**CZ3005 : Artificial Intelligence**

**Lab2 Introduction to Logic Programming – Prolog (Wk9)**

**LEE SUYEON / N1801612A / TS2**

**Example 1: on Family Tree**

#1. my-family-tree.pl

<code implementation>

1. male(jongho).
2. male(suhwan).
3. male(yunchang).
4. male(jaehoon).
5. male(chungbu).
6. female(suyeon).
7. female(jina).
8. female(jinsun).
9. female(jisu).
10. brother(suhwan,suyeon).
11. brother(yunchang,jina).
12. sister(suyeon,suhwan).
13. sister(jina,yunchang).
14. sister(jina,jinsun).
15. sister(jinsun,jina).
16. parent\_of(jina,suyeon).
17. parent\_of(jina,suhwan).
18. parent\_of(jongho,suhwan).
19. parent\_of(jongho,suyeon).
20. parent\_of(chungbu,jongho).
21. parent\_of(jinsun,jaehoon).
22. parent\_of(yunchang,jisu).
23. father(X,Y):-parent\_of(X,Y),male(X).
24. mother(X,Y):-parent\_of(X,Y),female(X).
25. son(X,Y):-parent\_of(Y,X),male(X).
26. daughter(X,Y):-parent\_of(Y,X),female(X).
27. grandfather(X,Z):-parent\_of(X,Y),parent\_of(Y,Z),male(X).
28. sibling(X,Y):-brother(X,Y);sister(X,Y);brother(Y,X);sister(Y,X).
29. aunt(X,Z):-sister(X,Y),parent\_of(Y,Z).
30. uncle(X,Z):-brother(X,Y),parent\_of(Y,Z).
31. cousin(X,Y):-parent\_of(A,X),parent\_of(B,Y),sibling(A,B).
32. spouse(X,Y):-parent\_of(X,Z),male(X),parent\_of(Y,Z),female(Y).

<query 1 : spouse(X,Y)>

?-spouse(X,Y)

**X** = jongho,  
**Y** = jina

**X** = jongho,  
**Y** = jina

**false**

?-trace.

**true**

?-trace,spouse(X,Y)

