Nanyang Technological University (NTU)

CZ3005 Artificial Intelligence

Lab Assignment 3

Adrian Goh Jun Wei

U1721134D

TSP1

Table of Content

Exercise 1: The Smart Phone Rivalry

First-Order Logic (FOL)

Some assumptions inferred:

- If X is a competitor of Y, Y is also a competitor of X
- Stevey is the boss of appy
- It is not unethical for a boss to steal business from non-rival companies

Statements	**First-Order Logics (FOL)**
sumsum, a competitor of *appy*	Company(sumsum) Company(appy) Competitors(sumsum, appy) ∀ x. ∀ y. (Competitors(x, y) ↔ Competitors(y, x))
sumsum ... developed some nice *smartphone technology* called *galactica-s3*	Develop(sumsum, galactica-s3) SmartphoneTech(galactica-s3)
galactica-s3 ... was stolen by *stevey*	Steal(stevey, galactica-s3)
stevey, who is a *boss*	Boss(stevey, appy)
unethical for a boss to steal business from rival companies	∀ a. ∀ b. ∀ c. ∀ d. (Boss(a, b) ∧ Steal(a, d) ∧ Business(d) ∧ Develop(c, d) ∧ Rival(b, c) ∧ Company(c) → Unethical(a))
competitor of *appy* is a *rival*	∀ x. (Competitors(x, appy) → Rival(x, appy))
Smartphone technology is *business*	∀ x. (SmartphoneTech(x) → Business(x))

Proof that Stevey is unethical

```
?- trace, unethical(stevey).
   Call: (11) unethical(stevey) ? creep
   Call: (12) boss(stevey, _11066) ? creep
   Exit: (12) boss(stevey, appy) ? creep
   Call: (12) steal(stevey, _11160) ? creep
   Exit: (12) steal(stevey, galactica-s3) ? creep
   Call: (12) business(galactica-s3) ? creep
   Call: (13) smartphonetech(galactica-s3) ? creep
   Exit: (13) smartphonetech(galactica-s3) ? creep
   Exit: (12) business(galactica-s3) ? creep
   Call: (12) develop(_11444, galactica-s3) ? creep
   Exit: (12) develop(sumsum, galactica-s3) ? creep
   Call: (12) rival(appy, sumsum) ? creep
   Call: (13) competitors(appy, sumsum) ? creep
   Call: (14) competitors(sumsum, appy) ? creep
   Exit: (14) competitors(sumsum, appy) ? creep
   Exit: (13) competitors(appy, sumsum) ? creep
   Exit: (12) rival(appy, sumsum) ? creep
   Call: (12) company(sumsum) ? creep
   Exit: (12) company(sumsum) ? creep
   Exit: (11) unethical(stevey) ? creep
true .
```

