Propositional Resolution

The propositional resolution calculus operates on a set of clauses and tests unsatisfiability.

Recall that for clauses I switch between the notation as a disjunction, e.g., \( P \lor Q \lor P \lor \neg R \), and the multiset notation, e.g., \( \{ P, Q, P, \neg R \} \). This makes no difference as we consider \( \lor \) in the context of clauses always modulo AC. Note that \( \bot \), the empty disjunction, corresponds to \( \emptyset \), the empty multiset. Clauses are typically denoted by letters \( C, D \), possibly with subscript.
Resolution Inference Rules

Resolution \((N \cup \{C_1 \lor P, C_2 \lor \neg P\}) \Rightarrow_{\text{RES}} (N \cup \{C_1 \lor P, C_2 \lor \neg P\} \cup \{C_1 \lor C_2\})\)

Factoring \((N \cup \{C \lor L \lor L\}) \Rightarrow_{\text{RES}} (N \cup \{C \lor L \lor L\} \cup \{C \lor L\})\)
2.6.1 Theorem (Soundness & Completeness)

The resolution calculus is sound and complete:

\( N \) is unsatisfiable iff \( N \Rightarrow^{\star}_{\text{RES}} N' \) and \( \bot \in N' \) for some \( N' \)
Resolution Reduction Rules

**Subsumption**

\[(N \cup \{C_1, C_2\}) \Rightarrow_{\text{RES}} (N \cup \{C_1\})\]
provided \(C_1 \subset C_2\)

**Tautology Deletion**

\[(N \cup \{C \lor P \lor \neg P\}) \Rightarrow_{\text{RES}} (N)\]

**Condensation**

\[(N \cup \{C_1 \lor L \lor L\}) \Rightarrow_{\text{RES}} (N \cup \{C_1 \lor L\})\]

**Subsumption Resolution**

\[(N \cup \{C_1 \lor L, C_2 \lor \text{comp}(L)\}) \Rightarrow_{\text{RES}} (N \cup \{C_1 \lor L, C_2\})\]
where \(C_1 \subseteq C_2\)
2.6.6 Theorem (Resolution Termination)

If reduction rules are preferred over inference rules and no inference rule is applied twice to the same clause(s), then $\Rightarrow^+_\text{RES}$ is well-founded.
The Overall Picture

<table>
<thead>
<tr>
<th>Application</th>
<th>System + Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>Algorithm + Implementation</td>
</tr>
<tr>
<td>Calculus + Strategy</td>
<td>Calculus + Logic + States + Rules</td>
</tr>
<tr>
<td>Logic</td>
<td>Syntax + Semantics</td>
</tr>
</tbody>
</table>
Conflict Driven Clause Learning (CDCL)

The CDCL calculus tests satisfiability of a finite set $N$ of propositional clauses.

I assume that $\bot \notin N$ and that the clauses in $N$ do not contain duplicate literal occurrences. Furthermore, duplicate literal occurrences are always silently removed during rule applications of the calculus. (Exhaustive Condensation.)
The CDCL calculus explicitly builds a candidate model for a clause set. If such a sequence of literals $L_1, \ldots, L_n$ satisfies the clause set $N$, it is done. If not, there is a false clause $C \in N$ with respect to $L_1, \ldots, L_n$.

Now instead of just backtracking through the literals $L_1, \ldots, L_n$, CDCL generates in addition a new clause, called *learned clause* via resolution, that actually guarantees that the subsequence of $L_1, \ldots, L_n$ that caused $C$ to be false will not be generated anymore.

This causes CDCL to be exponentially more powerful in proof length than its predecessor DPLL or Tableau.