64, Multi-threaded, version 8.2.2)' and contains the following text:

```
File Edit Settings Run Debug Help
sec, -1 clauses
?- tall(thu).
false.
?- short(thu).
true.
?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, -2 clauses
?- parent(X,hien).
X = vien ;
X = loc.
?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.02
sec, 2 clauses
?- handsome(vien).
true.
?- handsome(hien).
false.
?-
```

The right window is titled 'testAI.txt - Notepad' and contains the following text:

```
File Edit Format View Help
vien.
hien.
loc.
handsome(vien).
```

Below the windows, the text '\* Ví dụ 1:' is visible.

**\* Ví dụ 2:** Tính toán bằng Prolog:

Ta có công thức sau:

polynomial(X,Y):- X is Y+10.

The screenshot shows the SWI-Prolog environment with the following code in testAI.txt:

```
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, -2 clauses
?- parent(X,hien).
X = vien ;
X = loc.

?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.02
sec, 2 clauses
?- handsome(vien).
true.

?- handsome(hien).
false.

?- polynomial(X,3).
ERROR: Unknown procedure: polynomial/2 (DWIM could not correct goal)
?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, -3 clauses
?- polynomial(X,3).
X = 13.

?-
```

The Notepad window shows the rule definition:

```
polynomial(X,Y):- X is Y+10.
```

**\* Ví dụ 3:** Vẫn là tính toán nhưng công thức phức tạp hơn:

polynomial(X,Y):- X is Y\*4/2.

The screenshot shows the SWI-Prolog environment with the following code in testAI.txt:

```
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, -3 clauses
?- polynomial(X,3).
X = 13.

?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.02
sec, 0 clauses
?- polynomial(X,3,Z).
false.

?- polynomial(X,3,2).
false.

?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, 0 clauses
?- polynomial(X,2).
X = 4.

?-
```

The Notepad window shows the rule definition:

```
polynomial(X,Y):- X is Y*4/2.
```



**\* Ví dụ 4:** Cho cơ sở tri thức sau :

married(vien,thu).

parent(vien,hien).

parent(thu,hien).

parent(hien,vi).

parent(hien,toan).

grandParent(X,Y):-parent(Z,Y),parent(X,Z).

```
false.
?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, 0 clauses
?- polynomial(X,2).
X = 4.
?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, 5 clauses
?- grandParent(vien,X).
X = vi ;
X = toan.
?- grandParent(thu,X).
X = vi ;
X = toan.
?-
married(vien,thu).
parent(vien,hien).
parent(thu,hien).
```

```
thu
testAI.txt - Notepad
File Edit Format View Help
married(vien,thu).
parent(vien,hien).
parent(thu,hien).
parent(hien,vi).
parent(hien,toan).
grandParent(X,Y):-parent(Z,Y),parent(X,Z).
```

**\* Ví dụ 5:** Cho cơ sở tri thức:

vien.

hien.

sleep(hien).

dream(X):-sleep(X).

```
lần
v
ig
true
Unknown action: d (h for help)
Action? ;
true.
?- dream(vien).
true.
?- sleep(vien).
true.
?-
% c:/Users/VienLee/OneDrive - VNU-HCMUS/Desktop/testAI.txt compiled 0.00
sec, -1 clauses
?- dream(hien).
true.
?- dream(vien).
false.
?-
dream(X):-sleep(X).
```

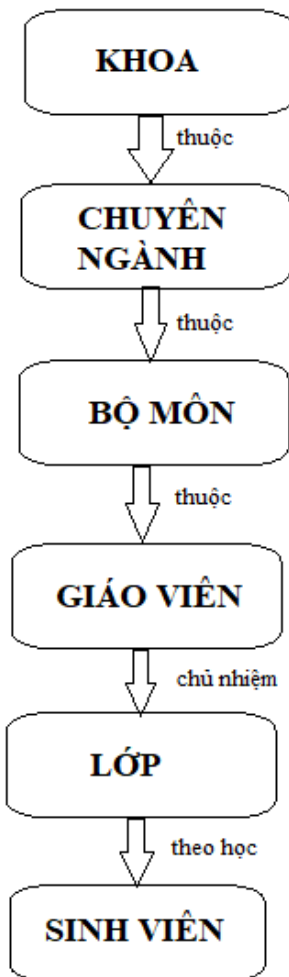
```
testAI.txt - Notepad
File Edit Format View Help
vien.
hien.
sleep(hien).
dream(X):-sleep(X).
```