Exercise 2: The Royal Family

Determine the line of succession

```
?- trace, sortedSuccessionList(queen_elizabeth, X).
   Call: (11) sortedSuccessionList(queen_elizabeth, _10556) ? creep
^  Call: (12) findall(_11108, offspring(_11108, queen_elizabeth), _11116) ? creep
   Call: (17) offspring(_11108, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_charles, queen_elizabeth) ? creep
   Redo: (17) offspring(_11108, queen_elizabeth) ? creep
   Exit: (17) offspring(princess_ann, queen_elizabeth) ? creep
   Redo: (17) offspring(_11108, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Redo: (17) offspring(_11108, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_edward, queen_elizabeth) ? creep
^  Exit: (12) findall(_11108, user:offspring(_11108, queen_elizabeth), [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
   Call: (12) sort_succession_list([prince_charles, princess_ann, prince_andrew, prince_edward], _10556) ? creep
   Call: (13) sort_succession_list([princess_ann, prince_andrew, prince_edward], _11682) ? creep
   Call: (14) sort_succession_list([prince_andrew, prince_edward], _11730) ? creep
   Call: (15) sort_succession_list([prince_edward], _11778) ? creep
   Call: (16) sort_succession_list([], _11826) ? creep
   Exit: (16) sort_succession_list([], []) ? creep
   Call: (16) insert(prince_edward, [], _11778) ? creep
   Exit: (16) insert(prince_edward, [], [prince_edward]) ? creep
   Exit: (15) sort_succession_list([prince_edward], [prince_edward]) ? creep
   Call: (15) insert(prince_andrew, [prince_edward], _11730) ? creep
^  Call: (16) not(precedes(prince_andrew, prince_edward)) ? creep
   Call: (17) precedes(prince_andrew, prince_edward) ? creep
   Call: (18) offspring(prince_andrew, _12220) ? creep
   Exit: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (18) male(prince_andrew) ? creep
   Exit: (18) male(prince_andrew) ? creep
   Call: (18) female(prince_edward) ? creep
   Fail: (18) female(prince_edward) ? creep
   Redo: (17) precedes(prince_andrew, prince_edward) ? creep
   Call: (18) offspring(prince_andrew, _12636) ? creep
   Exit: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (18) male(prince_andrew) ? creep
   Exit: (18) male(prince_andrew) ? creep
   Call: (18) male(prince_edward) ? creep
   Exit: (18) male(prince_edward) ? creep
   Call: (18) is_older(prince_andrew, prince_edward) ? creep
   Call: (19) older(prince_andrew, prince_edward) ? creep
   Exit: (19) older(prince_andrew, prince_edward) ? creep
   Exit: (18) is_older(prince_andrew, prince_edward) ? creep
   Exit: (17) precedes(prince_andrew, prince_edward) ? creep
^  Fail: (16) not(user:precedes(prince_andrew, prince_edward)) ? creep
   Redo: (15) insert(prince_andrew, [prince_edward], _11730) ? creep
   Exit: (15) insert(prince_andrew, [prince_edward], [prince_andrew, prince_edward]) ? creep
   Exit: (14) sort_succession_list([prince_andrew, prince_edward], [prince_andrew, prince_edward]) ? creep
   Call: (14) insert(princess_ann, [prince_andrew, prince_edward], _11682) ? creep
^  Call: (15) not(precedes(princess_ann, prince_andrew)) ? creep
   Call: (16) precedes(princess_ann, prince_andrew) ? creep
   Call: (17) offspring(princess_ann, _13582) ? creep
   Exit: (17) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (17) male(princess_ann) ? creep
   Fail: (17) male(princess_ann) ? creep
   Redo: (16) precedes(princess_ann, prince_andrew) ? creep
   Call: (17) offspring(princess_ann, _13906) ? creep
   Exit: (17) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (17) male(princess_ann) ? creep
   Fail: (17) male(princess_ann) ? creep
   Redo: (16) precedes(princess_ann, prince_andrew) ? creep
   Call: (17) offspring(princess_ann, _14230) ? creep
   Exit: (17) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (17) offspring(prince_andrew, queen_elizabeth) ? creep
```

```
   Exit: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (17) male(princess_ann) ? creep
   Fail: (17) male(princess_ann) ? creep
   Redo: (16) precedes(princess_ann, prince_andrew) ? creep
   Call: (17) offspring(princess_ann, _14230) ? creep
   Exit: (17) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (17) female(princess_ann) ? creep
   Exit: (17) female(princess_ann) ? creep
   Call: (17) female(prince_andrew) ? creep
