Nanyang Technological University
School of Computer Science and Engineering

CZ3005 - Lab 2

Artificial Intelligence

Phua Jia Sheng
SSP 2

1. The Smartphone Rivalary

(a) Translate the natural language statements above describing the dealing within the Smart Phone industry in to First Order Logic, (FOL).

company(sumSum)
company(appy)
smartphone technology(galactica_S3)

boss(stevey, appy)
competitor(sumSum, appy)
develop(sumSum,galactica_S3)
steal(stevey,galactica_S3)

competitor(X,appy) ⇒ $\mathrm{rival}(\mathrm{X},\mathrm{appy})$
∀x smartphone technology(X) ⇒ business(X)
∀x develop(X,Y) ⇒ own(X,Y)
steal(B,X) Λ own(D,X) Λ boss(B,C) Λ rival (D,C) ⇒ $\mathrm{unethical}(\mathrm{B})$

(b) Write these FOL statements as Prolog clauses.

```
%Entities
company(sumSum).
company(appy).
person(stevey).
smartphone_tech(galactica_S3).

%Relationships
boss(stevey, appy).
competitor(sumSum, appy).
develop(sumSum,galactica_S3).
steal(stevey,galactica_S3).

%Rules
/*1. A competitor of Appy is a rival*/
rival(X, appy) :-
  competitor(X, appy).

/*2. smartphone_tech is a business*/
business(X) :-
  smartphone_tech(X).

/*3. If a company develops a smartphone_tech, it owns the rights to it*/
own(X,Y) :-
  develop(X,Y).

/*4. It is unethical for a Boss (of company C)
to steal business (X) from rival companies (D).*/
unethical(B) :-
  steal(B,X), own(D,X), boss(B,C), rival(D,C).
```

Figure 1: Source code: smartphone.pl

(c) Using the prolog search engine, prove that Stevey is unethical. Show a trace of your proof.

```
[trace]  ?- unethical(stevey).
   Call: (8) unethical(stevey) ? creep
   Call: (9) steal(stevey, _5502) ? creep
   Exit: (9) steal(stevey, galactica_S3) ? creep
   Call: (9) own(_5500, galactica_S3) ? creep
   Call: (10) develop(_5500, galactica_S3) ? creep
   Exit: (10) develop(sumSum, galactica_S3) ? creep
   Exit: (9) own(sumSum, galactica_S3) ? creep
   Call: (9) boss(stevey, _5502) ? creep
   Exit: (9) boss(stevey, appy) ? creep
   Call: (9) rival(sumSum, appy) ? creep
   Call: (10) competitor(sumSum, appy) ? creep
   Exit: (10) competitor(sumSum, appy) ? creep
   Exit: (9) rival(sumSum, appy) ? creep
   Exit: (8) unethical(stevey) ? creep
true.
```

2. The Royal Family

(a) Male-priority succession

Since we are proving the goal successor(X, Elizabeth) through the sub-goal child(X, Elizabeth), and Prolog rules have the same priority as they are listed, in order to force Prolog to consider male successors first, we just list the male children in order of birth before female children in the child(X,Y) relationships.

```
%Entities
female(elizabeth).
male(charles).
female(ann).
male(andrew).
male(edward).

%Relationships
child(charles, elizabeth).
child(andrew, elizabeth).
child(edward, elizabeth).
child(ann, elizabeth).

%Rules
successor(X,elizabeth) :-
  child(X,elizabeth).
```

Figure 2: Source code: royalFam_Old.pl

The results are as follow:

```
[trace]  ?- successor(X,elizabeth).
   Call: (8) successor(_5320, elizabeth) ? creep
   Call: (9) child(_5320, elizabeth) ? creep
   Exit: (9) child(charles, elizabeth) ? creep
   Exit: (8) successor(charles, elizabeth) ? creep
X = charles ;
   Redo: (9) child(_5320, elizabeth) ? creep
   Exit: (9) child(andrew, elizabeth) ? creep
   Exit: (8) successor(andrew, elizabeth) ? creep
X = andrew ;
   Redo: (9) child(_5320, elizabeth) ? creep
   Exit: (9) child(edward, elizabeth) ? creep
   Exit: (8) successor(edward, elizabeth) ? creep
X = edward ;
   Redo: (9) child(_5320, elizabeth) ? creep
   Exit: (9) child(ann, elizabeth) ? creep
   Exit: (8) successor(ann, elizabeth) ? creep
X = ann.
```

(b) Succession based on birth order regardless of gender

For this question we just need to reorder the listing of child(X,Y) relationships to reflect birth order regardless of gender:

```
%Entities
female(elizabeth).
male(charles).
female(ann).
male(andrew).
male(edward).

%Relationships
child(charles, elizabeth).
child(ann, elizabeth).
child(andrew, elizabeth).
child(edward, elizabeth).

%Rules
successor(X,elizabeth) :-
  child(X,elizabeth).
```

Figure 3: Source code: royalFam_New.pl

The results are as follow:

```
[trace]  ?- successor(X,elizabeth).
   Call: (8) successor(_5320, elizabeth) ? creep
   Call: (9) child(_5320, elizabeth) ? creep
   Exit: (9) child(charles, elizabeth) ? creep
   Exit: (8) successor(charles, elizabeth) ? creep
X = charles ;
   Redo: (9) child(_5320, elizabeth) ? creep
   Exit: (9) child(ann, elizabeth) ? creep
   Exit: (8) successor(ann, elizabeth) ? creep
X = ann ;
   Redo: (9) child(_5320, elizabeth) ? creep
   Exit: (9) child(andrew, elizabeth) ? creep
   Exit: (8) successor(andrew, elizabeth) ? creep
X = andrew ;
   Redo: (9) child(_5320, elizabeth) ? creep
   Exit: (9) child(edward, elizabeth) ? creep
   Exit: (8) successor(edward, elizabeth) ? creep
X = edward.
```