

# BÁO CÁO ĐỒ ÁN 3

## LOGIC BẬC NHẤT

Môn học: Cơ sở trí tuệ nhân tạo

Lớp: Cử nhân tài năng

Thực hiện: Hoàng Thiên Nữ - Dương Nguyễn Thái Bảo



Tháng 11/2018

## I. Thông tin chung

### 1. Thông tin thành viên

MSSV	Họ và tên
1612880	Hoàng Thiên Nữ
1612840	Dương Nguyễn Thái Bảo

### 2. Phân công và đánh giá mức độ hoàn thành

STT	Công việc	Người thực hiện	Mức độ hoàn thành
1	Yêu cầu 1	Hoàng Thiên Nữ	100 %
2	Yêu cầu 2	Dương Nguyễn Thái Bảo	100 %
3	Code suy diễn tiến, lùi	Dương Nguyễn Thái Bảo	90 %
4	Code hợp giải	Hoàng Thiên Nữ	70 %

## II. Yêu cầu 1

### 1. Ngôn ngữ Prolog

#### a. Giới thiệu

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 theo logic". Xuất hiện từ năm 1972 (do Alain Colmerauer và Robert Kowalski thiết kế), mục tiêu của Prolog là 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.

Prolog được sử dụng nhiều trong các ứng dụng của trí tuệ nhân tạo và ngôn ngữ học trong khoa học máy tính. Đặc biệt, nó được sử dụng nhiều trong ngành xử lý ngôn ngữ tự nhiên (Natural Language Processing), vì đây là mục đích thiết kế ban đầu. Cú pháp và ngữ nghĩa của Prolog đơn giản và sáng sủa, nó được người Nhật xem là một trong những nền tảng để xây dựng máy tính thế hệ thứ 5 mà ở đó, thay vì phải mô tả cách giải quyết một bài trên máy tính, con người chỉ cần mô tả bài toán và máy tính sẽ hỗ trợ họ những việc còn lại.

#### b. Cú pháp

Chương trình Prolog là tập các mệnh đề xác định được viết theo kí hiệu có phần hơi khác với dạng bậc nhất chuẩn. Ở đây, ta có một số các kí hiệu chính được sử dụng trong Prolog như sau:

Ý nghĩa	Kí hiệu trong Prolog
AND	, (dấu phẩy)
OR	; (chấm phẩy)
NOT	Not
IF (suy ra)	:- (hai chấm và dấu gạch)
Kết thúc	. (chấm)

Chữ cái viết hoa	Biến
Chữ cái viết thường	Hằng

Các mệnh đề được biểu diễn dưới dạng mệnh đề Horn. Một mệnh đề Horn có dạng:

*Head:-Body.*

Trong đó:

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

live(X) :- not die(X).

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). Ngược lại, 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.

Với việc định nghĩa các mệnh đề như trên, ta có thể biểu diễn logic bậc nhất trong Prolog,

### Câu nguyên tố (Atoms)

Câu nguyên tố được hình thành từ một kí hiệu vị từ theo sau bởi danh sách đối số là các biểu thức logic. Câu nguyên tố sẽ chỉ đến một sự kiện (Facts).

Câu nguyên tố sẽ được biểu diễn bởi những mệnh đề Horn mà phần Body là rỗng.

Ví dụ:

likes(john, jenny).

cat(tom).

distance(hanoi, hcm, 2000).

Như ở các ví dụ trên, ta thông báo rằng:

- John thích Jenny.
- Tom là một con mèo.
- Khoảng cách từ Hà Nội đến Thành phố Hồ Chí Minh là 2000 km.

### Luật (Rules)

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 1:

dating(X, Y) :-

likes(X, Y), likes(Y, X).

Nếu hai người X và Y hẹn hò thì hai người đó phải thích nhau (trong trường hợp lí tưởng, bỏ qua các điều kiện xã hội).

Ví dụ 2:

```
light_is_on(X):-
    turn_on(X), electricity.
```

Nếu công tắc của đèn X sáng và tại đó có điện thì đèn X sẽ sáng.

### c. Truy vấn

### d. Cấu trúc dữ liệu

Prolog hỗ trợ kí hiệu danh sách và đại số. Danh sách trong Prolog có những tính chất sau:

- Ta có thể khai báo một danh sách với bằng cách đặt các phần tử bên trong ngoặc vuông, và chúng được cách nhau bởi dấu phẩy. Một danh sách có thể rỗng.
- Các phần tử của danh sách có thể khác nhau: một biến, một câu nguyên tố, hàm, ...
- Một danh sách này có thể chứa danh sách khác.

Ví dụ:

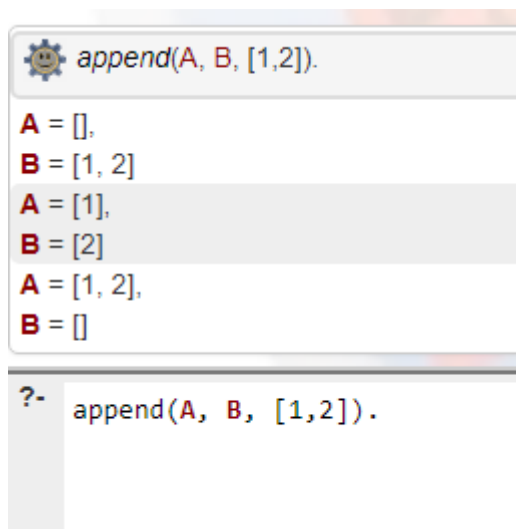
[]	Danh sách rỗng
[lan, xuan, anh]	Danh sách gồm 3 phần tử lan, xuan và anh
[mia, robber(honey_bunny), X, 2, mia]	Danh sách chứa các phần tử loại khác nhau
[mia, [vincent, jules], [butch, girlfriend(butch)]]	Danh sách có thể chứa danh sách khác.

