CS 511, Fall 2018, Handout 11

Resolution in Propositional Logic

Assaf Kfoury

18 September 2018

(last modified on: 24 September 2018)

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 φ is a logical contradiction. More generally, it tries to find reasons why a finite set Γ of wff's is not satisfiable.
- Like the tableaux method, it turns out that **resolution** is **refutation-complete**.
- As pointed out in Handout 10, **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 Γ a finite set of wff's and φ any wff (not restricted to $\varphi = \bot$).
- Later in the handout, we show that **resolution** can also be used to decide *satisfiability* of an arbitrary wff φ.
- 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 φ to be tested for non-satisfiability is in CNF.

- Before applying the method, we therefore need an efficient way of translating an arbitrary wff φ into another wff ψ in CNF.
- **Bad news**: Translating an arbitrary φ into an **equivalent** CNF ψ generally results in an exponential blow-up (see Handout 06).
- **Good news**: It is possible to efficiently translate an arbitrary wff φ into another wff ψ in CNF so that φ and ψ 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 φ is a propositional wff in CNF, we may write:

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

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

Efficient Transformation Into CNF

1. Already pointed out in Handout 06, 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 φ into the following wff ψ:

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

   produces a wff in CNF of size $\mathcal{O}(n)$ such that φ and ψ are equisatisfiable (though not equivalent), where $\{z_1, \ldots, z_n\}$ are fresh propositional variables.

   Exercise: Show that φ (in part 1 above) and ψ (in part 2) are equisatisfiable, *i.e.*, if there is truth-value assignment σ satisfying φ (resp. ψ), then there is a truth-value assignment σ' satisfying ψ (resp. φ).

3. An alternative translation of a wff φ into an equisatisfiable ψ 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 $i = 1, \ldots, n$. With these clauses, the initial wff φ implies $z_i \equiv x_i \wedge y_i$; in the new wff ψ we can view z_i as a name for "$x_i \wedge y_i$".

   Exercise: Look up "Tseitin transformation" on the Web for details, *e.g.* here .

4. A specific efficient algorithm, called CNF( ), to transform an arbitrary propositional wff φ into an equisatisfiable wff is presented next.

Efficient Transformation Into CNF

The definition of CNF( ) is by induction on wff's. Because it is inductive, it translates into a recursive algorithm, where Δ is a finite set of clauses:[1]

1. $\text{CNF}(p, \Delta) := \langle p, \Delta \rangle$
2. $\text{CNF}(\neg\varphi, \Delta) := \langle \neg\ell, \Delta' \rangle$ where $\text{CNF}(\varphi, \Delta) = \langle \ell, \Delta' \rangle$
3. $\text{CNF}(\varphi_1 \wedge \varphi_2, \Delta) := \langle p, \Delta' \rangle$ where
   $\text{CNF}(\varphi_1, \Delta) = \langle \ell_1, \Delta_1 \rangle$, $\text{CNF}(\varphi_2, \Delta_1) = \langle \ell_2, \Delta_2 \rangle$,
   p is a fresh atom (propositional variable),
   $\Delta' = \Delta_2 \cup \{\neg p \vee \ell_1,\ \neg p \vee \ell_2,\ \neg\ell_1 \vee \neg\ell_2 \vee p\}$
4. $\text{CNF}(\varphi_1 \vee \varphi_2, \Delta) := \langle p, \Delta' \rangle$ where
   $\text{CNF}(\varphi_1, \Delta) = \langle \ell_1, \Delta_1 \rangle$, $\text{CNF}(\varphi_2, \Delta_1) = \langle \ell_2, \Delta_2 \rangle$,
   p is a fresh atom (propositional variable),
   $\Delta' = \Delta_2 \cup \{\neg p \vee \ell_1 \vee \ell_2,\ \neg\ell_1 \vee p,\ \neg\ell_2 \vee p\}$

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

[1] Taken from Leonardo De Moura, "SMT Solvers: Theory and Implementation", Microsoft Research 2008.

Efficient Transformation Into CNF

Theorem

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

Proof.

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

Exercise

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

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

Exercise

Search the Web for improvements on the transformation $\text{CNF}()$.

Hint: How about introducing multi-arity $\wedge$ and multi-arity $\vee$?
But there are other possible improvements . . . .

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:

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

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

where all ℓ_i and ℓ'_j are literals, and $\boxed{\ell'_q = \neg \ell_p}$.

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:

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

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

where all ℓ_i and ℓ'_j are literals, and $\boxed{\ell'_q = \neg \ell_p}$.

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

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

New clause produced by **resolution** (below the line) is the **resolvent**. Note that ℓ_p and ℓ'_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 (ℓ_p, ℓ'_q) where $\ell'_q = \neg \ell_p$ in the special case when the two **antecedents** have each only one literal:

$$\frac{\ell_p \qquad \ell'_q}{\bot}$$

  In this case the **resolvent** is $\bot$ (**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 φ be a propositional wff. The following are equivalent statements:

1. φ *is formally derivable using* natural deduction.
2. φ *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 φ be a propositional wff. The following are equivalent statements:

1. φ *is formally derivable using* natural deduction*.*
2. φ *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 ψ be a propositional wff in CNF. The following are equivalent statements:

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

Soundness Proof | Refutation-Completeness Proof

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 $\boxed{\textbf{if } p, q \models p \vee q \textbf{ then } p, q \vdash_{\text{ND}} p \vee q}$.

     This follows from 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: $\boxed{\textbf{if } p, q \nvdash_{\text{ND}} p \vee q \textbf{ then } p, q \not\models p \vee q}$, which is again completeness for natural deduction stated contrapositively.

