

CS 512, Spring 2017, Handout 11

**Resolution in Propositional Logic**

Assaf Kfoury

8 February 2017

# Origins and background

- ▶ The **resolution method** was introduced around 1960 by Martin Davis (1928-) and Hilary Putnam (1926-2016), then gradually adapted and developed in later years.
- ▶ Like the tableaux method, the **resolution method** is said to be **refutation-based**. This means it tries to find reasons why a WFF  $\varphi$  is a logical contradiction. More generally, it tries to find reasons why a finite set  $\Gamma$  of WFF's is not satisfiable.
- ▶ Like the tableaux method, it turns out that **resolution** is **refutation-complete**.
- ▶ As pointed out in Handout 09, **refutation completeness** is not a serious limitation of the method, *e.g.*, it can also be used to decide any *semantic entailment*  $\Gamma \models \varphi$ , with  $\Gamma$  a finite set of WFF's and  $\varphi$  any WFF (not restricted to  $\varphi = \perp$ ).
- ▶ Later in the handout, we show that **resolution** can also be used to decide *satisfiability* of an arbitrary WFF  $\varphi$ .
- ▶ This handout is limited to the **resolution method** for *classical propositional logic*, its extension to *first-order logic* is taken up in a later handout.

# Efficient Transformation Into CNF

**Resolution** assumes that a WFF  $\varphi$  to be tested for non-satisfiability is in CNF.

- ▶ Before applying the method, we therefore need an efficient way of translating an arbitrary WFF  $\varphi$  into another WFF  $\psi$  in CNF.
- ▶ **Bad news:** Translating an arbitrary  $\varphi$  into an **equivalent** CNF  $\psi$  generally results in an exponential blow-up (see Handout 10).
- ▶ **Good news:** It is possible to efficiently translate an arbitrary WFF  $\varphi$  into another WFF  $\psi$  in CNF so that  $\varphi$  and  $\psi$  are **equisatisfiable** though not necessarily **equivalent**.

(There is more than one way of doing this – see next slide. For more on **equisatisfiability**, click [here](#) .)

- ▶ If  $\varphi$  is a propositional WFF in CNF, we may write:

$$\varphi = \{C_1, \dots, C_n\}, \quad \text{i.e., a finite set of clauses}$$

instead of  $\varphi = C_1 \wedge \dots \wedge C_n$  where each  $C_i$  is a disjunction of literals.

# Efficient Transformation Into CNF

1. Already pointed out in Handout 10, the transformation of the WFF:

$$\varphi = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \cdots \vee (x_n \wedge y_n)$$

into CNF produces an equivalent WFF of size  $\mathcal{O}(2^n)$ , an exponential blow-up.

2. However, the transformation of  $\varphi$  into the following WFF  $\psi$ :

$$\psi = (z_1 \vee \cdots \vee z_n) \wedge (\neg z_1 \vee x_1) \wedge (\neg z_1 \vee y_1) \wedge \cdots \wedge (\neg z_n \vee x_n) \wedge (\neg z_n \vee y_n)$$

produces a WFF in CNF of size  $\mathcal{O}(n)$  such that  $\varphi$  and  $\psi$  are equisatisfiable (though not equivalent), where  $\{z_1, \dots, z_n\}$  are fresh propositional variables.

**Exercise:** Show that  $\varphi$  (in part 1 above) and  $\psi$  (in part 2) are equisatisfiable, *i.e.*, if there is truth-value assignment  $\sigma$  satisfying  $\varphi$  (resp.  $\psi$ ), then there is a truth-value assignment  $\sigma'$  satisfying  $\psi$  (resp.  $\varphi$ ).

3. An alternative translation of a WFF  $\varphi$  into an equisatisfiable  $\psi$  is the so-called Tseitin transformation. The Tseitin transformation includes also the clauses  $z_i \vee \neg x_i \vee \neg y_i$  for every  $i = 1, \dots, n$ . With these clauses, the initial WFF  $\varphi$  implies  $z_i \equiv x_i \wedge y_i$ ; in the new WFF  $\psi$  we can view  $z_i$  as a name for “ $x_i \wedge y_i$ ”.

