BÁO CÁO ĐỒ ÁN 2

MÔN HỌC: CƠ SỞ TRÍ TUỆ NHÂN TẠO
BÀI TẬP PROLOG

Tên: Đỗ Thành Nhơn – 1512387
Nguyễn Thành Tân – 1512491
Lớp: 15CNTN

PHỤ LỤC

TÌM HIỂU PROLOG

TÌM HIỂU PROLOG

THÀNH PHẦN CƠ BẢN:

Facts (sự thật), rules (luật), queries (truy vấn).

Tập hợp facts và rules tạo thành database. Ngôn ngữ prolog tập trung vào tạo knowledgebase (cơ sở dữ liệu). Dùng bằng cách đặt các câu truy vấn.

Ví dụ về knowledgebase:

```
woman(mia).
woman(jody).
woman(yolanda).
playsAirGuitar(jody).
party.
```

Sau đó muốn truy vấn

```
?- woman(mia).
```

Prolog trả về

```
yes
```

Vì trong knowledgebase có woman(mia).

Nếu truy vấn playsAirGuitar(Yolanda) prolog sẽ trả về no vì không thể suy ra được từ knowledgebase (trường hợp suy ra sai cũng trả về no).

Ví dụ khác về knowledgebase

```
happy(yolanda).
listens2Music(mia).
listens2Music(yolanda):- happy(yolanda).
playsAirGuitar(mia):- listens2Music(mia).
playsAirGuitar(yolanda):- listens2Music(yolanda).
```

Knowledgebase này ngoài facts còn có rules. Ta định nghĩa rule như sau:

```
listens2Music(yolanda):- happy(yolanda).
```

Truy vấn listens2Music(yolanda) sẽ trả về yes vì ta có fact happy(yolanda) và có rule trên.

Facts và rules được gọi chung là clauses (mệnh đề). Một khái niệm khác là predicate (vị từ), nhìn vào kb trên ta có 3 predicates happy, listems2Music, playsAirguitar.

Xét một kb khác:

```
happy(vincent).
listens2Music(butch).
playsAirGuitar(vincent):-
   listens2Music(vincent),
   happy(vincent).
playsAirGuitar(butch):-
   happy(butch).
playsAirGuitar(butch):-
   listens2Music(butch).
```

Ta thấy rule

```
playsAirGuitar(vincent):-
   listens2Music(vincent),
   happy(vincent).
```

Có thêm dấu ‘,’ thể hiện toán tử *and.* Toán tử *or* được biểu diễn bằng ‘;’.

Xét một kb khác

```
woman(mia).
woman(jody).
woman(yolanda).

loves(vincent,mia).
loves(marsellus,mia).
loves(pumpkin,honey_bunny).
loves(honey_bunny,pumpkin).
```

Giả sử ta có một query như sau

```
?- woman(X).
```

Lúc này X được xem như một *biến* nên prolog sẽ trả về các giá trị như mia, jody, yolanda bằng cách *hợp nhất (unify)*. Cách làm hợp nhất sẽ được giới thiệu sau.

Xét tiếp kb sau:

```
loves(vincent,mia).
loves(marsellus,mia).
loves(pumpkin,honey_bunny).
loves(honey_bunny,pumpkin).

jealous(X,Y):- loves(X,Z), loves(Y,Z).
```

Biến còn có thể được dùng không chỉ trong queries mà còn trong kb. Nếu ta query jealous(vincent, marsellus) prolog thực hiện hợp nhất và sẽ trả về yes.

CÚ PHÁP:

Phần trước trình bày về các thành phần cơ bản của ngôn ngữ prolog như facts, rules, queries. Phần này ta sẽ đi sâu vào các thành phần cấu tạo nên facts, rules, queries.

Atoms (nguyên tử):

- Chuỗi các kí tự, số bắt đầu bằng chữ viết thường. Vd: mia, happy, listens2music,…
- Chuỗi các kí tự đặt biệt. Vd: ‘:-‘, ‘;’,…
- Chuỗi các kí tự bất kì đặt trong dấu ‘’. Vd: ‘sfkl3 sdf”,…

Numbers (số):

- Integer
- Float

Variables (biến)

- Chuỗi các kí tự, số bắt đầu bằng chữ viết in hoặc dấu gạch dưới. Vd: Vites, _G123,…

Kết hợp atoms, numbers, variables ta sẽ được complex term.

