3.3 Ground Resolution

Drawbacks of Gilmore's Algorithm:
- Nuclear with which ground terms one should instantiate variables to check whether a ground formula is satisfiable:
  - Try out all possible assignments of atomic ground formulas to \{TRUE, FALSE\} correspond to propositional variables.

Why is it not enough to check unsatisfiability of \( \varphi_1 \) or \( \varphi_2 \) or \( \varphi_3 \) or...

Ex:

\[
\begin{align*}
\varphi & : \quad p(0), \\
p(s(X)) & : \quad \neg p(X).
\end{align*}
\]

\[
\exists \neg p(s(s(0))).
\]

\[
\begin{align*}
\varphi_1 & : \quad p(0), \\
\varphi_2 & : \quad \forall X \quad p(X) \rightarrow p(s(X)) \\
\varphi & : \quad p(s(s(0)))
\end{align*}
\]

We have to check unsatisfiability of

\[
\varphi : \quad \forall X \quad p(0) \land (p(X) \rightarrow p(s(X))) \land \neg p(s(s(0)))
\]

\[
\begin{align*}
\varphi_1 : \quad p(0) \land (p(0) \rightarrow p(s(0))) \land \neg p(s(s(0)))
\end{align*}
\]

Satisfiable: \( p(0) : TRUE, p(s(0)) : TRUE, p(s(s(0))) : FALSE \)

\[
\begin{align*}
\varphi_2 : \quad p(0) \land (p(s(0)) \rightarrow p(s(s(0))) \land \neg p(s(s(0)))
\end{align*}
\]
Satisfiable: \( p(0): \text{TRUE}, \ p(5(0)): \text{FALSE}, \ p(5(5(0))): \text{FALSE} \)

\[ \exists x \ s(x(0)): \exists x: \ldots \]
also satisfiable

\( \Rightarrow \) all \( \exists x_i \) on their own are satisfiable
but \( \exists x_1 \land \exists x_2 \) is unsatisfiable

Reason: the same rule has to be applied several times with different instantiations.

Goal: Improve the 2nd drawback of Gilmore’s algorithm (i.e., check unsatisfiability of ground formulas)

Solution: Resolution (today: ground resolution)

3.3. Ground Resolution

Input: Formula \( \forall x_1, \ldots, x_n \ A \)
in Skolem NF
Goal: check unsatisfiability
First step: transform quantifier-free formula \( A \) to conjunctive normal form \( (\text{CNF}) \)

Def 3.3.1 (\( \text{CNF} \))
A formula \( A \) is in \( \text{CNF} \) iff it is quantifier-free and it has the following form:
\[
(L_1 \lor \ldots \lor L_i) \land \ldots \land (L'_1 \lor \ldots \lor L'_n)
\]
\[(L_{1,1} \lor \ldots \lor L_{1,n}) \land \ldots \land (L_{m,1} \lor \ldots \lor L_{m,n})\]

Here, \(L_{i,j}\) are literals, i.e., they are atomic or negated atomic formulas (i.e., they have the form \(p(t_1, \ldots, t_n)\) or \(\neg p(t_1, \ldots, t_n)\)).

For every literal \(L\), we define its negation \(\overline{L}\) as follows:

\[\overline{L} = \begin{cases} \neg A, & \text{if } L = A \in \text{At}(\Sigma, \Delta, \emptyset) \\ A, & \text{if } L = \neg A \text{ for } A \in \text{At}(\Sigma, \Delta, \emptyset) \end{cases}\]

A set of literals is called a clause.

Every formula \(\varphi\) in CNF corresponds to the following clause set:

\[\forall \{\forall \} = \{\{L_{1,1}, \ldots, L_{1,n}\}, \ldots, \{L_{m,1}, \ldots, L_{m,n}\}\}\]

So a clause stands for the universally quantified disjunction of its literals and a clause set corresponds to the conjunction of its clauses.

The empty clause is denoted \(\Box\) and it is unsatisfiable by definition.