Prolog hỗ trợ append cho danh sách. `append(X, Y, Z)` sẽ thành công nếu danh sách Z là kết quả nối của danh sách X và Y, `append` được định nghĩa trong Prolog như sau:

```
append([], Y, Y).
append([A|X], Y, [A|Z]) :- append(X, Y, Z).
```

Một số ví dụ sử dụng `append`:

Truy vấn `append(A, B, [1,2])` liệt xem hai dách nào nối nhau ra [1,2].



Bằng cách định nghĩa hàm Prefix (P, L) để liệt kê các tiền tố của L như sau:

```
Prefix(P, L) :- append(P, _, L).
```

Ta có được kết quả:

```
prefix(X,[a,b,c,d]).
X = []
X = [a]
X = [a, b]
X = [a, b, c]
X = [a, b, c, d]
?- prefix(X,[a,b,c,d]).
```

Tương tự với suffix:

```
Suffix(S, L) :- append(_, S, L).
```

Kết quả:

```
suffix(X,[a,b,c,d]).
X = [a, b, c, d]
X = [b, c, d]
X = [c, d]
X = [d]
X = []
?- suffix(X,[a,b,c,d]).
```

Ngoài ra, Prolog còn hỗ trợ một số các phép toán số học.

Ví dụ	Thể hiện trong Prolog
$6 + 2 = 8$	<code>8 is 6 + 2</code>
$6 * 2 = 12$	<code>12 is 6 * 2</code>
$6 - 2 = 4$	<code>4 is 6 - 2</code>
$6 - 8 = -2$	<code>-2 is 6 - 8</code>
$6 / 2 = 3$	<code>3 is 6 / 2</code>
$7 / 2 = 3$	<code>3 is 7 / 2</code>
1 là phần dư khi chia 7 cho 2	<code>1 is mod(7, 2)</code>

### e. Nguyên tắc hoạt động

Chương trình Prolog được thực thi thông qua suy diễn lùi theo chiều sâu, các mệnh đề được thử theo thứ tự đã khai báo trong cơ sở tri thức.

## 2. Môi trường lập trình Prolog – SWI-Prolog

### a. Cách cài đặt

SWI-Prolog có các bản cài đặt trên cả 3 hệ điều hành chính: Window, Mac, Linux.

Đối với bản Window và Mac chỉ cần tải file cài đặt tại link:

<http://www.swi-prolog.org/download/stable>

Sau đó cài đặt theo hướng dẫn.

Dưới đây là hướng dẫn cài đặt SWI-Prolog cho hệ điều hành Linux.

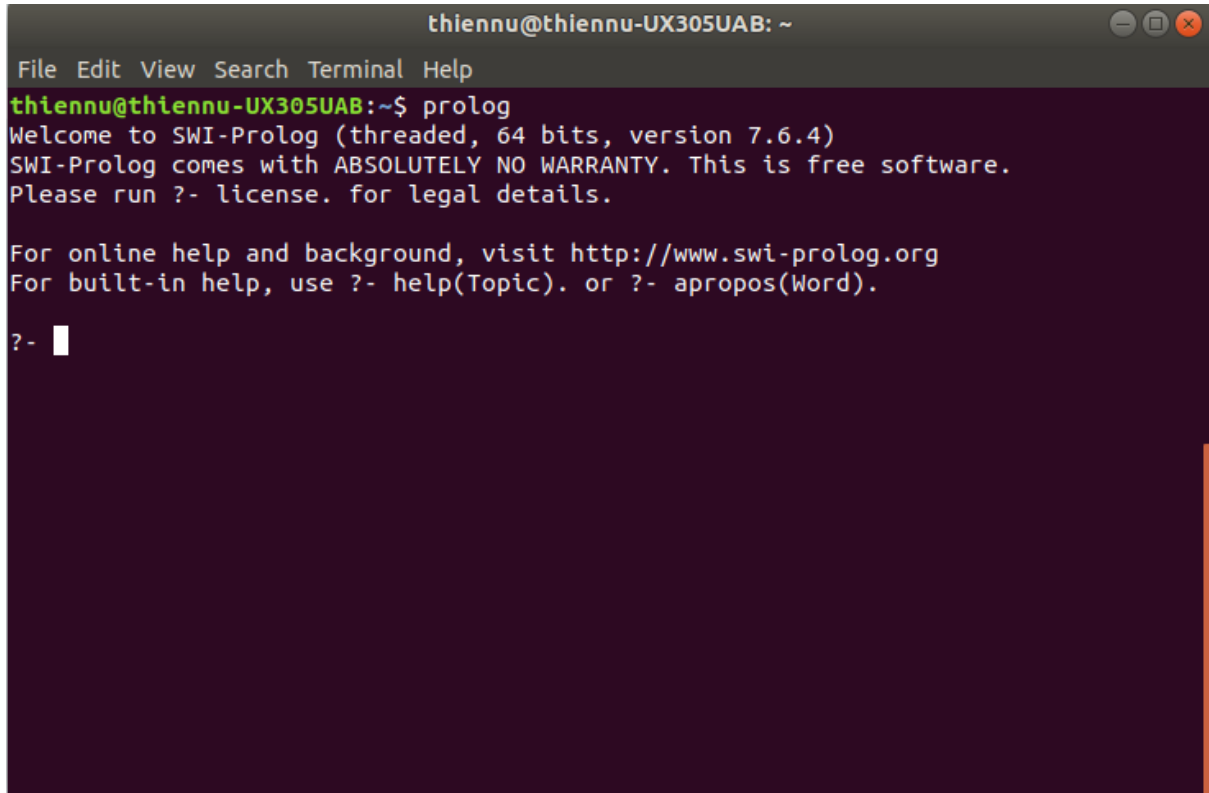
1. Cần thêm **ppa:swi-prolog/stable** vào nguồn phần mềm của hệ thống:

```
$ sudo add-apt-repository ppa:swi-prolog/stable  
$ sudo apt-get update
```