**Exercise:** Look up “Tseitin transformation” on the Web for more details.

4. A specific efficient algorithm, called  $\text{CNF}(\ )$ , to transform an arbitrary propositional WFF  $\varphi$  into an equisatisfiable WFF is presented next.

# Efficient Transformation Into CNF

The definition of  $\text{CNF}(\varphi, \Delta)$  is by induction on WFF's. Because it is inductive, it translates into a recursive algorithm, where  $\Delta$  is a finite set of clauses:<sup>1</sup>

1.  $\text{CNF}(p, \Delta) := \langle p, \Delta \rangle$
2.  $\text{CNF}(\neg\varphi, \Delta) := \langle \neg l, \Delta' \rangle$  where  $\text{CNF}(\varphi, \Delta) = \langle l, \Delta' \rangle$
3.  $\text{CNF}(\varphi_1 \wedge \varphi_2, \Delta) := \langle p, \Delta' \rangle$  where  
 $\text{CNF}(\varphi_1, \Delta) = \langle l_1, \Delta_1 \rangle$ ,  $\text{CNF}(\varphi_2, \Delta_1) = \langle l_2, \Delta_2 \rangle$ ,  
 $p$  is a fresh atom (propositional variable),  
 $\Delta' = \Delta_2 \cup \{ \neg p \vee l_1, \neg p \vee l_2, \neg l_1 \vee \neg l_2 \vee p \}$
4.  $\text{CNF}(\varphi_1 \vee \varphi_2, \Delta) := \langle p, \Delta' \rangle$  where  
 $\text{CNF}(\varphi_1, \Delta) = \langle l_1, \Delta_1 \rangle$ ,  $\text{CNF}(\varphi_2, \Delta_1) = \langle l_2, \Delta_2 \rangle$ ,  
 $p$  is a fresh atom (propositional variable),  
 $\Delta' = \Delta_2 \cup \{ \neg p \vee l_1 \vee l_2, \neg l_1 \vee p, \neg l_2 \vee p \}$

(If you prefer, every “:=” above can be replaced by “return”).

<sup>1</sup> Taken from Leonardo De Moura, “SMT Solvers: Theory and Implementation”, Microsoft Research 2008.

# Efficient Transformation Into CNF

## Theorem

Let  $\varphi$  be an arbitrary propositional WFF and let  $\text{CNF}(\varphi, \emptyset) = \langle \ell, \Delta \rangle$ .  
Then  $\varphi$  is satisfiable iff  $\{\ell\} \cup \Delta$  is satisfiable.

## Proof.

Left to you. *Hint:* You will need to use structural induction on  $\varphi$ . □

## Exercise

Carry out the transformation  $\text{CNF}(\varphi, \emptyset)$  where

$$\varphi := \neg((q_1 \vee \neg q_2) \wedge q_3)$$

## Exercise

Search the Web for improvements on the transformation  $\text{CNF}()$ .  
(How about introducing multi-arity  $\wedge$  and multi-arity  $\vee$ ?)

# Resolution Rule

- ▶ The rule is limited to propositional WFF's in CNF.
- ▶ The rule can be **used by itself** to establish that an arbitrary CNF is unsatisfiable.

# Resolution Rule

- ▶ The rule is limited to propositional WFF's in CNF.
- ▶ The rule can be **used by itself** to establish that an arbitrary CNF is unsatisfiable.
- ▶ CNF clauses are each a disjunction of literals (atoms and negated atoms).
- ▶ The **antecedents** of the **resolution rule** are two clauses of a CNF:

$$(\ell_1 \vee \cdots \vee \ell_{p-1} \vee \ell_p \vee \ell_{p+1} \cdots \vee \ell_m) \quad \text{and}$$

$$(\ell'_1 \vee \cdots \vee \ell'_{q-1} \vee \ell'_q \vee \ell'_{q+1} \cdots \vee \ell'_n)$$

where all  $\ell_i$  and  $\ell'_j$  are literals, and  $\ell'_q = \neg \ell_p$ .



