**Assignment 8**

**Problem Statement:**  
Implement the **Backward Chaining Algorithm**.

**Theory**

**1. Inference in AI**

* **Facts:** Known truths in the knowledge base.
* **Rules:** Conditional statements of the form:
* Inference engines use reasoning methods to prove or disprove a goal.

**2. Backward Chaining**

* **Backward Chaining** is a **goal-driven reasoning** method.
* Start with the goal → work backwards to check if it can be satisfied by known facts.
* If goal can be broken into sub-goals that match known facts → the goal is proven true.

**Steps of Backward Chaining:**

1. Take the goal (hypothesis).
2. If it is already a known fact → proved.
3. Otherwise, find a rule whose conclusion matches the goal.
4. Recursively prove each condition of that rule (sub-goals).
5. If all conditions are satisfied → conclude goal is true.
6. If no rule/facts support the goal → goal cannot be proved.

**Use cases:** Expert systems, medical diagnosis, legal reasoning, etc.

**Algorithm**

1. Input: Facts, Rules, Goal.
2. If the goal exists in the fact base and is true → return true.
3. Else, search for rules that conclude the goal.
4. For each such rule, try to prove all its conditions (sub-goals) recursively.
5. If all sub-goals succeed → conclude the goal is true.
6. Otherwise, return false.

**Code (C++ Implementation)**

#include <iostream>

#include <vector>

#include <string>

#include <map>

using namespace std;

// Rule structure

struct Rule {

vector<string> conditions; // IF conditions

string conclusion; // THEN conclusion

};

// Recursive function for backward chaining

bool backwardChaining(string goal, vector<Rule>& rules, map<string, bool>& facts) {

// If already known fact

if (facts.find(goal) != facts.end()) {

return facts[goal];

}

// Try to prove using rules

for (auto& rule : rules) {

if (rule.conclusion == goal) {

bool allTrue = true;

for (auto& cond : rule.conditions) {

if (!backwardChaining(cond, rules, facts)) {

allTrue = false;

break;

}

}

if (allTrue) {

facts[goal] = true;

cout << "Proved: " << goal << " using rule -> IF ";

for (size\_t i = 0; i < rule.conditions.size(); i++) {

cout << rule.conditions[i];

if (i != rule.conditions.size() - 1) cout << " AND ";

}

cout << " THEN " << goal << endl;

return true;

}

}

}

facts[goal] = false;

return false;

}

// ---------------- MAIN ----------------

int main() {

// Initial facts

map<string, bool> facts;

facts["Has\_Fever"] = true;

facts["Has\_Cough"] = true;

facts["Has\_Rash"] = false;

// Rules

vector<Rule> rules = {

{ {"Has\_Fever", "Has\_Cough"}, "Flu" },

{ {"Has\_Fever", "Has\_Rash"}, "Measles" }

};

cout << "Known facts:\n";

for (auto& f : facts)

if (f.second) cout << "- " << f.first << endl;

cout << "\n";

// Ask user for goal

string goal;

cout << "Enter the goal to prove (e.g., Flu, Measles): ";

cin >> goal;

if (backwardChaining(goal, rules, facts))

cout << "\nConclusion: " << goal << " is TRUE\n";

else

cout << "\nConclusion: " << goal << " cannot be proved.\n";

return 0;

}

**Sample Output**

Known facts:

- Has\_Cough

- Has\_Fever

Enter the goal to prove (e.g., Flu, Measles): Measles

Conclusion: Measles cannot be proved.

Another Example Run:

Known facts:

- Has\_Cough

- Has\_Fever

Enter the goal to prove (e.g., Flu, Measles): Flu

Proved: Flu using rule -> IF Has\_Fever AND Has\_Cough THEN Flu

Conclusion: Flu is TRUE

**Conclusion**

* The **Backward Chaining Algorithm** was successfully implemented.
* It starts with the **goal (hypothesis)** and works backwards to check if it can be derived from the known facts.
* For the given knowledge base:
  + Goal **Flu** can be proved since conditions (*Has\_Fever*, *Has\_Cough*) are true.
  + Goal **Measles** cannot be proved because *Has\_Rash* is false.
* Backward chaining is widely used in **expert systems** where reasoning is goal-driven (e.g., proving a diagnosis).