**WUMPUS WORLD**

import itertools

# Input KB

num\_sentences = int(input("Enter number of KB sentences: "))

kb\_sentences = []

all\_vars = set()

for i in range(num\_sentences):

s = input(f"Enter sentence {i+1} (use !, ∧, ∨, ->, ↔): ")

kb\_sentences.append(s)

# extract variable names (alphanumeric strings)

tokens = s.replace("(", " ").replace(")", " ").replace("!", " ").replace("∧", " ").replace("∨", " ").replace("->", " ").replace("<->", " ").split()

for t in tokens:

if t.isalnum() and not t.isdigit():

all\_vars.add(t)

# Input queries

queries\_input = input("Enter queries separated by comma: ")

queries = [q.strip() for q in queries\_input.split(",")]

for q in queries:

all\_vars.add(q)

# Prepare the KB expressions in Python syntax

kb\_py = []

for s in kb\_sentences:

s\_py = s

s\_py = s\_py.replace("∧", " and ").replace("∨", " or ").replace("!", " not ").replace("->", "<=").replace("<->", "==")

kb\_py.append(s\_py)

# Prepare queries in Python syntax

queries\_py = []

for q in queries:

q\_py = q

queries\_py.append(q\_py)

# Generate all truth assignments

var\_list = list(all\_vars)

# Print truth table header

header = " ".join(var\_list) + " " + " ".join([f"{s}" for s in kb\_sentences]) + " KB True?"

print("\nTruth Table:")

print(header)

print("-"\*len(header.expandtabs()))

# Fill truth table

for values in itertools.product([False, True], repeat=len(var\_list)):

env = dict(zip(var\_list, values))

kb\_values = [eval(expr, {}, env) for expr in kb\_py]

kb\_true = all(kb\_values)

row = " ".join([str(env[v]) for v in var\_list]) + " " + " ".join([str(val) for val in kb\_values]) + " " + str(kb\_true)

print(row)

# Show models where KB is True

print("\nModels where KB is True:")

for values in itertools.product([False, True], repeat=len(var\_list)):

env = dict(zip(var\_list, values))

if all(eval(expr, {}, env) for expr in kb\_py):

print({k:v for k,v in env.items()})

# Check entailment for queries

print("\nEntailment results:")

for q, q\_py in zip(queries, queries\_py):

entails = all(not all(eval(expr, {}, dict(zip(var\_list, vals))) for expr in kb\_py) or eval(q\_py, {}, dict(zip(var\_list, vals))) for vals in itertools.product([False, True], repeat=len(var\_list)))

print(f"Does KB entail {q}? --> {entails}")