**Thm 3.32 (Transformation to CNF)**

For every quantifier-free formula \(\varphi\), one can automatically construct an equivalent formula \(\varphi'\) in CNF.

**Proof:** First, replace sub-formulas \(\varphi_1 \leftrightarrow \varphi_2\) by
\((\forall_1 \rightarrow \forall_2) \land (\forall_2 \rightarrow \forall_3)\).

Then replace sub-formulas \(\forall_1 \rightarrow \forall_2\) by \(\neg \forall_1 \lor \forall_2\).

Then apply the following algorithm \(\text{CNF}(\psi)\):

- If \(\psi\) is atomic, then return \(\psi\).
- If \(\psi = \psi_1 \land \psi_2\), then \(\text{CNF}(\psi_1) \land \text{CNF}(\psi_2)\).
- If \(\psi = \psi_1 \lor \psi_2\), then compute
  
  \[
  \text{CNF}(\psi_1) = \bigwedge_{i \in \{1, \ldots, m\}} \psi_i^{11}
  \]
  
  \[
  \text{CNF}(\psi_2) = \bigwedge_{j \in \{1, \ldots, n\}} \psi_j^{11}
  \]

  Then return
  
  \[
  \bigwedge_{i \in \{1, \ldots, m\}} (\psi_i^{11} \lor \psi_i^{11})
  \]

- If \(\psi = \neg \forall_\eta\), then compute
  
  \[
  \text{CNF}(\forall_\eta) = \bigwedge_{i \in \{1, \ldots, \eta_1\}} (\lor \bigwedge_{j \in \{1, \ldots, \eta_j\}} \psi_{i,j})
  \]

  Applying De Morgan Laws results in
  
  \[
  \lor \bigwedge_{i \in \{1, \ldots, \eta_1\}} (\lor \bigwedge_{j \in \{1, \ldots, \eta_j\}} \psi_{i,j})
  \]

  Applying the distribution law yields the following formula that is returned:
  
  \[
  \bigwedge_{i \in \{1, \ldots, \eta_1\}} (\lor \bigwedge_{j \in \{1, \ldots, \eta_j\}} \psi_{i,j})
  \]

\[\text{CNF}(\psi)\]

is equivalent to

\[
(\forall_1 \lor \forall_2) \land (\forall_2 \lor \forall_3) \land (\forall_3 \lor \forall_4) \land (\forall_4 \lor \forall_5) \land (\forall_5 \lor \forall_6)
\]

Distribution Law
Ex. 3.3.3 \[ \Phi \] is the following formula with \[ p, q, r \in \Delta_0: \]

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

De Morgan laws yield:

\[ p \lor (q \land r) \]

Distribution law results in:

\[ (p \lor q) \land (p \lor r) \]

Remaining goal: Check unsatisfiability of a ground formula in CNF, i.e., of a set of ground clauses.

**Def 3.3.4 (Propositional Resolution)**

Let \( K_1, K_2 \) be ground clauses. Then the clause \( R \) is a **resolvent of** \( K_1 \) and \( K_2 \) iff there is a literal \( L \in K_1 \) with \( \overline{L} \in K_2 \) and \( R = (K_1 \setminus \{L\}) \cup (K_2 \setminus \{\overline{L}\}) \).

For a clause set \( K \) we define

\[ \text{Res}(K) = K \cup \{R \mid R \text{ is resolvent of two clauses from } K\} \]

Moreover, let
\[ \text{Res}^0(\neg X) = \neg X \]
\[ \text{Res}^{n+1}(\neg X) = \text{Res}(\text{Res}^n(\neg X)) \text{ for all } n \geq 0. \]

So the set of all clauses that can be deduced by resolution is
\[ \text{Res}^* (\neg X) = \bigcup_{n \geq 0} \text{Res}^n (\neg X) \]

Obviously, we have \( \Box \in \text{Res}^* (\neg X) \) iff there is a sequence of clauses \( K_1, \ldots, K_m \) such that the following holds for all \( 1 \leq i \leq m \):

- \( K_i \in \neg X \) or
- \( K_i \) is a resolvent of \( K_j \) and \( K_k \) for \( j, k < i \).