Complex terms: được xây dựng từ hàm tử và các đối số. Hàm tự phải là atom. Vd:

```
hide(X,father(father(father(butch))))
```

Hàm tử là hide, đối số là biến X và complex term khác father(father(butch)). Complex term này có hàm tử là father và đối số là complex term khác father(butch).

Số lượng đối số mà complex term có được gọi là arity. Hai complex term có cùng funtor nhưng khác arity thì khác nhau. Vd: happy(X), happy(Y,Z) được prolog hiểu là happy/1 và happy/2.

HỢP NHẤT

Phần này sẽ tìm hiểu cách prolog unify.

Ta có 3 loại term:

- Constants (atoms và numbers).
- Variables.
- Complex terms.

Khái niệm unification:

Hai term hợp nhất được với nhau nếu chúng là một hoặc có ít nhất 1 term là biến.

Ví dụ: mia và mia unify được, mia và X unify được, X và Y unify được,…

Ta có thể hiểu hợp nhất một biến và một term là khởi tạo biến đó thành term có cấu trúc giống như term ban đầu.

Prolog có built-in predicate cho phép xác định 2 term có unify được không

```
?- =(mia,mia).
```

Prolog sẽ trả về yes.

```
?- =(mia,vincent).
```

Prolog sẽ trả về no.

Hoặc

```
?- mia = mia.
yes
```

Ta đặt functor = vào giữa 2 đối số.

Ví dụ khác:

```
?- mia = X.

X = mia
yes
```

Lúc này prolog không chỉ trả về yes mà còn trả về cách thức để khởi tạo biến X sao cho có thể unify.

Nếu ta query

```
?- X = Y.
```

Kết quả trả về sẽ là

```
X = _5071
Y = _5071
yes
```

Lúc này prolog sẽ khởi tạo 2 biến X và Y thành 1 biến.

Ví dụ khác

```
?- k(s(g), t(k)) = k(X,t(Y)).

X = s(g)
Y = k
yes
```

Lúc này prolog sẽ khởi tạo X thành s(g) và Y thành k.

Unification (hợp nhất) không chỉ được dùng trong query mà còn trong knowledgebase. Bằng cách này, prolog giúp người sử dụng định nghĩa các đối tượng và các mối quan hệ dễ dàng hơn. Vd:

```
vertical(line(point(X,Y),point(X,Z))).

horizontal(line(point(X,Y),point(Z,Y))).
```

Định nghĩa predicate vertical và horizontal thông qua các biến, giúp kb tổng quát hơn.

Khi ta query

```
?- horizontal(line(point(1,1),point(2,Y))).
```

Prolog sẽ trả về

```
Y = 1 ;

no
```

Như đã chỉ ra ở trên.

TÌM KIẾM CHỨNG CỨ

Khi ta query, làm thế nào để Prolog biết câu hỏi đó đúng hay sai.

Câu trả lời ngắn gọn là prolog sẽ tìm kiếm đệ quy các khả năng để unify.

Ta xét kb sau:

```
f(a).
f(b).

g(a).
g(b).

h(b).

k(X) :- f(X), g(X), h(X).
```

Giả sử ta query

```
?- k(Y).
```

Prolog sẽ xử lý như thế nào?

Đầu tiên nó sẽ xem trong kb các định nghĩa predicate k và tìm thấy rule

```
k(X) :- f(X), g(X), h(X).
```

Lúc này prolog sẽ unify k(Y) với k(X). X và Y sẽ được khởi tạo chung 1 biến, giả sử _G34.

Câu query ban đầu trở thành

```
k(_G34) :- f(_G34), g(_G34), h(_G34).
```

Lúc này để tìm đối được thỏa mãn k để _G34 khởi tạo thành đối tượng đó, đối tượng đó phải thỏa mãn f, g, h.

Lúc này câu truy vấn sẽ trở thành:

```
?- f(_G34), g(_G34), h(_G34).
```

Prolog sẽ lần lượt tìm các đối tượng thỏa mãn tính chất f. Các đối tượng này trong kb là a, b.

