1. FOL FORWARD CHAINING

**INPUT:**

class KnowledgeBase:

def \_\_init\_\_(self):

# Store facts and rules

self.facts = set() # Set of known facts

self.rules = [] # List of rules (condition -> conclusion)

def add\_fact(self, fact):

""" Add a fact to the knowledge base """

self.facts.add(fact)

def add\_rule(self, condition, conclusion):

""" Add a rule to the knowledge base. Condition is a callable that checks if a rule is applicable. """

self.rules.append((condition, conclusion))

def forward\_reasoning(self):

""" Perform forward reasoning to derive new facts """

new\_facts = set(self.facts)

while True:

added = False

for condition, conclusion in self.rules:

if condition(self.facts): # If the condition is met based on current facts

if conclusion not in self.facts: # If conclusion is not already a fact

self.facts.add(conclusion)

new\_facts.add(conclusion)

added = True

if not added:

break # No new facts added, stop the reasoning

return new\_facts

def get\_input():

""" Function to get user input for facts and rules """

kb = KnowledgeBase()

print("Enter facts (type 'done' to finish):")

while True:

fact = input("Fact: ").strip()

if fact.lower() == 'done':

break

kb.add\_fact(fact)

print("\nEnter rules (condition -> conclusion, type 'done' to finish):")

while True:

rule\_input = input("Rule: ").strip()

if rule\_input.lower() == 'done':

break

# Example rule format: "IsHuman(John) -> HasLegs(John)"

if '->' in rule\_input:

condition, conclusion = rule\_input.split('->')

condition = condition.strip()

conclusion = conclusion.strip()

# Add a rule as a lambda function for condition -> conclusion

kb.add\_rule(lambda facts: condition in facts, conclusion)

else:

print("Invalid rule format. Please enter in the form: condition -> conclusion")

return kb

# Main interactive loop

def main():

print("Welcome to the Forward Reasoning System!\n")

kb = get\_input()

# Perform forward reasoning to derive new facts

kb.forward\_reasoning()

print("\nAll derived facts:")

for fact in kb.facts:

print(fact)

# Ask the user for a query

query = input("\nEnter a query to check if it's a fact (e.g., HasLegs(John)): ").strip()

if query in kb.facts:

print(f"Yes, {query} is a fact.")

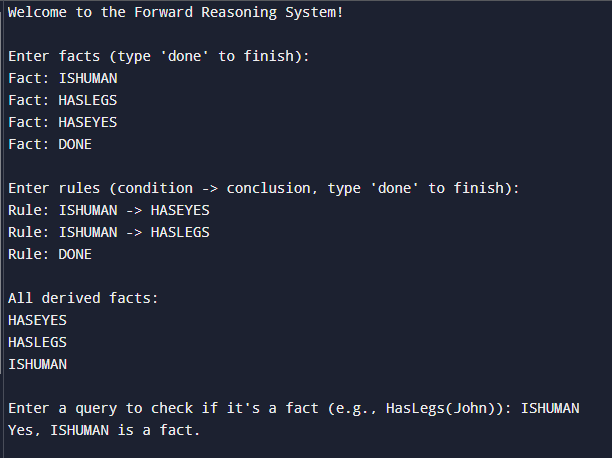
else:

print(f"No, {query} is not a fact.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**OUTPUT:**

****