   Fail: (17) female(prince_andrew) ? creep
   Fail: (16) precedes(princess_ann, prince_andrew) ? creep
^  Exit: (15) not(user:precedes(princess_ann, prince_andrew)) ? creep
   Call: (15) insert(princess_ann, [prince_edward], _13476) ? creep
^  Call: (16) not(precedes(princess_ann, prince_edward)) ? creep
   Call: (17) precedes(princess_ann, prince_edward) ? creep
   Call: (18) offspring(princess_ann, _14848) ? creep
   Exit: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (18) male(princess_ann) ? creep
   Fail: (18) male(princess_ann) ? creep
   Redo: (17) precedes(princess_ann, prince_edward) ? creep
   Call: (18) offspring(princess_ann, _15172) ? creep
   Exit: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (18) male(princess_ann) ? creep
   Fail: (18) male(princess_ann) ? creep
   Redo: (17) precedes(princess_ann, prince_edward) ? creep
   Call: (18) offspring(princess_ann, _15496) ? creep
   Exit: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (18) female(princess_ann) ? creep
   Exit: (18) female(princess_ann) ? creep
   Call: (18) female(prince_edward) ? creep
   Fail: (18) female(prince_edward) ? creep
   Fail: (17) precedes(princess_ann, prince_edward) ? creep
^  Exit: (16) not(user:precedes(princess_ann, prince_edward)) ? creep
   Call: (16) insert(princess_ann, [], _14742) ? creep
   Exit: (16) insert(princess_ann, [], [princess_ann]) ? creep
   Exit: (15) insert(princess_ann, [prince_edward], [prince_edward, princess_ann]) ? creep
   Exit: (14) insert(princess_ann, [prince_andrew, prince_edward], [prince_andrew, prince_edward, princess_ann]) ? creep
   Exit: (13) sort_succession_list([princess_ann, prince_andrew, prince_edward], [prince_andrew, prince_edward, princess_ann]) ? creep
   Call: (13) insert(prince_charles, [prince_andrew, prince_edward, princess_ann], _10556) ? creep
^  Call: (14) not(precedes(prince_charles, prince_andrew)) ? creep
   Call: (15) precedes(prince_charles, prince_andrew) ? creep
   Call: (16) offspring(prince_charles, _16350) ? creep
   Exit: (16) offspring(prince_charles, queen_elizabeth) ? creep
   Call: (16) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (16) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (16) male(prince_charles) ? creep
   Exit: (16) male(prince_charles) ? creep
   Call: (16) female(prince_andrew) ? creep
   Fail: (16) female(prince_andrew) ? creep
   Redo: (15) precedes(prince_charles, prince_andrew) ? creep
   Call: (16) offspring(prince_charles, _16766) ? creep
   Exit: (16) offspring(prince_charles, queen_elizabeth) ? creep
   Call: (16) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (16) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (16) male(prince_charles) ? creep
   Exit: (16) male(prince_charles) ? creep
   Call: (16) male(prince_andrew) ? creep
   Exit: (16) male(prince_andrew) ? creep
   Call: (16) is_older(prince_charles, prince_andrew) ? creep
   Call: (17) older(prince_charles, prince_andrew) ? creep
   Fail: (17) older(prince_charles, prince_andrew) ? creep

   Redo: (16) is_older(prince_charles, prince_andrew)
 ? creep
   Call: (17) older(prince_charles, _17320) ? creep
   Exit: (17) older(prince_charles, princess_ann) ? creep
   Call: (17) is_older(princess_ann, prince_andrew) ? creep
   Call: (18) older(princess_ann, prince_andrew) ? creep
   Exit: (18) older(princess_ann, prince_andrew) ? creep
   Exit: (17) is_older(princess_ann, prince_andrew) ? creep
   Exit: (16) is_older(prince_charles, prince_andrew) ? creep
   Exit: (15) precedes(prince_charles, prince_andrew) ? creep
^  Fail: (14) not(user:precedes(prince_charles, prince_andrew)) ? creep
   Redo: (13) insert(prince_charles, [prince_andrew, prince_edward, princess_ann], _10556) ? creep
   Exit: (13) insert(prince_charles, [prince_andrew, prince_edward, princess_ann], [prince_charles, prince_andrew, prince_edward, princess_ann])
 ? creep
   Exit: (12) sort_succession_list([prince_charles, princess_ann, prince_andrew, prince_edward], [prince_charles, prince_andrew, prince_edward, princess_ann])
 ? creep
   Exit: (11) sortedSuccessionList(queen_elizabeth, [prince_charles, prince_andrew, prince_edward, princess_ann]) ? creep
X = [prince_charles, prince_andrew, prince_edward, princess_ann].
```