2. Cài đặt

```
$ sudo apt-get install swi-prolog
```

Sau khi cài đặt ta chạy lệnh prolog và được kết quả như sau:

A terminal window titled 'thiennu@thiennu-UX305UAB: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is 'thiennu@thiennu-UX305UAB:~\$' and the command 'prolog' has been entered. The output shows the SWI-Prolog welcome message: 'Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4). SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software. Please run ?- license. for legal details.' It also provides links for online help and background, and instructions for built-in help. The prompt is now '?- ' with a cursor.

```
thiennu@thiennu-UX305UAB: ~
File Edit View Search Terminal Help
thiennu@thiennu-UX305UAB:~$ prolog
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

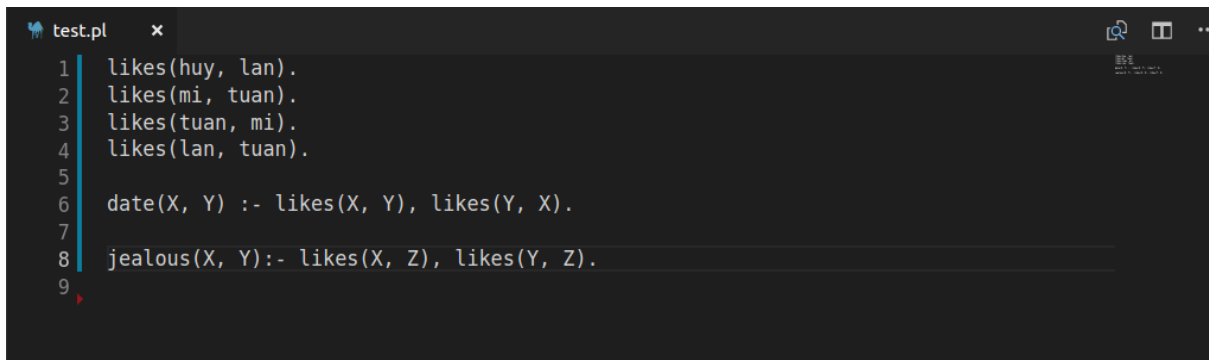
For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- 
```

## b. Chạy một chương trình Prolog trên Linux

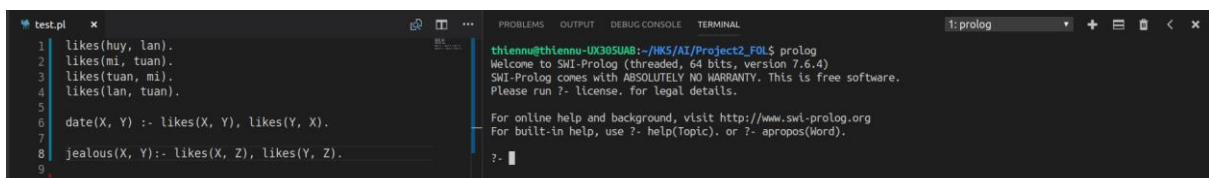
Bước 1: Tạo file script có dạng “filename.pl”.

File này lưu các cơ sở tri thức mà ta tạo nên.

A code editor window titled 'test.pl' showing a Prolog script. The script contains four facts, two rules, and a query. The facts are 'likes(huy, lan).', 'likes(mi, tuan).', 'likes(tuan, mi).', and 'likes(lan, tuan)'. The first rule is 'date(X, Y) :- likes(X, Y), likes(Y, X).' and the second rule is 'jealous(X, Y):- likes(X, Z), likes(Y, Z).' The query is 'jealous(X, Y):- likes(X, Z), likes(Y, Z).' The script is numbered 1 through 9.

```
test.pl
1 likes(huy, lan).
2 likes(mi, tuan).
3 likes(tuan, mi).
4 likes(lan, tuan).
5
6 date(X, Y) :- likes(X, Y), likes(Y, X).
7
8 jealous(X, Y):- likes(X, Z), likes(Y, Z).
9
```

Bước 2: Mở prolog trên terminal bằng lệnh: `$ prolog`.

A terminal window titled 'thiennu@thiennu-UX305UAB: ~/KS/AI/Project2\_F01\$' showing the Prolog environment. The prompt is 'thiennu@thiennu-UX305UAB:~/KS/AI/Project2\_F01\$' and the command 'prolog' has been entered. The output shows the SWI-Prolog welcome message. The prompt is now '?- ' with a cursor.

```
thiennu@thiennu-UX305UAB: ~/KS/AI/Project2_F01$ prolog
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- 
```

Bước 3: Thêm tập dữ liệu cơ sở tri thức: `['filename.pl']`.