**Call:***spouse*(\_4316, \_4312)

**Call:***parent\_of*(\_4316, \_4600)

**Exit:***parent\_of*(jina, suyeon)

**Call:***male*(jina)

**Fail:***male*(jina)

**Redo:***parent\_of*(\_4316, \_4604)

**Exit:***parent\_of*(jina, suhwan)

**Call:***male*(jina)

**Fail:***male*(jina)

**Redo:***parent\_of*(\_4316, \_4604)

**Exit:***parent\_of*(jongho, suhwan)

**Call:***male*(jongho)

**Exit:***male*(jongho)

**Call:***parent\_of*(\_4312, suhwan)

**Exit:***parent\_of*(jina, suhwan)

**Call:***female*(jina)

**Exit:***female*(jina)

**Exit:***spouse*(jongho, jina)

**X** = jongho,  
**Y** = jina

**Redo:***parent\_of*(\_4312, suhwan)

**Exit:***parent\_of*(jongho, suhwan)

**Call:***female*(jongho)

**Fail:***female*(jongho)

**Redo:***parent\_of*(\_4316, \_4998)

**Exit:***parent\_of*(jongho, suyeon)

**Call:***male*(jongho)

**Exit:***male*(jongho)

**Call:***parent\_of*(\_4312, suyeon)

**Exit:***parent\_of*(jina, suyeon)

**Call:***female*(jina)

**Exit:***female*(jina)

**Exit:***spouse*(jongho, jina)

**X** = jongho,  
**Y** = jina

**Redo:***parent\_of*(\_4312, suyeon)

**Exit:***parent\_of*(jongho, suyeon)

**Call:***female*(jongho)

**Fail:***female*(jongho)

**Redo:***parent\_of*(\_4316, \_5366)

**Exit:***parent\_of*(chungbu, jongho)

**Call:***male*(chungbu)

**Exit:***male*(chungbu)

**Call:***parent\_of*(\_4312, jongho)

**Exit:***parent\_of*(chungbu, jongho)

**Call:***female*(chungbu)

**Fail:***female*(chungbu)

**Redo:***parent\_of*(\_4316, \_5366)

**Exit:***parent\_of*(jinsun, jaehoon)

**Call:***male*(jinsun)

**Fail:***male*(jinsun)

**Redo:***parent\_of*(\_4316, \_5366)

**Exit:***parent\_of*(yunchang, jisu)

**Call:***male*(yunchang)

**Exit:***male*(yunchang)

**Call:***parent\_of*(\_4312, jisu)

**Exit:***parent\_of*(yunchang, jisu)

**Call:***female*(yunchang)

**Fail:***female*(yunchang)

**Fail:***spouse*(\_4316, \_4312)

**false**

<query 2 : uncle(X,myself)>

?-uncle(X,suyeon)

**X** = yunchang

**false**

?-trace.

**true**

?-trace,uncle(X,suyeon)

**Call:***uncle*(\_4152, suyeon)

**Call:***brother*(\_4152, \_4424)

**Exit:***brother*(suhwan, suyeon)

**Call:***parent\_of*(suyeon, suyeon)

**Fail:***parent\_of*(suyeon, suyeon)

**Redo:***brother*(\_4152, \_4428)

**Exit:***brother*(yunchang, jina)

**Call:***parent\_of*(jina, suyeon)

**Exit:***parent\_of*(jina, suyeon)

**Exit:***uncle*(yunchang, suyeon)

**X** = yunchang

**Redo:***parent\_of*(jina, suyeon)

**Fail:***parent\_of*(jina, suyeon)

**Fail:***uncle*(\_4152, suyeon)

**false**

#2. using family.pl

(a)

<parent\_of(X,Y) trace>

Added “parent\_of(X,Y):-parent\_of(X,Z),sibling(Y,Z).” definition to find unspecified “parent\_of(X,Y)” in given family facts.

?-parent\_of(X,Y).

**X** = warren,  
**Y** = jerry

**X** = maryalice,  
**Y** = jerry

**X** = warren,  
**Y** = kather

**X** = warren,  
**Y** = stuart

**X** = maryalice,  
**Y** = kather

**X** = maryalice,  
**Y** = stuart

(removed duplicates)

?- trace.

**true**

?-trace,parent\_of(X,Y)

