Warm-Up Problem

Is the following true or false?

$$\emptyset \vDash (p \to (q \to p))$$

Propositional Logic: Resolution

Carmen Bruni

Lecture 6

Based on work by J. Buss, A. Gao, L. Kari, A. Lubiw, B. Bonakdarpour, D. Maftuleac, C. Roberts, R. Trefler, and P. Van Beek

Conjunctive Normal Form

Conjunctive normal form (CNF):

- A *literal* is a (propositional) variable or the negation of a variable.
- A *clause* is a disjunction of literals. (literals connected by $\vee$)
- A formula is in *conjunctive normal form* if it is a conjunction of clauses.

In other words, a formula is in CNF if and only if

- its only connectives are $\neg$, $\vee$ and/or $\wedge$,
- $\neg$ applies only to variables, and
- $\vee$ applies only to subformulas with no occurrence of $\wedge$.

For CNF Formulas, it is usually easier to read by dropping brackets and only using them around clauses and the whole formula (possible by associativity since we're only dealing with $\wedge$ and $\vee$ binary connectives and only care up to logical equivalence):

CNF examples

Some CNF formulas:

- $\big((p \vee (\neg q)) \wedge \big(r \wedge ((\neg r) \vee (p \vee q))\big)\big)$
- $((\neg r) \vee (p \vee q))$ (Only one clause.)
- $((\neg r) \wedge (p \wedge q))$ (Three singleton clauses.)

With simpler bracketing:

- $\big((p \vee (\neg q)) \wedge r \wedge ((\neg r) \vee p \vee q)\big)$
- $((\neg r) \vee p \vee q)$ (Only one clause.)
- $((\neg r) \wedge p \wedge q)$ (Three singleton clauses.)

CNF examples

Some formulas not in CNF:

- $\big(\neg\big((\neg p) \wedge q\big)\big)$ ($\neg$ applied to $\wedge$.)
- $\big(p \vee (r \wedge q)\big)$ ($\vee$ applied to $\wedge$.)
- $\big(\neg(\neg q)\big)$ ($\neg$ applied to $\neg$.)
- $(p \to q)$ (Uses $\to$.)

Converting to CNF

1. Eliminate implication and equivalence.
   Replace $(\alpha \to \beta)$ by $((\neg\alpha) \vee \beta)$
   Replace $(\alpha \leftrightarrow \beta)$ by $((\neg\alpha) \vee \beta) \wedge (\alpha \vee (\neg\beta))$.
   (*Now only* $\wedge$, $\vee$ *and* $\neg$ *appear as connectives.*)
2. Apply De Morgan's and double-negation laws as often as possible.
   Replace $(\neg(\alpha \vee \beta))$ by $((\neg\alpha) \wedge (\neg\beta))$.
   Replace $(\neg(\alpha \wedge \beta))$ by $((\neg\alpha) \vee (\neg\beta))$.
   Replace $(\neg(\neg\alpha))$ by α.
   (*Now negation only occurs in literals.*)
3. Transform into a conjunction of clauses using distributivity.
   Replace $(\alpha \vee (\beta \wedge \gamma))$ by $((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$.
   (*One could stop here, but....*)
4. Simplify using idempotence, contradiction, excluded middle and Simplification I & II.

Practice

Show that the CNF of

$$\big((p \leftrightarrow q) \rightarrow \big((\neg p) \wedge r\big)\big)$$

is

$$\big((p \vee q \vee r) \wedge ((\neg q) \vee (\neg p))\big)$$

Working with CNF Formulas

In a clause, there is no need to have any variable twice. [Why?]
Also, the order of the literals in the clause does not matter.

Thus we can think of a clause as simply a set of literals.

Similarly, in a CNF formula, no clause [set] need appear more than once, and the order of clauses does not matter.

Thus we can think of a CNF formula as simply a set of clauses.

For example, the formula $\Big((p \vee q) \wedge \big((q \vee (\neg r)) \wedge s\big)\Big)$
can be described by the set of clauses $\{p, q\}$, $\{q, (\neg r)\}$ and $\{s\}$.

We shall use these observations in our next topic, "Resolution".

What Is a "Proof"?

A *proof* is a formal demonstration that a statement is true.

- A proof is generally syntactic, rather than semantic.
- Generically, a proof consists of a sequence of formulas.
- *Inference rules* justify subsequent lines of the proof which are *inferred* by the previous lines.