Prolog sẽ xây dựng tập cơ sở tri thức dựa trên suy diễn lùi.

```

test.pl
1 likes(huy, lan).
2 likes(mi, tuan).
3 likes(tuan, mi).
4 likes(lan, tuan).
5
6 date(X, Y) :- likes(X, Y), likes(Y, X).
7
8 jealous(X, Y) :- likes(X, Z), likes(Y, Z).
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

thiennu@thiennu-UX305UAB:~/HKS/AI/Project2_FOL$ prolog
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['test.pl'].
true.
    
```

Bước 4: Đặt câu hỏi với cơ sở tri thức đã xây dựng.

```

thiennu@thiennu-UX305UAB:~/HKS/AI/Project2_FOL$ prolog
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['test.pl'].
true.

?- likes(mi, tuan).
true.

?- date(X, Y).
X = mi,
Y = tuan ;
X = tuan,
Y = mi ;
false.

?- jealous(lan, mi).
true.
    
```

## c. Một số ví dụ

**Ví dụ 1: Mối quan hệ yêu đương.**

- likes(X, Y): Thể hiện X thích Y.
- date(X, Y): 2 người hẹn hò với nhau thì 2 người đó sẽ thích nhau.
- jealous(X, Y): 2 người ghen tị với nhau nếu họ thích cùng một người.

```

thiennu@thiennu-UX305UAB:~/HKS/AI/Project2_FOL$ prolog
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['test.pl'].
true.

?- likes(mi, tuan).
true.

?- date(X, Y).
X = mi,
Y = tuan ;
X = tuan,
Y = mi ;
false.

?- jealous(lan, mi).
true.
    
```

**Ví dụ 2: Các mối quan hệ trên list.**

Đây là ví dụ sử dụng cấu trúc list của Prolog. Ta định nghĩa các quan hệ sau:

- Last(X, List): X là phần tử cuối cùng của List. Quan hệ được định nghĩa với dạng đệ quy, với base case là nếu List chỉ có 1 phần tử X thì X là phần tử cuối cùng.
- Nextto(X, Y, List): kiểm tra 2 phần tử có đứng cạnh nhau trong list không.
- Reverse(List1, List2): List1 là nghịch đảo của List2.
- Sublist(X, List): X là list con của List.
- Prefix(X, List): X là tiền tố của List.



```
test.pl x
12 /* Example 2 */
13 /* last */
14 /* last(X, List) -> "X is the last element in List" */
15 last(X, [X]).
16 last(X, [_|Y]) :- last(X, Y).
17
18 /* nextto */
19 /* nextto(X,Y,List) -> "X and Y are consecutive elements in List" */
20 nextto(X,Y,[X,Y|_]).
21 nextto(X,Y,[_|Z]) :- nextto(X,Y,Z).
22
23 /* reverse */
24 /* reverse(List1, List2) -> "List2 is the reversing order of List1" */
25 reverse([], []).
26 reverse([H|T], List) :- reverse(T, Z), append(Z, [H], List).
27
28 /* sublist */
29 /* sublist(S, List) -> "S is a sublist of List which appears consecutively,
30 /* and in the same order"
31 sublist([X|L], [X|M]) :- prefix(L, M).
32 sublist(L, [_|M]) :- sublist(L, M).
33
34 /* prefix */
35 /* prefix(X, List) -> "X is prefix of List" */
36 prefix([], _).
37 prefix([X|L], [X|M]) :- prefix(L, M).
38
```

**Ví dụ 3: Giải bài toán 8 hâu.**

Có thể giải bài toán 8 hậu bằng cách áp dụng đệ quy trong prolog.

```

/* Example 3 */
sel(X, [X|Y], Y).
sel(U, [X|Y], [X|Y]) :- sel(U,Y,Y).

safe([ ]).
safe([X|Y]) :- check(X,Y), safe(Y).

check(_, [ ]).
check(P, [Q|R]) :-
    not_on_diag(P,Q), check(P,R).

not_on_diag(P(X1,Y1),P(X2,Y2)) :-
    DX is X1-X2, DY is Y1-Y2,
    MDY is Y2-Y1, DX=+DY, DX=-MDY.

queens(Rows, [Col|RestCols], Points) :-
    sel(Row,Rows,RestRows),
    safe([P(Row,Col) | Points]),
    queens(RestRows,RestCols,
        [P(Row,Col) | Points]).

queens([ ], [ ], Points) :-
    print('Solution: '),print(Points),nl.

```

**Ví dụ 4: Tính giai thừa bằng đệ quy trong Prolog.**

Định nghĩa tính giai thừa bằng đệ quy.

Quan hệ  $\text{fact}(N, R): R = N!$

```
62 |  
63 |  
64 | /* Example 4 */  
65 | fact(0,1).  
66 | fact(N,R):- fact(N1,R1),N is N1 + 1,R is R1 * N.  
67 |  
68 |
```

**Ví dụ 5: Đồ thị trong Prolog.**

Ta có thể dùng Prolog để xây dựng 1 đồ thị vô hướng với các quan hệ sau:

