**Assignment 8**

**Title: Implementation of Backward Chaining Algorithm**

**Theory**

**Backward Chaining** is an **inference strategy** used in **Artificial Intelligence** and **Expert Systems** to reason from goals (or hypotheses) backward to known facts.

It starts with a **goal (conclusion)** and recursively checks whether there are rules and facts in the knowledge base that can support (prove) that goal.  
If a rule’s **consequent** matches the goal, the algorithm attempts to prove all its **antecedents**.  
If all antecedents are proven true, then the goal is confirmed.

Backward chaining is typically used in **diagnostic systems**, **medical expert systems**, and **logic-based reasoning**, where we ask:

“Can this conclusion be proved from what we know?”

**Definition**

Backward Chaining is a **goal-driven reasoning approach** that works as follows:

1. Start with the **goal (query)** to prove.
2. Search for rules where this goal appears as the **conclusion (THEN part)**.
3. For each such rule, recursively prove all its **antecedents (IF conditions)**.
4. If all antecedents can be proven using facts or other rules, the goal is proven true.
5. If no rules or facts can prove the goal, it fails.

**Logical form:**

If (A ∧ B) ⇒ C

Goal: Prove C

Check: Can A and B be proven?

**Explanation of Implementation**

This program demonstrates **Backward Chaining** using a medical diagnosis example, where the system tries to prove if a patient has **Flu** based on known symptoms and rules.

**Step 1: Knowledge Representation**

Knowledge is represented as a set of **rules** and **facts**:

* **Rule Structure:**  
  IF antecedents → THEN consequent

*Example:*

IF Fever ∧ Cough THEN ViralInfection

IF ViralInfection ∧ BodyAche THEN FluSymptoms

IF FluSymptoms THEN Flu

* **Facts (Known Information):**  
  {Fever, Cough, BodyAche}

**Step 2: Indexing Rules**

To efficiently find rules that produce a particular conclusion, the algorithm creates an **index** mapping each consequent to its rules.

*Pseudocode:*

FUNCTION index\_rules(rules):

FOR each rule in rules:

index[rule.consequent].append(rule)

RETURN index

**Step 3: Backward Proving Procedure**

The **core recursive function** backward\_prove(goal) works as follows:

*Pseudocode:*

FUNCTION backward\_prove(goal):

IF goal is a known fact:

RETURN true

IF goal has already been proved:

RETURN true

IF goal is being visited:

RETURN false // prevents infinite loops

FIND all rules where consequent = goal

IF no such rules:

RETURN false

FOR each rule producing goal:

FOR each antecedent in rule:

IF backward\_prove(antecedent) == false:

FAIL this rule

IF all antecedents proved:

ADD goal to facts

RETURN true

RETURN false

The function recursively checks antecedents until either:

* The goal is confirmed (proved), or
* No more rules can prove it.

**Step 4: Reasoning Trace**

During execution, the algorithm maintains a **trace log** showing the reasoning process:

* Known facts identified
* Rules being tested
* Antecedents being proved or failed
* Final conclusion

This makes the reasoning transparent, similar to expert systems like MYCIN.

**Output**

**Knowledge Base:**

1. IF Fever ∧ Cough THEN ViralInfection

2. IF ViralInfection ∧ BodyAche THEN FluSymptoms

3. IF FluSymptoms THEN Flu

**Facts (Symptoms):**

Fever, Cough, BodyAche

**Goal:**

Flu

**Sample Output:**

Backward Chaining Trace:

TRY PROVING: Flu

Try rule: FluSymptoms ⇒ Flu

TRY PROVING: FluSymptoms

Try rule: ViralInfection ∧ BodyAche ⇒ FluSymptoms

TRY PROVING: ViralInfection

Try rule: Fever ∧ Cough ⇒ ViralInfection

TRY PROVING: Fever

KNOWN FACT: Fever

-> antecedent PROVED: Fever

TRY PROVING: Cough

KNOWN FACT: Cough

-> antecedent PROVED: Cough

RULE SUCCEEDED -> proved: ViralInfection

-> antecedent PROVED: ViralInfection

TRY PROVING: BodyAche

KNOWN FACT: BodyAche

-> antecedent PROVED: BodyAche

RULE SUCCEEDED -> proved: FluSymptoms

-> antecedent PROVED: FluSymptoms

RULE SUCCEEDED -> proved: Flu

Result: PROVED for goal: Flu

Final facts: Fever Cough BodyAche ViralInfection FluSymptoms Flu

**Conclusion**

The **Backward Chaining Algorithm** was successfully implemented using C++.  
The system started from the **goal (Flu)** and worked backward through rules and known facts to confirm the diagnosis.

This method efficiently models **logical reasoning** used in **expert systems**, where conclusions are derived by recursively proving sub-goals.  
Thus, backward chaining is a key inference mechanism in AI for **diagnostic, planning, and decision-making systems**.