Propositional Logic: Semantics and Normal Forms

The Propositional SAT Problem
Semantic Equivalence

Definition (Semantic entailment)
If, for all valuations in which all $\phi_1, \phi_2, \cdots, \phi_n$ evaluates to T, $\psi$ also evaluates to T, we say that

$$\phi_1, \phi_2, \cdots, \phi_n \models \psi$$

holds and call $\models$ the semantic entailment relation.
Semantic Equivalence

- **Definition (Semantic entailment)**
  If, for all valuations in which all $\phi_1, \phi_2, \cdots, \phi_n$ evaluates to T, $\psi$ also evaluates to T, we say that

  $$\phi_1, \phi_2, \cdots, \phi_n \models \psi$$

  holds and call $\models$ the *semantic entailment* relation.

- **Definition (Semantic equivalence)**
  Two propositional logical formulas $\phi$ and $\psi$ are *semantically equivalent* iff. $\phi \models \psi$ and $\psi \models \phi$. In that case we write, $\phi \equiv \psi$. 
Validity and Satisfiability

Definition (Validity)
We say a formula \( \phi \) is valid iff \( \models \phi \) holds.

Definition (Satisfiability)
We say a formula \( \phi \) is satisfiable iff there exists a valuation in which it evaluates to \( T \).

Proposition
A formula \( \phi \) is satisfiable iff \( \neg \phi \) is not valid.
Conjunctive Normal Forms

▶ **atom**: proposition, e.g., \( p, q \)
▶ **literal**: atom or the negation of an atom, e.g., \( p, \neg p \)
▶ **clause**: disjunction of literals

**Definition**
A formula is in Conjunctive Normal Form (CNF) if it is a conjunction of clauses.

**Example**
\[
(\neg q \lor p \lor r) \land (\neg p \lor r \lor \neg r) \land q
\]
Every formula can be transformed into an *equivalent* formula in CNF.
Transforming formulas into CNF

- **IMPL_FREE**: remove all implications by replacing $\phi \rightarrow \eta$ with $\neg \phi \lor \eta$.

- **NNF**: transform formula into *negation normal form* (NNF), i.e., negation occur only in front of atoms by
  - applying De-Morgan law $\neg (\phi \lor \eta) \equiv \neg \phi \land \neg \eta$ and $\neg (\phi \land \eta) \equiv \neg \phi \lor \neg \eta$.
  - removing double negations $\neg \neg \phi \equiv \phi$.

- **DISTR**: push all occurrences of $\lor$ inside $\land$ by applying the distributive law $(\phi_1 \land \phi_2) \lor \phi_3 \equiv (\phi_1 \lor \phi_3) \land (\phi_1 \lor \phi_3)$. 
Satisfiability and Validity of Formulas in CNF

- Satisfiability in CNF is NP-Complete.
  - For 3-SAT, monotone 3-SAT, and monotone 3-2-SAT
  - 2-SAT can be solved in polynomial time.

Validity in CNF can be solved in linear time.

\[ \varphi = c_1 \land c_2 \land \cdots \land c_n \text{ is valid iff each } c_i \text{ is valid.} \]

\[ L_1 \lor L_2 \lor \cdots \lor L_m \text{ is valid iff there exist } i, j \text{ such that } L_i = \neg L_j. \]
Satisfiability and Validity of Formulas in CNF

- Satisfiability in CNF is NP-Complete.
  - For 3-SAT, monotone 3-SAT, and monotone 3-2-SAT
  - 2-SAT can be solved in polynomial time.

- Validity in CNF can be solved in linear time.
  - $\phi = c_1 \land c_2 \land \cdots \land c_n$ is valid iff each $c_i$ is valid.
  - $L_1 \lor L_2 \lor \cdots \lor L_m$ is valid iff there exist $i, j$ such that $L_i$ is $\neg L_j$. 
Disjunctive Normal Forms

Definition
A formula is in Disjunctive Normal Form (DNF) if it is a disjunction of conjunctions.

Example

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

- Every formula can be transformed into an equivalent formula in DNF.
- Checking satisfiability of formulas in DNF can be solved in linear time.
- Validity of formulas in DNF is co-NP complete.
The Propositional SAT Problem

The SAT Problem

- NP-complete, worst-case exponential time, however, large instances can be solved in practice.
- SAT solvers widely used in verification.
- Hardness of purely randomly generated SAT instances depends primarily upon the ratio of \# clauses to \# variables.
  - Ration too large, easily unsatisfiable (too constrained).
  - Ratio too small, easily satisfiable (many solutions).
  - For random 3SAT, ratio $\approx 4.2$ hardest in an early study. Later study shows dependence on SAT solvers.
Algorithms for SAT

- Modern variants of the DPLL (Davis-Putnam-Logemann-Loveland) algorithm.
  - complete, backtracking,
- Stochastic local search algorithms, e.g., WALKSAT.
DPLL

- **Basic Backtracking**: Choose a literal, assign a truth value to it, simplify the formula, and then recursively checking if the simplified formula is satisfiable;
  - if so, the original formula is satisfiable;
  - otherwise, assume the opposite truth value, redo simplification and recursive check. This is known as the splitting rule.
- The simplification step: removes all clauses which become true, and all literals that become false.
- Optimizations
  - **Unit propagation**: If a clause is a unit clause, the truth value of the literal is determined. In practice, this often leads to deterministic cascades of units, thus avoiding a large part of the naive search space.
  - **Pure literal elimination**: If all occurrences of a propositional variable are positive (or negative), it is called pure. Not very useful due to cost of detecting.
The Propositional SAT Problem

DPLL

- **Basic Backtracking**: Choose a literal, assign a truth value to it, simplify the formula, and then recursively checking if the simplified formula is satisfiable;
  - if so, the original formula is satisfiable;
  - otherwise, assume the opposite truth value, redo simplification and recursive check. This is known as the splitting rule.

- **The simplification step**: removes all clauses which become true, and all literals that become false.
The Propositional SAT Problem

DPLL

- **Basic Backtracking**: Choose a literal, assign a truth value to it, simplify the formula, and then recursively checking if the simplified formula is satisfiable;
  - if so, the original formula is satisfiable;
  - otherwise, assume the opposite truth value, redo simplification and recursive check. This is known as the splitting rule.

- **The simplification step**: removes all clauses which become true, and all literals that become false.

- **Optimizations**
  - **Unit propagation**: If a clause is a unit clause, the truth value of the literal is determined. In practice, this often leads to deterministic cascades of units, thus avoiding a large part of the naive search space.
  - **Pure literal elimination**: If all occurrences of a propositional variable are positive (or negative), it is called pure. Not very useful due to cost of detecting.