L07 Resolution
Chapter 8

=== ignoring the following for fall 2022 ===
Midterm
Propositional logic

- Syntax
- Semantics
- \Sigma \models \sigma

Proofs - manual
Tableaux proof
Resolution proof/refutation (30) [after def 8.16 not in exam]
New Concepts:

- Understand definitions
- Create examples
- Proofs.

Prep work for Monday class: Before Monday, study chapter until the concept "resolution tree proof". Also try to prove the following lemma (8.9 in 2nd edition of the book): If the formula (i.e., set of clauses) S = {C1, C2} is satisfiable and C is a resolvent of C1 and C2, then C is satisfiable. Using working backwards. It also needs the skill of "prove by cases."
=== end of ignoring for 2022 ===

1. Motivation

Resolution is a proof method used in modern Logic Programming and in theorem prover. It is simple and efficient.

In scratch below, we show intuition about resolution, resolution based proof, resolution refutation, and the form of propositions we discuss: propositions of conjunctive normal form (but with
a new syntax)

Resolution

$\{A\to B,\ A\} \models B$

$\left.\begin{matrix}A\to B\\ A\end{matrix}\right\} B$

$A\to B;\ A;\ B$ — resol^n proof of B

$\{A\to B,\ B\to C,\ (C\wedge D)\to e,\ A\} \cup \{D\}$

$A;\ A\to B;\ B;\ B\to C;\ C;\ D;\ (C\wedge D);\ (C\wedge D)\to e;\ e$

$\{\{A\},\ \{\neg A\}\}$

$A,\ \neg A,\ \square$ resol^n

$\underline{A\to B}\quad \underline{A} \Rightarrow B$

$\{\neg A\vee B\}\quad \{A\} \Rightarrow B$

$\{\neg A, B, C\} \Rightarrow \{B, C, D, e\}$

$\{A, D, e\}$

$\{B, C\}$

$\{\neg A, B\}$ | $A\to B$ | Case A $\quad A;\ A\to B;\ B \left.\right\}$ $B\vee C$

$\{A, C\}$ | $A\vee C$ | Case C

$\left.\begin{matrix}A\\ C\\ A\to B\end{matrix}\right\rangle B \quad \boxed{B\wedge C}$

$\{\{A\},\ \{\neg A, B\},\ \dots\ \}$

meaning: ~~A A {¬A, B}~~
$A \wedge (\neg A\vee B) \wedge \cdots$

$\{A\}\ \{\neg A\}$

$\{\}$

you're either at home or not

$\vee c \vee d \vee e$

$\vee ?$

$\{\underline{A}\to C,\ B\to C,\ \underline{A\vee B}\}\quad A\to \overset{C}{B} \quad \overset{C}{B}$ true

case A is true $\left(\begin{matrix}B \text{ is true}\\ B \text{ is false}\end{matrix}\right)$

case B is true $\left(\begin{matrix}A \text{ is true}\\ A \text{ is false}\end{matrix}\right)$ $B\to C$ $\quad$ C is true.

C is true

Read definitions of the new syntax of “proposition,” resolution proof/refutation, soundness and completeness results about resolution proof.

2. Go through the definitions and theorems:

Definition 8.1 (i) A literal l is a propositional letter p or its negation $\neg p$. If l is p or $\neg p$, we write $\bar{l}$ for $\neg p$ or p, respectively. The propositional letters are also called positive literals and their negations negative literals.

(ii) A clause C is a finite set of literals (which you should think of as the disjunction of its elements). As we think of C as being true iff one of its elements is true, the empty clause $\Box$ is always false - it has no true element.

(iii) A formula S is a (not necessarily finite) set of clauses (which you should think of as the conjunction of its elements). As we think of a formula S as being true if everyone of its elements is true, the empty formula $\emptyset$ is always true - it has no false element.

