Querying Family Relations

Database is reused from the last assignment.

Objectives :- Different kind of queries to be correctly answered by Program.

- **Query Type 1**

Input: " Who is the R of X? ".
Output: " The R of X is Y ".

Where R is a predicate and X is given like "who is the father of suresh"
Output should be ramesh is father of suresh. For this we simply defined a rule named part1, in which we have used apply/2 predicate(we could have also used call/2) which both of which takes goal or predicate as a first argument and depending to the syntax second argument is either lists of arguments for that predicate or simply arguments separated by commas.

- **Query Type 2**

Input: "Whose R is X".
Output: "X is R of Y".
Same idea as used in solving query type 1.

- **Query Type 3**

Input: "How is X related to Y".
Output: X is Y's [series of relations].

For this query type Breadth First Search is implemented along with Dynamic Database Programming using assert & retract predicates. Idea was adapted from slides of IIT Delhi mentioned in references.

- part3 predicate is created with arity 3 which takes query in the input form as written above as first argument. Third argument of part3 is a list containing the required relations which needs to be considered for the output if found. In short our program will only verify those goals which were stated in the list. Second argument is the output format. Part3 predicate calls two goals namely bfs and reverse.
- bfs predicate is created with arity 6 which instantiate start node, goal node, variable R initially declared zero however it can also take values as 1 and 2 and three lists one which contains the relations another which contains the path that needs to be reversed for correct output and one more list for path of nodes not necessarily the solution.
- bfs predicate has three subparts each gets triggered based on the values of R. When R is set to 0 which is by default then the bfs_main predicate is called and it would not explore other possibilities. When R is set to 1 then control gets back after executing bfs_main predicate. In case the goals for R==1 section is also false then control will move to the last section and execute the final lines of bfs predicate.
- We are generating a goal_vertex fact using builtin predicate assert at the run time so as the rules can later be verified if it has reached the goal_vertex or not from source node. In the next goal for the same bfs rule, it is checked if any queue fact with arity 2 exists in the program or not which was dynamically generated, If any such fact exists then predicate bfs_main predicate gets triggered.
- bfs_main predicate calls neighbour predicate in which it is checked whether any family relation can verify it or not with the help of check predicate. If it can be then queue fact is again generated and control gets back to bfs_main but if its not then all the generated queue facts are dynamically deleted using retractall builtin predicate and search fails naturally.

References:-

- http://www.cse.iitd.ac.in/~saroj/LFP/LFP_2013/L12.pdf
- http://stackoverflow.com/questions/8219555/what-is-the-difference-between-and-in-prolog
- http://stackoverflow.com/questions/24136833/how-can-i-use-a-functor-name-like-a-variable-in-prolog
- http://stackoverflow.com/questions/10563818/prolog-passing-a-function-as-a-variable-how-to-add-arguments
- http://www.swi-prolog.org/pldoc/man?section=metacall
- http://kti.ms.mff.cuni.cz/~bartak/prolog.old/learning/LearningProlog10.html
- http://www.swi-prolog.org/pldoc/doc_for?object=current_predicate/2
- http://www.cse.unsw.edu.au/~billw/dictionaries/prolog/functor.html