_G34 được khởi tạo thành a. Lúc này câu query sẽ trở thành:

```
?- g(a),h(a).
```

Prolog tiếp tục xét theo thứ tự, vì g(a) có trong kb nên câu query trở thành

```
?- h(a).
```

Lúc này trong kb chỉ có 1 fact về predicate h là h(b) nên không có cách nào để thỏa mãn h(a).

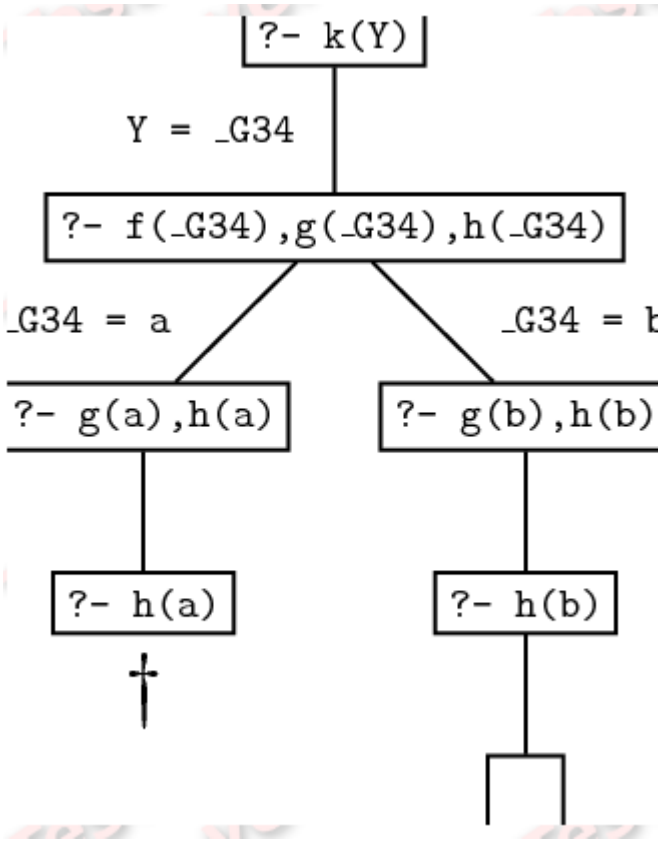
Prolog quay lui trên cây tìm kiếm đến trường hợp mà nó còn có thể khởi tạo _G34 thành giá trị khác, cụ thể là b. Lúc này câu truy vấn là:

```
?- g(b),h(b).
```

g(b) có trong kb nên câu truy vấn trở thành

```
?- h(b).
```

h(b) có trong kb nên cau truy vấn ban đầu thỏa và Y được khởi tạo thành b. Nói cách khác ta có k(b). Cây tìm kiếm



Xét kb khác

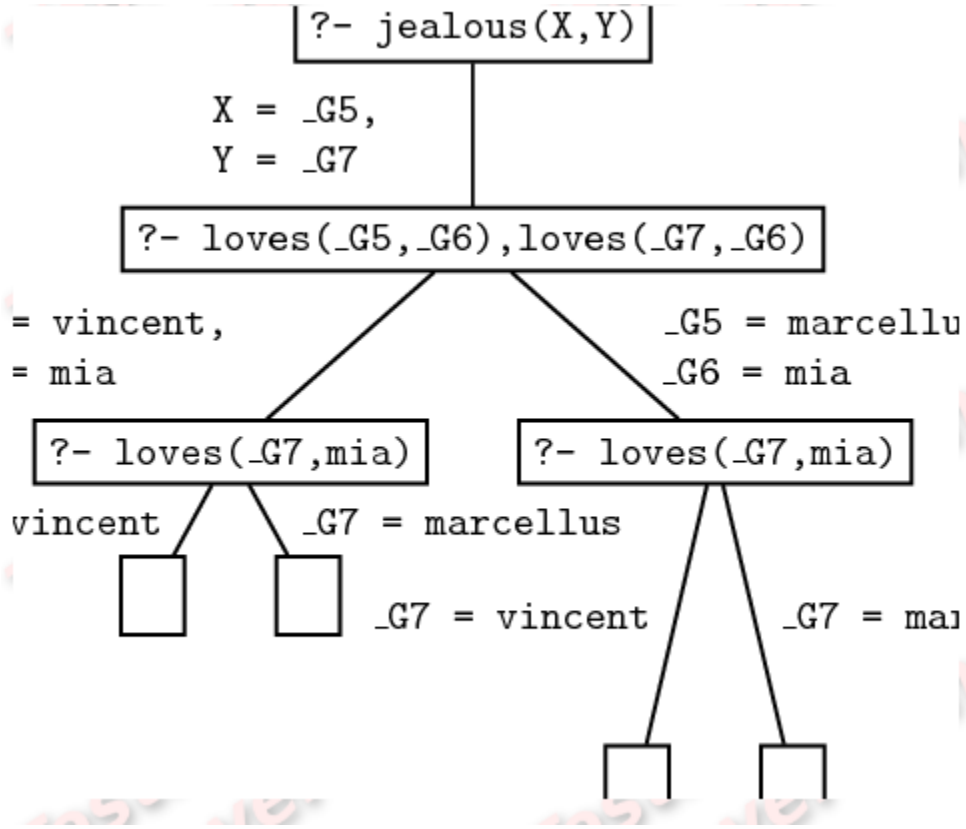
```
loves(vincent,mia).
loves(marcellus,mia).

jealous(A,B):- loves(A,C), loves(B,C).
```

