3.3 Ground Resolution

Drawbacks of Gilmore’s Algorithm:
- How can one find a suitable instantiation of variables by ground terms? \[ \text{Sect 3.4} \]
- How can one check satisfiability of a propositional formula efficiently? \[ \text{Sect 3.3} \]

Resolution: main proof technique in logic prog.
First: ground resolution (for formulas without variables)

To check a formula $\forall X_1, \ldots, X_n\, \forall$ in Skolem NF for unsatisfiability by resolution, one first has to transform $\forall$ into conjunctive normal form (CNF).
Such formulas can then be represented as clause sets.

**Def 3.3.1 (CNF, Clause, Literal)**

A formula $\forall$ is in conjunctive normal form iff it is quantifier-free and has the form

$$(L_{1,\overline{1}} \lor \ldots \lor L_{1,n}) \land \ldots \land (L_{m,\overline{1}} \lor \ldots \lor L_{m,n}).$$

Here, $L_{i,\overline{1}}$ are literals, i.e., atomic formulas or negated atomic formulas of the form $p(t_1, \ldots, t_k)$ or $\neg p(t_1, \ldots, t_k)$.

For every literal $L$, its negation $\overline{L}$ is defined as

$$\overline{L} = \begin{cases} \neg A, & \text{if } L = A \in \text{At}(\exists, \Delta, \forall) \\ A, & \text{if } L = \neg A, A \in \text{At}(\exists, \Delta, \forall) \end{cases}$$

A clause is a set of literals and it represents the
universally quantified disjunction of the literals. A clause set represents the conjunction of its clauses. So every formula $\forall_i \psi_i$ in CNF can be represented as a clause set

$$\forall_i \psi_i = \bigwedge \{ \{ \psi_{i,1}, \ldots, \psi_{i,m_i} \} \mid \psi_{i,1}, \ldots, \psi_{i,m_i} \in \psi_i \}$$

Therefore, we also speak of satisfiability and entailment of clause sets. The empty clause is denoted $\Box$. It is unsatisfiable by definition (empty disjunction).

**Theorem 3.3.2 (Transformation to CNF)**

Every quantifier-free formula $\forall_i \psi_i$ can be transformed into an equivalent formula $\forall_i \psi_i'$ in CNF automatically.

**Proof:** First replace all sub-formulas $\forall_i \psi_i \leftrightarrow \forall_i \psi_i'$ by

$$(\forall_i \psi_i \rightarrow \forall_i \psi_i') \land (\forall_i \psi_i' \rightarrow \forall_i \psi_i)$$

Then replace all sub-formulas $\forall_i \psi_i \rightarrow \forall_i \psi_i'$ by $\forall_i \psi_i \lor \forall_i \psi_i'$.

Afterwards, use the following algorithm CNF:

- **Input:** $\forall_i \psi_i$ (quantifier-free, without $\leftrightarrow$, $\lor$, $\land$)
- **Output:** Equivalent formula in CNF

  - If $\forall_i \psi_i$ is an atomic formula, then return $\forall_i \psi_i$.
  - If $\forall_i \psi_i = \forall_i \psi_i \land \forall_i \psi_i'$, then return $\text{CNF}(\forall_i \psi_i) \land \text{CNF}(\forall_i \psi_i')$.
  - If $\forall_i \psi_i = \forall_i \psi_i \lor \forall_i \psi_i'$, then compute
    
    $$\text{CNF}(\forall_i \psi_i) = \forall_i \psi_i' \lor \cdots \lor \forall_i \psi_{i,m_i}'$$
    
    $$\text{CNF}(\forall_i \psi_i') = \forall_i \psi_i'' \lor \cdots \lor \forall_i \psi_{i,m_i}''$$
    
    Return $\bigwedge_{i \in \{1, \ldots, m_i\}} \bigwedge_{j \in \{1, \ldots, m_j\}} (\forall_i \psi_i' \lor \forall_i \psi_i'' \lor \cdots \lor \forall_i \psi_{i,m_i}'' \lor \forall_i \psi_{i,m_j}''')$

