

# Propositional Logic Resolution

# Clause representation of CNF formulas

CNF:

$$(L_{1,1} \vee \dots \vee L_{1,n_1}) \wedge \dots \wedge (L_{k,1} \vee \dots \vee L_{1,n_k})$$

Representation as set of sets of literals:

$$\underbrace{\{L_{1,1}, \dots, L_{1,n_1}\}}_{\text{clause}}, \dots, \{L_{k,1}, \dots, L_{1,n_k}\}$$

- ▶ **Clause** = set of literals (disjunction).
- ▶ A formula in CNF can be viewed as a **set of clauses**
- ▶ Degenerate cases:
  - ▶ The empty clause stands for  $\perp$ .
  - ▶ The empty set of clauses stands for  $\top$ .

# The joy of sets

We get “for free”:

- ▶ **Commutativity:**

$A \vee B \equiv B \vee A$ , both represented by  $\{A, B\}$

- ▶ **Associativity:**

$(A \vee B) \vee C \equiv A \vee (B \vee C)$ , both represented by  $\{A, B, C\}$

- ▶ **Idempotence:**

$(A \vee A) \equiv A$ , both represented by  $\{A\}$

Sets are a convenient representation of conjunctions and disjunctions that build in associativity, commutativity and idempotence

## Resolution — The idea

Input: Set of clauses  $F$

Question: Is  $F$  unsatisfiable?

Algorithm:

Keep on “resolving” two clauses from  $F$  and adding the result to  $F$  until the empty clause is found

**Correctness:**

If the empty clause is found, the initial  $F$  is unsatisfiable

**Completeness:**

If the initial  $F$  is unsatisfiable, the empty clause can be found.

Correctness/Completeness of **syntactic** procedure (**resolution**)  
w.r.t. **semantic** property (**unsatisfiability**)

# Resolvent

## Definition

Let  $L$  be a literal. Then  $\bar{L}$  is defined as follows:

$$\bar{L} = \begin{cases} \neg A_i & \text{if } L = A_i \\ A_i & \text{if } L = \neg A_i \end{cases}$$

## Definition

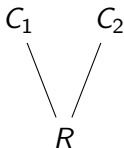
Let  $C_1, C_2$  be clauses and let  $L$  be a literal such that  $L \in C_1$  and  $\bar{L} \in C_2$ . Then the clause

$$(C_1 - \{L\}) \cup (C_2 - \{\bar{L}\})$$

is a **resolvent** of  $C_1$  and  $C_2$ .

The process of deriving the resolvent is called a **resolution step**.

Graphical representation of resolvent:



If  $C_1 = \{L\}$  and  $C_2 = \{\bar{L}\}$  then the empty clause is a resolvent of  $C_1$  and  $C_2$ . The special symbol  $\square$  denotes the empty clause.

Recall:  $\square$  represents  $\perp$ .

# Resolution proof

## Definition

A **resolution proof** of a clause  $C$  from a set of clauses  $F$  is a sequence of clauses  $C_0, \dots, C_n$  such that

- ▶  $C_i \in F$  or  $C_i$  is a resolvent of two clauses  $C_a$  and  $C_b$ ,  $a, b < i$ ,
- ▶  $C_n = C$

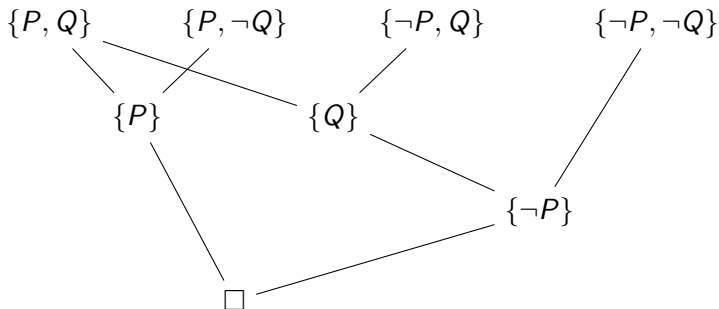
Then we can write  $F \vdash_{Res} C$ .

