Lab Sheet 3-TicTac Toe game

1. Define an ontology in first-order logic for tic-tac-toe using aima package. The ontology should contain situations, actions, squares, players, marks (X, O, or blank), and the notion of winning, losing, or drawing a game. Also define the notion of a forced win (or draw): a position from which a player can force a win (or draw) with the right sequence of actions. Write axioms for the domain. (Note: The axioms that enumerate the different squares and that characterize the winning positions are rather long. You need not write these out in full, but indicate clearly what they look like.)

The sample for creating the ontology is given below:

from aima.logic import expr

# Define constants for players, marks, and actions

X, O, BLANK = 'X', 'O', ' '

# Define symbols for the domain

Situation, Action, Square, Player, Mark = expr('Situation'), expr('Action'), expr('Square'), expr('Player'), expr('Mark')

# Define winning combinations (axioms for winning)

winning\_combinations = [

# Rows

[(Square(1, 1), Mark(X)), (Square(1, 2), Mark(X)), (Square(1, 3), Mark(X))],

[(Square(2, 1), Mark(X)), (Square(2, 2), Mark(X)), (Square(2, 3), Mark(X))],

[(Square(3, 1), Mark(X)), (Square(3, 2), Mark(X)), (Square(3, 3), Mark(X))],

# Columns

[(Square(1, 1), Mark(X)), (Square(2, 1), Mark(X)), (Square(3, 1), Mark(X))],

[(Square(1, 2), Mark(X)), (Square(2, 2), Mark(X)), (Square(3, 2), Mark(X))],

[(Square(1, 3), Mark(X)), (Square(2, 3), Mark(X)), (Square(3, 3), Mark(X))],

# Diagonals

[(Square(1, 1), Mark(X)), (Square(2, 2), Mark(X)), (Square(3, 3), Mark(X))],

[(Square(1, 3), Mark(X)), (Square(2, 2), Mark(X)), (Square(3, 1), Mark(X))]

]

# Define axioms for winning

def is\_winning(situation, player):

for combination in winning\_combinations:

if all((expr\_in\_situation(situation, square, mark) and mark == Mark(player)) for square, mark in combination):

return True

return False

# Define axioms for forced win (or draw)

def can\_force\_win(situation, player):

# Define axioms for forced win here

pass

# Define function to check if a given square and mark are in a situation

def expr\_in\_situation(situation, square, mark):

return expr('In')(square, situation) & expr('Mark')(square, mark)

# Example usage

current\_situation = Situation()

current\_situation.append(expr('In')(Square(1, 1), Mark(X)))

current\_situation.append(expr('In')(Square(1, 2), Mark(O)))

current\_situation.append(expr('In')(Square(2, 2), Mark(X)))

print(is\_winning(current\_situation, X))

print(is\_winning(current\_situation, O))

Write a program so as to simulate the tic tac toe game using the above definitions and ontology. You can use necessary inerfaces and packages to make the game more attractive!!