Lecture Summary: Propositional Logic and Sudoku

Generated by LectureMate

February 22, 2025

1 Introduction

The lecture began with a light-hearted reference to *The Hobbit* and a discussion about the meaning of greetings. The main focus of the lecture was propositional logic, specifically evaluating formulas in conjunctive normal form (CNF).

2 Propositional Logic

2.1 Formulas in CNF

A formula in CNF is defined as a conjunction of clauses, where each clause is a disjunction of literals. A literal can be an atom or its negation. For example, a formula can be expressed as:

$$(p \lor \neg p) \land (q \lor r)$$

The evaluation of such formulas involves determining if there exists a valuation (assignment of truth values) that makes the formula true.

2.2 Valuation

A valuation is represented as a list of true literals. Unlike traditional mappings from variables to truth values, this approach allows for unknown variables. The lecture emphasized that to check the satisfiability of a formula, one can try all possible assignments of true and false to the variables. For n variables, there are 2^n possible assignments, which can be computationally expensive.

2.3 Proof Theory Perspective

From a proof theory perspective, a valuation can be seen as assumptions that allow for the proof of a formula. The lecture provided an example of a CNF formula and discussed how to find values for variables that satisfy the formula.

3 Algorithm for Satisfiability

The lecture introduced a recursive algorithm for determining satisfiability in CNF. The algorithm works as follows:

- Choose a variable and assign it a truth value (true or false).
- Simplify the formula based on this assignment.
- Recursively apply the algorithm to the simplified formula.
- If a satisfying assignment is found, return it; otherwise, backtrack and try the opposite assignment.

3.1 Example

An example was provided to illustrate the algorithm. The formula:

$$(\neg A \lor \neg C \lor \neg D) \land (A \lor C) \land (\neg D)$$

was analyzed step-by-step, demonstrating how to assign truth values to variables and simplify the formula.

4 Optimizations

The lecture discussed optimizations to improve the efficiency of the algorithm:

- Prioritize pure literals (variables that appear only positively or negatively).
- Start with the simplest clauses to reduce complexity.

5 Sudoku as a Boolean Problem

The latter part of the lecture focused on applying the discussed concepts to solve Sudoku puzzles using propositional logic. The Sudoku grid is a 9x9 matrix where each row, column, and 3x3 subgrid must contain the digits 1 to 9 without repetition.

5.1 Encoding Sudoku

To encode Sudoku as a Boolean formula:

- Define atomic statements for each cell, e.g., $F_{i,j,n}$ indicates that cell (i, j) contains the digit n.
- Formulate constraints to ensure that:
  - No cell contains two values.
  - Each row contains every digit.
  - Each column contains every digit.
  - Each 3x3 subgrid contains every digit.

5.2 Example Constraints

The constraints can be expressed as:

$$\neg(F_{i,j,n} \land F_{i,j,n'}) \quad \text{for } n \neq n'$$

This ensures that no cell has two different digits. Additionally, the existence of each digit in rows, columns, and subgrids can be expressed using existential quantifiers.

6 Conclusion

The lecture concluded with a reminder of the importance of understanding the algorithm and its applications, particularly in solving complex problems like Sudoku. Students were encouraged to practice implementing the algorithm and to explore the encoding of Sudoku in their upcoming tutorials.