# Resolution Rule

- ▶ The rule is limited to propositional WFF's in CNF.
- ▶ The rule can be **used by itself** to establish that an arbitrary CNF is unsatisfiable.
- ▶ CNF clauses are each a disjunction of literals (atoms and negated atoms).
- ▶ The **antecedents** of the **resolution rule** are two clauses of a CNF:

$$(\ell_1 \vee \cdots \vee \ell_{p-1} \vee \ell_p \vee \ell_{p+1} \cdots \vee \ell_m) \quad \text{and}$$
$$(\ell'_1 \vee \cdots \vee \ell'_{q-1} \vee \ell'_q \vee \ell'_{q+1} \cdots \vee \ell'_n)$$

where all  $\ell_i$  and  $\ell'_j$  are literals, and  $\ell'_q = \neg\ell_p$ .

- ▶ The **resolution rule** applied to the pair  $(\ell_p, \ell'_q)$  where  $\ell'_q = \neg\ell_p$  is:

$$\frac{(\ell_1 \vee \cdots \vee \ell_{p-1} \vee \ell_p \vee \ell_{p+1} \cdots \vee \ell_m) \quad (\ell'_1 \vee \cdots \vee \ell'_{q-1} \vee \ell'_q \vee \ell'_{q+1} \cdots \vee \ell'_n)}{\ell_1 \vee \cdots \vee \ell_{p-1} \vee \ell_{p+1} \cdots \vee \ell_m \vee \ell'_1 \vee \cdots \vee \ell'_{q-1} \vee \ell'_{q+1} \cdots \vee \ell'_n}$$

New clause produced by **resolution** (below the line) is the **resolvent**. Note that  $\ell_p$  and  $\ell'_q$  are **not** mentioned in the **resolvent**, so that the size of the resolvent is less than the size of the two antecedents together.

# Resolution Rule

- ▶ The **resolution rule** applied to the pair  $(\ell_p, \ell'_q)$  where  $\ell'_q = \neg\ell_p$  in the special case when the two **antecedents** have each only one literal:

$$\frac{\ell_p \quad \ell'_q}{\perp}$$

In this case the **resolvent** is  $\perp$  (**falsity**).

## Exercise

Show that MP (*modus ponens*) can be viewed as a special case of the **resolution** rule.

# Resolution Rule: how to use it

- ▶ Before some examples, how strong is **resolution**?

# Resolution Rule: how to use it

- ▶ Before some examples, how strong is **resolution**?
- ▶ **Resolution** is a **sound** and **refutation-complete** system of formal proofs for CNF's, *i.e.*, **resolution is strong enough!**

# Resolution Rule: how to use it

- ▶ Before some examples, how strong is **resolution**?
- ▶ **Resolution** is a **sound** and **refutation-complete** system of formal proofs for CNF's, *i.e.*, **resolution is strong enough!**

From [LCS, Chapt 1], we already know:

## Theorem

Let  $\varphi$  be a propositional WFF. The following are equivalent statements:

1.  $\varphi$  is valid, *i.e.*, formally derivable using **natural deduction**.
2.  $\varphi$  is a tautology, *i.e.*, entries of last column of its **truth-table** are all **T**.
3.  $\neg\varphi$  is a contradiction, *i.e.*,  $\perp$  is formally derivable from  $\neg\varphi$  using **natural deduction**.
4.  $\neg\varphi$  is unsatisfiable, *i.e.*, entries of last column of its **truth-table** are all **F**.

