Programming Assignment 5: Logical Agent based on First Order Logic

Problem Description

A knowledge base (KB) consists of real world rules and is populated with the rules written in their own original form expressed in First Order Logic (FOL). Develop a logical agent to infer FOL queries using Forward Chaining. Implement the algorithm UNIFY(x,y,θ) that produces unifiers for the given knowledge base for given x and y values.

You are given two logical problems in this document. Formulate the predicates, write the rules and formulate queries using appropriate predicates from the vocabulary of the problem domain using human intelligence and knowledge of the real world. Test the implementation of above technique using these sets of rules and facts as specified in the logic problem description.

Implementation

Use Python version 2.7.13 (Windows 10) for implementing your solution. Only standard Python Libraries should be used. Support from external sources or libraries such as github will not be accepted in your submissions. Each student must design own solution and write own code.
[Refer handout to understand the malpractice policies.]

You can create objects of FOL such as **predicates** and **functions** appropriately. Specify two quantifiers as **For_every** and **There_exists** to define universal and existential quantifiers.

objects

1. **predicate** - object consisting of name and arguments of a predicate. Facts are the instances of predicates and rules. Also represent the variable names used in the predicate. Remember that these names of variables are one form of data for your algorithms which will take different values on substitution.

2. **rule**- is an object consisting of a body and head (body==>head) where body consists of sentences written in FOL.

3. **unifier**- It represents the substitution and is represented as a list of pair of values as discussed in the class.

Modules to implement FOL

A. Knowledge Base related

1. readPredicate(): returns a predicate(X, Y, Z,...) with a list of variables and constants. The name of the predicate is specified by the user. The function must take from each line of the file

named predicateFile#.txt, the names of predicate and variables. Consider the following file predicateFile0.txt with the description of the predicates

Friend(X,Y): X is friend of Y
Sells(X, Y, Z): X sells Z to Y
Person(X): X is a person

The function returns the name of the predicate as "Friend" and the list of variable symbols as "["X","Y"] for the first time it is invoked. When repeatedly invoked this function it returns the other predicate "Sells" and variables ["X","Y","Z"], and later predicate "Person" and variable ["X"]. Remember that names of variables are important for substitution and unification processes.

2. populate_FOL_KB(KB, filename), takes as input an empty KB and adds the rules from file *filename* to KB. It returns the new KB. You should initialize the KB to an empty structure and populate one by one reading the rules from the file ruleFile#.txt. Initial KB receives rules in their original form. For example, a rule describing "brothers are siblings" expressed in FOL is

$\forall$ X $\forall$Y (Brother(X,Y)$\Rightarrow$Sibling(X,Y))

This rule is expressed in file ruleFile#.txt as

For_every X **For_every** Y (Brother(X,Y)$\Rightarrow$Sibling(X,Y))

Once the rule is populated in the KB, one can process the rule for different purposes such testing whether the rule is a horn/definite clause or not(as needed). It is expected that the user writes all rules in the file rulesFile#.txt after manually standardizing the variables. You are expected to process the quantifiers For_every and There_exists appropriately.

B. Modules for Inference

1. Unify(x, y, theta): Implement algorithm given in Fig. 9.1 as discussed in the class. The function takes as input x and y, where x and y can be the compound statements, lists, variables or constants. theta is the set of pairs of variable/ value required for substituion.

2. ForwardChaining_FOL(KB, facts, query): returns the truth value of the query. Refer FOL_FC_ASK( ) algorithm given in Fig. 9.3 of the text book and implement the algorithm.

Data generation

Write the predicates clearly for the following two logic problems in files predicateFile#.txt with their corresponding English description.

Logic Problem 1

Marcus was a human. Marcus was a Pompeian. Marcus was born in 40 A.D. All humans are mortal. All Pompeians died when the volcano erupted in 79 A.D. No mortal lives longer than 150 years. It is now 2017. Alive means not dead. If someone dies, then he is dead at all later times. All Pompeians were Roman. Ceaser was a ruler. All Romans were either loyal to Ceaser or hated him. Everyone is loyal to someone. People only try to assassinate rulers they are not loyal to. Marcus assassinated Ceaser. Is Marcus alive now? Who assassinated Ceaser? Was Marcus loyal to Ceaser? Who was the ruler? When did the volcano erupt? Was Marcus dead in 60 A.D.? Did Marcus hate Ceaser? Was Marcus alive in 35 A.D.?