Determine the line of succession (modified)

The original rules are as follow:

```
%% Rule 1: Male child will always come before female child
precedes(X, Y):-
       offspring(X, A), offspring(Y, A),
       male(X), female(Y),
       not(queen(Y)).

%% Rule 2: Older male child will come before younger male child
precedes(X, Y):-
       offspring(X, A), offspring(Y, A),
       male(X), male(Y),
       is_older(X, Y).

%% Rule 3: Older female child will come before younger female child
precedes(X, Y):-
       offspring(X, A), offspring(Y, A),
       female(X), female(Y),
       is_older(X, Y),
       not(queen(X)), not(queen(Y)).
```

Succession rules to determine the original line of succession

To update the rules such that gender does not plays a part, we simply have to:

- Remove Rule 1
- Merge Rule 2 and Rule 3

```
%% Rule 1: Older child will come younger child
precedes(X, Y):-
       offspring(X, A), offspring(Y, A),
       is_older(X, Y),
       not(queen(X)), not(queen(Y)).
```

Updated succession rules to determine line to determine line of succession

```
?- trace, sortedSuccessionList(queen_elizabeth, X).
   Call: (11) sortedSuccessionList(queen_elizabeth, _2942) ? creep
^  Call: (12) findall(_3504, offspring(_3504, queen_elizabeth), _3512) ? creep
   Call: (17) offspring(_3504, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_charles, queen_elizabeth) ? creep
   Redo: (17) offspring(_3504, queen_elizabeth) ? creep
   Exit: (17) offspring(princess_ann, queen_elizabeth) ? creep
   Redo: (17) offspring(_3504, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Redo: (17) offspring(_3504, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_edward, queen_elizabeth) ? creep
^  Exit: (12) findall(_3504, user:offspring(_3504, queen_elizabeth), [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
   Call: (12) sort_succession_list([prince_charles, princess_ann, prince_andrew, prince_edward], _2942) ? creep
   Call: (13) sort_succession_list([princess_ann, prince_andrew, prince_edward], _4078) ? creep
   Call: (14) sort_succession_list([prince_andrew, prince_edward], _4126) ? creep
   Call: (15) sort_succession_list([prince_edward], _4174) ? creep
   Call: (16) sort_succession_list([], _4222) ? creep
   Exit: (16) sort_succession_list([], []) ? creep
   Call: (16) insert(prince_edward, [], _4174) ? creep
   Exit: (16) insert(prince_edward, [], [prince_edward]) ? creep
   Exit: (15) sort_succession_list([prince_edward], [prince_edward]) ? creep
   Call: (15) insert(prince_andrew, [prince_edward], _4126) ? creep
^  Call: (16) not(precedes(prince_andrew, prince_edward)) ? creep
   Call: (17) precedes(prince_andrew, prince_edward) ? creep
   Call: (18) offspring(prince_andrew, _4616) ? creep
   Exit: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (18) is_older(prince_andrew, prince_edward) ? creep
   Call: (19) older(prince_andrew, prince_edward) ? creep
   Exit: (19) older(prince_andrew, prince_edward) ? creep
   Exit: (18) is_older(prince_andrew, prince_edward) ? creep
^  Call: (18) not(queen(prince_andrew)) ? creep
   Call: (19) queen(prince_andrew) ? creep
   Fail: (19) queen(prince_andrew) ? creep
^  Exit: (18) not(user:queen(prince_andrew)) ? creep
^  Call: (18) not(queen(prince_edward)) ? creep
   Call: (19) queen(prince_edward) ? creep
   Fail: (19) queen(prince_edward) ? creep
^  Exit: (18) not(user:queen(prince_edward)) ? creep
   Exit: (17) precedes(prince_andrew, prince_edward) ? creep