# Resolution Rule: how to use it

- ▶ Before some examples, how strong is **resolution**?
- ▶ **Resolution** is a **sound** and **refutation-complete** system of formal proofs for CNF's, *i.e.*, **resolution is strong enough!**

From [LCS, Chapt 1], we already know:

## Theorem

Let  $\varphi$  be a propositional WFF. The following are equivalent statements:

1.  $\varphi$  is valid, *i.e.*, formally derivable using **natural deduction**.
2.  $\varphi$  is a tautology, *i.e.*, entries of last column of its **truth-table** are all **T**.
3.  $\neg\varphi$  is a contradiction, *i.e.*,  $\perp$  is formally derivable from  $\neg\varphi$  using **natural deduction**.
4.  $\neg\varphi$  is unsatisfiable, *i.e.*, entries of last column of its **truth-table** are all **F**.

We can specialize preceding theorem to CNF's and restrict it to parts 3 and 4, to express the soundness (part 3  $\Rightarrow$  part 4) and refutation-completeness (part 4  $\Rightarrow$  part 3) of resolution:

## Theorem

Let  $\psi$  be a propositional WFF in CNF. The following are equivalent statements:

3.  $\psi$  is a contradiction, *i.e.*,  $\perp$  is derivable from  $\psi$  using **resolution**, in shorthand  $\psi \vdash_{\text{res}} \perp$ .
4.  $\psi$  is unsatisfiable, *i.e.*, entries of last column of its **truth-table** are all **F**.

# Resolution Rule: how to use it

- ▶ Observe carefully how **refutation-completeness** is used:

1. From the two clauses  $\{p, q\}$ , representing the CNF  $p \wedge q$ , we can**NOT** apply **resolution** to formally derive  $p \vee q$ , even though we know  $p, q \models p \vee q$ .

**This is why resolution is not complete, but only refutation-complete.**

2. Completeness of another formal-proof system, such as **natural deduction**, means that  $\boxed{\text{if } p, q \models p \vee q \text{ then } p, q \vdash_{\text{ND}} p \vee q}$ .

**This is the standard statement of completeness for natural deduction.**

3. From outside the **resolution**-based formal-proof system, *i.e.*, at the meta-level, we know:  $\boxed{\text{if } p, q \not\vdash_{\text{ND}} p \vee q \text{ then } p, q \not\models p \vee q}$ , which is again completeness for natural deduction stated contrapositively.

**Can we use resolution to show that if  $p, q \not\vdash_{\text{ND}} p \vee q$  then  $p, q \not\models p \vee q$ ??**

4. Yes, this is possible. At the meta-level,  $p, q \not\vdash_{\text{ND}} p \vee q$  means the same thing as  $\{p, q, \neg(p \vee q)\}$  is **contradictory** (why?), and **resolution** can derive this **contradiction**, which will in turn imply  $p, q \not\models p \vee q$ , which will also imply  $p, q \models \neg(p \vee q)$  (why?).

# Resolution Rule: how to use it

Suppose we want to decide whether WFF  $\psi$  is formally derivable from a finite **knowledge base**  $KB = \{\varphi_1, \dots, \varphi_n\}$ , *i.e.*, whether  $KB \vdash_{ND} \psi$  using natural deduction (or some other formal proof system).

The following are the steps to show that  $KB \vdash_{res} \psi$  using **resolution** instead:



# Resolution Rule: how to use it

Suppose we want to decide whether WFF  $\psi$  is formally derivable from a finite **knowledge base**  $KB = \{\varphi_1, \dots, \varphi_n\}$ , *i.e.*, whether  $KB \vdash_{ND} \psi$  using natural deduction (or some other formal proof system).