(iv) An assignment $\mathcal{A}$ is a consistent set of literals, i.e., one not containing both p and $\neg p$ for any propositional letter p. (This, of course, is just the (partial) truth assignment in which those $p \in \mathcal{A}$ are assigned T and those q with $\bar{q} \in \mathcal{A}$ are assigned F.) A complete assignment is one containing p or $\neg p$ for every propositional letter p. It corresponds to what we called a truth assignment in Definition 3.1.

(v) $\mathcal{A}$ satisfies S, $\mathcal{A} \models S$, iff $\forall C \in S (C \cap \mathcal{A} \neq \emptyset)$, i.e., the valuation induced by $\mathcal{A}$ makes every clause in S true.

(vi) A formula S is (un)satisfiable if there is an (no) assignment $\mathcal{A}$ that satisfies it.

Definition 8.2 (i) $p, q, r, \neg p, \bar{q}(= \neg q), \bar{r}$ and $\neg\bar{q}(= q)$ are literals.

(ii) $\{p, r\}$, $\{\neg q\}$ and $\{q, \neg r\}$ are clauses.

(iii) $S = \{\{p, r\}, \{q, \neg r\}, \{\neg q\}, \{\neg p, t\}, \{s, \neg t\}\}$ is a formula that, in our original notation system, would be written as $((p \vee r) \wedge (q \vee \neg r) \wedge (\neg q) \wedge (\neg p \vee t) \wedge (s \vee \neg t))$.

(iv) If $\mathcal{A}$ is given by $\{p, q, r, s, t\}$, i.e., the (partial) assignment such that $\mathcal{A}(p) = T = \mathcal{A}(q) = \mathcal{A}(r) = \mathcal{A}(s) = \mathcal{A}(t)$, then $\mathcal{A}$ is an assignment not satisfying the formula S in (iii). S is, however, satisfiable.

Definition 8.3 [Resolution] From clauses C_1 and C_2 of the form $\{l\} \sqcup C_1'$ and $\{\bar{l}\} \sqcup C_2'$, infer $C = C_1' \cup C_2'$ which is called a resolvent of C_1 and C_2. (Here l is any literal and $\sqcup$ means that we are taking a union of disjoint sets.) We may also call C_1 and C_2 the parent and C their child and say that we resolved on (the literal) l.

Definition 8.4 A (resolution) deduction or proof of C from a given formula S is a finite sequence $C_1, C_2, ..., C_n = C$ of clauses such that each C_i is either a member of S or a resolvent of clauses C_j, C_k for $j, k < i$. If there is such a deduction, we say that C is (resolution) provable from S and write $S \vdash_{\mathcal{R}} C$. A deduction of $\square$ from S is called a (resolution) refutation of S. If there is such a deduction we say that S is (resolution) refutable and write $S \vdash_{\mathcal{R}} \square$.

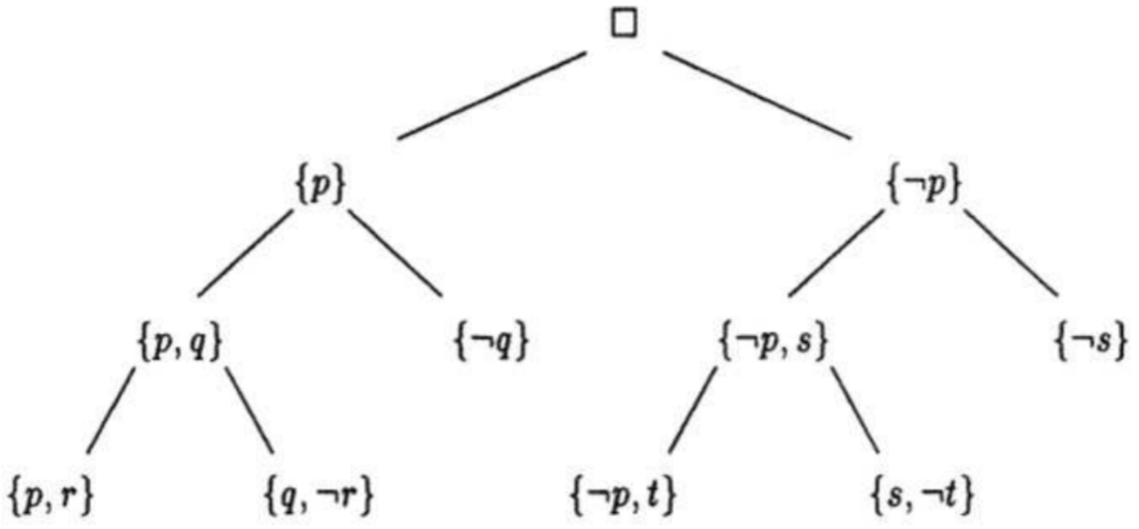
Definition 8.5 A resolution tree proof of C from S is a labeled binary tree T with the following properties:

(i) The root of T is labeled C.

(ii) The leaves of T are labeled with elements of S.

(iii) If any nonleaf node σ is labeled with C_2 and its immediate successors σ_0, σ_1 are labeled with C_0, C_1, respectively, then C_2 is a resolvent of C_0 and C_1.

A resolution tree refutation of the formula $S = \{\{p, r\}, \{q, \neg r\}, \{\neg q\}, \{\neg p, t\}, \{\neg s\}, \{s, \neg t\}\}$, i.e., a resolution tree proof of $\Box$ from S:



Lemma 8.9 If the formula (i.e., set of clauses) $S = \{C_1, C_2\}$ is satisfiable and C is a resolvent of C_1 and C_2, then C is satisfiable. Indeed, any assignment $\mathcal{A}$ satisfying S satisfies C.

Theorem 8.10 [Soundness of resolution] If there is a resolution refutation of S, then S is unsatisfiable.

Lemma 8.11 For any formula T and any literal l, let $T(l) = \{C \in \mathcal{R}(T) \mid l, \bar{l} \notin C\}$. If T is unsatisfiable, then so is $T(l)$.

Theorem 8.12 [Completeness of resolution] If S is unsatisfiable, then there is a resolution refutation of S.

reading and understanding them using techniques we studied earlier

A
-A

Example (satisfies)

{{A}, {A, B} }   // A /\ (A \/B)
S={-A, B}   // -A = T, B=T

S={A, B}   // A = T, B=T

empty clause {} not satisfiable // always F
empty formula{} always satisfiable //

For all C in S …   ⇒ for every C if C in S then …

Example (resolvent)
C1 = {A} C2 ={-A} resolvent of them is {}
C1={A, B} C2 = {-A, C} : resolvent is {B, C} (C1'={B} C2'={C})

C1 = {A, B} C2 = {-A, -B}:

- a resolvent (on A) - {B, -B} (C1 = {A} U {B}, C2 = {-A} U {-B}, C1' = {B}, C2'={-B})
- a resolvent (on B) - {A, -A}

3. Proof of lemma 8.12 (8.9 in 2nd edition)

Lemma 8.12 (Lemma 8.9 in 2nd edition) If the formula (i.e., set of clauses) S = {C1, C2} is satisfiable and C is a resolvent of C1 and C2, then C is satisfiable.

We use working backwards (application of definitions to an instance etc.), **prove by cases**, and show how to prove **existence** of sth.

Proof

(b1) If the formula S = {C1, C2} is satisfiable, C is satisfiable.
QED

4. Some notations useful

Given a formula S, literal l, S^l denote the clauses of the formula that are simplified and whose “truth values” are unknown **when l is taken as “true.”** Also remember a goal in **resolution refutation** is to get the unsatisfied clause - empty clause, hence we can ignore the clauses that are satisfied/satisfiable.

S={ {-A, B}, {A, -C, D}, {D, F} }

$S^{\neg A}$: $\neg A$ is true, so, the first clause is removed because it is “true”, the 2nd is simplified: {-C, D} by removing A (because it is “false”)

What is S^A?

