Programming Language, Assignment - 4

Tulsi Jain

1. Investigating Prolog

1.1

Reorder Fact;

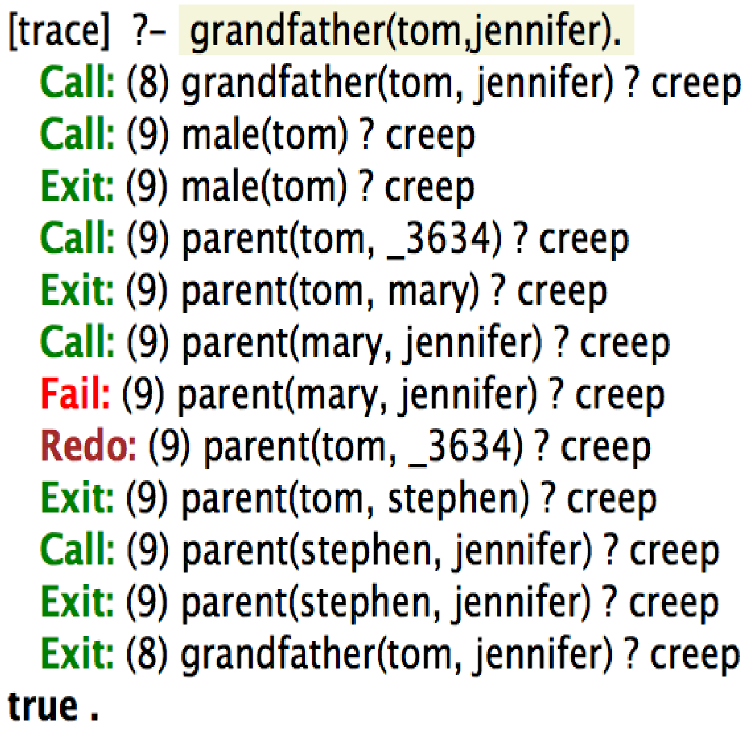
```
male(tom).
male(brian).
male(kevin).
male(zhane).
male(fred).
male(jake).
male(bob).
male(stephen).
male(paul).

parent(tom,stephen).
parent(stephen,jennifer).
parent(tom,mary).
parent(mary,sarah).
parent(melissa,brian).
parent(bob,jane).
parent(paul,kevin).
parent(jake,bob).
parent(zhane,melissa).
parent(stephen,paul).
parent(emily,bob).
parent(zhane,mary).
```

1.2

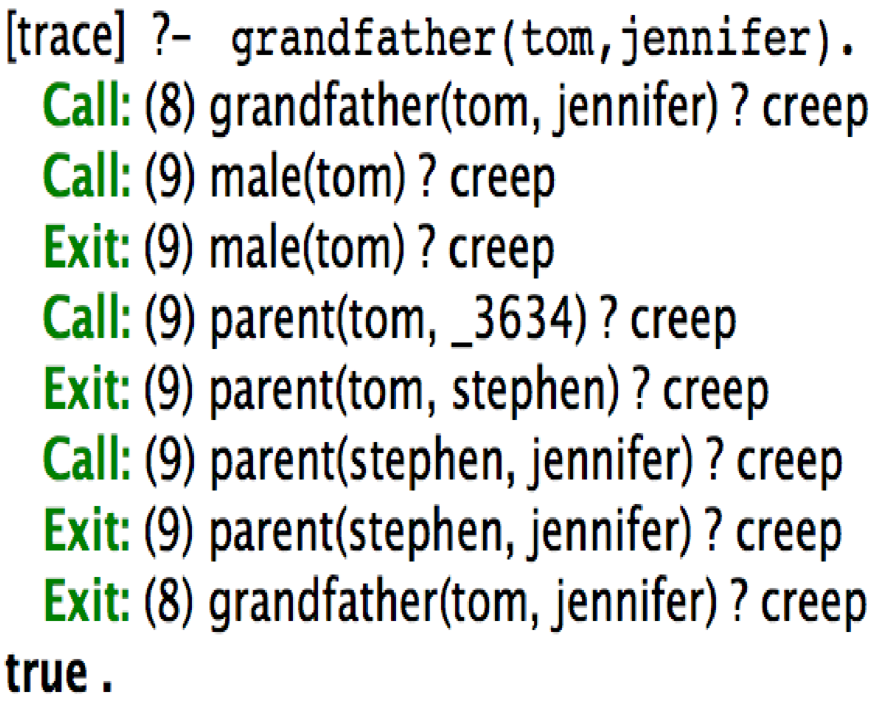
Changes above affects the execution time because of Prolog backtracking behaviour. First, **male(Tom)** would satisfy without any backtracking. As per original ordering **parent(tom,mary)** would be discovered before **parent(tom,stephen).)** but this will fail to **parent(mary,jennifer)**. Hence this would cause backtracking and more execution time. Prolog would again look for in which tom is parent. Our reordering allows **parent(tom,stephen))** to discover first and this also satisfies **parent(stephen,jennifer)**. I have also moved **male(tom)**, **parent(tom,stephen)** and **parent(stephen,jennifer)** on top.