The following are the steps to show that  $KB \vdash_{res} \psi$  using **resolution** instead:

- ▶ Negate  $\psi$  and add  $\neg\psi$  to KB.
- ▶ Transform  $KB \cup \{\neg\psi\}$  into a single CNF, thus obtaining a finite set of clauses.
- ▶ Apply the **resolution rule** repeatedly, until there is no resolvable pair of clauses.  
*(The procedure is bound to terminate – why?)*
- ▶ Every time the **resolution rule** is applied, add the **resolvent** (which is a new clause) to the knowledge base.
- ▶ If  $\perp$  (the empty clause) is produced, stop and report that the original  $\psi$  is formally derivable from  $\varphi_1, \dots, \varphi_n$ , *i.e.*,  $\varphi_1, \dots, \varphi_n \vdash_{res} \psi$ .
  
- ▶ An example is on the next slide.

# Resolution Rule: small example

Is the WFF  $\neg P$  derivable from the **knowledge base**  $\{P \rightarrow Q, Q \rightarrow R, \neg R\}$ ?

- ▶ Negate the initial WFF  $\neg\neg P = P$  and add it to the **knowledge base**.
- ▶ Transform all WFF's in the **knowledge base** into CNF:  $\{\neg P \vee Q, \neg Q \vee R, \neg R, P\}$ .
- ▶ Putting down every clause in the **knowledge base** first, then applying the resolution rule repeatedly, we obtain:

# Resolution Rule: small example

Is the WFF  $\neg P$  derivable from the **knowledge base**  $\{P \rightarrow Q, Q \rightarrow R, \neg R\}$ ?

- ▶ Negate the initial WFF  $\neg\neg P = P$  and add it to the **knowledge base**.
- ▶ Transform all WFF's in the **knowledge base** into CNF:  $\{\neg P \vee Q, \neg Q \vee R, \neg R, P\}$ .
- ▶ Putting down every clause in the **knowledge base** first, then applying the resolution rule repeatedly, we obtain:

$$1 \quad \neg P \vee Q$$

$$2 \quad \neg Q \vee R$$

$$3 \quad \neg R$$

$$4 \quad P$$

# Resolution Rule: small example

Is the WFF  $\neg P$  derivable from the **knowledge base**  $\{P \rightarrow Q, Q \rightarrow R, \neg R\}$ ?

- ▶ Negate the initial WFF  $\neg\neg P = P$  and add it to the **knowledge base**.
- ▶ Transform all WFF's in the **knowledge base** into CNF:  $\{\neg P \vee Q, \neg Q \vee R, \neg R, P\}$ .
- ▶ Putting down every clause in the **knowledge base** first, then applying the resolution rule repeatedly, we obtain:

$$1 \quad \neg P \vee Q$$

$$2 \quad \neg Q \vee R$$

$$3 \quad \neg R$$

$$4 \quad P$$

$$5 \quad Q \qquad \qquad \qquad \text{resolve 1, 4}$$

$$6 \quad R \qquad \qquad \qquad \text{resolve 2, 5}$$

$$7 \quad \perp \qquad \qquad \qquad \text{resolve 3, 6}$$

- ▶ Stop and report that the initial WFF  $\neg P$  is formally derivable from  $\{P \rightarrow Q, Q \rightarrow R, \neg R\}$ .

# Resolution Rule: how to use it

Suppose we want to decide whether propositional WFF  $\varphi$  is **satisfiable**.

The following are the steps of a procedure to decide satisfiability using **resolution**:

# Resolution Rule: how to use it

Suppose we want to decide whether propositional WFF  $\varphi$  is **satisfiable**.

The following are the steps of a procedure to decide satisfiability using **resolution**:

- ▶ Transform  $\varphi$  into CNF, to obtain a finite set of clauses, the initial **knowledge base**.
- ▶ Apply the **resolution rule** repeatedly, until there is no resolvable pair of clauses.  
*(The procedure is bound to terminate – why?)*
- ▶ Every time the **resolution rule** is applied, add the **resolvent** (a new clause) to the **knowledge base**.
- ▶ If  $\perp$  (the empty clause) is produced, stop and report that the original  $\varphi$  is **unsatisfiable**.
- ▶ If there are no more resolvable pair of clauses (and  $\perp$  is not produced), stop and report that the original  $\varphi$  is **satisfiable**.
  
