Report on Backward Chaining

Introduction to Backward Chaining

Backward chaining is a goal-driven reasoning technique commonly used in rule-based expert systems. Unlike forward chaining, which starts with facts, backward chaining starts with a goal (or hypothesis) and works backward to check whether the goal can be proven using known facts and rules. This method is efficient when the objective is to confirm or deny a specific query rather than generate all possible conclusions.

Objective of the Assignment

The objective of this assignment is to implement a backward chaining inference engine in C++. The system demonstrates how reasoning begins with a desired conclusion and traces back through inference rules to determine if the conclusion is supported by known facts. A realistic example in the medical diagnosis domain has been used.

Rule-Based Systems and Logic

Rule-based systems consist of a knowledge base (facts) and an inference engine (rules). A typical rule has the form: IF conditions THEN conclusion. In backward chaining, the system takes a goal (e.g., 'Flu') and looks for rules that conclude it. If such a rule is found, the system attempts to prove its conditions (sub-goals) recursively. If all conditions can be proven from facts or other rules, the goal is confirmed true.

Implementation in C++

The backward chaining system is implemented in C++ with the following features: - A Rule structure representing conditions and a conclusion. - A set of known facts that serve as the knowledge base. - A recursive backwardChaining function that attempts to prove a goal by checking: 1. If the goal is already in the known facts (base case). 2. If there exists a rule whose conclusion matches the goal, in which case it recursively attempts to prove each of the rule's conditions. The medical diagnosis scenario uses rules such as: 1. IF Fever AND Cough THEN Flu 2. IF Flu THEN RestNeeded 3. IF Flu THEN SeeDoctor

Explanation of the Code

1. The program starts with a set of facts representing symptoms observed in a patient (e.g., Fever, Cough). Example: {"Fever", "Cough"} 2. A set of rules is defined with conditions and a conclusion. 3. The backwardChaining function attempts to prove a goal: - If the goal is a fact, it is proven immediately. - Otherwise, it searches for rules where the conclusion matches the goal. - For each such rule, it recursively tries to prove all conditions (sub-goals). 4. If all sub-goals are proven, the goal itself is proven. 5. If no rule or fact supports the goal, it cannot be proven. Example Run (Goal = SeeDoctor): - To prove SeeDoctor → check rule: IF Flu THEN SeeDoctor → sub-goal = Flu - To prove Flu → check rule: IF Fever AND Cough THEN Flu → check Fever and Cough - Both Fever and Cough are known facts → Flu proven → SeeDoctor proven.

Conclusion

Backward chaining is a fundamental reasoning method for goal-oriented queries. It is efficient when the objective is to validate a specific conclusion, making it useful in systems like Prolog, legal reasoning systems, and diagnostic assistants. This assignment showed how backward chaining can be implemented in C++ using recursion, sets for fact storage, and rules for inference. The medical diagnosis example highlights how backward chaining works step by step to verify goals.