- $\text{Edge}(u, v)$ : cạnh  $(u, v)$ .
- $\text{Connected}(u, v)$ : có cạnh nối trực tiếp từ  $u$  đến  $v$ .
- $\text{Path}(A, B, \text{Path})$ : Path là đường đi từ  $A$  đến  $B$ .
- $\text{Travel}(A, B, \text{Visited}, \text{Path})$ : Đi từ  $A$  đến  $B$  bằng thuật toán DFS.

```

82
83 /* Example 5 */
84 edge(1,2).
85 edge(1,4).
86 edge(1,3).
87 edge(2,3).
88 edge(2,5).
89 edge(3,4).
90 edge(3,5).
91 edge(4,5).
92
93 connected(X,Y) :- edge(X,Y) ; edge(Y,X).
94 path(A,B,Path) :-
95     travel(A,B,[A],0),
96     reverse(0,Path).
97
98 travel(A,B,P,[B|P]) :-
99     connected(A,B),
100     travel(A,B,Visited,Path) :-
101         connected(A,C),
102         C \== B,
103         \member(C,Visited),
104         travel(C,B,[C|Visited],Path).
105
thiennuthienmu-UX385UAB-:/HS/AI/Project2_FPL$ prolog
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- ['test.pl'].
true.

?- path(1, 5, P).
P = [1, 2, 5] ;
P = [1, 2, 3, 5] ;
P = [1, 2, 3, 4, 5] ;
P = [1, 4, 5] ;
P = [1, 4, 3, 5] ;
P = [1, 4, 3, 2, 5] ;
P = [1, 3, 5] ;
P = [1, 3, 4, 5] ;
P = [1, 3, 2, 5] ;
false.

?-
    
```

### 3. Bài tập cây phả hệ cho Hoàng gia Anh

Code Prolog được đính kèm vào file nộp.

#### Bộ câu hỏi truy vấn

1. Liệt kê những người có giới tính là nam?

```

?- male(X).
X = phillip ;
X = charles ;
X = mark_phillips ;
X = timothy_laurence ;
X = andrew ;
X = edward ;
X = william ;
X = harry ;
X = peter_phillips ;
X = mike ;
X = james ;
X = george.
    
```

2. Harry có phải là con của Charles không?

```

?- child(harry, charles).
true.
    
```

3. Cha của Isla phillips là ai?

```

?- father(X, isla).
X = peter_phillips.
    
```

4. Diana và Charles có phải là vợ chồng không?

```

?- married(diana, charles).
false.
    
```

5. Chồng của Anne là ai?

```

?- husband(X, anne).
X = timothy_laurence.
    
```

6. Edward đã ly dị với ai?

```

?- divorced(edward, X).
false.
    
```

7. Liệt kê các con trai của Elizabeth?

```

?- findall(X, son(X, elizabeth), Son).
Son = [charles, andrew, edward].
    
```

8. Con của William là ai?

```

?- findall(X, child(X, william), Children).
Children = [george, charlotte].
    
```

9. Các con gái của Autumn Kelly?

```
?- findall(X, daughter(X, autumn), Daughter).
Daughter = [savannah, isla].
```

10. Liệt kê các cháu của Anne?

```
?- findall(X, grandchild(X, anne), GrandChildren).
GrandChildren = [savannah, isla, miagrace].
```

11. Ai là ông của James?

```
?- grandfather(X, james).
X = phillip ;
false.
```

12. Ai là bà của Mia Grace?

```
?- grandmother(X, miagrace).
X = anne .
```

13. Dì của Savannah là?

```
?- aunt(X, savannah).
X = zara .
```

14. Anh em ruột của Andrew?

```
?- findall(X, sibling(andrew, X), Siblings).
Siblings = [charles, anne, andrew, edward].
```

15. Eugene là dì của ai?

```
?- findall(X, aunt(eugene, X), NieceNephew).
NieceNephew = [].
```

16. Beatrice và James có phair quan hệ dì cháu không?

```
?- aunt(beatrice, james).
false.
```

17. Ai vừa là cậu của William và Peter Phillips?

```
?- uncle(X, william), uncle(X, peter_phillips).
X = andrew ;
X = edward ;
false.
```

18. Cháu gái của Phillip mà còn độc thân?

```
?- findall(X, (granddaughter(X, phillip), not(married(X, Y))), FA_granddaughter).
FA_granddaughter = [beatrice, eugene, louise_mountbatten].
```

19. Mia Grace Tindall và George có cùng ông hay không?

```
?- grandfather(X, miagrace), grandfather(X, george).
false.
```

20. Anh trai của Zara Phillips là ai?

```
?- findall(X, brother(X, zara), Brother).
Brother = [peter_phillips].
```

### III. Yêu cầu 2

Tóm tắt cơ sở tri thức về chủ đề nhà hàng:

Các đối tượng thuộc các tập cơ bản nhất là:

**Person**: người. Mỗi người có thể là một đầu bếp (**Chef**), người ăn chay (**Vegetarian**) hoặc người bình thường.

***Ingredient***: nguyên liệu. Nguyên liệu có thể có thuộc kiểu nguyên liệu ngọt (***Sweet***), mặn (***Salty***), chua (***Sour***), cay (***Spicy***), là thịt (***Meat***), là hải sản (***Seafood***), rau (***Vegetables***), trái cây (***Fruit***) hoặc nguyên liệu bình thường.

***Place***: nguồn gốc nguyên liệu.

***Drink***: đồ uống. Đồ uống có thể thuộc loại bình thường hoặc có cồn (***Alcoholic***).

***Dish***: món ăn.

***Event***: sự kiện/ loại buổi tiệc.

Các quan hệ cơ bản nhất trong chủ đề:

source(I, S): nguyên liệu I có nguồn gốc từ nơi S.

dish\_in\_event(D, E): món D được phục vụ trong buổi tiệc E.

ingredient\_in\_dish(I, D): nguyên liệu I được dùng trong món D.

drink\_in\_event(D, E): đồ uống D được phục vụ trong buổi tiệc E.