Logic Problem 2

Consider the following models of the wumpus world. The percepts are breeze, stench and glitter to represent the presence of pit, wumpus and gold. A wumpus creates stench in all its 4 horizontal and vertical neighboring squares. A pit sends breeze in all 4 of its horizontal and vertical squares. The glitter percept is only true in the square containing gold. The termination condition of the game is that the logical agent gets either caught by the Wumpus, or falls in the pit or finds the gold. Produce message appropriately about one of the three conditions reached. The agent moves using its knowledge either available initially, through the facts (percepts' truth values when logical agent reaches the particular square) and by intermediate additions of inferences in the knowledge base. The environment is partially observable and the logical agent is able to sense only the square which it is in, but has a complete access to its evolving knowledge base through which it uses its logic to move right, move left, move forward or backward. For example, if it senses a stench in a square, it must step backward as there could be a wumpus in any of the three squares on its left, right or front. You can refer to the wumpus world description in FOL given in section 8.3 for more clarity. As a programmer, you are expected to create the wumpus world rules in FOL, and display the path taken by the logical agent using the inference algorithms. You can either hardcode the percepts for these two models alone with specified number of wumpuses and pits, in their specified locations and capture the stench and breeze as two dimensional arrays, or you can randomize the environment creation. Remember that the logical agent is able to perceive the percepts only in the square it is in.

