Lab Assignment 8

Subject: Artificial Intelligence **Guided by:** Dr. Anuradha Yenkikar

Student name: Neha Sawant

Experiment Name: Implement backward chaining algorithm.

Objective:

To implement the backward chaining algorithm, a fundamental technique in artificial intelligence used for reasoning in rule-based systems. The goal is to infer new information based on a specific goal, starting from known facts and applying rules recursively.

Problem Statement:

In this exercise, you will simulate a simple expert system that uses backward chaining to determine if a specific goal can be inferred from a set of known facts and defined rules. The system will attempt to prove the goal by checking if it can be derived from the existing knowledge.

Requirements:

- Programming Language: Python

- Environment: Any Python IDE (e.g., PyCharm, Jupyter Notebook)

- Python Version: 3.6 or higher

Code Explanation:

1] Code Overview

The provided code defines a backward chaining algorithm that attempts to infer a goal based on initial facts and rules.

2] Components of the Code

- Facts: A list of known facts. In this case, the facts are `["A", "B"]`.

- Rules: A list of tuples, where each tuple contains a list of conditions and a conclusion.

  - For example, `(["A", "B"], "C")` means that if both A and B are true, then C can be inferred.

- backward_chaining(goal, facts, rules):

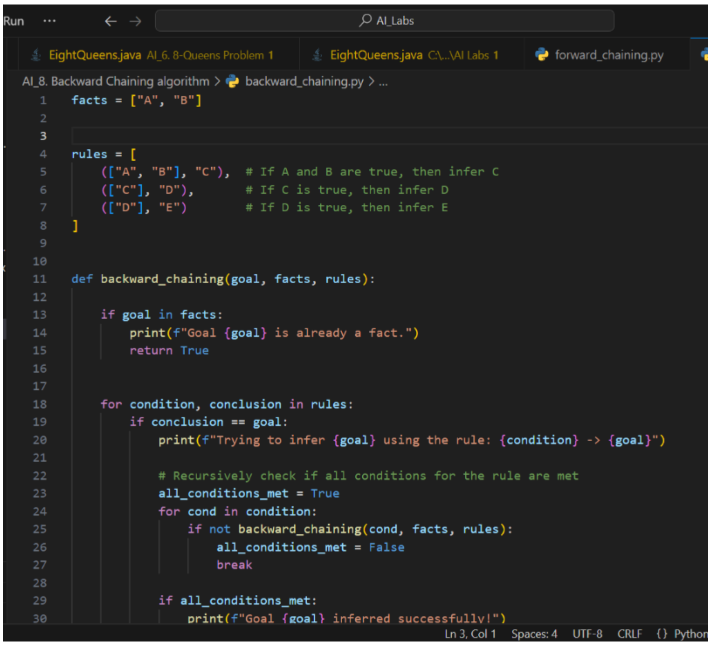
  This function implements the backward chaining algorithm:

  - It checks if the goal is already a known fact.

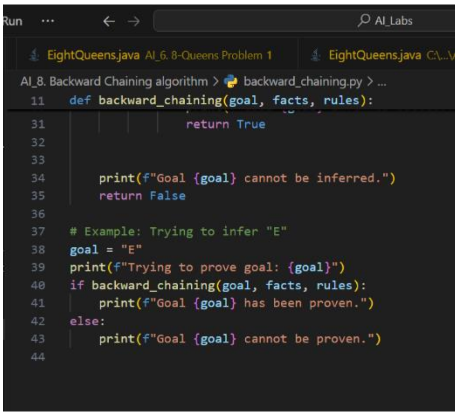
  - If not, it iterates through the rules to find one that concludes the goal and checks if all conditions for that rule can be satisfied recursively.

3] Code Implementation

Here’s the complete implementation of the backward chaining algorithm:


Run
AI_Labs
EightQueens.java AI_6. 8-Queens Problem 1
EightQueens.java C:\...\AI Labs 1
forward_chaining.py
AI_8. Backward Chaining algorithm > backward_chaining.py > ...
Ln 3, Col 1 Spaces: 4 UTF-8 CRLF {} Python