To display resolution proofs, we often use diagrams:

\[
\begin{array}{ccc}
K_1 & \text{\scriptsize{\u2013}} & K_2 \\
\downarrow & & \downarrow \\
R & \text{\scriptsize{\u2013}} & \\
\end{array}
\]

Means that \( R \) is resolvent of \( K_1 \) and \( K_2 \)

\textbf{Ex 3.3.5} \( \Box \) can be derived

We now have to show that
\[ \Box \in \text{Res}^* (\neg X) \] iff \( \neg X \) is unsatisfiable

\( \text{syntax} \) can be checked \( \Rightarrow \) : Soundness

\( \text{semantics} \) of resolution \( \Leftarrow \) : Completeness
To prove soundness of ground resolution, we show that adding resolvents preserves equivalence.

**Lemma 33.6 (Propositional Resolution Lemma)**

Let \( \mathcal{K} \) be a set of ground clauses. If \( \mathcal{K}_1, \mathcal{K}_2 \in \mathcal{K} \) and \( R \) is resolvent of \( \mathcal{K}_1 \) and \( \mathcal{K}_2 \), then \( \mathcal{K} \) and \( \mathcal{K} \cup \{ R \} \) are equivalent.

**Proof:** :

\( \subseteq \): If there is a structure \( S \) with \( S \models \mathcal{K} \cup \{ R \} \), then also \( S \models \mathcal{K} \).

\( \supseteq \): Let \( S \models \mathcal{K} \).

There is a literal \( \ell \in \mathcal{K}_1 \), \( \bar{\ell} \in \mathcal{K}_2 \), \( R = (\mathcal{K}_1 \setminus \{ \ell \}) \cup (\mathcal{K}_2 \setminus \{ \bar{\ell} \}) \).

Assume that \( S \not\models \mathcal{K} \cup \{ R \} \), i.e., \( S \not\models R \).

If \( S \models \ell \), then \( S \models \mathcal{K}_1 \) implies \( S \models \mathcal{K}_2 \), which in turn implies \( S \models \mathcal{K}_2 \setminus \{ \bar{\ell} \} \). Thus, \( S \not\models R \). 

If \( S \models \bar{\ell} \), then in a similar way one can show \( S \models \mathcal{K}_1 \setminus \{ \ell \} \). Thus, \( S \not\models R \).

\( \blacksquare \)

**Theorem 33.7 (Soundness and Completeness of propositional resolution)**

Let \( \mathcal{K} \) be a (possibly infinite) set of ground clauses. Then: \( \forall \mathcal{E} \in \text{Res}^*(\neg \mathcal{K}) \) implies \( \mathcal{K} \) is satisfiable.
Then: \[ \emptyset \in \text{Res}^\ast (\mathcal{X}) \text{ iff } \mathcal{X} \text{ is unsatisfiable.} \]

Proof: \( \Rightarrow \) (Soundness)
Resolution Lemma 3.36 states that \( \mathcal{X} \) and \( \text{Res} (\mathcal{X}) \) are equivalent.
By induction, one can show that \( \mathcal{X} \) is equivalent to \( \text{Res}^n (\mathcal{X}) \) for all \( n \in \mathbb{N} \).
\[ \emptyset \in \text{Res}^\ast (\mathcal{X}) \]
\( \forall \) there is an \( n \in \mathbb{N} \) such that \( \emptyset \in \text{Res}^n (\mathcal{X}) \)
\( \forall \) \( \text{Res}^n (\mathcal{X}) \) is unsatisfiable
\( \forall \) \( \mathcal{X} \) is unsatisfiable.