- ▶ An example is on the next slide.

# Resolution Rule: small example

Let  $\varphi := (q_1 \vee q_2 \vee q_3) \wedge (q_2 \vee \neg q_3 \vee \neg q_4) \wedge (\neg q_2 \vee q_5)$ , which is already a CNF.

- ▶ Is  $\varphi$  satisfiable?

# Resolution Rule: small example

Let  $\varphi := (q_1 \vee q_2 \vee q_3) \wedge (q_2 \vee \neg q_3 \vee \neg q_4) \wedge (\neg q_2 \vee q_5)$ , which is already a CNF.

- ▶ Is  $\varphi$  satisfiable?
- ▶ Write down  $\varphi$  as a set of clauses, the initial **knowledge base**:  
 $\{q_1 \vee q_2 \vee q_3, q_2 \vee \neg q_3 \vee \neg q_4, \neg q_2 \vee q_5\}$ .
- ▶ Put down every clause in the **knowledge base** first, then apply resolution repeatedly:



# Resolution Rule: small example

Let  $\varphi := (q_1 \vee q_2 \vee q_3) \wedge (q_2 \vee \neg q_3 \vee \neg q_4) \wedge (\neg q_2 \vee q_5)$ , which is already a CNF.

- ▶ Is  $\varphi$  satisfiable?
- ▶ Write down  $\varphi$  as a set of clauses, the initial **knowledge base**:  
 $\{q_1 \vee q_2 \vee q_3, q_2 \vee \neg q_3 \vee \neg q_4, \neg q_2 \vee q_5\}$ .
- ▶ Put down every clause in the **knowledge base** first, then apply resolution repeatedly:

$$1 \quad q_1 \vee q_2 \vee q_3$$

$$2 \quad q_2 \vee \neg q_3 \vee \neg q_4$$

$$3 \quad \neg q_2 \vee q_5$$

# Resolution Rule: small example

Let  $\varphi := (q_1 \vee q_2 \vee q_3) \wedge (q_2 \vee \neg q_3 \vee \neg q_4) \wedge (\neg q_2 \vee q_5)$ , which is already a CNF.

► Is  $\varphi$  satisfiable?

► Write down  $\varphi$  as a set of clauses, the initial **knowledge base**:

$\{q_1 \vee q_2 \vee q_3, q_2 \vee \neg q_3 \vee \neg q_4, \neg q_2 \vee q_5\}$ .

► Put down every clause in the **knowledge base** first, then apply resolution repeatedly:

$$1 \quad q_1 \vee q_2 \vee q_3$$

$$2 \quad q_2 \vee \neg q_3 \vee \neg q_4$$

$$3 \quad \neg q_2 \vee q_5$$

$$4 \quad q_1 \vee q_3 \vee q_5$$

resolve 1, 3

$$5 \quad \neg q_3 \vee \neg q_4 \vee q_5$$

resolve 2, 3

$$6 \quad q_1 \vee \neg q_4 \vee q_5$$

resolve 4, 5

► there are no more resolvable pairs of clauses, stop and report  $\varphi$  is satisfiable.

## Resolution Rule: another small example

Let  $\psi := (p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_3) \wedge (\neg p_1 \vee \neg p_3)$ , already a CNF.

- ▶ Is  $\psi$  satisfiable?

# Resolution Rule: another small example

Let  $\psi := (p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_3) \wedge (\neg p_1 \vee \neg p_3)$ , already a CNF.

- ▶ Is  $\psi$  satisfiable?
- ▶ Write down  $\varphi$  as a set of clauses, the initial **knowledge base**:  
 $\{p_1 \vee p_2, p_1 \vee \neg p_2, \neg p_1 \vee p_3, \neg p_1 \vee \neg p_3\}$ .
- ▶ Put down every clause in the **knowledge base** first, then apply the resolution rule:

# Resolution Rule: another small example

Let  $\psi := (p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_3) \wedge (\neg p_1 \vee \neg p_3)$ , already a CNF.