Nếu ta query

```
?- jealous(X,Y).
```

Cây tìm kiếm sẽ là



Như ta thấy trên cây tìm kiếm, có 4 kết quả:

1. X = _G5 = vincent and Y = _G7 = vincent
2. X = _G5 = vincent and Y = _G7 = marcellus
3. X = _G5 = marcellus and Y = _G7 = vincent
4. X = _G5 = marcellus and Y = _G7 = marcellus

ĐỆ QUY

Predicates có thể được định nghĩa 1 cách đệ quy trong kb.

Xét kb sau:

```
is_digesting(X,Y) :- just_ate(X,Y).
is_digesting(X,Y) :-
        just_ate(X,Z),
        is_digesting(Z,Y).

just_ate(mosquito,blood(john)).
just_ate(frog,mosquito).
just_ate(stork,frog).
```

Predicate is_digesting được định nghĩa đệ quy theo rule thứ 2.

Lưu ý có trường hợp "cơ sở" để đệ quy dừng nhờ luật thứ 1.

Giả sử ta có câu query

```
?- is_digesting(stork,mosquito).
```

Prolog sẽ dựa vào predicate is_digesting/2 trong kb và duyệt từ trên xuống dưới.

Dựa vào rule đầu tiên, prolog sẽ unify X với stork và Y với mosquito. Câu truy vấn lúc này trở thành:

```
?- just_ate(stork,mosquito).
```

Prolog tiếp tục tìm kiếm trong kb predicate just_ate/2 để xem có thể thỏa mãn được câu query trên hay không. Nhưng trong kb chỉ có toàn facts về just_ate/2 và không có fact nào thỏa mãn nen prolog sẽ sử dụng rule thứ 2

```
is_digesting(X,Y) :-
        just_ate(X,Z),
        is_digesting(Z,Y).
```

Khởi tạo X thành stork và Y thành mosquito, câu truy vấn trở thành:

```
?- just_ate(stork,Z),
is_digesting(Z,mosquito).
```

Xét từ trái qua phải, prolog xử lý just_ate(stork,Z) trước. Xét trong kb, chỉ có khởi tạo Z thành frog thì mới thỏa mãn. Do đó câu truy vấn lúc này trở thành:

```
?- is_digesting(frog,mosquito).
```

Vì trong kb có

```
?- just_ate(frog,mosquito).
```

Và

```
is_digesting(X,Y) :- just_ate(X,Y).
```

Nên khởi tạo X thành frog, Y thành mosquito ta được is_digesting(frog,mosquito).

Do đó prolog trả về yes cho câu truy vấn ban đầu.

TRIỂN KHAI

CÁCH THỨC TRIỂN KHAI

Download từ distribution của Ubuntu.

```
nakt@nakt-computer:~$ sudo apt-get install swi-prolog
```

Khởi động

```
nakt@nakt-computer:~$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.2)
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).

?-
```

VÍ DỤ

Bộ cơ sở dữ liệu (kb).

```
woman(thao).
woman(tuyen).
man(nam).
child(nhi).
```

Load kb

Query

```
?- woman(nam).
false.

?- man(nam).
true.

?- woman(thao).
true.
```

Bộ kb khác

```
happy(X) :- highpoint(X).
happy(X) :- like(Y,X).
like(thao, nam).
highpoint(tuyen).
```


happy(X) :- highpoint(X).
happy(X) :- like(Y,X).