  - Use the distributivity law
- If $\Phi = \neg \Phi_0$, then compute

$$\text{CNF}(\Phi) = \bigwedge_{i \in \{1, \ldots, n \}} \left( \bigvee_{j \in \{1, \ldots, m \}} \neg \Phi_{i,j} \right)$$

De Morgan's law states that the negation of this formula is

$$\bigvee_{i \in \{1, \ldots, n \}} \left( \bigwedge_{j \in \{1, \ldots, m \}} \Phi_{i,j} \right)$$

Due to the distributivity law, we return

$$\bigwedge_{j \in \{1, \ldots, m \}} \left( \bigvee_{i \in \{1, \ldots, n \}} \bigvee_{k \in \{1, \ldots, n \}} \Phi_{i,j,k} \right)$$

**Ex 3.3** Let $p, q, r \in \Delta_0$.

Transform the following formula into CNF:

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

↓ De Morgan

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

↓ Distributivity

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

Alg. of Gilmore: To check unsatisf. of $\Psi$, consider $E(\Psi)$ and prove unsatisf. of these propositional formulas.

Therefore: Now introduce a technique to prove unsatisfiability of prop. formulas in CNF.

In other words: Prove unsatisf. of clause sets
Without variables.

Resolution: \((L_1 \lor L) \land (L_2 \lor \neg L)\)

implies \(L_1 \lor L_2\)

Def 33.4 (Propositional Resolution)

Let \(K_1, K_2\) be two clauses without variables. Then, the clause \(R\) is a resolvent of \(K_1\) and \(K_2\) iff there exists a \(L \in K_1\) with \(\neg L \in K_2\) and \(R = (K_1 \setminus \{L\}) \lor (K_2 \setminus \{\neg L\})\).

For a clause set \(Y\), we define

\[ \text{Res}(\neg Y) = Y \lor \{ R \mid R \text{ is resolvent of two clauses from } Y \} \]

We define

\[ \text{Res}^0(\neg Y) = Y \]

\[ \text{Res}^{n+1}(\neg Y) = \text{Res}(\text{Res}^n(\neg Y)) \quad \text{for all } n \geq 0 \]

Moreover:

\[ \text{Res}^d(\neg Y) = \bigcup_{n \geq 0} \text{Res}^n(\neg Y) \]

Idea: Construct \(\text{Res}^d(\neg Y)\) until one obtains \(\square\). Since adding resolvents is equivalence-preserving, this means that \(Y\) is unsatisfiable.

Clearly: \(\square \in \text{Res}^d(\neg Y)\) iff there is a sequence of clauses \(K_1, \ldots, K_m\) with \(K_m = \square\) where for all \(1 \leq i \leq m\), we have

\[ \bullet K_i \in Y \quad \text{or} \quad \bullet \neg K_i \in Y \]
\( K_\alpha \) is a resolvent of \( K_\delta, K_\kappa \) for \( \delta, \kappa < \alpha \).

To denote resolution proofs:

\[
\begin{array}{c}
\text{\( K_1 \)} \\
\text{\( K_2 \)}
\end{array}
\xrightarrow{R}
\begin{array}{c}
\text{\( R \)}
\end{array}
\]

This means that \( R \) is resolvent of \( K_1 \) and \( K_2 \).

Ex. 3.35 Let \( \Delta = \{ p, q \} \), \( \Sigma_0 = \{ a, b \} \), \( \Sigma_1 = \{ f \} \), \( \Sigma_2 = \{ g \} \).