**Call:***parent\_of*(\_4720, \_4716)

**Exit:***parent\_of*(warren, jerry)

**X** = warren,  
**Y** = jerry

**Redo:***parent\_of*(\_4720, \_4716)

**Exit:***parent\_of*(maryalice, jerry)

**X** = maryalice,  
**Y** = jerry

**Redo:***parent\_of*(\_4720, \_4716)

**Call:***parent\_of*(\_4720, \_5766)

**Exit:***parent\_of*(warren, jerry)

**Call:***sibling*(\_4716, jerry)

**Call:***brother*(\_4716, jerry)

**Fail:***brother*(\_4716, jerry)

**Redo:***sibling*(\_4716, jerry)

**Call:***sister*(\_4716, jerry)

**Exit:***sister*(kather, jerry)

**Exit:***sibling*(kather, jerry)

**Exit:***parent\_of*(warren, kather)

**X** = warren,  
**Y** = kather

**Redo:***sibling*(\_4716, jerry)

**Call:***brother*(jerry, \_4716)

**Exit:***brother*(jerry, stuart)

**Exit:***sibling*(stuart, jerry)

**Exit:***parent\_of*(warren, stuart)

**X** = warren,  
**Y** = stuart

**Redo:***brother*(jerry, \_4716)

**Exit:***brother*(jerry, kather)

**Exit:***sibling*(kather, jerry)

**Exit:***parent\_of*(warren, kather)

**X** = warren,  
**Y** = kather (=> duplicated result)

**Redo:***sibling*(\_4716, jerry)

**Call:***sister*(jerry, \_4716)

**Fail:***sister*(jerry, \_4716)

**Fail:***sibling*(\_4716, jerry)

**Redo:***parent\_of*(\_4720, \_6874)

**Exit:***parent\_of*(maryalice, jerry)

**Call:***sibling*(\_4716, jerry)

**Call:***brother*(\_4716, jerry)

**Fail:***brother*(\_4716, jerry)

**Redo:***sibling*(\_4716, jerry)

**Call:***sister*(\_4716, jerry)

**Exit:***sister*(kather, jerry)

**Exit:***sibling*(kather, jerry)

**Exit:***parent\_of*(maryalice, kather)

**X** = maryalice,  
**Y** = kather

**Redo:***sibling*(\_4716, jerry)

**Call:***brother*(jerry, \_4716)

**Exit:***brother*(jerry, stuart)

**Exit:***sibling*(stuart, jerry)

**Exit:***parent\_of*(maryalice, stuart)

**X** = maryalice,  
**Y** = stuart

<AND-OR tree>

parent\_of(X,Y)

parent\_of(X,Z) and sibling(Y,Z)

sibling(Y,Z)

parent\_of(X,Z)

X = warren

Z = jerry

brother(Y,Z)

sister(Z,Y)

X = marryalice

Z = jerry

Y = jerry

Z = kather

Y = jerry

Z = kather

succeed

succeed

Y = jerry

Z = stuart

X = marryalice

Z = kather

X = marryalice

Z = stuart

X = warren

Z = kather

X = warren

Z = stuart

succeed

succeed

succeed

succeed

(b) after reordering the facts

?-parent\_of(X,Y)

**X** = maryalice,  
**Y** = jerry

**X** = warren,  
**Y** = jerry

**X** = maryalice,  
**Y** = kather

**X** = maryalice,  
**Y** = stuart

**X** = warren,  
**Y** = kather

**X** = warren,  
**Y** = stuart

(removed duplicates)

?-trace.

**true**

?-trace,parent\_of(X,Y)

