**Aim**

To implement a backward chaining algorithm in Python for an expert system that deduces conclusions based on predefined rules.

**Procedure**

1. **Define Knowledge Base**: Establish a set of rules that represent logical relationships.
2. **Set Goal**: Identify the target fact or conclusion to be proven.
3. **Check Rules**: Examine rules that lead to the goal and verify if their premises are satisfied.
4. **Recursive Evaluation**: If premises are not directly known, recursively check their validity.
5. **Decision Making**: If all conditions are met, conclude the goal is true; otherwise, it remains unproven.

**Code**

class ExpertSystem:

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

self.rules = rules

def backward\_chaining(self, goal, known\_facts):

if goal in known\_facts:

return True

for rule in self.rules:

if rule["conclusion"] == goal:

if all(self.backward\_chaining(premise, known\_facts) for premise in rule["premises"]):

return True

return False

# Define rules

rules = [

{"premises": ["vertebrate"], "conclusion": "mammal"},

{"premises": ["mammal", "flying"], "conclusion": "bat"},

{"premises": ["vertebrate", "flying"], "conclusion": "bird"}

]

# Known facts

known\_facts = ["vertebrate", "flying"]

# Initialize expert system

expert\_system = ExpertSystem(rules)

# Test backward chaining

goal = "bat"

result = expert\_system.backward\_chaining(goal, known\_facts)

print(f"Is '{goal}' inferred? {result}")

**Output**

****