\( \Leftarrow \) (Completeness)
\( \mathcal{X} \) is unsatisfiable
\( \forall \) there is a finite subset \( \mathcal{X}' \subseteq \mathcal{X} \) that is unsatisfiable
We prove \( \emptyset \in \text{Res}^\ast (\mathcal{X}') \) by induction on the number \( n \) of different atomic formulas in \( \mathcal{X}' \).

Ind Base: \( n = 0 \)
There are only 2 clause sets without atomic formulas:

\[ \mathcal{X}' = \emptyset \] is valid (holds in every structure)
or
\[ \mathcal{X}' = \{ \square \} \] is unsatisfiable
Then \( \emptyset \in \text{Res}^0 (\mathcal{X}') \subseteq \text{Res}^\ast (\mathcal{X}') \).

Ind Step: \( n > 0 \)
Let \( A \) be an atomic formula occurring in \( \mathcal{X}' \).
Let $\mathcal{X}^+$ result from $\mathcal{X}'$ by omitting all clauses that contain $A$. Moreover, $\neg A$ is removed from all remaining clauses:

$$\mathcal{X}^+ = \{ \{ A \} \mid A \in \mathcal{X}', \ A \neq A \}$$

$$\mathcal{X}^- = \{ \{ A \} \mid A \in \mathcal{X}', \ \neg A \neq A \}$$

Clearly, $A$ does not occur anymore in $\mathcal{X}^+$ and $\mathcal{X}^-$. Thus, $\mathcal{X}^+$ and $\mathcal{X}^-$ contain at most $n-1$ atomic formulas. $\mathcal{X}^+$ is unsatisfiable:

If $S \models \mathcal{X}^+$ then $S$ could be extended to a structure $S'$ with $S' \models A$. Then: $S' \models \mathcal{X}'$. By the unsatisfiability of $\mathcal{X}'$.

**Induction Hypothesis:**

$$\Box \in \text{Res}^d (\mathcal{X}^+), \ \Box \in \text{Res}^d (\mathcal{X}^-)$$

This means that there is a sequence of clauses $K_1, \ldots, K_m$ with $K_m = \Box$ and for all $1 \leq i \leq m$:

- $K_i \in \mathcal{X}^+$ or
- $K_i$ is a resolvent of $K_j$ and $K_k$ for $j, k \leq i$

If those clauses $K_i \in \mathcal{X}^+$ that were used in the resolution proof are also contained in $\mathcal{X}'$, then this is already a resolution proof from $\mathcal{X}'$, i.e., $\Box \in \text{Res}^d (\mathcal{X}')$.

Otherwise: re-insert $\neg A$ into those clauses $K_i$ where it had been removed. This yields again a resolution proof from $\mathcal{X}'$ ending in $\{ \neg A \}$. 

from $S'$ ending in $\neg A_j$.

(Reason: $K_j \lor K_k \lor K_j \lor \{\neg A_j \lor K_k\} \lor K_k$)

$\Rightarrow \{\neg A\} \in \text{Res}^d(S',\neg X')$.

Similarly, there is a resolution proof of $\Box$ from $S'$.
If this proof used only clauses from $S'$, then we directly have $\Box \in \text{Res}^d(S',\neg X')$.
Otherwise, re-insert $A$ into the clauses from $S'$:

$\Rightarrow \{A\} \in \text{Res}^d(S',\neg X')$.
One last resolution step yields $\square \in \text{Res}^d(S',\neg X')$:

$\{A\} \rightarrow \{\neg A\}$

Now we can improve the algorithm of Gilmore to the Ground Resolution Algorithm:

Advantage over Gilmore's Alg: better check for unsatisfiability

Same disadvantage as Gilmore: step from predicate to propositional logic is done via Herbrand-Expansion (instantiate variables by all possible ground terms)
Ground Res. Alg. is sound and complete:

- if \{q_1, \ldots, q_n\} \models q, then alg. terminates and returns "true"
- if \{q_1, \ldots, q_n\} \not\models q, then alg. does not return "true" (but it doesn't terminate in general)

Now: Improve the step from pred. to prop. logic
(avoid a blind guess by which ground terms one has to instantiate variables)