1. Propositional Logic Foundations

- **Syntax and Structure:** We learned that propositional logic is built from variables (like p and q) combined with operators: AND, OR, NOT, and IMPLIES. The order of operations is important, with NOT binding most tightly, followed by AND/OR, and finally IMPLIES.

- **Encoding Real-World Problems:** We explored how to translate English constraints into logic. For example, a Sudoku grid can be encoded by creating variables representing "Cell (i, j) contains number n" and writing logic rules to ensure every row, column, and block contains a unique number.

2. Semantics and Proof Systems

- **Satisfiability and Validity:** A formula is "satisfiable" if there is at least one assignment of True/False to the variables that makes the whole formula True. A formula is "valid" (or a tautology) if it is True under every possible assignment.

- **Natural Deduction:** We studied a system of rules to prove arguments. Key rules include Modus Ponens (if p implies q, and p is true, then q is true) and Proof by Contradiction.

- **Soundness and Completeness:** The proof system is "Sound," meaning any statement we prove is actually true. It is also "Complete," meaning if a statement is semantically true, our proof system is capable of proving it.

3. Normal Forms and Complexity

- **CNF and DNF:** To solve problems algorithmically, we convert formulas into standard shapes. Conjunctive Normal Form (CNF) is a series of OR statements connected by ANDs. Disjunctive Normal Form (DNF) is the opposite.

- **Tseitin Encoding:** Converting complex formulas to CNF can usually cause the formula size to explode exponentially. We learned Tseitin Encoding, which introduces new temporary variables to keep the CNF size small and manageable.

- **Horn Formulas:** We identified a special type of formula called a Horn Formula (clauses with at most one positive literal)[12]. Unlike general logic problems which are very hard (NP-Complete), Horn formulas can be solved very quickly (Polynomial time)[13].

4. SAT Solving Algorithms

- **Resolution:** We learned the Resolution rule, which is a method to derive new conclusions by canceling out conflicting variables (e.g., combining "p OR q" with "NOT p OR r" gives "q OR r").

- **DPLL Algorithm:** This is the core logic behind modern SAT solvers. It works by assigning values to variables one by one (Decisions), automatically forcing other values when a clause has only one option left (Unit Propagation), and undoing assignments when a contradiction is found (Backtracking).

- **Clause Learning:** We explored how modern solvers improve DPLL by analyzing conflicts. When the solver hits a dead end, it "learns" a new rule (Conflict Clause) to prevent making the same mistake again in a different part of the search.