4.1 Resolution

Resolution Rule: Let \( \varphi \in \text{CNF} \), i.e., \( \varphi = \{ C_1, \ldots, C_m \} \) is a set of clauses on variables \( \{ p_0, \ldots, p_{n-1} \} \). Let \( C_1 = \{ \ell, \ell_1, \ldots, \ell_r \}, C_2 = \{ \bar{\ell}, \ell'_1, \ldots, \ell'_s \} \) be clauses in \( \varphi \) such that \( \ell \in C_1 \) and \( \bar{\ell} \in C_2 \), for some literal \( \ell \). Then the clause, \( C = \{ \ell_1, \ldots, \ell_r, \ell'_1, \ldots, \ell'_s \} \) is a resolvent of \( C_1, C_2 \), also called the resolvent of \( C_1, C_2 \) with pivot \( \ell \).

Example 4.1 Let \( \varphi = \{ C_1, C_2, C_3, C_4, C_5 \} \)
where \( C_1 = \{ -1, 2 \}, C_2 = \{ 1, -2 \}, C_3 = \{ -1, -3 \}, C_4 = \{ -2, 3 \}, C_5 = \{ 2, 3 \} \)
Resolution steps:
\[
\begin{align*}
C_6 & \overset{\text{def}}{=} \text{Res}(C_2, C_3) = \{ -2, -3 \} \\
C_7 & \overset{\text{def}}{=} \text{Res}(C_4, C_5) = \{ 3 \}
\end{align*}
\]
[A clause with exactly one variable is called a unit clause. In this case, \( p_3 \) must be true. Whenever we have a unit clause, we should use it wherever possible, i.e., in this case we can remove “\( \neg p_3 \)” from all clauses.]
\[
\begin{align*}
C_8 & \overset{\text{def}}{=} \text{Res}(C_6, C_7) = \{ -2 \} \\
C_9 & \overset{\text{def}}{=} \text{Res}(C_8, C_1) = \{ -1 \} \\
C_{10} & \overset{\text{def}}{=} \text{Res}(C_8, C_5) = \{ 3 \}
\end{align*}
\]
Note that we have shown that \( \varphi \vdash \neg p_1 \land \neg p_2 \land p_3 \). Furthermore, observe that any truth assignment that assigns 0, 0, 1 to \( p_1, p_2, p_3 \) satisfies \( \varphi \). Thus, \( \varphi \) is satisfiable and we have characterized all of its satisfying assignments, i.e., \( \varphi \equiv (\neg p_1 \land \neg p_2 \land p_3) \).

4.2 Two optimizations for Resolution

1. For any \( \varphi \in \text{CNF} \), if two clauses \( C_1 \) and \( C_2 \) from \( \varphi \) have 2 possible pivots \( p_1 \) and \( p_2 \), don’t bother resolving the clauses because the resolvent will result in a clause of the form \( \{ p_2, \neg p_2, \ell_3, \ldots \} \) which is worthless since this clause is equivalent to \( \top \), and thus provides no information.

2. If you have two clauses \( C \) and \( T \) s.t. \( C \subseteq T \), then you can get rid of \( T \) since \( C \) is stronger, i.e., \( (C \rightarrow T) \in \text{VALID} \). For example: \( C = \{ p_1 \lor \neg p_2 \} \) and \( T = \{ p_1 \lor \neg p_2 \lor p_4 \} \).

4.3 Definition of “proof”

Definition 4.2 [Proof] If \( \varphi \) is a set of clauses and \( \alpha \in \text{P}_{\text{fmla}} \), we say that, \( \varphi \proves \alpha \) (\( \varphi \vdash \alpha \)) iff there is a resolution proof of \( \Box \) from \( \varphi \cup \neg \alpha \). (Here we are assuming that \( \neg \alpha \) is a set of clauses, i.e.,
we put it in CNF.)