^  Fail: (16) not(user:precedes(prince_andrew, prince_edward)) ? creep
   Redo: (15) insert(prince_andrew, [prince_edward], _4126) ? creep
   Exit: (15) insert(prince_andrew, [prince_edward], [prince_andrew, prince_edward]) ? creep
   Exit: (14) sort_succession_list([prince_andrew, prince_edward], [prince_andrew, prince_edward]) ? creep
   Call: (14) insert(princess_ann, [prince_andrew, prince_edward], _4078) ? creep
^  Call: (15) not(precedes(princess_ann, prince_andrew)) ? creep
   Call: (16) precedes(princess_ann, prince_andrew) ? creep
   Call: (17) offspring(princess_ann, _5766) ? creep
   Exit: (17) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (17) is_older(princess_ann, prince_andrew) ? creep
   Call: (18) older(princess_ann, prince_andrew) ? creep
   Exit: (18) older(princess_ann, prince_andrew) ? creep
   Exit: (17) is_older(princess_ann, prince_andrew) ? creep
^  Call: (17) not(queen(princess_ann)) ? creep
   Call: (18) queen(princess_ann) ? creep
   Fail: (18) queen(princess_ann) ? creep
^  Exit: (17) not(user:queen(princess_ann)) ? creep
^  Call: (17) not(queen(prince_andrew)) ? creep
   Call: (18) queen(prince_andrew) ? creep
   Fail: (18) queen(prince_andrew) ? creep
^  Exit: (17) not(user:queen(prince_andrew)) ? creep
   Exit: (16) precedes(princess_ann, prince_andrew) ? creep
^  Fail: (15) not(user:precedes(princess_ann, prince_andrew)) ? creep
   Redo: (14) insert(princess_ann, [prince_andrew, prince_edward], _4078) ? creep
   Exit: (14) insert(princess_ann, [prince_andrew, prince_edward], [princess_ann, prince_andrew, prince_edward]) ? creep
   Exit: (13) sort_succession_list([princess_ann, prince_andrew, prince_edward], [princess_ann, prince_andrew, prince_edward]) ? creep
   Call: (13) insert(prince_charles, [princess_ann, prince_andrew, prince_edward], _2942) ? creep
^  Call: (14) not(precedes(prince_charles, princess_ann)) ? creep
   Call: (15) precedes(prince_charles, princess_ann) ? creep
   Call: (16) offspring(prince_charles, _6916) ? creep
   Exit: (16) offspring(prince_charles, queen_elizabeth) ? creep
   Call: (16) offspring(princess_ann, queen_elizabeth) ? creep
   Exit: (16) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (16) is_older(prince_charles, princess_ann) ? creep
   Call: (17) older(prince_charles, princess_ann) ? creep
   Exit: (17) older(prince_charles, princess_ann) ? creep
   Exit: (16) is_older(prince_charles, princess_ann) ? creep
^  Call: (16) not(queen(prince_charles)) ? creep
   Call: (17) queen(prince_charles) ? creep
   Fail: (17) queen(prince_charles) ? creep
^  Exit: (16) not(user:queen(prince_charles)) ? creep
^  Call: (16) not(queen(princess_ann)) ? creep
   Call: (17) queen(princess_ann) ? creep
   Fail: (17) queen(princess_ann) ? creep
^  Exit: (16) not(user:queen(princess_ann)) ? creep
   Exit: (15) precedes(prince_charles, princess_ann) ? creep
^  Fail: (14) not(user:precedes(prince_charles, princess_ann)) ? creep
   Redo: (13) insert(prince_charles, [princess_ann, prince_andrew, prince_edward], _2942) ? creep
   Exit: (13) insert(prince_charles, [princess_ann, prince_andrew, prince_edward], [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
   Exit: (12) sort_succession_list([prince_charles, princess_ann, prince_andrew, prince_edward], [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
   Exit: (11) sortedSuccessionList(queen_elizabeth, [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
X = [prince_charles, princess_ann, prince_andrew, prince_edward].
```