**Call:***parent\_of*(\_4904, \_4900)

**Exit:***parent\_of*(maryalice, jerry)

**X** = maryalice,  
**Y** = jerry

**Redo:***parent\_of*(\_4904, \_4900)

**Exit:***parent\_of*(warren, jerry)

**X** = warren,  
**Y** = jerry

**Redo:***parent\_of*(\_4904, \_4900)

**Call:***parent\_of*(\_4904, \_5950)

**Exit:***parent\_of*(maryalice, jerry)

**Call:***sibling*(\_4900, jerry)

**Call:***brother*(\_4900, jerry)

**Fail:***brother*(\_4900, jerry)

**Redo:***sibling*(\_4900, jerry)

**Call:***sister*(\_4900, jerry)

**Exit:***sister*(kather, jerry)

**Exit:***sibling*(kather, jerry)

**Exit:***parent\_of*(maryalice, kather)

**X** = maryalice,  
**Y** = kather

**Redo:***sibling*(\_4900, jerry)

**Call:***brother*(jerry, \_4900)

**Exit:***brother*(jerry, kather)

**Exit:***sibling*(kather, jerry)

**Exit:***parent\_of*(maryalice, kather)

**X** = maryalice,  
**Y** = kather (=> duplicated result)

**Redo:***brother*(jerry, \_4900)

**Exit:***brother*(jerry, stuart)

**Exit:***sibling*(stuart, jerry)

**Exit:***parent\_of*(maryalice, stuart)

**X** = maryalice,  
**Y** = stuart

**Redo:***sibling*(\_4900, jerry)

**Call:***sister*(jerry, \_4900)

**Fail:***sister*(jerry, \_4900)

**Fail:***sibling*(\_4900, jerry)

**Redo:***parent\_of*(\_4904, \_7058)

**Exit:***parent\_of*(warren, jerry)

**Call:***sibling*(\_4900, jerry)

**Call:***brother*(\_4900, jerry)

**Fail:***brother*(\_4900, jerry)

**Redo:***sibling*(\_4900, jerry)

**Call:***sister*(\_4900, jerry)

**Exit:***sister*(kather, jerry)

**Exit:***sibling*(kather, jerry)

**Exit:***parent\_of*(warren, kather)

**X** = warren,  
**Y** = kather

**Redo:***sibling*(\_4900, jerry)

**Call:***brother*(jerry, \_4900)

**Exit:***brother*(jerry, kather)

**Exit:***sibling*(kather, jerry)

**Exit:***parent\_of*(warren, kather)

**X** = warren,  
**Y** = kather (=> duplicated result)

**Redo:***brother*(jerry, \_4900)

**Exit:***brother*(jerry, stuart)

**Exit:***sibling*(stuart, jerry)

**Exit:***parent\_of*(warren, stuart)

**X** = warren,  
**Y** = stuart

conclusion) The query result set is same. However the order of query results are different after reordering, which means the traces are not identical. Prolog uses DFS and the search engine gives a result as soon as it hit one. Therefore, if the order of each rule’s appearance changes, the order of the results may change. In that case, the traces may also not identical.

**Exercise 1: The Smart Phone Rivalry**

1. First Order Logic

company(sumSum).

company(appy).

smartPhoneTech(galacticaS3).

developed(galacticaS3, sumSum).

boss(stevey).

competitor(sumSum, appy).

steal(stevey, galacticaS3).

∀ Product smartPhoneTech(Product) ⇒ business(Product)

∀ Company (competitor(Company, appy) ∨ competitor(appy, Company)) ∧ company(Company) ⇒ rival(Company)

∀ (X, Prod, Comp) boss(X) ∧ steal(X, Prod) ∧ business(Prod) ∧ developed(Biz, Comp) ^ rival(Comp) ⇒ unethical(X)

2. prolog code implementation

1. company(sumSum).
2. company(appy).
3. smartPhoneTech(galacticaS3).
4. developed(galacticaS3,sumSum).
5. boss(stevey).
6. competitor(sumSum,appy).
7. steal(stevey,galacticaS3).
8. business(Product):-smartPhoneTech(Product).
9. rival(Company):-(competitor(Company,appy);competitor(appy,Company)),company(Company).
10. unethical(X):-boss(X),steal(X,Prod),business(Prod),developed(Prod,Comp),rival(Comp).

3.

?-unethical(X)

**X** = stevey

**false**

?-trace.

**true**

?trace,unethical(X)

**Call:***unethical*(\_4716)

**Call:***boss*(\_4716)

**Exit:***boss*(stevey)

**Call:***steal*(stevey, \_4986)

**Exit:***steal*(stevey, galacticaS3)

**Call:***business*(galacticaS3)

**Call:***smartPhoneTech*(galacticaS3)

**Exit:***smartPhoneTech*(galacticaS3)

**Exit:***business*(galacticaS3)

**Call:***developed*(galacticaS3, \_4986)

**Exit:***developed*(galacticaS3, sumSum)

**Call:***rival*(sumSum)

**Call:***competitor*(sumSum, appy)

**Exit:***competitor*(sumSum, appy)

**Call:***company*(sumSum)

**Exit:***company*(sumSum)

**Exit:***rival*(sumSum)

**Exit:***unethical*(stevey)

**X** = stevey

**Redo:***rival*(sumSum)

**Call:***competitor*(appy, sumSum)

**Fail:***competitor*(appy, sumSum)

**Fail:***rival*(sumSum)

**Fail:***unethical*(\_4716)

**false**

**Exercise 2: The Royal Family**

1.

<code implementation>

1. female(elizabeth).
2. female(ann).
3. male(charles).
4. male(andrew).
5. male(edward).
6. parent\_of(elizabeth,charles).
7. parent\_of(elizabeth,ann).
8. parent\_of(elizabeth,andrew).
9. parent\_of(elizabeth,edward).
10. older\_than(charles,ann).
11. older\_than(ann,andrew).
12. older\_than(andrew,edward).
13. youngest(edward).
14. old\_succ(X,Y):-(parent\_of(Y,X),male(X),((older\_than(X,Z),parent\_of(Y,Z));youngest(X)));(parent\_of(Y,X),female(X),((older\_than(X,Z),parent\_of(Y,Z));youngest(X))).

