**Reasoning with Propositional Logic**

Wumpus World Problem is taken for demonstrating the reasoning with propositional logic. A knowledge-based agent is developed which will solve the Wumpus World Problem with the help of propositional logic.

The Wumpus world is a 4\*4 grid connected with passageways. So, there are 16 grids connected to each other. We have a knowledge-based agent which will go forward in this world. The grid may have Wumpus which will kill anyone who enters it. It may also have bottomless pits and if agent falls in Pits, then he will be stuck there forever. The exciting thing is that there is one grid which has a heap of gold. So, the agent goal is to find the gold without fallen into Pits or eaten by Wumpus.

**There are also some components which can help the agent to navigate the path. These components are given as follows:**

1. The grids adjacent to the Wumpus grid have stench smell.
2. The grids adjacent to PITs has are breezy.

There are some actions an agent can do:

1. Move Up
2. Move Right

In our system we have initial knowledge base:

KB = [ ('S',1,1,False),('B',1,1,False),('W',1,1,False),('P',1,1,False)]

Which means at the beginning of the game our agent will start from grid (1,1) and there will be no stench, no breeze, no Wumpus and no pit in grid (1,1). Here S , B , W and P stands for stench, breeze, Wumpus and pit respectively.

We have applied some propositional rules like Modus Ponens, And Elimination and Unit Resolution to find the safe grids so that agent does not get killed by Wumpus or pit. Some rule we have applied in this system are:

(R1) Si,j Wi,j+1 Wi,j-1 Wi-1,j Wi+1,j

(R2) Bi,j Pi,j+1 Pi,j-1 Pi-1,j Pi+1,j

(R3) Si,j Wi,j+1 V Wi,j-1 V Wi-1,j V Wi+1,j

(R4) Bi,j Pi,j+1 V Pi,j-1 V Pi-1,j V Pi+1,j

**The demonstrated Python code of Reasoning with Propositional Logic for solving Wumpus World Problem is as below:**

# S = Stentch , B = Breeze , W = Wumpus , P = Pit , G = Gold , A = Players's initial position

grid = [('S',1,2), ('S',1,4), ('S',2,3),('B',2,1), ('B',2,3), ('B',3,2), ('B',3,4), ('B',4,1), ('B',4,3),

('W',1,3),('P',3,1), ('P',3,3) , ('G',2,3) ]

KB = [ ('S',1,1,False),('B',1,1,False),('W',1,1,False),('P',1,1,False)]

xx = [1,0]

yy = [0,1]

gd = []

#checking for grid boundary conditions

def in\_boundary(x,y):

if(x>=0 and x<=4 and y>=0 and y<=4):

return True

else:

return False

def start(y,x,KB):

print('Entering into Safe Grid : ('+str(y)+','+str(x)+')')

pos = ('A',y,x,True)

KB.append(pos)

#if gold is found return the grid

if( ('G', y , x) in grid ):

global gd

gd = (y,x)

return

if( ('W', y , x) in grid ):

printf('Killed by Wumpus')

return

if( ('P', y , x) in grid ):

printf('Fell into Pit')

return

for i in range(2):

p\_y = y+yy[i]

p\_x = x+xx[i]

if( in\_boundary(p\_y,p\_x) ):

#applying Modus Ponens, And Elimination, Unit Resolution

if( ('S', y , x ) not in grid and ('W',p\_y,p\_x,False) not in KB):

if(('W',p\_y,p\_x,True) in KB):

KB.remove( ('W',p\_y,p\_x,True) )

KB.append( ('W',p\_y,p\_x,False) )

if( ('B', y, x) not in grid and ('P',p\_y,p\_x,False) not in KB):

if(('P',p\_y,p\_x,True) in KB):

KB.remove( ('W',p\_y,p\_x,True) )

KB.append( ('P',p\_y,p\_x,False) )

if( ('S', y , x ) in grid and ('W',p\_y,p\_x,False) not in KB):

KB.append( ('W',p\_y,p\_x,True) )

if( ('B', y, x ) in grid and ('P',p\_y,p\_x,False) not in KB):

KB.append( ('P',p\_y,p\_x,True) )

if(pos in KB):

KB.remove(pos)

for i in range(2):

p\_y = y+yy[i]

p\_x = x+xx[i]

if( in\_boundary(p\_y,p\_x) and gd == [] ):

if ( ('W',p\_y,p\_x,False) in KB and ('P',p\_y,p\_x,False) in KB):

start(p\_y,p\_x, KB)

#Main

print('Initial Knowldege Base: ')

print(KB)

print()

start(1,1,KB)

print('\n!! Gold Found in '+str(gd)+' !!\n')

print('Knowldege Base After Execution: ')

print(KB)

**A sample input and output is as below:**