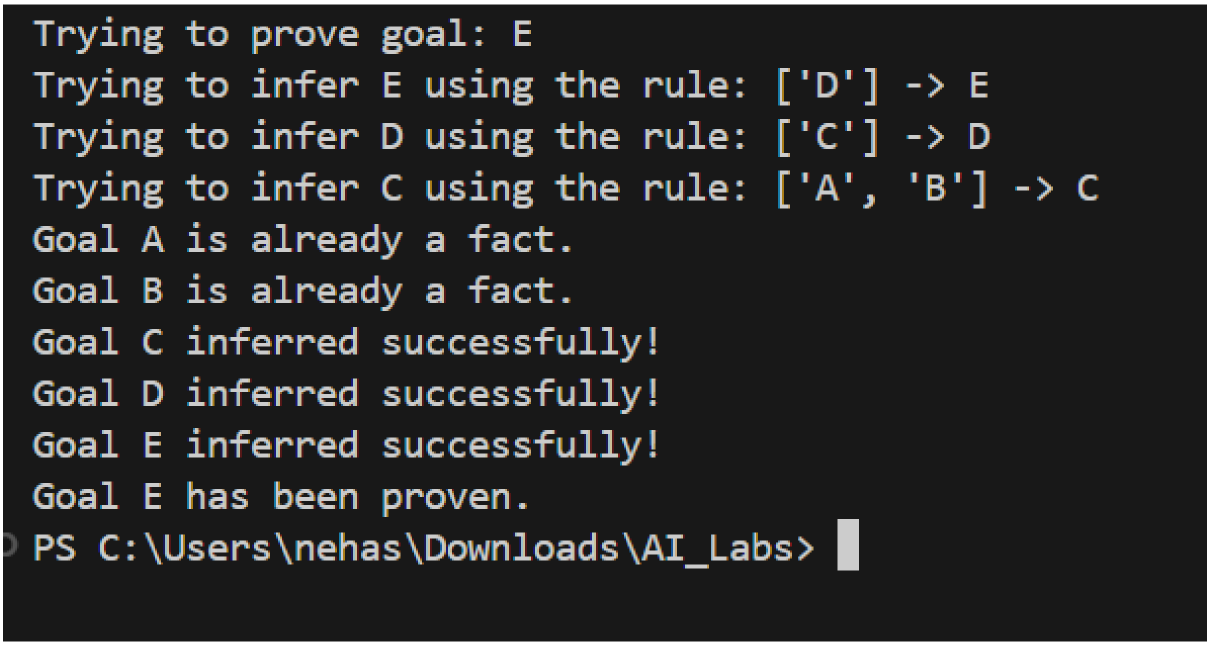
```python
facts = ["A", "B"]


rules = [
    (["A", "B"], "C"),  # If A and B are true, then infer C
    (["C"], "D"),       # If C is true, then infer D
    (["D"], "E")        # If D is true, then infer E
]


def backward_chaining(goal, facts, rules):

    if goal in facts:
        print(f"Goal {goal} is already a fact.")
        return True


    for condition, conclusion in rules:
        if conclusion == goal:
            print(f"Trying to infer {goal} using the rule: {condition} -> {goal}")

            # Recursively check if all conditions for the rule are met
            all_conditions_met = True
            for cond in condition:
                if not backward_chaining(cond, facts, rules):
                    all_conditions_met = False
                    break

            if all_conditions_met:
                print(f"Goal {goal} inferred successfully!")
```


Run
AI_Labs
EightQueens.java AI_6. 8-Queens Problem 1
EightQueens.java C:\...\
AI_8. Backward Chaining algorithm > backward_chaining.py > ...

```python
def backward_chaining(goal, facts, rules):
                return True


        print(f"Goal {goal} cannot be inferred.")
        return False

# Example: Trying to infer "E"
goal = "E"
print(f"Trying to prove goal: {goal}")
if backward_chaining(goal, facts, rules):
    print(f"Goal {goal} has been proven.")
else:
    print(f"Goal {goal} cannot be proven.")
```

Output:

The program will output messages indicating whether the goal can be inferred and any intermediary steps taken to infer it. An example output might look like this:



```
Trying to prove goal: E
Trying to infer E using the rule: ['D'] -> E
Trying to infer D using the rule: ['C'] -> D
Trying to infer C using the rule: ['A', 'B'] -> C
Goal A is already a fact.
Goal B is already a fact.
Goal C inferred successfully!
Goal D inferred successfully!
Goal E inferred successfully!
Goal E has been proven.
PS C:\Users\nehas\Downloads\AI_Labs>
```

Working of the Code:

1. Initialization: The initial facts and rules are defined.

2. Inference Process:

- The algorithm first checks if the goal is already a known fact.

- If not, it searches for a rule that concludes the goal and recursively checks if all conditions for that rule can be satisfied.

- If all conditions are met, it concludes that the goal can be inferred.

3. Result Display: Finally, it prints whether the goal has been proven and any relevant inference steps.

Conclusion:

In this lab, I have successfully implemented the backward chaining algorithm to infer new information in a rule-based system. This exercise has enhanced my understanding of rule-based reasoning and how to implement logical inference using programming.