?-old\_succ(X,elizabeth)

**X** = charles

**X** = andrew

**X** = edward

**X** = ann

**false**

?-trace.

**true**

?-trace,old\_succ(X,elizabeth)

**Call:***old\_succ*(\_5434, elizabeth)

**Call:***parent\_of*(elizabeth, \_5434)

**Exit:***parent\_of*(elizabeth, charles)

**Call:***male*(charles)

**Exit:***male*(charles)

**Call:***older\_than*(charles, \_5706)

**Exit:***older\_than*(charles, ann)

**Call:***parent\_of*(elizabeth, ann)

**Exit:***parent\_of*(elizabeth, ann)

**Exit:***old\_succ*(charles, elizabeth)

**X** = charles

**Redo:***old\_succ*(charles, elizabeth)

**Call:***youngest*(charles)

**Fail:***youngest*(charles)

**Redo:***parent\_of*(elizabeth, \_5434)

**Exit:***parent\_of*(elizabeth, ann)

**Call:***male*(ann)

**Fail:***male*(ann)

**Redo:***parent\_of*(elizabeth, \_5434)

**Exit:***parent\_of*(elizabeth, andrew)

**Call:***male*(andrew)

**Exit:***male*(andrew)

**Call:***older\_than*(andrew, \_6090)

**Exit:***older\_than*(andrew, edward)

**Call:***parent\_of*(elizabeth, edward)

**Exit:***parent\_of*(elizabeth, edward)

**Exit:***old\_succ*(andrew, elizabeth)

**X** = andrew

**Redo:***old\_succ*(andrew, elizabeth)

**Call:***youngest*(andrew)

**Fail:***youngest*(andrew)

**Redo:***parent\_of*(elizabeth, \_5434)

**Exit:***parent\_of*(elizabeth, edward)

**Call:***male*(edward)

**Exit:***male*(edward)

**Call:***older\_than*(edward, \_6448)

**Fail:***older\_than*(edward, \_6448)

**Redo:***old\_succ*(edward, elizabeth)

**Call:***youngest*(edward)

**Exit:***youngest*(edward)

**Exit:***old\_succ*(edward, elizabeth)

**X** = edward

**Redo:***old\_succ*(\_5434, elizabeth)

**Call:***parent\_of*(elizabeth, \_5434)

**Exit:***parent\_of*(elizabeth, charles)

**Call:***female*(charles)

**Fail:***female*(charles)

**Redo:***parent\_of*(elizabeth, \_5434)

**Exit:***parent\_of*(elizabeth, ann)

**Call:***female*(ann)

**Exit:***female*(ann)

**Call:***older\_than*(ann, \_6806)

**Exit:***older\_than*(ann, andrew)

**Call:***parent\_of*(elizabeth, andrew)

**Exit:***parent\_of*(elizabeth, andrew)

**Exit:***old\_succ*(ann, elizabeth)

**X** = ann

**Redo:***old\_succ*(ann, elizabeth)

**Call:***youngest*(ann)

**Fail:***youngest*(ann)

**Redo:***parent\_of*(elizabeth, \_5434)

**Exit:***parent\_of*(elizabeth, andrew)

**Call:***female*(andrew)

**Fail:***female*(andrew)

**Redo:***parent\_of*(elizabeth, \_5434)

**Exit:***parent\_of*(elizabeth, edward)

**Call:***female*(edward)

**Fail:***female*(edward)

**Fail:***old\_succ*(\_5434, elizabeth)

**false**

2.

<code implementation>

1. female(elizabeth).
2. female(ann).
3. male(charles).
4. male(andrew).
5. male(edward).
6. parent\_of(elizabeth,charles).
7. parent\_of(elizabeth,ann).
8. parent\_of(elizabeth,andrew).
9. parent\_of(elizabeth,edward).
10. older\_than(charles,ann).
11. older\_than(ann,andrew).
12. older\_than(andrew,edward).
13. youngest(edward).
14. new\_succ(X,Y):-parent\_of(Y,X),((older\_than(X,Z),parent\_of(Y,Z));youngest(X)),(male(X);female(X)).