We regard the clause set \( \mathcal{Y} = \{ K_1, K_2, K_3, K_4 \} \) with

\[
\begin{align*}
K_1 & : \{ \neg p(f(a)), q(b) \} \\
K_2 & : \{ p(f(a)) \} \\
K_3 & : \{ p(g(b,a)) \} \\
K_4 & : \{ \neg p(g(b,a)), q(b) \}
\end{align*}
\]

The resolution calculus is sound and complete:

Completeness: If \( \mathcal{Y} \) is unsat., then \( \Box \in \text{Res}^*(\mathcal{Y}) \).

Soundness: If \( \Box \in \text{Res}^*(\mathcal{Y}) \), then \( \mathcal{Y} \) is unsat.

To prove soundness, we need the following lemma.

Lemma 3.36 (Propositional Resolution Lemma)

Let \( \mathcal{Y} \) be a set of clauses without variables.

If \( \mathcal{Y}_1, \mathcal{Y}_2 \in \mathcal{Y} \) and \( R \) is a resolvent of \( \mathcal{Y}_1 \) and \( \mathcal{Y}_2 \),

then \( \mathcal{Y}_1 \) and \( \mathcal{Y}_2 \cup \{ R \} \) are equivalent.

Proof: \( \supseteq \): Every structure that satisfies \( \mathcal{Y}_1 \cup \{ R \} \) also satisfies \( \mathcal{Y}_1 \).

Thus: \( \mathcal{Y}_1 \cup \{ R \} \models \mathcal{Y}_1 \).
"⇒": Let $S$ be a structure with $S \models \mathcal{K}$.

Let $L \in \mathcal{K}_1$, $\bar{L} \in \mathcal{K}_2$, $R = (\mathcal{K}_1 \setminus \{L\}) \cup (\mathcal{K}_2 \setminus \{\bar{L}\})$.

Assume $S \not\models \mathcal{K} \cup \{R\}$. Thus: $S \not\models R$

If $S \not\models L$, then $S \models \bar{L}_2$ implies $S \models \bar{L}_2 \setminus \{\bar{L}\}$

Therefore: $S \models R$

If $S \not\models \bar{L}$, then $S \models L$ and $S \models \mathcal{K}_2$ implies $S \models \mathcal{K}_2 \setminus \{L\}$

Therefore: $S \models R$.

Thus $\Rightarrow$ (Soundness andCompleteness of Prop. Resolution)

Let $\mathcal{K}$ be a set of clauses without variables.

Then $\mathcal{K}$ is unsatisfiable iff $\exists \in \text{Res}^n(\mathcal{K})$.

Proof: Soundness "⇒":

By the resolution lemma 3.3.6, $\mathcal{K}$ and $\text{Res}(\mathcal{K})$ are equivalent. By induction on $n$, one can show that $\mathcal{K}$ and $\text{Res}^n(\mathcal{K})$ are equivalent for all $n \in \mathbb{N}$.

$\exists \in \text{Res}^n(\mathcal{K})$

there exists an $n \in \mathbb{N}$ with $\exists \in \text{Res}^n(\mathcal{K})$

$\Rightarrow \text{Res}^n(\mathcal{K})$ is unsat.

$\Rightarrow \mathcal{K}$ is unsat.

Completeness "⇒":

If $\mathcal{K}$ is unsatisfiable, then there is a finite subset $\mathcal{K}' \subseteq \mathcal{K}$ which is also unsat. (by the compactness theorem of prop. logic).

Let $n$ be the number of different atomic formulas in $\mathcal{K}'$. We use induction on $n$.
Ind. Base: $n = 0$
There are only two clause sets without atomic formulas:
\[ X^1 = \emptyset \]  
- empty conjunction: valid, i.e., true in every interpretation
\[ X^2 = \{ \bot \} \]. Then \( \bot \in \text{Res}^* (X^2) \subseteq \text{Res}^* (X^1) \).

