AIM:

To implement Forward and Backward Chaining in Ai using Python Programming Language.

Rules:

* + If A and B then C
  + If C and D then E
  + If E then F
* **Initial facts**: A, B, D

Code:

# Knowledge Base

rules = [

{"if": ["A", "B"], "then": "C"},

{"if": ["C", "D"], "then": "E"},

{"if": ["E"], "then": "F"}

]

facts = {"A", "B", "D"}

# Forward Chaining

def forward\_chaining(rules, facts):

inferred = set(facts)

added = True

while added:

added = False

for rule in rules:

if set(rule["if"]).issubset(inferred) and rule["then"] not in inferred:

inferred.add(rule["then"])

print(f"Inferred: {rule['then']}")

added = True

return inferred

# Backward Chaining

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

if goal in known\_facts:

print(f"{goal} is already known.")

return True

for rule in rules:

if rule["then"] == goal:

print(f"Trying to prove: {goal} using rule {rule}")

if all(backward\_chaining(fact, rules, known\_facts) for fact in rule["if"]):

known\_facts.add(goal)

print(f"Proved: {goal}")

return True

print(f"Failed to prove: {goal}")

return False

# --- Run Forward Chaining ---

print("=== Forward Chaining ===")

final\_facts = forward\_chaining(rules, facts)

print("Final facts:", final\_facts)

# --- Run Backward Chaining ---

print("\n=== Backward Chaining ===")

goal = "F"

known\_facts = set(facts)

result = backward\_chaining(goal, rules, known\_facts)

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

Output:

=== Forward Chaining ===

Inferred: C

Inferred: E

Inferred: F

Final facts: {'F', 'C', 'D', 'E', 'B', 'A'}

=== Backward Chaining ===

Trying to prove: F using rule {'if': ['E'], 'then': 'F'}

Trying to prove: E using rule {'if': ['C', 'D'], 'then': 'E'}

Trying to prove: C using rule {'if': ['A', 'B'], 'then': 'C'}

A is already known.

B is already known.

Proved: C

D is already known.

Proved: E

Proved: F

Can we derive F? Yes

Result:

Thus the Forward and Backward Chaining Implementation is verified Successfully.