Key result: Lemma 8.19: S is satisfiable iff S^l or $S^{\bar{l}}$ is satisfiable. (recall the satisfiability result about resolvant and its two parent clauses)

This is used to prove completeness results. This is also used in the modern STA solver algorithms.

Study the definitions and main results.

Exercise: prove Lemma 8.19

Definition 8.13 If S is a formula and l a literal, we let

$$S^l = \{C - \{\bar{l}\} \mid C \in S \wedge l \notin C\}$$

Let $S = \{\{p\}, \{\neg q\}, \{\neg p, \neg q\}\}$. The analysis in which we eliminate first p and then q can be represented below:

$$\begin{tikzcd}
 & & S \arrow[dl, no head] \arrow[dr, no head] & \\
 & S^p = \{\{\neg q\}\} \arrow[dl, no head] \arrow[dr, no head] & & \{\square, \{\neg q\}\} = S^{\bar{p}} \\
S^{pq} = \{\square\} & & \emptyset = S^{p\bar{q}} &
\end{tikzcd}$$

This notion helps prove the completeness result

Theorem 8.17 [Completeness] If S is unsatisfiable, then there is a resolution refutation of S (equivalently, $\square \in \mathcal{R}(S)$).

Lemma 8.14 S is satisfiable if and only if either S^l or $S^{\bar{l}}$ is satisfiable.

5. Proof by induction

(intuition why the theorem below is true.)

Definition. A clause C is **satisfied** by an assignment A if the intersection of A and C is not empty. A clause C is **satisfiable** if there exists an assignment A such that C is satisfied by A.

Question a: What are the new concepts and their parameters that are defined in the definitions above? What are the concepts and parameters that are used to define the new concepts?

Question b: Prove **theorem**: if there is a resolution refutation of a formula S, the S is unsatisfiable. You may use the following claim 1 directly.

Claim 1: If the formula S = {C1, C2} is satisfiable and C is a resolvent of C1 and C2, then C is satisfiable.

You must follow the following steps to prove this theorem.

Proof

1. Let C_1, ..., C_n be the resolution refutation of S. We will prove the following statement by induction on the length n of the resolution refutation:

Claim 2: For any i \in 1..n, if an assignment satisfies S, then the assignment satisfies C_i. (i.e., any assignment satisfying S satisfies C_i).

Base case (i = 1): prove: if an assignment satisfies S, then the assignment satisfies C_1.

[... put your proof here ...]

Inductive hypothesis: Consider a number k (> 1 but less than n). We assume for any i \in 1.. k-1, if an assignment satisfies S, then the assignment satisfies C_1. [This statement is true. You can use it directly]

[nothing should go here]

inductive proof: now we prove that for k, if an assignment satisfies S, then the assignment satisfies C_k.

case 1: C_k is from S:

[... fill the proof of: if an assignment satisfies S, then the assignment satisfies C_k.]

case 2: C_k is not from S:

[... fill the proof of: if an assignment satisfies S, then the assignment satisfies C_k.] [Hint: you can use the inductive hypothesis above in your proof here]

by case 1 and 2, we proved: for k, if an assignment satisfies S, then the assignment satisfies C_k.

Therefore (from the basic case and inductive proof), for any i \in 1..n, if an assignment satisfies S, then the assignment satisfies C_i.

2. Now we prove the theorem by contradiction. [You may use Claim 2, and claim 1 directly. Hint: what is C_n? recall the definition.]

[... Fill your proof by contradiction here....]

QED

Working to prove case 1:

(1) Assume an assignment A **satisfies** S

  (2) **for any** clause C of S, A \cap C != {}   by (1) and definition of “satisfies”

  (3) C_1 \in S // why? Because C_1, …, C_n is a resolution refutation of S.

  (4) A \cap C_1 != { }   by (2) and (3)

(5) A **satisfies** C_1

(6) **if** … **then** …

Typo in inductive hypothesis: C_1 should be C_i

Exercise: another proof by induction to practice