Backward Chaining

class BackwardChaining:

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

self.facts = set() # Known facts

self.rules = [] # Inference rules

def add\_fact(self, fact):

self.facts.add(fact)

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

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

def backward\_chain(self, goal):

if goal in self.facts:

return True

for conditions, conclusion in self.rules:

if conclusion == goal:

# Check if all conditions can be satisfied

if all(self.backward\_chain(cond) for cond in conditions):

return True

return False

def get\_input\_backward():

bc = BackwardChaining()

n = int(input("Enter number of facts: "))

for \_ in range(n):

fact = input("Enter fact: ")

bc.add\_fact(fact)

m = int(input("Enter number of rules: "))

for \_ in range(m):

rule\_input = input("Enter rule in format 'if fact1, fact2 then conclusion': ").split(" then ")

conditions = set(rule\_input[0].split(", "))

conclusion = rule\_input[1].strip()

bc.add\_rule(conditions, conclusion)

goal = input("Enter the goal to prove: ")

result = bc.backward\_chain(goal)

print(f"Can we prove {goal}? {'Yes' if result else 'No'}")

# Main driver

get\_input\_backward()

OUTPUT:

Enter number of facts: 2

Enter fact: A

Enter fact: B

Enter number of rules: 2

Enter rule in format 'if fact1, fact2 then conclusion': A, B then C

Enter rule in format 'if fact1, fact2 then conclusion': C then D

Enter the goal to prove: D

Can we prove D? Yes