Ind. Step: $n > 0$
Let $A$ be an atomic formula that occurs in the unsat. clause set $X^1$.
Let $X^+$ result from $X^1$ by
- removing all clauses that contain literal $A$
- drop $\neg A$ from all remaining clauses
Thus: $X^+ = \{ K \setminus \{ \neg A \} \mid K \in X^1, A \in K \}$
Similarly: $X^- = \{ K \setminus \{ A \} \mid K \in X^1, \neg A \in K \}$
$X^+$ is unsat.: If $S \models X^+$ then extend $S$ to a structure $S'$ with $S' \models A$.
Then $S' \models X^1 \quad \bot$ to the unsat. of $X^1$
Similarly, $X^-$ is unsat.
Since $X^+$ and $X^-$ do not contain $A$, we can apply the ind. hypothesis, which yields:
\[ \bot \in \text{Res}^* (X^+) \quad \text{and} \quad \bot \in \text{Res}^* (X^-) \quad \bot \in \text{Res}^* (X^+) \]
means that there is a sequence \( \ell_1, \ldots, \ell_m \) with \( \bot = \ell_m \) where
for all \( 1 \leq i \leq m \):
\[ \bullet \ell_i \in X^+ \quad \text{or} \quad \ell_i \text{ is resolvent of } \ell_j \text{ and } \ell_k \text{ for } j, k < i \]
If all these \( \mathcal{K}_i \) are also contained in \( \mathcal{K}'_i \), then we have proved \( \Box \in \text{Res}^* (\mathcal{K}'_i) \subseteq \text{Res}^* (\mathcal{K}) \).

Otherwise, add \( \neg A \) again to all clauses where it had been removed.

Then, obtain a resolution proof for \( \{ \neg A \} \in \text{Res}^* (\mathcal{K}') \)

Reason: \( C_j \backslash / C_k \quad C_j \cup \neg A \quad C_k \cup \neg A \)

\( C_i \quad C_i \cup \neg A \)

Similarly: \( \Box \in \text{Res}^* (\mathcal{K}') \) implies \( \Box \in \text{Res}^* (\mathcal{K}') \) or \( \{ A \} \in \text{Res}^* (\mathcal{K}') \).

Thus: \( \{ A \} \cup \{ \neg A \} \in \text{Res}^* (\mathcal{K}') \)

implies \( \Box \in \text{Res}^* (\mathcal{K}') \)

\( \ldots \)

\( \{ \neg A \} \quad \{ A \} \)

\( \Box \)

Now the algo. of Gilmore can be improved to the ground resolution algorithm.
Sound: if alg. returns "true," then \( \{y_1, \ldots, y_k\} \models \gamma \)

complete: if \( \{y_1, \ldots, y_k\} \models \gamma \), then alg. terminates and returns "true"

if \( \{y_1, \ldots, y_k\} \not\models \gamma \), then the alg. might not terminate

\Rightarrow \text{ semi-decision procedure}

\text{Step 5: Advantage over Gilmore's alg.}

\Rightarrow \text{ better check for unsat. of propositional clause sets}

\text{Step 4: Still inefficient, since we don't know how to instantiate variables by ground terms in a "clever" way.}

\textbf{Ex 3.38:} Show unsatisfiability of

\[ \forall x, y \ (\neg p(x) \lor \neg p(f(a)) \lor q(y)) \land p(y) \land (\neg p(g(b,x)) \lor \neg q(b)) \]

Corresponding clause set \( J^c(\gamma) \):

\[ \{\neg p(x), \neg p(f(a)), q(y)\}, \{p(y), \{\neg p(g(b,x)), \neg q(b)\}\} \]

\[ U_1: [x/f(a), y/b] \quad U_2: [y/f(a)] \quad U_3: [y/g(b,a)] \quad U_4: [x/a] \]

\[ \{\neg p(f(a)), q(b)\} \quad \{p(f(a))\} \quad \{p(g(b,a))\} \quad \{\neg q(b)\} \]

\[ \{ q(b) \} \quad \{ \neg q(b) \} \]

- Instantiations can unify several literals of the same clause.
- We can use several different instantiations of the same
clause.

How can one find such suitable instantiations automatically?