# Q1.2

For grandfather (tom, jennifer) there are 3 conditions to be satisfied.

# HOMEWORK 4

1. **male(tom):**

So we put that rule first among all the other male rules. So that when the tree traverses down the male(X), male(tom) is the first rule it sees and backtracks true.

# parent(tom, stephen):

Next condition to be satisfied is if tom is parent to someone who is in turn parent to jennifer. We see that parent(tom, stephen) satisfies that and we put that as the first rule under parent(X,Y)

We know that X == tom through unification from first condition. Hence, parent(tom, stephen) returns true.

# parent(stephen, jennifer):

Our next condition is simple. Through unification we know that we have to look for parent(stephen, jennifer) under parent(X,Y). So, we put that rule as the second one. And since the parent(tom, stephen) was already traversed it won't be traversed now. So, the next rule should be parent(stephen, jennifer).

Order of rest of the rules doesnt matter. In this manner with minimum traversal and backtracking, we find grandfather(tom, jennifer).

# Providing trace of before and after the changes made. BEFORE

[trace] 7 ?- grandfather(tom, jennifer).

Call: (10) grandfather(tom, jennifer) ? creep Call: (11) male(tom) ? creep

Exit: (11) male(tom) ? creep

Call: (11) parent(tom, \_2968) ? creep Exit: (11) parent(tom, mary) ? creep Call: (11) parent(mary, jennifer) ? creep Fail: (11) parent(mary, jennifer) ? creep Redo: (11) parent(tom, \_3144) ? creep Exit: (11) parent(tom, stephen) ? creep

Call: (11) parent(stephen, jennifer) ? creep Exit: (11) parent(stephen, jennifer) ? creep

Exit: (10) grandfather(tom, jennifer) ? creep true .

# AFTER

[trace] 2 ?- grandfather(tom, jennifer).

Call: (10) grandfather(tom, jennifer) ? creep Call: (11) male(tom) ? creep

Exit: (11) male(tom) ? creep

Call: (11) parent(tom, \_5232) ? creep Exit: (11) parent(tom, stephen) ? creep

Call: (11) parent(stephen, jennifer) ? creep Exit: (11) parent(stephen, jennifer) ? creep Exit: (10) grandfather(tom, jennifer) ? creep

true .

As can be clearly seen, we avoided the step of parent(tom, marry) traversal by re- ordering the rules a bit.

# Q1.3

Assuming that the universe of facts is not complete, we cannot define a new rule ‘grandmother’ that can call into the given goals to correctly derive an answer. This is because for the rule of grandmother(X,Y), we would need to establish that X is a female. But there is no such rule provided originally. And, we cannot also assume that if X is not male then X is female.

So, just based on the given goals we cannot introduce the new rule of grandmother. We would need some more facts added to the ones already given to be able to do that.

# Q3

1. **Doesn’t Unify.**

Reason – d and c are not the same functors. No bindings.

# Doesn’t Unify.

Reason - Initially, X takes the value 4. (4 = X)

Then it tries to take the value of 76 (X = 76), which it can’t as a value can only have one binding. Hence, it fails.

# Doesn’t Unify

Reason – Initially,

# Unifies

The functors are the same,

The number of arguments is the same,

First argument Variable X is unified with value 4, (X = 4)

Second argument we see that one is a variable Y while the other is a function and one of its argument is Y itself. Hence an Occurs check takes place, and the two don’t unify. (Y = b(3, 1, Y))

Bindings – Between b(1, X) and b(X,Y) {left associative} 1 = X, X = Y

Between b(X,Y) and b(Y,1) X = Y, Y = 1

We get, X = Y, Y = 1;

# Doesn’t unify

Reason – Between a(1,X) & b(X,Y) {left associative} No bindings are a and b are two different functors.

1. **Unifies**

Bindings - a(X, c(2, B, D)) & a(4, c(A, 7, C))

The functors are same,

The number of arguments is the same, First argument gives us X = 4,

Second argument is a functor which is same, The number of arguments inside c is same, First inner argument gives us 2 = A, Second inner argument gives us B = 7,

Third inner argument gives us D = C, We have, X = 4; A = 2; B = 7; D = C;

# Doesn’t Unify

Reason - Between e(c(2, D)) & e(c(8, D))

Functors are same,

Number of arguments is same, Inner functor is same,

Number of arguments of inner functor is same,

1. **Unifies**

First argument gives us 2 = 8, which doesn’t unify as both are different

values and they unify with themselves only. No bindings.

Bindings – between X & e(f(6, 2), g(8, 1))

We get, X = e(f(6, 2), g(8, 1)) as the only binding.

1. **Unifies**

Bindings – between b(X, g(8, X)) & b(f(6, 2), g(8, f(6, 2))),

Functors are same,

Number of arguments is same, First argument gives us X = f(6,2)

Second argument are same functors with equal number of arguments, We get 8 = 8 and X = f(6,2) as the binding.

# Doesn’t Unify

Reason - Between a(1, b(X, Y)) & a(Y, b(2, c(6, Z), 10))

Functors are same with equal number of arguments, First argument gives us Y =1,

Second argument are functors with unequal number of arguments, Hence No bindings.

# Unifies

Bindings - Between d(c(1, 2, 1)) & d( c(X, Y, X))

Functor is same with equal number of arugments, Inner functor is same with equal number of arguments, First inner argument gives us X = 1,

Second inner argument gives us Y = 2,

Third argument gives us X = 1, which matches. Hence, we have,

X = 1, Y = 2

# Q6

1. x is contained locally to obj1.
2. y is contained locally to obj2.
3. z is contained locally to obj3.
4. x is contained locally to obj4.

**5. x =** 20

**6. x =** 20

**7. x =** 20

**8. x =** 10

**9.** No y field for obj4.

# 10. y = 5

**11. y =** 5

**12. z =** 30