Query

```
?- happy(nam).
true.

?- happy(thao).
false.

?- happy(tuyen).
true ;
false.
```

XÂY DỰNG BỘ CÂU HỎI CHO HỆ TRI THỨC

1. ?- male(X). // Ai là nam
    a. X = phillip ;
    b. X = charles ;
    c. X = captain_Mark_Phillips ;
    d. X = timothy_Laurance ;
    e. X = andrew ;
    f. X = edward ;
    g. X = william ;
    h. X = harry ;
    i. X = peter ;
    j. X = mike ;
    k. X = james ;
    l. X = george.

2. ?- female(X). // Ai là nữ
    a. X = queen_Ezlizabeth_II ;
    b. X = diana ;
    c. X = camilla ;
    d. X = anne ;
    e. X = sarah ;
    f. X = sophie ;
    g. X = kate ;
    h. X = autumn ;
    i. X = zara ;
    j. X = beatrice ;
    k. X = eugenie ;
    l. X = louise ;

m. X = charlotte ;
n. X = savannah ;
o. X = isla ;
p. X = mia_Grace.

3. ?- mother(X,andrew). // Ai là mẹ của Prince andrew
    a. X = queen_Ezlizabeth_II

4. ?- wife(X,william). // Ai là vợ của Price William
    a. X = kate
5. ?- husband(Y,diana). // Ai là chồng của công nương Diana
    a. Y = charles

6. ?- child(X,diana).// AI là con của Diana
    a. X = william ;
    b. X = harry.
7. ?- father(X,Y). // Nêu những cặp cha con
    a. X = phillip,
    b. Y = charles ;
    c. X = phillip,
    d. Y = anne ;
    e. X = phillip,
    f. Y = andrew ;
    g. X = phillip,
    h. Y = edward ;
    i. X = charles,
    j. Y = william ;
    k. X = charles,
    l. Y = harry ;
    m. X = captain_Mark_Phillips,
    n. Y = peter ;
    o. X = captain_Mark_Phillips,
    p. Y = zara ;
    q. X = andrew,
    r. Y = beatrice ;
    s. X = andrew,
    t. Y = eugenie ;
    u. X = edward,

v. Y = lousie ;
w. X = edward,
x. Y = james ;
y. X = william,
z. Y = george ;
aa. X = william,
bb. Y = charlotte ;
cc. X = peter,
dd. Y = savannah ;
ee. X = peter,
ff. Y = isla ;
gg. X = mike,
hh. Y = mia_Grace ;

8. ?- wife(queen_Ezlizabeth_II,mia_Grace). //Nữ hoàng ezlizabeth có phải vợ MiaGrace.
   a. false.
9. ?- sibling(X,harry).// Ai là anh em của harry
   a. X = william ;
10. ?- daughter(X,william). // Ai là con gái của william
   a. X = charlotte ;
11. ?- son(X,kate). // ai là con gái của kate.
   a. X = george.

12. ?- grandparent(X,mia_Grace).// ai là ông bà của Mia Grace
   a. X = captain_Mark_Phillips ;
   b. X = anne ;

13. ?- grandmother(X,james) . //Ai là bà của James
   a. X = queen_Ezlizabeth_II

14. ?-grandfather(x,peter).// Ai là ông của Peter.
   a. X= phillips
15. ?- grandchild(X,diana).// Ai là cháu của Diana
   a. X = george ;
   b. X = charlotte
16. ?- grandson(X,phillip).// Ai là cháu trai của Phillips
   a. X = william ;
   b. X = harry ;
   c. X = peter ;

   d. X = james ;

17. ?- granddaughter(X,anne).// ai là cháu gái của Anne
   a. X = savannah ;
   b. X = isla ;
   c. X = mia_Grace ;
18. ?- sister(X,isla).//Ai là chị gái của Isla
   a. X = savannah ;
19. ?- niece(X,savannah). //Ai là cháu gái Savannah.
   a. false.
20. ?- nephew(X,diana). //Ai là cháu họ của Diana.
   a. X = peter ;
   b. X = james ;