Chapter 4: Propositional Calculus: Resolution and BDDs

October 18, 2008
4.1 Resolution
4.1 Resolution

Definition
A formula is in **conjunctive normal form (CNF)** if it is a conjunction of disjunctions of literals.

Examples

(a) $p \land (\neg p \lor q \lor \neg r) \land (\neg q \lor q \lor r) \land (\neg q \lor p)$
Formula is in CNF

(b) $(\neg p \lor q \lor r) \land \neg (p \lor \neg r) \land q$
This formula is not in CNF
Theorem
Every propositional formula can be transformed into an equivalent formula in CNF.

Proof.
(Algorithm)
1. eliminate all connectives other than \( \neg \), \( \lor \), and \( \land \).
2. push all negations inward using De Morgan’s laws:
   \[
   \neg (A \lor B) \equiv \neg A \land \neg B \\
   \neg (A \land B) \equiv \neg A \lor \neg B
   \]
3. eliminate double negations
4. use distributivity to eliminate conjunctions within disjunctions:
   \[
   A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)
   \]
Example
Transform the formula

\[(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)\]

into an equivalent formula in CNF.

Solution:

\[(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)\]
\[\equiv (\neg p \lor q) \rightarrow (\neg \neg q \lor \neg p)\]
\[\equiv \neg (\neg p \lor q) \lor (\neg \neg q \lor \neg p)\]
\[\equiv (\neg \neg p \land \neg q) \lor (\neg \neg q \lor \neg p)\]
\[\equiv (p \land \neg q) \lor (q \lor \neg p)\]
\[\equiv (p \lor q \lor \neg p) \land (\neg q \lor q \lor \neg p)\]
Definition
A clause is a set of literals which is assumed (implicitly) to be a disjunction of those literals.

Example

\[ \neg p \lor q \lor \neg r \iff \{\neg p, q, \neg r\} \]

- **unit clause**: clause with only one literal; e.g. \{\neg q\}
- **clausal form** of a formula: implicit conjunction of clauses.
Example

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

\[ \iff \]

\[ \{ \{ p \}, \{ \neg p, q, \neg r \}, \{ \neg q, q, \neg r \}, \{ \neg q, p \} \} \]

- Abbreviated notation:

\[ \{ p, \bar{p}q\bar{r}, \bar{q}q\bar{r}, \bar{q}p \} \]

Notation:

- \( l \)-literal, \( l^c \)-complement of \( l \)
- \( C \)-clause (a set of literals)
- \( S \)-a clausal form (a set of clauses)
Properties of Clausal Forms

(1) If \( l \) appears in some clause of \( S \), but \( l^c \) does not appear in any clause, then, if we delete all clauses in \( S \) containing \( l \), the new clausal form \( S' \) is satisfiable if and only if \( S \) is satisfiable.

Example

Satisfiability of

\[
S = \{pq\,\bar{r}, p\bar{q}, \bar{p}q\}
\]

is equivalent to satisfiability of

\[
S' = \{p\bar{q}, \bar{p}q\}
\]
(2) Suppose \( C = \{l\} \) is a unit clause and we obtain \( S' \) from \( S \) by deleting \( C \) and \( l^c \) from all clauses that contain it. Then, \( S \) is satisfiable if and only if \( S' \) is satisfiable.

Example

\[
S = \{ p, \overline{pq}\overline{r}, \overline{qq}\overline{r}, q\overline{p} \}
\]

is satisfiable if and only if

\[
S' = \{ q\overline{r}, \overline{qq}\overline{r}, q \}
\]

is satisfiable.
(3) If $S$ contains two clauses $C$ and $C'$, such that $C \subseteq C'$, we can delete $C'$ without affecting the (un)satisfiability of $S$.

Example

$$S = \{p, \bar{p}q\bar{r}, \bar{q}q\bar{r}, q\bar{p}\}$$

is satisfiable if and only if

$$S' = \{p, \bar{q}q\bar{r}, q\bar{p}\}$$

is satisfiable.
(4) If a clause $C$ in $S$ contains a pair of complementary literals $l, l^c$, then $C$ can be deleted from $S$ without affecting its (un)satisfiability.

Example

$$S = \{p, \overline{pq}r, \overline{qq}r, q\overline{p}\}$$

is satisfiable if and only if

$$S' = \{p, \overline{pq}r, q\overline{p}\}$$

is such.
Definition
The empty clause will be denoted □. The empty set of clauses (i.e. the empty clausal form) will be denoted ∅.

Caution: We have to be careful not to confuse the empty clause with the empty clausal form.

For example,

\[ S = \{ p\bar{q}, \bar{p}qr, \square \} \]

is a nonempty clausal form \((S \neq \emptyset)\) which does contain the empty clause.
Resolution Rule

Suppose $C_1, C_2$ are clauses such that $l \in C_1$, $l^c \in C_2$. The clauses $C_1$ and $C_2$ are said to be **clashing clauses** and they clash on the complementary literals $l, l^c$.

$C$, the **resolvent** of $C_1, C_2$ is the clause

$$Res(C_1, C_2) = (C_1 - \{l\}) \cup (C_2 - \{l^c\})$$

$C_1$ and $C_2$ are called the **parent clauses** of $C$. 

$$C = (C_1 - \{l\}) \cup (C_2 - \{l^c\})$$
Example

The clauses

\[ C_1 = \bar{p}q\bar{r}, \quad C_2 = \bar{q}p \]

clash on \( p, \bar{p} \).

\[ \text{Res}(C_1, C_2) = q\bar{r} \cup \bar{q} = q\bar{r}\bar{q} \]

\( C_1, C_2 \) also clash on \( q, \bar{q} \), so, another way to find a resolvent for these two clauses is

\[ \text{Res}(C_1, C_2) = \bar{p}\bar{r} \cup p = \bar{p}\bar{r}p \]
Theorem
Resolvent C is satisfiable if and only if the parent clauses $C_1, C_2$ are simultaneously satisfiable.