?-new\_succ(X,elizabeth)

**X** = charles

**X** = ann

**X** = andrew

**X** = edward

**false**

?-trace.

**true**

?-trace,new\_succ(X,elizabeth)

**Call:***new\_succ*(\_4976, elizabeth)

**Call:***parent\_of*(elizabeth, \_4976)

**Exit:***parent\_of*(elizabeth, charles)

**Call:***older\_than*(charles, \_5248)

**Exit:***older\_than*(charles, ann)

**Call:***parent\_of*(elizabeth, ann)

**Exit:***parent\_of*(elizabeth, ann)

**Call:***male*(charles)

**Exit:***male*(charles)

**Exit:***new\_succ*(charles, elizabeth)

**X** = charles

**Redo:***new\_succ*(charles, elizabeth)

**Call:***female*(charles)

**Fail:***female*(charles)

**Redo:***new\_succ*(charles, elizabeth)

**Call:***youngest*(charles)

**Fail:***youngest*(charles)

**Redo:***parent\_of*(elizabeth, \_4976)

**Exit:***parent\_of*(elizabeth, ann)

**Call:***older\_than*(ann, \_5632)

**Exit:***older\_than*(ann, andrew)

**Call:***parent\_of*(elizabeth, andrew)

**Exit:***parent\_of*(elizabeth, andrew)

**Call:***male*(ann)

**Fail:***male*(ann)

**Redo:***new\_succ*(ann, elizabeth)

**Call:***female*(ann)

**Exit:***female*(ann)

**Exit:***new\_succ*(ann, elizabeth)

**X** = ann

**Redo:***new\_succ*(ann, elizabeth)

**Call:***youngest*(ann)

**Fail:***youngest*(ann)

**Redo:***parent\_of*(elizabeth, \_4976)

**Exit:***parent\_of*(elizabeth, andrew)

**Call:***older\_than*(andrew, \_5990)

**Exit:***older\_than*(andrew, edward)

**Call:***parent\_of*(elizabeth, edward)

**Exit:***parent\_of*(elizabeth, edward)

**Call:***male*(andrew)

**Exit:***male*(andrew)

**Exit:***new\_succ*(andrew, elizabeth)

**X** = andrew

**Redo:***new\_succ*(andrew, elizabeth)

**Call:***female*(andrew)

**Fail:***female*(andrew)

**Redo:***new\_succ*(andrew, elizabeth)

**Call:***youngest*(andrew)

**Fail:***youngest*(andrew)

**Redo:***parent\_of*(elizabeth, \_4976)

**Exit:***parent\_of*(elizabeth, edward)

**Call:***older\_than*(edward, \_6348)

**Fail:***older\_than*(edward, \_6348)

**Redo:***new\_succ*(edward, elizabeth)

**Call:***youngest*(edward)

**Exit:***youngest*(edward)

**Call:***male*(edward)

**Exit:***male*(edward)

**Exit:***new\_succ*(edward, elizabeth)

**X** = edward

**Redo:***new\_succ*(edward, elizabeth)

**Call:***female*(edward)

**Fail:***female*(edward)

**Fail:***new\_succ*(\_4976, elizabeth)

**false**

<difference between prolog rules of old succession and new succession >

old\_succ(X,Y):-(parent\_of(Y,X),**male(X)**,((older\_than(X,Z),parent\_of(Y,Z));youngest(X)));(parent\_of(Y,X),**female(X)**,((older\_than(X,Z),parent\_of(Y,Z));youngest(X))).

new\_succ(X,Y):-parent\_of(Y,X),((older\_than(X,Z),parent\_of(Y,Z));youngest(X)),(**male(X);female(X)**).

As codes above, old succession rule first find all male successors according to the order of birth, and then search for female successors. On the other hand, new succession rule search for successors according to the order of birth, regardless of gender. Therefore new rule set gender as a last search element using (male ‘or’(;) female).

(End)