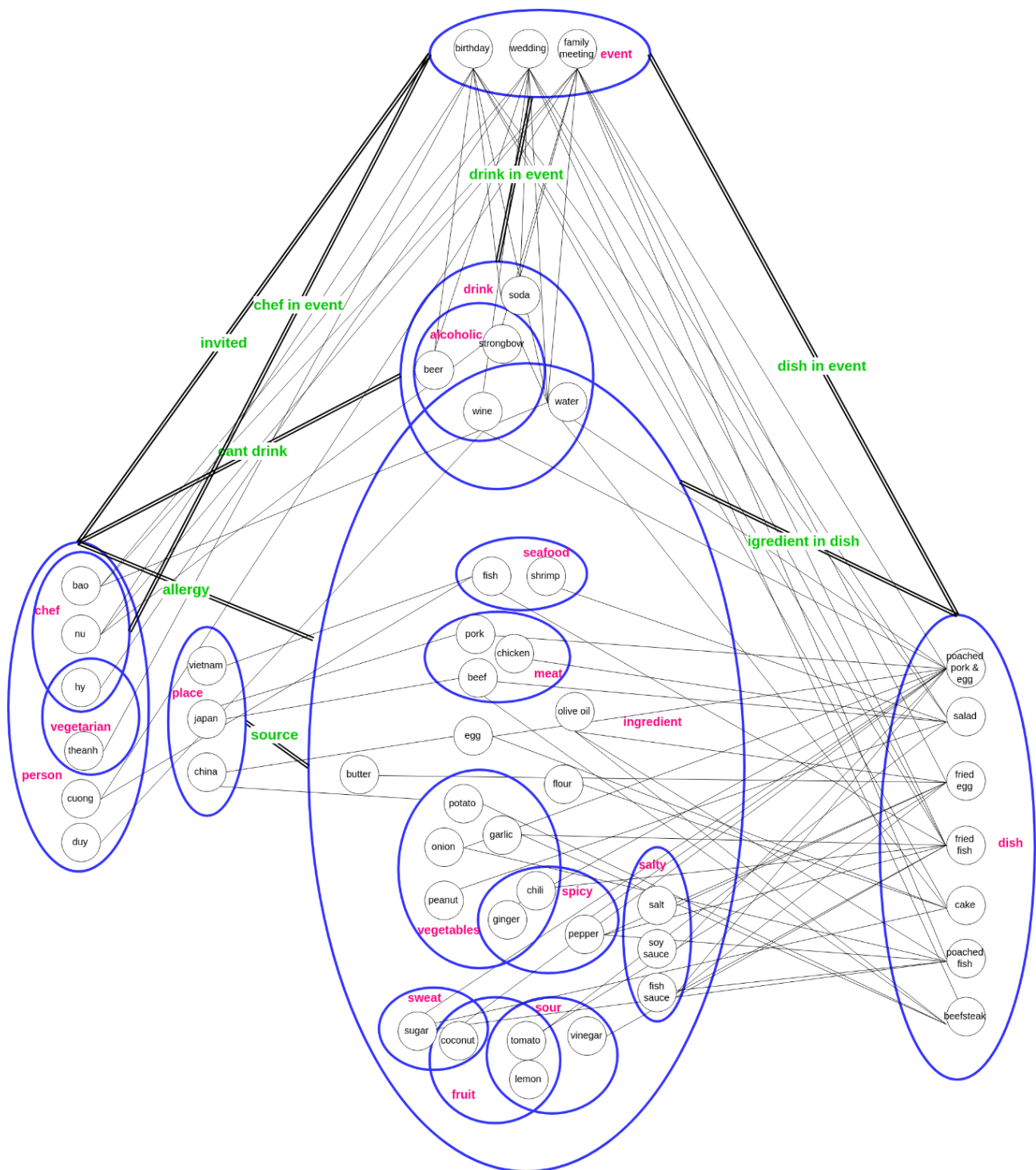
allergy(P, I): người P bị dị ứng với nguyên liệu I.

chef\_in\_event(C, E): đầu bếp C phụ trách buổi tiệc E.

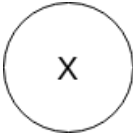
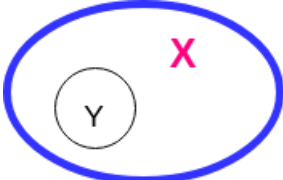
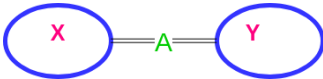
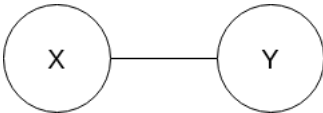
invited(E, P): người P được mời tới buổi tiệc E.

cant\_drink(P, D): người P không uống được loại đồ uống D.

Sơ đồ các đối tượng và quan hệ cơ bản:



*Chú thích:*

STT	Ký hiệu	Ý nghĩa
1		Đối tượng X.
2		Đối tượng Y trong tập hợp X.
3		Các phần tử trong tập X và Y nếu có quan hệ với nhau thì đó là quan hệ loại A.
4		Đối tượng X có quan hệ với đối tượng Y. Loại quan hệ được xác định bằng quan hệ của 2 tập hợp như chú thích [3].

### Các quan hệ nâng cao:

`ingredient_in_event(I, E)`: nguyên liệu I được dùng trong sự kiện E, khi có món D làm từ nguyên liệu I được phục vụ trong sự kiện E.

`cant_eat(P, D)`: người P không ăn được món D, khi có nguyên liệu I trong món D khiến người P bị dị ứng; hoặc người P ăn chay và món D có nguyên liệu làm từ thịt hoặc cá.

`cant_join(P, E)`: người P không thể tham gia sự kiện E, khi có món D trong sự kiện E mà người P không ăn được.

`not_fun_event(E)`: bữa tiệc E không vui, nếu có một người P được mời nhưng không thể tham gia hoặc không uống được đồ uống có cồn D (nếu có) trong bữa tiệc.

