3.1 The method of resolution

Now we will describe a method for proving that a formula in CNF is unsatisfiable. Recall that a formula in CNF has the form \((\ell_{1,1} \lor \ldots, \ell_{1,n_1}) \land (\ell_{1,2} \lor \ldots, \ell_{1,n_2}) \land \ldots \land (\ell_{1,k} \lor \ldots, \ell_{1,n_k})\), where the \(\ell_{i,j}\) are literals. Since the order of the literals \(\ell_{i,1} \ldots \ell_{i,n_i}\) does not matter inside each conjunct, we can think of them as simply being a set of literals, which we call a clause. It is this observation which motivates the following definition, which essentially is just new terminology for parts of a formula in CNF.

**Definition 3.1.** A clause is a set of literals. A clause is satisfied by a valuation \(v\) if at least one of the literals in the clause is true according to the valuation. A set of clauses is satisfiable iff there is some valuation making each clause in the set satisfied.

We let \(\square\) denote the empty clause.

For example, the CNF formula \(\{(p \lor \neg q \lor r) \land (\neg p \lor q \lor r) \land (\neg p \lor \neg q \lor \neg r)\}\) is associated to the set of clauses \(\{\{p, \neg q, r\}, \{\neg p, q, r\}, \{\neg p, \neg q, \neg r\}\}\), and this set of clauses is satisfiable iff the original formula is.

Now suppose we have two disjunctions of literals where some propositional variable \(p\) appears in the first, and \(\neg p\) appears in the second. For example, \(p \lor q \lor r\) and \(\neg p \lor q \lor \neg t\). If both these disjunctions are true, then we can conclude that at least one of the remaining literals other than \(p\) and \(\neg p\) is true. This is because if \(p\) is false, then \(q\) or \(r\) is true (to make the first disjunction true), and if \(p\) is true, then \(q\) or \(\neg t\) is true. Hence, from \(p \lor q \lor r\) and \(\neg p \lor q \lor \neg t\) we can conclude \(q \lor r \lor \neg t\). This idea is what leads us to make the definition of a resolvent of two clauses.

**Definition 3.2.** Suppose \(c_1\) and \(c_2\) are clauses. We say that a third clause \(c\) is a resolvent of \(c_1\) and \(c_2\) if there is a propositional variable \(p\) such that \(c_1\) contains \(p\), \(c_2\) contains \(\neg p\), and \(c = (c_1 \setminus \{p\}) \cup (c_2 \setminus \{\neg p\})\).

We denote this by the diagram

\[
\begin{array}{c}
\text{c} \\
\downarrow \\
\text{c_1} \quad \text{c_2}
\end{array}
\]

Hence, for example:

- \(\{r, \neg s, t\}\) is a resolvent of \(\{\neg p, r, t\}\) and \(\{p, \neg s, r, t\}\).
- The empty clause \{\} is a resolvent of \{q\} and \{¬q\}.
- \{p, ¬p\} is a resolvent of \{p, ¬q\} and \{¬p, q\}.
- \{¬q, q\} is a resolvent of \{p, ¬q\} and \{¬p, q\}.

Note that the empty clause is unsatisfiable; there is no literal in this clause which can be made true by any valuation.

By our reasoning above, we have the following fact:

**Lemma 3.3.** Suppose \{c_1, c_2\} is a set of clauses and c is a resolvent of \{c_1\} and \{c_2\}. If \{c_1, c_2\} is satisfied by the valuation \(v\), then so is \(c\).

**Definition 3.4.** A proof by resolution of a clause \(c\) from a set of clauses \(C\) is a finite tree whose vertices are each assigned a clause such that

1. The root is assigned \(c\)
2. If a vertex assigned \(c\) has children it has exactly two children and \(c\) is a resolvent of the clauses assigned to them.
3. If a vertex has no children it is assigned a clause from \(C\).

We give an example of a proof by resolution of \(□\) from the set of clauses \{\{p, q, ¬r\}, \{¬p\}, \{p, q, r\}, \{p, ¬q\}\}:

```
□
/\     /
\  /\  /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\
\ /\ /\ /\  \\

\{p\}    \{¬p\}
\{p, q\}  \{¬p, q\}
\{p, q, r\}  \{p, q, ¬r\}
```

The following theorem is essentially soundness for proofs by resolution.

**Theorem 3.5** (Soundness for proofs by resolution). If there is a proof by resolution of a clause \(c\) from a set of clauses \(C\), then if \(C\) is satisfied by the valuation \(v\), then so is \(c\).