- ▶ Is  $\psi$  satisfiable?
- ▶ Write down  $\varphi$  as a set of clauses, the initial **knowledge base**:  
 $\{p_1 \vee p_2, p_1 \vee \neg p_2, \neg p_1 \vee p_3, \neg p_1 \vee \neg p_3\}$ .
- ▶ Put down every clause in the **knowledge base** first, then apply the resolution rule:
  - 1  $p_1 \vee p_2$
  - 2  $p_1 \vee \neg p_2$
  - 3  $\neg p_1 \vee p_3$
  - 4  $\neg p_1 \vee \neg p_3$

# Resolution Rule: another small example

Let  $\psi := (p_1 \vee p_2) \wedge (p_1 \vee \neg p_2) \wedge (\neg p_1 \vee p_3) \wedge (\neg p_1 \vee \neg p_3)$ , already a CNF.

► Is  $\psi$  satisfiable?

► Write down  $\varphi$  as a set of clauses, the initial **knowledge base**:

$\{p_1 \vee p_2, p_1 \vee \neg p_2, \neg p_1 \vee p_3, \neg p_1 \vee \neg p_3\}$ .

► Put down every clause in the **knowledge base** first, then apply the resolution rule:

$$1 \quad p_1 \vee p_2$$

$$2 \quad p_1 \vee \neg p_2$$

$$3 \quad \neg p_1 \vee p_3$$

$$4 \quad \neg p_1 \vee \neg p_3$$

$$5 \quad p_1$$

resolve 1, 2

$$6 \quad p_3$$

resolve 3, 5

$$7 \quad \neg p_3$$

resolve 4, 5

$$8 \quad \perp$$

resolve 6, 7

► stop and report  $\psi$  is unsatisfiable.

# Resolution Rule: improvements in using it

After each application of the **resolution rule**:

- ▶ Simple improvement : **remove repeated literals** in the resolvent.
- ▶ Simple improvement : if the resolvent contains **complementary literals**, **discard the resolvent** instead of adding it to knowledge base.  
In this case, the resolvent is a tautology, satisfied by every truth-value assignment.
- ▶ Advanced improvements : see DPLL-based SAT solvers . . . (in a later handout).

# Resolution Rule: improvements in using it

After each application of the **resolution rule**:

- ▶ Simple improvement : **remove repeated literals** in the resolvent.
- ▶ Simple improvement : if the resolvent contains **complementary literals**, **discard the resolvent** instead of adding it to knowledge base.  
In this case, the resolvent is a tautology, satisfied by every truth-value assignment.
- ▶ Advanced improvements : see DPLL-based SAT solvers . . . (in a later handout).

Two important **heuristics** in choosing the next resolution step:

- ▶ Prefer a resolution involving a **unit clause** (a clause with one literal), because it produces a shorter clause as a resolvent.
- ▶ Use the so-called **set-of-support rule**, *i.e.*, prefer a resolution involving the **negated goal** or any **clause derived from the negated goal**, because we are trying to produce a contradiction that follows from the **negated goal** and these are the most “relevant” clauses.



# Resolution Rule: proof of soundness

## Theorem

Let  $\psi$  be a CNF,  $\psi = \{C_1, \dots, C_n\}$ , where every clause  $C_i$  is a finite disjunct of literals. Pose  $\Psi_0 = \psi$  and apply **resolution** repeatedly to  $\Psi_0$  to obtain the sequence of CNF's:

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \quad \dots \quad \Psi_p \quad \text{for some } p \geq 1.$$

**If  $\perp \in \Psi_p$  then  $\psi = \Psi_0$  is unsatisfiable.**

*(Leave aside whether the sequence is bound to terminate. Yes, it is bound to terminate!)*