Example 4.3 We want to show that \{p \to q, q \to r\} \vdash p \to r. Thus \(\varphi = \{C_1, C_2\}\) where \(C_1 = \{\neg p, q\}\), \(C_2 = \{\neg q, r\}\) and \(\alpha = (\neg p \lor r), \) so \(\neg \alpha = \{C_3, C_4\}\) where \(C_3 = \{p\}\), \(C_4 = \{\neg r\}\). Observe that \(\varphi \cup \neg \alpha \vdash \square:\)
\[
\varphi \cup \neg \alpha = \{\neg p \lor q, \neg q \lor r, p, \neg r\}
\]
\(C_5 \overset{\text{def}}{=} \text{Res}(C_1, C_3) = \{q\}\)
\(C_6 \overset{\text{def}}{=} \text{Res}(C_5, C_2) = \{r\}\)
\(\square = \text{Res}(C_6, C_4)\)

Resolution is always trying to prove \(\square\), i.e., it uses proof by contradiction.

Xilin’s Comment: Resolution can prove that \((p \to q) \land (q \to r) \vdash (p \to r)\) directly, i.e., we didn’t have to do a proof by contradiction in that case: \(\{\neg p, r\} = \text{Res}(C_1, C_2)\)

4.4 Resolution is Complete for Prop Logic

Recall that we already proved that resolution is sound: If \(\varphi \vdash \square\) then \(\varphi\) is unsatisfiable.

Theorem 4.4 (Completeness Theorem for Propositional Logic) If \(\varphi\) is unsatisfiable then \(\varphi \vdash \square\).

It thus follows that if \(\varphi \vdash \alpha\) then \(\varphi \cup \neg \alpha\) is unsatisfiable and thus \(\varphi \cup \neg \alpha \vdash \square\), i.e., \(\varphi \vdash \alpha\).

Thus, \(\varphi \vdash \alpha\) iff \(\varphi \models \alpha\). Thus, we no longer need “semantically implies”.

From now in Prop Logic, we will only use “\(\models\)” to mean “models”, e.g., \(\mathcal{A} \models \varphi\).

Proof: (of Thm. 4.4) Let \(\varphi = \{C_1, C_2, \ldots, C_m\}\) over \(\text{var}(\varphi) = \{p_0, p_1, \ldots, p_{n-1}\}\), \(\varphi \in \text{UNSAT}\).

Goal: Show that from \(\varphi\) we can prove a contradiction using resolution (by induction on \(n\)).

Base case: \(n=0\): Since \(\varphi \in \text{UNSAT}\) and \(\text{VAR}(\varphi) = \emptyset, \square \in \varphi\).

Inductive case: Assume \(\text{indhyp}\): if \(\psi \in \text{UNSAT}\) and \(\text{var}(\Sigma) \subseteq \{p_0, p_1, \ldots, p_{n-2}\}\) then \(\psi \vdash \square\).

Idea: In order to use \(\text{indhyp}\) we need to get rid of \(p_{n-1}\). Note that in any possible truth assignment, \(p_n\) is either true or false.

Let \(\psi_0 \overset{\text{def}}{=} \varphi[\bot/p_{n-1}], \psi_1 \overset{\text{def}}{=} \varphi[\top/p_{n-1}],\) i.e., substitute \(\bot, \top\) for \(p_{n-1}\) in \(\varphi\).

Note that \(\psi_0, \psi_1 \in \text{UNSAT}\). We can see this because, for example, if \(\mathcal{A} \models \psi_0\), then \((\mathcal{A} \cup \{(p_{n-1}, 0)\}) \models \varphi\).

Thus, by \(\text{indhyp}\), \(\psi_0 \vdash \square\) and \(\psi_1 \vdash \square\).

Now we claim that from \(\psi_0 \vdash \square\) we can conclude that \(\varphi \vdash \{p_{n-1}\}\) and from \(\psi_1 \vdash \square\) and we can conclude that \(\varphi \vdash \{\neg p_{n-1}\}\). To see this, look at the transformation from \(\varphi\) to \(\psi_0\). What happens
is that any clause in which \( p_{n-1} \) does not occur is left alone, any clause having \( \neg p_{n-1} \) is removed because it is equivalent to \( \top \), and for any clause having \( p_{n-1} \), we delete the literal \( p_{n-1} \) because we know it is false. Thus, if we are trying to change the resolution proof of \( \psi_0 \vdash \Box \) to a similar proof from \( \varphi \), some of the clauses may have an extra \( p_{n-1} \). Thus, the final step in the proof might be \( \Box \), or it might be \( \{ \neg p_{n-1} \} \). Thus \( \varphi \vdash \{ p_{n-1} \} \).

