Ling Xiang
cse465
HW3
3/11/2019

1. a. antecedent is the condition that forms the body of the query.
   consequent is the conclusion that forms the head of the query.

   b. Resolution means to write a rule to compute a certain computation.

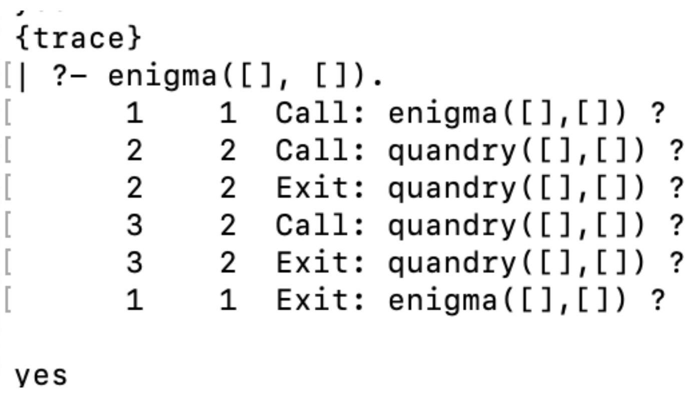
for example, in the relatives.pl, parent(X, Y) :- father(X, Y).
parent(X, Y) :- mother(X, Y).

means that parent(X, Y) needs X to be the father / mother of Y, in order to becomes Y's parent.

Unification means to match two terms.

For example, a = a will returns yes. It unifies two atoms to be the same.

c. Backtracking means the process. When fails, prolog will traces its steps backwards to the previous goal, and tries to find another query to make the query success. For example, the enigma prolog below. I trace the first one, it will send final answer from quandry([], []) back to enigma([], []).

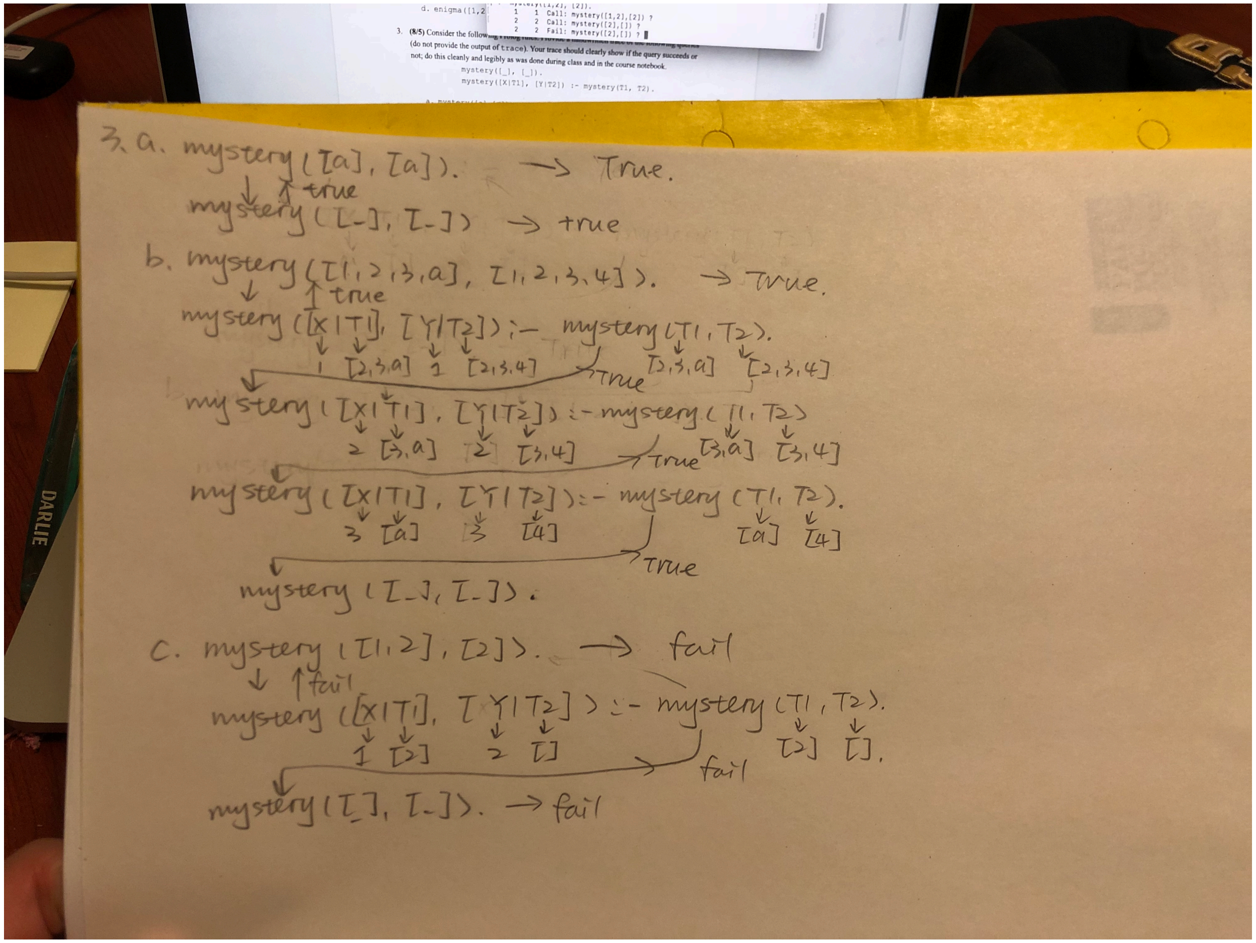


```
{trace}
[| ?- enigma([], []).
[      1    1  Call: enigma([],[]) ?
[      2    2  Call: quandry([],[]) ?
[      2    2  Exit: quandry([],[]) ?
[      3    2  Call: quandry([],[]) ?
[      3    2  Exit: quandry([],[]) ?
[      1    1  Exit: enigma([],[]) ?

yes
```

d. K is K + 1 is not assign, it is unification. It will try to unify if K is K + 1 which is false. To make this one succeed. Use another variable, S is K + 1.

2. true, true, false, true.



```
d. enigma([1,2
        1    1  Call: mystery([1,2],[2]) ?
        2    2  Call: mystery([2],[]) ?
        2    2  Fail: mystery([2],[]) ?
```

3. (8/5) Consider the following
(do not provide the output of `trace`). Your trace should clearly show if the query succeeds or not; do this cleanly and legibly as was done during class and in the course notebook.

```
mystery([_], [_]).
mystery([X|T1], [Y|T2]) :- mystery(T1, T2).
```

```
3. a. mystery([a], [a]).   -> True.
        ↓  ↑ true
      mystery([_], [_])  -> true

   b. mystery([1,2,3,a], [1,2,3,4]).   -> True.
        ↓  ↑ true
      mystery([X|T1], [Y|T2]) :- mystery(T1, T2).
               1 [2,3,a]  1 [2,3,4]   True  [2,3,a]  [2,3,4]
      mystery([X|T1], [Y|T2]) :- mystery(T1, T2)
               2 [3,a]   2 [3,4]    True  [3,a]  [3,4]
      mystery([X|T1], [Y|T2]) :- mystery(T1, T2).
               3 [a]     3 [4]       True           [a]  [4]
        mystery([_], [_]).

   c. mystery([1,2], [2]).   -> fail
        ↓  ↑ fail
      mystery([X|T1], [Y|T2]) :- mystery(T1, T2).
               1 [2]     2 []       fail    [2]  [].
      mystery([], [_]).  -> fail
```