COMP 348: Principles of Programming Languages
Assignment No.1 on Logical Programming

Duc Nguyen - Antonio Verdicchio - Zahra Nikbakht

Gina Cody School of Computer Science and Software Engineering
Concordia University, Montreal, QC, Canada

Summer 2020

Contents

1 Question 4

1.1 Modified Query

```
team(X), member(S, X),
|    findall([A,B], takes_course(S, A, B, _), L1),
|    list_to_set(L1, L),
|    length(L, NN),
|    write(S), write('has taken'), write(NN),
|    write(' course '), nl, fail.
```

2 Question 5

2.1 Explaination:

all_courses2('4000123', L) returns all the courses that the student with ID 4000123 has taken. But all_courses2(4000123, L) returns an empty list. This is because when we wrote our knowledge base, we enforced student IDs to be in form of quoted atoms (we wrapped IDs in single quotation marks). However, in the second query, the student ID is a number, and a number cannot be unified with a quoted atom. Therefore, the second query returns an empty list because it cant be unified with any of take_course statements ('4000123' = 4000123 is false).

3 Question 6:

Indicate which of the following pairs of terms can be unied together. If they cant be unied, please provide the reason for it. In case of error, indicate the error. If they can be unied successfully, wherever relevant, provide the variable instantiations that lead to successful unication. (Note that : indicates unication)

3.1 food(bread, X) = Food(Y, soup)

In this case, unification will not be successful since **food** is a functor and **Food** is not. For this unification to be successful, both sides would have to contain matching functors such as **food**.

3.2 Bread = soup

Since we have a variable and an atom, the variable will be instantiated to the value of the atom, thus the terms unify.

3.3 Bread = Soup

Since we have two variables, the variable will be instantiated to the value of the other variable or vice versa, thus the terms unify.

3.4 food(bread, X, milk) = food(Y, salad, X)

In this case the functors match, but the arguments do not. Y will be instantiated to 'bread', but X will be instantiated to either 'salad' or 'milk', meaning it cannot match with both of the atoms. This means the terms will not unify. If there were three variables, lets say X,Y,Z instead of X,Y,Y, then the terms would unify.

3.5 manager(X) = Y

In this case unification will be successful and **Y** will be instantiated to **manager(X)**.

3.6 meal(healthyFood(bread), drink(milk)) = meal(X,Y)

In this case unification will be successful with **X** instantiated to **healthyFood(bread)** and **Y** instantiated to **drink(milk)**.

3.7 meal(eat(Z), drink(milk)) = [X]

In this case, unification is not successful as the arities and functors do not match.

3.8 [eat(Z), drink(milk)] = [X, Y | Z]

In this case unification is successful with **X** being instantiated to **eat**([ ]), **Y** being instantiated to **drink(milk)**, and **Z** being instantiated to [ ].

3.9 f(X, t(b, c)) = f(l, t(Z, c))

In this case unification is successful with **X** being instantiated to **1**, and and **Z** being instantiated to **b**.

3.10 ancestor(french(jean), B) = ancestor(A, scottish(joe))

In this case unification is successful with **A** being instantiated to **french(jean)**, and and **B** being instantiated to **scottish(joe)**.

3.11 meal(healthyFood(bread), Y) = meal(X, drink(water))

In this case unification is successful with **X** being instantiated to **healthyFood(bread)**, and and **Y** being instantiated to **drink(water)**.

3.12 [H|T] = [a, b, c]

In this case unification is successful with **H** being instantiated to **a**, and **T** being instantiated to [**b, c**].

3.13 [H, T] = [a, b, c]

In this case unification is not successful as the number of elements in the left list does not equal the number of elements in the right list. The list on the left would have to be composed of 3 variables for unification to be successful.