## C) XÂY DỰNG CƠ SỞ TRI THỨC VỚI SWI-PROLOG

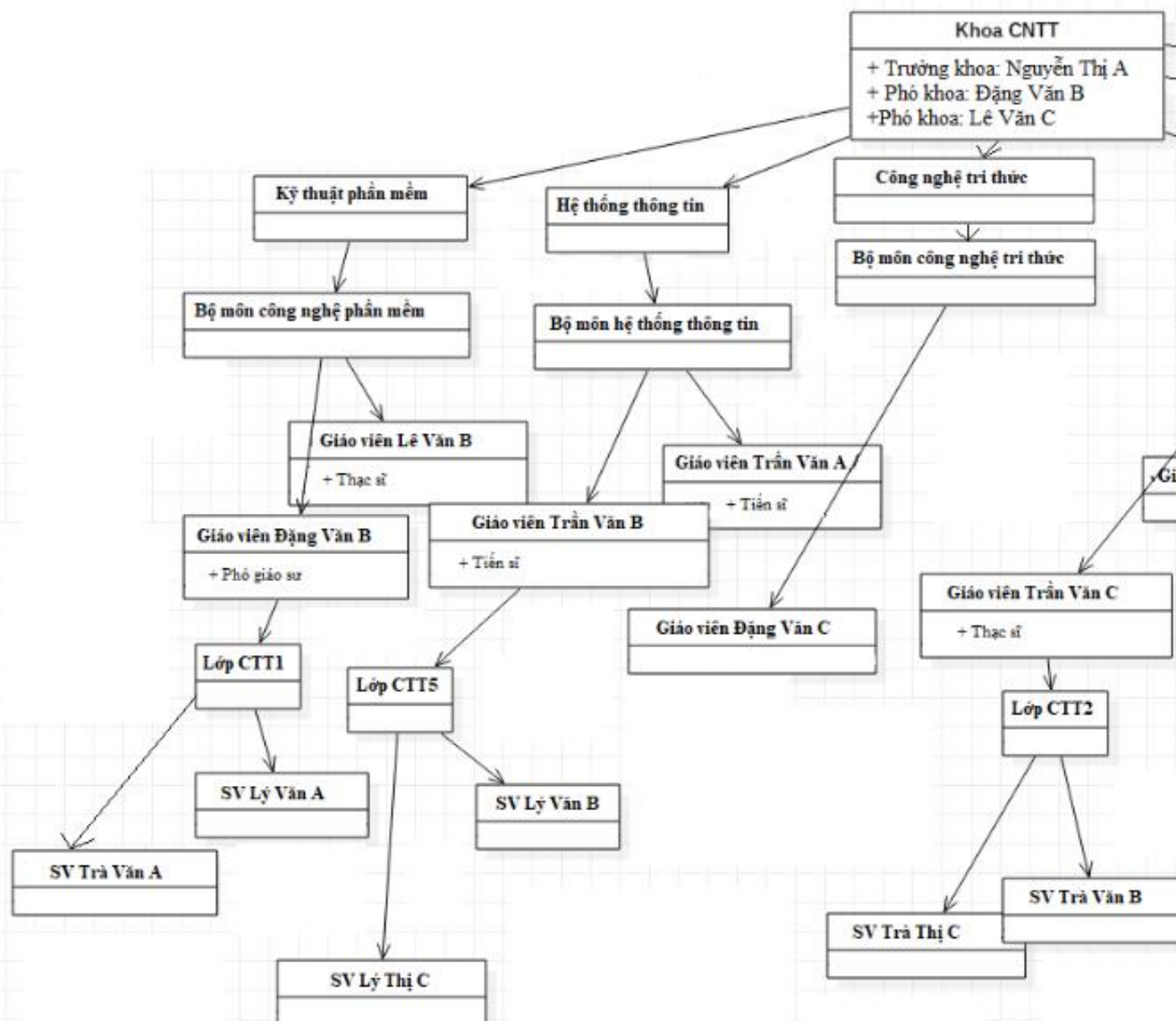
### I) Xây dựng cơ sở tri thức

#### CƠ SỞ DỮ LIỆU KHOA CÔNG NGHỆ THÔNG TIN TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

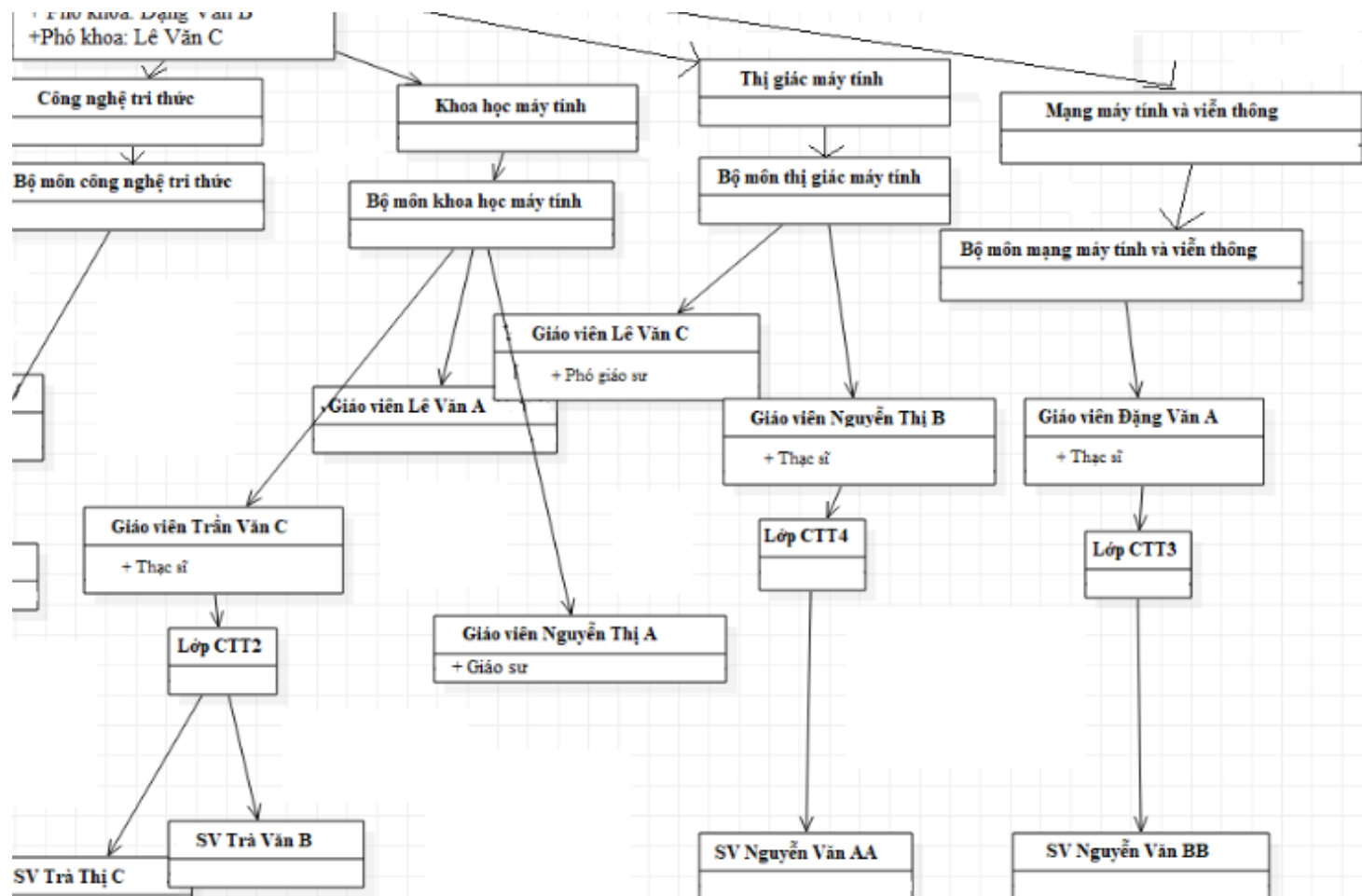
##### 1) Sơ đồ quan hệ



2) Sơ đồ quan hệ chi tiết  
Phần 1:



## Phần 2:



Activate Windows

## II) Vài ví dụ về truy vấn của cơ sở tri thức

### \* Ví dụ 1

5/ Khoa có các lớp nào?

```
atabase.pl cor
?- lop(X).
X = ctt1 ;
X = ctt2 ;
X = ctt3 ;
X = ctt4 ;
X = ctt5.
```

### \* Ví dụ 2

9/ Các giáo sư của khoa?

```
?- giaoSu(X).
X = nguyenThiA.
```

### \* Ví dụ 3

13/ Khoa có các chuyên ngành nào?

```
?- chuyenNganh(X).
X = kyThuatPhanMem ;
X = heThongThongTin ;
X = congNgheTriThuc ;
X = khoaHocMayTinh ;
X = thiGiacMayTinh ;
X = mangMayTinhVaVienThong.

?- ■
```