# Resolution Rule: proof of soundness

## Theorem

Let  $\psi$  be a CNF,  $\psi = \{C_1, \dots, C_n\}$ , where every clause  $C_i$  is a finite disjunct of literals. Pose  $\Psi_0 = \psi$  and apply **resolution** repeatedly to  $\Psi_0$  to obtain the sequence of CNF's:

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \quad \dots \quad \Psi_p \quad \text{for some } p \geq 1.$$

**If  $\perp \in \Psi_p$  then  $\psi = \Psi_0$  is unsatisfiable.**

*(Leave aside whether the sequence is bound to terminate. Yes, it is bound to terminate!)*

## Proof.

Every time **resolution** is applied to some  $\Psi_i$ , we have:

$$\frac{(C \vee p) \quad (D \vee \neg p)}{(C \vee D)}$$

Resolvent  $(C \vee D)$  is satisfied by any truth-value assignment satisfying  $C$  or  $D$ .

Hence, if  $\Psi_i$  is satisfiable, then so is  $\Psi_{i+1} = \Psi_i \cup \{(C \vee D)\}$ .

Hence, **resolution preserves satisfiability** at every step from  $\Psi_0$  to  $\Psi_p$ .

Hence, if  $\Psi_p$  is unsatisfiable, then so is  $\Psi_0$ .

But  $\perp \in \Psi_p$  means  $\Psi_p$  is unsatisfiable, implying desired conclusion. □

# Resolution Rule: proof of refutation-completeness

## Theorem

Let  $\psi$  be a CNF,  $\psi = \{C_1, \dots, C_n\}$ , where every clause  $C_i$  is a finite disjunct of literals. Pose  $\Psi_0 = \psi$  and apply **resolution** repeatedly to  $\Psi_0$  to obtain the sequence of CNF's:

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \quad \dots \quad \Psi_p \quad \text{for some } p \geq 1.$$

**If  $\psi = \Psi_0$  is unsatisfiable, then  $\perp \in \Psi_p$ .**

*(Leave aside whether the sequence is bound to terminate. Yes, it is bound to terminate!)*

# Resolution Rule: proof of refutation-completeness

## Theorem

Let  $\psi$  be a CNF,  $\psi = \{C_1, \dots, C_n\}$ , where every clause  $C_i$  is a finite disjunct of literals. Pose  $\Psi_0 = \psi$  and apply **resolution** repeatedly to  $\Psi_0$  to obtain the sequence of CNF's:

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \quad \dots \quad \Psi_p \quad \text{for some } p \geq 1.$$

**If  $\psi = \Psi_0$  is unsatisfiable, then  $\perp \in \Psi_p$ .**

(Leave aside whether the sequence is bound to terminate. Yes, it is bound to terminate!)

## Proof.

The proof is by induction and the question is what to do the induction on. Define the *number of excess literals* in a clause  $C$ :

$$\text{excess}(C) := \begin{cases} 0 & \text{if } |C| = 0 \text{ or } 1, \\ |C| - 1 & \text{if } |C| \geq 2, \end{cases}$$

where  $|C|$  is the number of literals in  $C$ . For a CNF  $\psi = \{C_1, \dots, C_n\}$ , define  $\text{excess}(\psi) = \text{excess}(C_1) + \dots + \text{excess}(C_n)$ . An appropriate induction is on the measure  $\text{excess}(\psi)$ . All details omitted. □

## Exercise

Provide the details of the induction in [Refutation-Completeness Proof](#).

## Exercise

Search the Web for an (infinite) family of propositional WFF's on which the **resolution method** outperforms the **tableaux method** (as presented in Handout 09). Run the two methods on the smallest member of this set to show that the **tableaux method** takes more steps to terminate.

*Hint:* Consider the WFF  $\Psi$ , which is in CNF, in the last exercise in Handout 09.

## Exercise

Provide a detailed comparison of the **tableaux method** and the **resolution method**.