Note:  $F$  can be finite or infinite

## Resolution proof as DAG

A resolution proof can be shown as a DAG with the clauses in  $F$  as the leaves and  $C$  as the root:

### Example





## A linear resolution proof

0:  $\{P, Q\}$

1:  $\{P, \neg Q\}$

2:  $\{\neg P, Q\}$

3:  $\{\neg P, \neg Q\}$

4:  $\{P\}$  (0, 1)

5:  $\{Q\}$  (0, 2)

6:  $\{\neg P\}$  (3, 5)

7:  $\square$  (4, 6)

## Correctness of resolution

### Lemma (Resolution Lemma)

Let  $R$  be a resolvent of two clauses  $C_1$  and  $C_2$ . Then  $C_1, C_2 \models R$ .

**Proof** By definition  $R = (C_1 - \{L\}) \cup (C_2 - \{\bar{L}\})$  (for some  $L$ ).

Let  $\mathcal{A} \models C_1$  and  $\mathcal{A} \models C_2$ . There are two cases.

If  $\mathcal{A} \models L$  then  $\mathcal{A} \models C_2 - \{\bar{L}\}$  (because  $\mathcal{A} \models C_2$ ), thus  $\mathcal{A} \models R$ .

If  $\mathcal{A} \not\models L$  then  $\mathcal{A} \models C_1 - \{L\}$  (because  $\mathcal{A} \models C_1$ ), thus  $\mathcal{A} \models R$ .

### Theorem (Correctness of resolution)

Let  $F$  be a set of clauses. If  $F \vdash_{Res} C$  then  $F \models C$ .

**Proof** Assume there is a resolution proof  $C_0, \dots, C_n = C$ .

By induction on  $i$  we show  $F \models C_i$ . IH:  $F \models C_j$  for all  $j < i$ .

If  $C_i \in F$  then  $F \models C_i$  is trivial. If  $C_i$  is a resolvent of  $C_a$  and  $C_b$ ,  $a, b < i$ , then  $F \models C_a$  and  $F \models C_b$  by IH and  $C_a, C_b \models C_i$  by the resolution lemma. Thus  $F \models C_i$ .

### Corollary

Let  $F$  be a set of clauses. If  $F \vdash_{Res} \square$  then  $F$  is unsatisfiable.

# Completeness of resolution

## Theorem (Completeness of resolution)

Let  $F$  be a set of clauses. *If  $F$  is unsatisfiable then  $F \vdash_{Res} \square$ .*

**Proof** If  $F$  is infinite, there must be a finite unsatisfiable subset of  $F$  (by the Compactness Lemma); in that case let  $F$  be that finite subset. The proof of  $F \vdash_{Res} \square$  is by induction on the number of atoms in  $F$ .

## Corollary

*A set of clauses  $F$  is unsatisfiable iff  $F \vdash_{Res} \square$ .*

# Resolution is only refutation complete

Not everything that is a consequence of a set of clauses  
can be derived by resolution.

## Exercise

Find  $F$  and  $C$  such that  $F \models C$  but not  $F \vdash_{Res} C$ .

How to prove  $F \models C$  by resolution?

Prove  $F \cup \{\neg C\} \vdash_{Res} \square$

## A resolution algorithm

Input: A CNF formula  $F$ , i.e. a finite set of clauses

**while** there are clauses  $C_a, C_b \in F$  and resolvent  $R$  of  $C_a$  and  $C_b$   
such that  $R \notin F$   
**do**  $F := F \cup \{R\}$

Lemma

*The algorithm terminates.*

**Proof** There are only finitely many clauses over a finite set of atoms.

Theorem

*The initial  $F$  is unsatisfiable iff  $\square$  is in the final  $F$*

**Proof**  $F_{init}$  is unsat. iff  $F_{init} \vdash_{Res} \square$  iff  $\square \in F_{final}$   
because the algorithm enumerates all  $R$  such that  $F_{init} \vdash R$ .

Corollary

*The algorithm is a decision procedure for unsatisfiability of CNF formulas.*