`same_ingredient_events(I, E1, E2)`: nguyên liệu I có được dùng trong cả 2 sự kiện E1 và E2 không.

`same_ingredients_event(I1, I2, E)`: nguyên liệu I1 và I2 có cùng được dùng trong sự kiện E hay không.

`same_ingredient_dishes(I, D1, D2)`: nguyên liệu I có được dùng chung trong 2 món D1 và D2 không.

same\_ingredients\_dish(I1, I2, D): nguyên liệu I1 và I2 có được dùng chung trong món D hay không.

same\_dish\_events(D, E1, E2): món D có được phục vụ trong cả 2 sự kiện E1 và E2 hay không.

same\_dishes\_event(D1, D2, E): món D1 và D2 có cùng được phục vụ trong sự kiện E hay không.

high\_grade\_ingredient(I): nguyên liệu I là cao cấp, nếu nó có nguồn gốc từ Nhật Bản.

high\_grade\_dish(D): món D là cao cấp, nếu nó có sử dụng 1 nguyên liệu cao cấp.

high\_grade\_event(E): sự kiện E là cao cấp, nếu nó phục vụ 1 món cao cấp.

same\_chef\_events(C, E1, E2): đầu bếp C có phụ trách cả 2 sự kiện E1 và E2 không.

same\_chefs\_event(C1, C2, E): 2 đầu bếp C1 và C2 có cùng phụ trách sự kiện E không.

same\_source\_ingredients(S, I1, I2): 2 nguyên liệu I1 và I2 có chung nguồn gốc là S không.

source\_in\_dish(S, D): món D có dùng nguyên liệu nào có nguồn gốc từ S không.

same\_source\_dishes(S, D1, D2): món D1 và D2 có dùng nguyên liệu cùng đến từ S không.

same\_sources\_dish(S1, S2, D): món D có dùng nguyên liệu đến từ cả S1 và S2 không.

source\_in\_event(S, E): sự kiện E có dùng nguyên liệu đến từ S không.

same\_source\_events(S, E1, E2): 2 sự kiện E1 và E2 có cùng dùng nguyên liệu đến từ S1 và S2 không.

same\_sources\_event(S1, S2, E): sự kiện E có dùng nguyên liệu đến từ S1 và S2 không.

### Bộ câu hỏi test cho cơ sở tri thức nhà hàng:

1. Cường có phải là đầu bếp?

```
?- chef(cuong).
false.
```

2. Đầu bếp có những ai?

```
?- findall(X, chef(X), Chef).
Chef = [bao, nu, hy].
```

3. Những nguyên liệu nào đến từ Trung Quốc?

```
?- findall(X, source(X, china), FromChina).
FromChina = [egg, potato, onion, chili, garlic, ginger, peanut].
```

4. Hy có bị dị ứng với “Hành” hay không?

```
?- allergy(hy, garlic).
false.
```

5. Ai là người ăn chay?

```
?- findall(X, vegetarian(X), Vegetarian).
Vegetarian = [hy, theanh].
```

6. Làm món trứng chiên thì cần những nguyên liệu gì?

```
?- findall(X, ingredient_in_dish(X, fried_egg), I).
I = [egg, salt, pepper, butter, fish_sauce, tomato].
```

7. Cá chiên có cần để thịt hay không?

```
?- ingredient_in_dish(meat(X), fried_fish).
false.
```

8. Có được uống rượu trong sinh nhật không?

```
?- drink_in_event(wine, birthday).
false.
```

9. Bao là đầu bếp trong những sự kiện nào?

```
?- findall(E, chef_in_event(bao, E), Event).
Event = [family_meeting, wedding].
```

10. Hy không thể ăn được những món nào?

```
?- findall(X, cant_eat(hy, X), FoodHy).
FoodHy = [fried_egg, poached_pork_and_egg, beefsteak, salad, salad, poached_fish, fried_fish, salad].
```

11. Tiệc đám cưới có vui không?

```
?- not_fun_event(wedding).
true .
```

12. Thế Anh có đến được buổi tiệc sinh nhật không?

```
?- cant_join(theanh, birthday).
false.
```

13. Người ăn chay thì không ăn được những thứ gì?

```
?- findall(X, (vegetarian(P), cant_eat(P, X)), V).
V = [fried_egg, poached_pork_and_egg, beefsteak, salad, salad, poached_fish, fried_fish, salad, poached_pork_and_egg...].
```

14. Liệt kê những món cao cấp (high\_grade\_dish)?

```
?- findall(D, high_grade_dish(D), Dish).
Dish = [beefsteak, salad].
```

15. Những nguyên liệu nào nấu được thịt kho trứng mà có nguồn gốc từ việt nam?

```
?- findall(X, (ingredient_in_dish(X, fried_egg), source(X, vietnam)), I).
I = [].
```

16. Đám cưới thì cần những nguyên liệu từ nguồn nào?



```
?- ingredient_in_event(I, wedding).
I = fish ;
I = olive_oil ;
I = garlic ;
I = fish_sauce ;
I = vinegar ;
I = chili ;
I = pepper ;
I = beef ;
I = shrimp ;
I = chicken ;
I = tomato ;
I = peanut ;
I = vinegar ;
I = beef ;
I = olive_oil ;
I = garlic ;
I = potato.
```