By a similar argument, using the proof that \( \psi_1 \vdash \Box \), we conclude that \( \varphi \vdash \{ \neg p_{n-1} \} \). Then using one more resolution step, \( \varphi \vdash \Box = \text{Res}(\{ p_{n-1} \}, \{ \neg p_{n-1} \}) \).

Example 4.5  [to help understand the proof of Thm. 4.4]

In Example 4.3 we showed that \( \varphi \vdash \Box \) where

\[
\begin{align*}
\varphi &= \{ \neg p \lor q \}, \{ \neg q \lor r \}, \{ p \}, \{ \neg r \} = \{ A, B, C, D \} \\
\psi_0 &= \varphi[\bot/r] = \{ \neg p, q \}, \{ \neg q, r \}, \{ p \}, \{ \top \} = \{ \neg p \lor q \}, \{ \neg q \}, \{ p \} = \{ A, B_0, C \} \\
\psi_1 &= \varphi[\top/p_{n-1}] = \{ \neg p, q \}, \{ \neg q \lor \top \}, \{ p \}, \{ \bot \} = \{ \neg p \lor q \}, \{ p \}, \{ \Box \} = \{ A, C, D_0 \}
\end{align*}
\]

Then \( \psi_0 \vdash \Box \), with proof: \( E_0 = \text{Res}(A, B_0) = \{ \neg p \}; \Box = \text{Res}(E_0, C) \). If we pull this back to a proof from assumption \( \varphi \), we get:

\( \varphi \vdash \{ r \} \), with proof: \( E = \text{Res}(A, B) = \{ \neg p, r \}; \{ r \} = \text{Res}(E, C) \).

Also, \( \psi_1 \vdash \Box \) because \( D_0 = \Box \). If we pull this back to a proof from assumption \( \varphi \), we get \( \varphi \vdash \{ \neg r \} \) because \( D = \{ \neg r \} \in \varphi \).

Putting the two together with one more resolution step: \( \Box = \text{Res}(\{ r \}, \{ \neg r \}) \), we get \( \varphi \vdash \Box \) as desired. \( \Box \)

4.5 First Order Logic

First order logic is enough to model all of mathematics. It will allow us to talk about all mathematical objects and computer science objects of interest.

Let’s start with graphs. We will model a graph, \( G = (V, E) \) as a non-empty universe, \( V \), the vertices, and a binary relation \( E \subseteq V^2 \).

Here are a few graphs, with both the drawing and the set notation: \( G = (|G|, E^G) \), where \( |G| \) is the universe of \( G \), i.e., the vertices, and \( E^G \) is the set of edges of \( G \).
$G_0 = \{V = \{0, 1, 2\}, E^{G_0} = \{(0, 1), (2, 2)\}\}$

$G_1 = \{V = \{0, 1, 2\}, E^{G_1} = \{(0, 1), (1, 2), (2, 0)\}\}$

$G_\infty = \{V = \{0, 1, 2, 3, \ldots\}, E^{G_\infty} = \{(0, 1), (1, 2), (2, 3), \ldots\}\}$

$U_0$ (Undirected graph)

$U_0 = \{V = \{0, 1, 2, 3, 4\}, E^{U_0} = \{(0, 1), (1, 0), (1, 2), (2, 1), (3, 4), (4, 3)\}\}$
Here are some first-order (FO) formulas about graphs:

\[ \varphi_1 \equiv \forall x \exists y E(x, y) \quad \text{“Every vertex has an outgoing edge.”} \]

\[ G_1, G_N, U_0 \models \varphi_1; \quad G_0, G_2 \not\models \varphi_1 \]

\[ \varphi_2 \equiv \forall x \exists y z w (E(y, x) \land E(z, x) \land E(w, y) \land \text{distinct}(y, z, w)) \] where \[ \text{distinct}(y, z, w) \leftrightarrow (y \neq z \land z \neq w \land y \neq w) \]

“Every vertex has indegree \leq 2.”

\[ G_0, G_1, G_N, U_0 \models \varphi_1; \quad G_2 \not\models \varphi_1 \]

\[ \varphi_3 = \forall x y (E(x, y) \rightarrow E(y, x) \land \neg E(x, x)) \quad \text{“This graph is undirected and loop free.”} \]

\[ G_0, G_1, G_N, G_2 \not\models \varphi_3; \quad U_0 \models \varphi_3 \]