Proof.
(⇐) Suppose $C_1$ and $C_2$ are simultaneously satisfiable, and let $v$ be a truth-assignment which makes all formulas in $C_1$ and $C_2$ true. Let $l, l^c$ be the pair of clashing literals used in resolving $C_1$ and $C_2$. Then, either

- $v(l) = T, v(l^c) = F$; or
- $v(l) = F, v(l^c) = T$
If $v(l) = T$, then $C_2$ can be satisfied only if $v(l') = T$, for some literal $l'$ different from $l^c$.
Since $l'$ still appears in $Res(C_1, C_2)$, the resolvent clause will be satisfied by $v$. The other possibility is handled analogously.
Suppose the resolvent $C$ is satisfiable. Then, for some truth-assignment $v$ and some literal $l' \in C$, we have

$$v(l') = T$$

By resolution, this $l'$ was originally either in $C_1$ or in $C_2$ (or, maybe, both). Then, it is not difficult to see that it is possible to extend this assignment $v$ to the deleted literals $l$ and $l^c$ so that both clauses are satisfied by $v$. \qed
Resolution Algorithm

**Input:** \( S \) - a set of clauses  
**Output:** “\( S \) is satisfiable” or “\( S \) is not satisfiable”

1. Set \( S_0 := S \).
2. Suppose \( S_i \) has already been constructed.
3. To construct \( S_{i+1} \), choose a pair of clashing literals and clauses \( C_1, C_2 \) in \( S \) (if there are any) and derive
   
   \[
   C := Res(C_1, C_2) \\
   S_{i+1} := S_i \cup \{ C \}
   \]

4. If \( C = \Box \), output “\( S \) is not satisfiable”; if \( S_{i+1} = S_i \), output “\( S \) is satisfiable”.
5. Otherwise, set \( i := i + 1 \) and go back to Step 2.
Example
Determine whether

\[ S = \{ \bar{p}q, \bar{q}\bar{r}s, p, r, \bar{s} \} \]

is satisfiable.

Solution:

1. \[ S_0 = \{ \bar{p}q, \bar{q}\bar{r}s, p, r, \bar{s} \} \]
2. \[ C_1 = \bar{p}q, C_2 = p, C = q, \quad S_1 = \{ \bar{p}q, \bar{q}\bar{r}s, p, r, \bar{s}, q \} \]
3. \[ C_1 = \bar{q}\bar{r}s, C_2 = q, C = \bar{r}s, \quad S_2 = \{ \bar{p}q, \bar{q}\bar{r}s, p, r, \bar{s}, q, \bar{r}s \} \]
4. \[ C_1 = r, C_2 = \bar{r}s, C = s, \quad S_3 = \{ \bar{p}q, \bar{q}\bar{r}s, p, r, \bar{s}, q, \bar{r}s, s \} \]
5. \[ C_1 = \bar{s}, C_2 = s, C = \square \]

\[ S \] is not satisfiable.
In the preceding example, we can use facts about sets of clauses (1)-(4), mentioned earlier, in order to keep the sets $S_i$ shorter; the drawback is that this approach requires a large number of checks before reducing the set $S_i$ to a simplified set $S'_i$ in each step.

1. $S_0 = \{\overline{pq}, \overline{qrs}, p, r, s\}$
2. $C_1 = \overline{pq}, C_2 = p, C = q, \quad S_1 = \{\overline{pq}, \overline{qrs}, p, r, s, q\}$ which can be reduced to $S'_1 = \{\overline{qrs}, p, r, s, q\}$
3. $C_1 = \overline{qrs}, C_2 = q, C = r, \quad S_2 = \{\overline{qrs}, p, r, s, q, \overline{rs}\}$ which can be reduced to $S'_2 = \{p, r, s, q, \overline{rs}\}$
4. $C_1 = r, C_2 = \overline{rs}, C = s, \quad S_3 = \{p, r, s, q, \overline{rs}, s\}$ which can be reduced to $S'_3 = \{p, r, s, q, s\}$
5. $C_1 = \overline{s}, C_2 = s, C = \square$
Example

Show that

\[(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)\]

is a valid formula.

Solution: We will show that

\[\neg[(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)]\]

is not satisfiable

(1) Transform the formula into CNF:

\[\neg[(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)] \equiv (p \rightarrow q) \land \neg(\neg q \rightarrow \neg p)\]

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

\[\equiv (\neg p \lor q) \land \neg q \land p\]
(2) Show, using resolution, that

\[ S = \{ \bar{p}q, \bar{q}, p \} \]

1. \( S_0 = \{ \bar{p}q, \bar{q}, p \} \)
2. \( C_1 = \bar{p}q, \quad C_2 = \bar{q}, \quad C = \bar{p}, \quad S_1 = \{ \bar{p}q, \bar{q}, p, \bar{p} \} \)
3. \( C_1 = p, \quad C_2 = \bar{p}, \quad C = \Box \)

**Definition**
A derivation of \( \Box \) from \( S \) is called a **refutation** of \( S \).
Soundness and Completeness

Theorem
If the set of clauses labeling the leaves of a resolution tree is satisfiable, then the clause at the root is satisfiable.

Proof.
This is a simple consequence of a theorem proved earlier.

Theorem
(Soundness) If the empty clause □ is derived from a set of clauses, then the set of clauses is unsatisfiable.

Theorem
(Completeness) If a set of clauses is unsatisfiable, then the empty clause □ can be derived from it using resolution algorithm.