17. Nữ có uống được rượu không?

```
?- cant_drink(nu, wine).
false.
```

18. Bao và Nữ có làm đầu bếp trong cùng tiệc đám cưới không?

```
?- same_chefs_event(bao, nu, wedding).
false.
```

19. Những món ăn nào được sử dụng trong tiệc gia đình và đến từ Trung Quốc?

```
?- findall(D, (dish_in_event(D, family_meeting), source_in_dish(china, D)), Dish).
Dish = [fried_egg, fried_fish, fried_fish, poached_fish, poached_pork_and_egg, poached_pork_and_egg, poached_pork_and_egg, salad].
```

20. Những event mà cùng sử dụng nguồn nguyên liệu đến từ Việt Nam?

```
?- findall((E1, E2), (same_ingredient_events(I, E1, E2), source(I, vietnam), E1 \= E2), Event).
Event = [(birthday, family_meeting), (birthday, wedding), (birthday, family_meeting), (family_meeting, birthday), (family_meeting, wedding), (wedding, birthday), (wedding, family_meeting), (wedding, family_meeting), (...)]...
```

## IV. Yêu cầu 3

### 1. Kịch bản chương trình

1. Để chạy toàn bộ chương trình, chạy file “main.py”:

```
thiennu@thiennu-UX305UAB:~/HK5/AI/Project2_FOL$ python3 main.py
Enter file input of knowledge base: familytree.pl
0. Exit
1. Forward
2. Backward
3. Resolution
Which do you want to choose? 1
?-
```

2. Chương trình sẽ yêu cầu nhập file để nạp cơ sở tri thức.

4. Tiếp đó là chọn phương thức suy diễn. Các phương thức suy diễn có cú pháp tương tự Prolog, tuy nhiên không thể đặt câu hỏi dưới dạng các phép toán and. Kết quả trả về là danh sách các biến có thể thay thế được vào câu trả lời, nếu không có biến thì sẽ trả về True/False.
5. Để kết thúc dùng lệnh “halt”.

## 2. Demo các thuật toán suy diễn

### Thuật toán forward:

```
thiennu@thiennu-UX305UAB:~/HK5/AI/Project2_FOL$ python3 main.py
Enter file input of knowledge base: familytree.pl
0. Exit
1. Forward
2. Backward
3. Resolution
Which do you want to choose? 1
?- male(X).
{'X': 'mike'}
{'X': 'mark_phillips'}
{'X': 'william'}
{'X': 'timothy_laurence'}
{'X': 'charles'}
{'X': 'james'}
{'X': 'peter_phillips'}
{'X': 'george'}
{'X': 'harry'}
{'X': 'andrew'}
{'X': 'phillip'}
{'X': 'edward'}
True
?- grandchild(X, elizabeth).
{'X': 'eugene'}
{'X': 'zara'}
{'X': 'william'}
{'X': 'james'}
{'X': 'beatrice'}
{'X': 'louise_mountbatten'}
{'X': 'harry'}
{'X': 'peter_phillips'}
True
?- husband(charles, diana).
False
?-
```

### Thuật toán backward:

```
thiennu@thiennu-UX305UAB:~/HK5/AI/Project2_FOL$ python3 main.py
Enter file input of knowledge base: familytree.pl
0. Exit
1. Forward
2. Backward
3. Resolution
Which do you want to choose? 2
?- male(X).
{'X': 'james'}
{'X': 'mark_phillips'}
{'X': 'timothy_laurence'}
{'X': 'charles'}
{'X': 'andrew'}
{'X': 'edward'}
{'X': 'william'}
{'X': 'peter_phillips'}
{'X': 'harry'}
{'X': 'mike'}
{'X': 'george'}
{'X': 'phillip'}
True
?- grandchild(X, elizabeth).
{'X': 'james'}
{'X': 'beatrice'}
{'X': 'zara'}
{'X': 'peter_phillips'}
{'X': 'eugene'}
{'X': 'william'}
{'X': 'harry'}
{'X': 'louise_mountbatten'}
True
?- husband(charles, diana).
False
?- halt
```

### Thuật toán suy diễn Hợp giải:

Hợp giải chưa được cài đặt đồng nhất nên hiện tại chỉ chạy được một số cơ sở tri thức nhỏ.

Giá sử lấy cơ sở tri thức ở ví dụ 1.

```
thiennu@thiennu-UX305UAB:~/HK5/AI/Project2_FOL$ python3 main.py
Enter file input of knowledge base: script.pl
0. Exit
1. Forward
2. Backward
3. Resolution
Which do you want to choose? 3
?- likes(tuan, X).
{'X': 'mi'}
True
?- date(X, Y).
{'X': 'tuan', 'Y': 'mi'}
{'X': 'mi', 'Y': 'tuan'}
True
?- jealous(X, Y).
{'X': 'lan', 'Y': 'lan'}
{'X': 'lan', 'Y': 'mi'}
{'X': 'tuan', 'Y': 'tuan'}
{'X': 'mi', 'Y': 'lan'}
{'X': 'mi', 'Y': 'mi'}
{'X': 'huy', 'Y': 'huy'}
True
?- likes(tuan, mi).
{}
True
?- likes(tuan, lan).
False
?-
```

### 3. Một số hạn chế chưa làm được

- Các câu hỏi cũng như các mệnh đề trong cơ sở chỉ được liên kết với nhau bởi phép and. Chưa xử lý được phép or.
- Phần hợp giải chỉ mới hoàn thành chạy brute-force. Chưa áp dụng phương pháp đồng nhất vào hợp giải. Do vậy, quá trình nạp cơ sở tri thức của hợp giải khá lâu.
- Thuật toán hợp giải chưa xử lý được câu hỏi dưới dạng nhiều mệnh đề nối với nhau.