Figure 1.1: Before modifying order



```
[trace]  ?-  grandfather(tom,jennifer).
   Call: (8) grandfather(tom, jennifer) ? creep
   Call: (9) male(tom) ? creep
   Exit: (9) male(tom) ? creep
   Call: (9) parent(tom, _3634) ? creep
   Exit: (9) parent(tom, mary) ? creep
   Call: (9) parent(mary, jennifer) ? creep
   Fail: (9) parent(mary, jennifer) ? creep
   Redo: (9) parent(tom, _3634) ? creep
   Exit: (9) parent(tom, stephen) ? creep
   Call: (9) parent(stephen, jennifer) ? creep
   Exit: (9) parent(stephen, jennifer) ? creep
   Exit: (8) grandfather(tom, jennifer) ? creep
true .
```

Figure 1.2: After modifying order



```
[trace]  ?-  grandfather(tom,jennifer).
   Call: (8) grandfather(tom, jennifer) ? creep
   Call: (9) male(tom) ? creep
   Exit: (9) male(tom) ? creep
   Call: (9) parent(tom, _3634) ? creep
   Exit: (9) parent(tom, stephen) ? creep
   Call: (9) parent(stephen, jennifer) ? creep
   Exit: (9) parent(stephen, jennifer) ? creep
   Exit: (8) grandfather(tom, jennifer) ? creep
true .
```

1.3

No, with given facts and rules we can't represent grandmother rule because we would not able to satisfy female condition. If a person is not a male then we can not really say this is female.

Below mentioned part is not solution but a further extension. This is only possible under the following assumptions:

1. our world is close (universe of facts is complete)
2. A person must have a gender
3. A person must only and only one gender
4. Only two gender possible i.e. male and female.

Rule

```
grandmother(X,Y) :- parent(X,Z), parent(Z,Y), \+ male(X).
```

Output is X = emily, Y = jane.

1.4

1.4.1

```
brother(X,Y) :- male(X), parent(Z,X), parent(Z,Y), \+(X=Y).
uncle(X, Y) :- brother(X, K), parent(K, Y).
```

Output is X = stephen, Y = sarah.

1.4.2

Define new facts as follows:

```
female(melissa).
female(mary).
female(sarah).
female(jane).
female(emily).
female(jennifer).
sister(X,Y) :- female(X), parent(Z, X), parent(Z, Y), \+(X=Y).
aunt(X, Y) :- sister(X, K), parent(K, Y).
```

Output is X = melissa, Y = sarah.
Output is X = mary, Y = jennifer.
Output is X = mary, Y = paul.
Output is X = mary, Y = brian.
Output is X = jennifer, Y = kevin.

2. Prolog Rules

See coding file

3. Unification

3.1 d(15) & c(X)

No, different functor

3.2 42 & 23

No, different constant

3.3 a(X, b(3, 1, Y)) a(4, Y)

No, X = 4 but b(3, 1, Y) ↪ Y leads to infinite recursion

3.4 a(X, c(2, B, D)) a(4, c(A, 7, C))

Yes, X =4, A =2, B =7, C=D

3.5 a(X, c(2, A, X)) a(4, c(A, 7, C))

No, A is already unified with 2, can not unified with 7

3.6 e(c(2, D)) & e(c(8, D))

No, 2 can not be unified with 8

3.7 X e(f(6, 2), g(8, 1))

Yes, X = e(f(6, 2), g(8, 1))

3.8 b(X, g(8, X)) & b(f(6, 2), g(8, f(6, 2)))

Yes, X = f(6, 2)

3.9 a(1, b(X, Y)) & a(Y, b(2, c(6, Z), 10))

No, b takes two argument of LHS and three argument on RHS

3.10 d(c(1, 2, 1)) & d(c(X, Y, X))

Yes , X = 1, and Y = 2

4. Nani Search

Point 9. I implemented a new feature that as follows: To go into cellar you either need a only transporter or key and door that connects to present location. For example. If you have a transporter you can go to cellar without any other requirement. Else you need a key and then you can go to cellar from kitchen because kitchen has a door to cellar but despite of having a key you can not go to cellar from office as there is not door to cellar from office.

5. OOLs

5.1

```
output base class
show base class
output derived class
show derived class
output derived class
show base class
output base class
show base class
```

5.2

Used Virtual Table

```
bp− >output();
bp2− >output();
```

Not Used Virtual Table

```
b.show();
b.output();
bp− >show();
bp2− >output();
bp2− >show();
d.show();
d.output();
```

5.3

Base Class

Base Instance	Method	Version
vptr, →	output →	Base Version

Derived Class

Derived Instance	Method	Version
vptr, →	output →	Derived Version
	show →	Derived Version

6. Clone Object

6.1 obj1

x(20)

6.2 obj2

y(5)

6.3 obj3

z(30)

6.4 obj4

x(10)

6.5 obj1.x

20

6.6 obj2.x

20

6.7 obj3.x

20

6.8 obj4.x

10

6.9 obj4.y

Undefined in both obj1 and obj2

6.10 obj2.y

5

6.11 obj3.y

5

6.12 obj3.z

30