### \* Ví dụ 4

## 18/ Các tiến sĩ của Ngành Hệ Thống Thông Tin.

```
?- tienSiCuaNganh(X,heThongThongTin).  
X = tranVanA ;  
X = tranVanB ;  
false.  
~
```

### \* Ví dụ 5

## 20/ Phó khoa là ai?

```
?- phoKhoa(X).  
X = dangVanB ;  
X = leVanC.
```

```
?- ;■
```

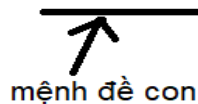
## D) CÀI ĐẶT HỆ THỐNG NGÔN LOGIC BẰNG NGÔN NGỮ LẬP TRÌNH

Ngôn ngữ lập trình sử dụng: C/C++

1. Phương pháp được áp dụng trong :
  - Sử dụng phương pháp suy diễn lùi.
    - \* Chuyển các mệnh đề về dạng chuẩn CNF.
    - \* Cú pháp biểu diễn theo chuẩn của ngôn ngữ Prolog.
2. Sơ lược các bước thực hiện:
  - Đọc file pl với cú pháp biểu diễn theo chuẩn của ngôn ngữ Prolog (nhập vào các sự kiện và luật vào cơ sở tri thức)
  - Chuyển mệnh đề trong KB theo dạng chuẩn CNF.
  - Nhập mệnh đề hỏi vào.

### DẠNG HỘI CHUẨN CNF

$(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$

  
mệnh đề con

- Phân tích lần lượt từng mệnh đề con nối rời nhau -> Từng sự kiện với mỗi mệnh đề con->
  - o Nếu sự kiện đúng ta lưu các biến tương ứng và pop\_back() sự kiện
  - o Nếu sự kiện sai thì pop\_back() sự kiện
  - o Nếu sự kiện tìm thấy trong luật thì phân giải theo luật thành các mệnh đề con mới (dạng chuẩn CNF) sau đó áp dụng phân phối or vào and thêm vào stack các mệnh đề con lớn.

+ Nếu 1 mệnh đề con không có tìm thấy sự kiện nào trong KB thì mệnh đề hỏi sai

Lập và giao các biến được lưu ở các mệnh đề con ta thu được kết quả truy vấn cuối cùng.

Kiểm chứng kết quả của hệ thống tự cài đặt với kết quả của Prolog :

Cây phả hệ của Hoàng gia Anh

```
C:\Users\root\source\repos\Project2\Debug\Project2.exe
husband(p_Charles,p_Diana).
> Chung minh = False !
=====

husband(p_Charles,c_p_Bowles).
Chung minh = True !
=====

mother(X,p_Andrew).
=====
X = q_ElizabethII
Chung minh = True !
=====

daughter(i_Phillips,a_Kelly).
=====
Chung minh = True !
=====

grandchild(X,p_Charles).
=====
X = p_George
X = p_Charlotte
Chung minh = True !
=====

grandson(X,p_Charles).
Chung minh = False !
=====

grandchild(X,q_ElizabethII).
=====
X = p_William
X = p_Harry
X = peter_Phillips
X = z_Phillips
X = p_Beatrice
X = p_Eugenie
X = j_v_Severn
X = l_l_m_Windsor
Chung minh = True !
=====
Activate Windows
```



## Cơ sở tri thức tự thu thập.

```
C:\Users\root\source\repos\Project2\Debug\Project2.exe

chuyenNganh(X).
=====
X = kyThuatPhanMem
X = heThongThongTin
X = congNgheTriThuc
X = khoaHocMayTinh
X = thiGiacMayTinh
X = mangMayTinhVaVienThong
Chung minh = True !
=====

sinhVienCuaLop(X,ctt1).
=====
X = lyVanA
X = traVanA
Chung minh = True !
=====

giaovienThuocBoMon(X,thiGiacMayTinh_bm).
=====
X = leVanC
X = nguyenThiB
Chung minh = True !
=====

boMonCuaNganh(X,khoaHocMayTinh).
=====
X = khoaHocMayTinh_bm
Chung minh = True !
=====

sinhVienTheoChuyenNganh(X,kyThuatPhanMem).
Chung minh = False !
=====

sinhVienCuaLop(traVanA,X).
=====
X = ctt1
Chung minh = True !
=====

giaoSuu(X).
=====
X = nguyenThiA
Chung minh = True !
```

Activate Windows  
Go to Settings to activate Windows.