**Proof.** In class, proved by induction on height of the proof. The key fact one needs to use for the inductive step is Lemma 3.3 above.

In class we also proved the completeness theorem for proofs by resolution. We began with the following definition:
Definition 3.6. Suppose $C$ is a set of clauses. Then we define

$$C(p = \text{True}) = \{c \setminus \{\neg p\} : c \in C \land p \notin c\}.$$ 

and

$$C(p = \text{True}) = \{c \setminus \{p\} : c \in C \land \neg p \notin c\}.$$ 

For example, if $C = \{\{p, s\}, \{\neg p, q\}, \{\neg q, s\}, \{\neg p, q, \neg s\}\}$, then $C(p = \text{True}) = \{\{q\}, \{\neg q, s\}, \{q, \neg s\}\}$, and $C(p = \text{False}) = \{\{s\}, \{\neg q, s\}\}$.

Lemma 3.7. Suppose $C$ is a set of clauses. Then $C(p = \text{True})$ is satisfiable iff there is a valuation satisfying $C$ such that $p$ is true. Further, $C(p = \text{False})$ is satisfiable iff there is a valuation satisfying $C$ such that $p$ is false.

Theorem 3.8 (Completeness for proofs by resolution). If $C$ is an unsatisfiable set of clauses, then there is a proof by resolution of the empty clause from $C$.

Proof. Given in class. The rough idea is to prove the theorem by induction on the number of propositional variables $\{p_1, \ldots, p_n\}$ used in the set of clauses. Assuming $C$ is an unsatisfiable set of clauses in the variables $\{p_1, \ldots, p_n, p_{n+1}\}$, both $C(p_{n+1} = \text{True})$ and $C(p_{n+1} = \text{False})$ are unsatisfiable by Lemma 3.7.

By assumption, you then have proofs by resolution of $\Box$ from $C(p_{n+1} = \text{True})$ and $C(p_{n+1} = \text{False})$. These don’t work as proofs from $C$, but the only difference is that clauses from $C(p_{n+1} = \text{True})$ may be missing $\neg p_{n+1}$ and clauses from $C(p_{n+1} = \text{False})$ may be missing $p_{n+1}$. So if we put those variables back in any clauses that we removed them from, the worst-case scenario is that we now have proofs from $C$ of $p_{n+1}$ and $\neg p_{n+1}$. But then one more resolution gets us $\Box$.

3.2 A Hilbert-style formal proof system

One disadvantage of proofs of resolution is that they bear little resemblance to the mathematical proofs we generally use in practice. Next, we describe a formal proof system which is closer to the proof we usually use in mathematics. The Hilbert-style system we will describe has 3 logical axioms and 1 rule. This system deals only with proofs of formulas involving the connectives $\neg$ and $\rightarrow$.

Definition 3.9. For any formulas $\phi, \psi, \theta$, the following are logical axioms:

1. $\phi \rightarrow (\psi \rightarrow \phi)$.
2. $(\phi \rightarrow (\psi \rightarrow \theta)) \rightarrow ((\phi \rightarrow \psi) \rightarrow (\phi \rightarrow \theta))$
3. $((\neg \phi \rightarrow \neg \psi) \rightarrow ((\neg \phi \rightarrow \psi) \rightarrow \phi))$.

In addition, we have the rule (modus ponens) that if $\phi$ and $\phi \rightarrow \psi$ are true, then $\psi$ is true, which we represent with the diagram:
Now we can define a formal proof in this system as follows:

**Definition 3.10.** A *Hilbert-style proof* of a formula $\theta$ from a set of formulas $S$ is a finite tree whose vertices are each assigned a formula such that

1. The root is assigned $\theta$

2. If a vertex assigned $\psi$ has children it has exactly two children, one is assigned some formula $\phi$ and the other is assigned $\phi \rightarrow \psi$ (so this corresponds to an instance of modus ponens).

3. If a vertex has no children it is assigned a formula from $S$ or a logical axiom.

If there is such a proof $\theta$ from $S$, we write $S \vdash \theta$.

We give an example showing that $\{\neg p \rightarrow q, \neg p \rightarrow \neg q, p \rightarrow q\} \vdash q$

We give another example showing that from the empty set of formulas we can prove $p \rightarrow p$ (and similarly we can prove $\phi \rightarrow \phi$ for every formula $\phi$).