Inference Rules

In general, an inference rule is written as

$$\frac{\alpha_1 \quad \alpha_2 \quad \ldots \quad \alpha_n}{\beta} \ .$$

That is, given α_1, α_2, ..., α_n, we can infer β

Examples of possible rules (remember rules are syntactic NOT semantic!):

$\frac{\alpha \quad \beta}{(\alpha \wedge \beta)}$ A kind of definition of $\wedge$.

$\frac{(\alpha \wedge \beta)}{(\alpha \vee \beta)}$ Rules need not be equivalences.

Notation

We notate "there is a proof with premises Σ and conclusion φ" by

$$\Sigma \vdash \varphi$$

In this course, we are doing two types of proof systems: Resolution and Natural Deduction. These will be denoted by

$$\Sigma \vdash_{\mathsf{Res}} \varphi \qquad \text{and} \qquad \Sigma \vdash_{\mathsf{ND}} \varphi$$

Overview of Resolution

Resolution is one of the most widely used systems for computer-aided proofs.

It has two distinctive features.

- It applies only to formulas in Conjunctive Normal Form.

  Thus we do some preliminary work before starting an actual proof.

- It is used to prove contradictions. That is, a proof aims to conclude a special "contradiction formula" $\perp$.

  For this reason, Resolution is sometimes called a "refutation" system.

A Note and Warning

In CNF, we treat $\bot$ as a clause containing no literals, i.e., the set $\emptyset$. Since it contains no true literal, it is false.

Be forewarned however that this notation only makes sense when dealing with CNF formulas. In some sense, we really should have a symbol like $\bot_{\text{Res}}$ but we won't do this. Just keep this slide in mind if you start replacing $\emptyset$ with $\bot$ when not dealing with CNF formulas.

The "Resolution" System and Rule

The *Resolution* inference rule:

$$\frac{(\alpha \vee p) \quad ((\neg p) \vee \beta)}{(\alpha \vee \beta)}$$

for any Propositional variable p and formulas α and β. Special Cases:

Unit resolution:

$$\frac{(\alpha \vee p) \quad (\neg p)}{\alpha}$$

Contradiction:

$$\frac{p \quad (\neg p)}{\bot}$$

Resolution is a refutation system; a proof is finished when one derives a contradiction $\bot$. Remember: proof systems are syntactic NOT semantic!

In this case, the original premises are refuted.

Example of Using Resolution

To prove: $\{p,\ q,\ ((\neg p) \vee (\neg q))\} \vdash_{Res} \bot$.

1.	p	premise
2.	q	premise
3.	$((\neg p) \vee (\neg q))$	premise

Consider lines 1 and 3....

We have the formulas (1) p and (3) $((\neg p) \vee (\neg q))$.
Apply unit resolution, yielding the formula $(\neg q)$.

Example of Using Resolution

To prove: $\{p,\ q,\ ((\neg p) \vee (\neg q))\} \vdash_{Res} \bot$.

1.	p	premise
2.	q	premise
3.	$((\neg p) \vee (\neg q))$	premise
4.	$(\neg q)$	1, 3

Consider lines 1 and 3....

We have the formulas (1) p and (3) $((\neg p) \vee (\neg q))$.
Apply unit resolution, yielding the formula $(\neg q)$.

We have the formulas (2) q and (4) $(\neg q)$.
Apply the contradiction rule, yielding $\bot$.

Example of Using Resolution

To prove: $\{p,\ q,\ ((\neg p) \vee (\neg q))\} \vdash_{Res} \bot$.

1.	p	premise
2.	q	premise
3.	$((\neg p) \vee (\neg q))$	premise
4.	$(\neg q)$	1, 3
5.	$\bot$	2, 4

Consider lines 1 and 3....

We have the formulas (1) p and (3) $((\neg p) \vee (\neg q))$.
Apply unit resolution, yielding the formula $(\neg q)$.

We have the formulas (2) q and (4) $(\neg q)$.
Apply the contradiction rule, yielding $\bot$.

Done!

Resolution Using Set Notation

In a Resolution proof, we consider only CNF formulas.

Each step of the proof produces one clause from two previous clauses.

Implicitly, all clauses together can be considered a single CNF formula (join the clauses by $\wedge$s).