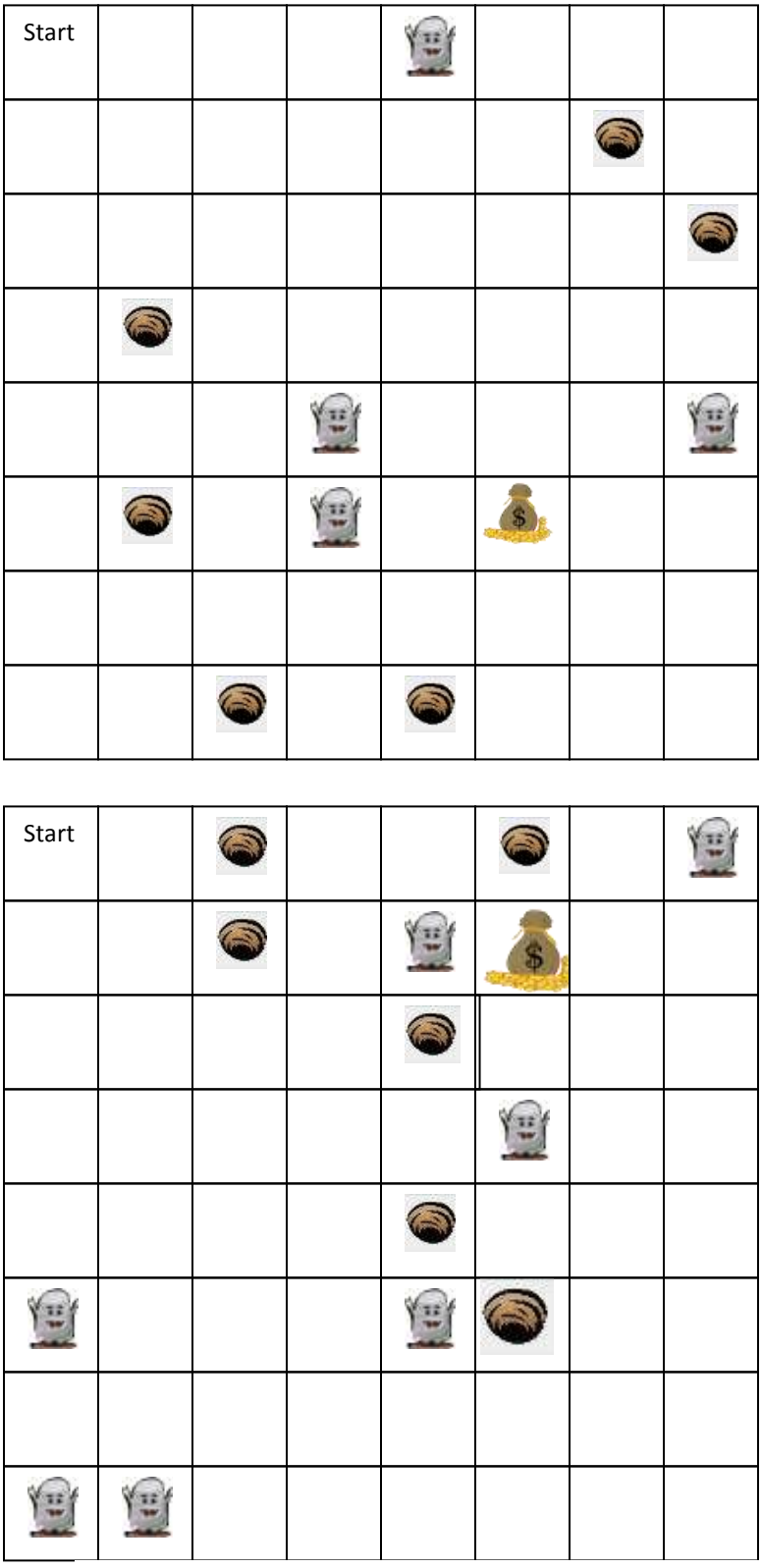


Fig.1. Wumpus World - two different models

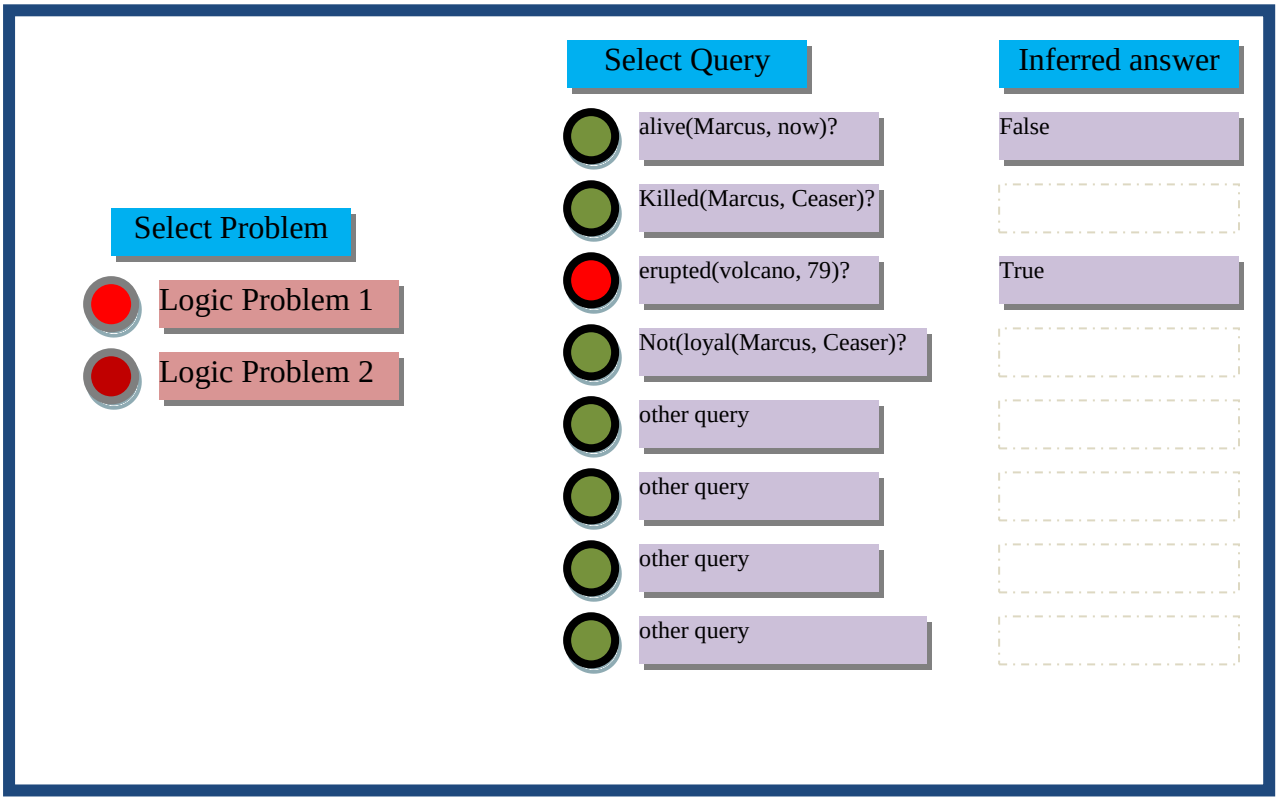
File handling

Generate English descriptions of predicates and rules in FOL in the files *manually* by typing the details in files *predicateFile#.txt* and *ruleFile#.txt* respectively where # is the number of the logic problem. The naming must follow the following scheme.

Logic Problem	English description of predicates	Rules in FOL
1	predicateFile1.txt	ruleFile1.txt
2	predicateFile2.txt	ruleFile2.txt

Graphics

Create a graphical user interface (GUI) to use buttons for the selection of logic problem (1-2), and queries formulated by you.



Leave the answers to the queries once invoked/selected on the screen so as to enable the visibility of the answers to all the queries till the end. Ensure that the selections display highlighted buttons (as shown with Red in the above sample GUI). When a different logic problem is selected, clear the query and inferred answer area and refresh with set of queries pertaining to the selected logic problem. If all the queries are not fit in the space, use next button to refresh the previous set of queries and display remaining set of queries pertaining to the same problem. If the logic problem 2 of wumpus world is selected, then after the queries are displayed,

prompt the user to see the path. Display the logical path taken by the logical agent in problem 2 after refreshing the query area.

Driver

The driver must integrate all functionalities and execute the functions appropriately using the selections made through the above GUI.

Writeup, evaluation and submission

Write up details will be made available two days before the submission. Evaluation will be out of 16 marks (8% weight). Students are advised to inform me immediately, if any discrepancy exists in this document. The assignment is due for submission on November 15, 2018 (Thursday) by 6:30 p.m. The students are expected to read the text book chapters 8 and 9 thoroughly and clarify all their doubts pertaining to the problem specification, explanations given above, conceptual understanding, doubts related to individual logic problem or the data structures to be used, and the doubts relating other aspects of implementation.

Please feel free to meet me and discuss your doubts.

Vandana
November 5, 2018