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

  4. Yes, this is possible. At the meta-level, $p, q \nvdash_{\text{ND}} p \vee q$ 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 ψ is formally derivable from a finite **knowledge base** KB $= \{\varphi_1, \ldots, \varphi_n\}$, *i.e.*, whether KB $\vdash_{\text{ND}}$ ψ using natural deduction (or some other formal proof system).

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

Resolution Rule: how to use it

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

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

- Negate ψ 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 ψ is formally derivable from $\varphi_1, \ldots, \varphi_n$, *i.e.*, $\varphi_1, \ldots, \varphi_n \vdash_{\text{res}} \psi$.

- An example is on the next slide.

Resolution Rule: small example

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

- 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** $\{P \to Q, Q \to R, \neg R\}$?

- 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 $\neg P \vee Q$

2 $\neg Q \vee R$

3 $\neg R$

4 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	$\neg P \vee Q$	
2	$\neg Q \vee R$	
3	$\neg R$	
4	P	
5	Q	resolve $1, 4$
6	R	resolve $2, 5$
7	$\perp$	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 φ 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 φ is **satisfiable**.

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

- Transform φ 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 φ is **unsatisfiable**.
- If there are no more resolvable pair of clauses (and $\perp$ is not produced), stop and report that the original φ 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 φ satisfiable?

[2] *Hint*: In contrast to the tableaux method, the resolution method does not give an immediate obvious way to define a satisfying truth-value assignment.

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 φ satisfiable?
- Write down φ 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:

[2] *Hint*: In contrast to the tableaux method, the resolution method does not give an immediate obvious way to define a satisfying truth-value assignment.

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 φ satisfiable?
- Write down φ 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 $q_1 \vee q_2 \vee q_3$

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

3 $\neg q_2 \vee q_5$

[2] *Hint*: In contrast to the tableaux method, the resolution method does not give an immediate obvious way to define a satisfying truth-value assignment.

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 φ satisfiable?
- Write down φ 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	$q_1 \vee q_2 \vee q_3$	
2	$q_2 \vee \neg q_3 \vee \neg q_4$	
3	$\neg q_2 \vee q_5$	
4	$q_1 \vee q_3 \vee q_5$	resolve $1, 3$
5	$\neg q_3 \vee \neg q_4 \vee q_5$	resolve $2, 3$
6	$q_1 \vee \neg q_4 \vee q_5$	resolve $4, 5$

- there are no more resolvable pairs of clauses, stop and report φ is satisfiable.

  Exercise: Extract a truth-value assignment for the initial φ from the resolution proof. Does your method for extracting a truth-value assignment work in general, *i.e.*, for any initial wff? [2]

[2] *Hint*: In contrast to the tableaux method, the resolution method does not give an immediate obvious way to define a satisfying truth-value assignment.

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 ψ 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 ψ satisfiable?
- Write down φ 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 ψ satisfiable?
- Write down φ 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 ψ satisfiable?
- Write down φ 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	$\neg p_1 \vee \neg p_3$	
5	p_1	resolve $1, 2$
6	p_3	resolve $3, 5$
7	$\neg p_3$	resolve $4, 5$
8	$\perp$	resolve $6, 7$

- stop and report ψ 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:

- Give preference to 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.*, give preference to 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 ψ be a CNF, $\psi = \{C_1, \ldots, C_n\}$, where every clause C_i is a finite disjunct of literals. Pose $\Psi_0 = \psi$ and apply **resolution** *repeatedly to Ψ_0 to obtain the sequence of CNF's:*

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \qquad \cdots \qquad \Psi_p \qquad \text{for some } p \geqslant 1.$$

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 ψ be a CNF, $\psi = \{C_1, \ldots, C_n\}$, where every clause C_i is a finite disjunct of literals. Pose $\Psi_0 = \psi$ and apply **resolution** *repeatedly to Ψ_0 to obtain the sequence of CNF's:*

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \qquad \cdots \qquad \Psi_p \qquad \text{for some } p \geqslant 1.$$

If $\bot \in \Psi_p$ 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 Ψ_i, we have:

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

Resolvent $(C \vee D)$ is satisfied by any truth-value assignment satisfying C or D.
Hence, if Ψ_i is satisfiable, then so is $\Psi_{i+1} = \Psi_i \cup \{(C \vee D)\}$.
Hence, **resolution preserves satisfiability** at every step from Ψ_0 to Ψ_p.
Hence, if Ψ_p is unsatisfiable, then so is Ψ_0.
But $\bot \in \Psi_p$ means Ψ_p is unsatisfiable, implying desired conclusion. $\square$

Back to Resolution: how to use it

Resolution Rule: proof of refutation-completeness

Theorem

Let ψ be a CNF, $\psi = \{C_1, \ldots, C_n\}$, where every clause C_i is a finite disjunct of literals. Pose $\Psi_0 = \psi$ and apply **resolution** *repeatedly to Ψ_0 to obtain the sequence of CNF's:*

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \qquad \cdots \qquad \Psi_p \qquad \textit{for some } p \geqslant 1.$$

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

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

Resolution Rule: proof of refutation-completeness

Theorem

Let ψ be a CNF, $\psi = \{C_1, \ldots, C_n\}$, where every clause C_i is a finite disjunct of literals. Pose $\Psi_0 = \psi$ and apply **resolution** *repeatedly to Ψ_0 to obtain the sequence of CNF's:*

$$\Psi_0 \quad \Psi_1 \quad \Psi_2 \qquad \cdots \qquad \Psi_p \qquad \text{for some } p \geqslant 1.$$

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:

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

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

Back to Resolution: how to use it

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 10). 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 Ψ, which is in CNF, in the last exercise in Handout 10.

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

(THIS PAGE INTENTIONALLY LEFT BLANK)