EXPERIMENT -9

Objective: IMPLEMENT BACKWARD CHAINING ALGORITHM

Theory :

Backward Chaining is a goal-driven reasoning technique used in expert systems.

It starts from a goal (hypothesis) and works backwards to determine if the goal can be proven from known facts.

Principle:

Based on Modus Tollens / backward reasoning:

If “IF P THEN Q” is true, and we want to prove Q, then check whether P is true.

Components:

1) Knowledge Base (KB): A set of rules (IF <conditions> THEN <conclusion>).
2) Working Memory (WM): Contains known facts.
3) Goal: The fact to be proven true.

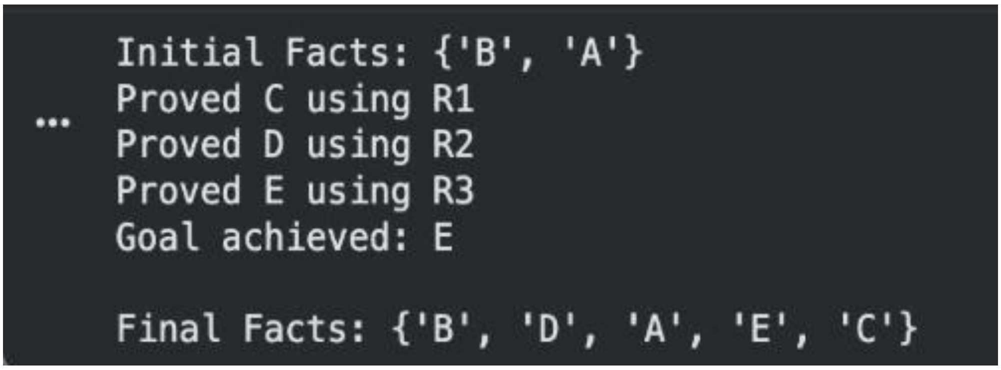
Algorithm :

1. Start with the goal you want to prove.
2. If the goal is already in the known facts → success.
3. Else, find rules whose conclusion matches the goal.
4. For each such rule, make sub-goals from its conditions.
5. Recursively try to prove all sub-goals.
6. If all sub-goals are true → goal is proven.
7. If no rule can prove the goal → fail.

Code :

```
rules = {
    "R1": ({"A", "B"}, "C"),
    "R2": ({"C"}, "D"),
    "R3": ({"D"}, "E")
  }
facts = {"A", "B"}
goal = "E"
  def
    backward_chain(goal):
    if goal in facts:
      return True
    for rule, (conditions, conclusion) in
  rules.items():
      if conclusion == goal:
        if all(backward_chain(c) for c
  in conditions):
          facts.add(goal)
          print(f"Proved {goal} using
  {rule}")
          return True
    return False
print("Initial Facts:", facts) if
backward_chain(goal):
    print("Goal achieved:", goal)
  else:
    print("Goal cannot be proven.")
  print("\nFinal Facts:", facts)
```

Output -



```
...  Initial Facts: {'B', 'A'}
     Proved C using R1
     Proved D using R2
     Proved E using R3
     Goal achieved: E

     Final Facts: {'B', 'D', 'A', 'E', 'C'}
```