Often, one writes clauses as sets. That is, the formula

$$\Big((p \vee q) \wedge \big(((\neg p) \vee r) \wedge \big((\neg q) \vee ((\neg r) \vee p)\big)\big)\Big)$$

has the set of clauses

$$\Big\{(p \vee q), \big((\neg p) \vee r\big), \big((\neg q) \vee \big((\neg r) \vee p\big)\big)\Big\}$$

which become the three sets of literals

$$\{p, q\}, \quad \{(\neg p), r\}, \quad \text{and} \quad \{(\neg q), (\neg r), p\} \ .$$

Set notation works due to associativity, commutativity and idempotency.

Set-Notation Example

Expressed using the set notation, the previous example becomes

1.	$\{p\}$	premise
2.	$\{q\}$	premise
3.	$\{(\neg p), (\neg q)\}$	premise
4.	$\{(\neg q)\}$	1, 3
5.	$\bot$	2, 4

(Note that $\{\}$ and $\bot$ are the same thing.)

Preparing for a Resolution Proof

To use Resolution successfully, we must account for two features.

- Resolution only yields contradictions.

→ Rather than proving $\Sigma \vdash_{Res} \varphi$, prove $\Sigma \cup \{\neg\varphi\} \vdash_{Res} \bot$ instead.

- The resolution rule only applies to disjunctions ($\vee$).

→ Before applying resolution, first
  - convert each formula to CNF, and
  - separate all formulas at the $\wedge$s.

Resolution Example II

To prove: $\{p,\ q\} \vdash_{Res} (p \wedge q)$.

Preliminary step 1: move the conclusion to the premises, negating it. This gives premises

$$\{p,\ q,\ \neg(p \wedge q)\}$$

Resolution Example II

To prove: $\{p,\ q\} \vdash_{Res} (p \wedge q)$.

Preliminary step 1: move the conclusion to the premises, negating it. This gives premises

$$\{p,\ q,\ \neg(p \wedge q)\}$$

Preliminary step 2: Convert the premises to CNF.

$$\{p,\ q,\ ((\neg p) \vee (\neg q))\}$$

Resolution Example II

To prove: $\{p,\ q\} \vdash_{Res} (p \wedge q)$.

Preliminary step 1: move the conclusion to the premises, negating it. This gives premises

$$\{p,\ q,\ \neg(p \wedge q)\}$$

Preliminary step 2: Convert the premises to CNF.

$$\{p,\ q,\ ((\neg p) \vee (\neg q))\}$$

Step 3: Write down the list of premises

1.	p	premise
2.	q	premise
3.	$((\neg p) \vee (\neg q))$	premise (from negated goal)

and then do the actual resolution, as above.

The Resolution Proof Procedure

To prove φ from Σ, via a Resolution refutation:

1. Convert each formula in Σ to CNF.
2. Convert $(\neg\varphi)$ to CNF.
3. Split the CNF formulas at the $\wedge$s, yielding a set of clauses.
4. From the resulting set of clauses, keep applying the resolution inference rule until either:
   - The empty clause $\bot$ results.
     In this case, φ is proven from Σ.
   - The rule can no longer be applied to give a new formula.
     In this case, φ cannot be proven from Σ.

Example: Resolution

To show: $\{(p \to q), (q \to r)\} \vdash_{\mathsf{Res}} (p \to r)$.

Example: Resolution

To show: $\{(p \to q), (q \to r)\} \vdash_{\mathsf{Res}} (p \to r)$.

Convert each premise formula to CNF.

We get $((\neg p) \vee q)$ and $((\neg q) \vee r)$.

Convert the **negation** of the goal formula to CNF:

Replacing the $\to$ yields $\neg((\neg p) \vee r)$; then
De Morgan yields $(p \wedge (\neg r))$.

Splitting the $\wedge$ yields four clauses: $((\neg p) \vee q)$, $((\neg q) \vee r)$, p and $(\neg r)$.

Example, cont'd

Now we can make inferences, starting from our premises.

1.	$((\neg p) \vee q)$	premise
2.	$((\neg q) \vee r)$	premise
3.	p	premise (from negated conclusion)
4.	$(\neg r)$	premise (from negated conclusion)

Example, cont'd

Now we can make inferences, starting from our premises.

1.	$((\neg p) \vee q)$	premise
2.	$((\neg q) \vee r)$	premise
3.	p	premise (from negated conclusion)
4.	$(\neg r)$	premise (from negated conclusion)
5.	q	1, 3 (variable p)

Example, cont'd

Now we can make inferences, starting from our premises.

1.	$((\neg p) \vee q)$	premise
2.	$((\neg q) \vee r)$	premise
3.	p	premise (from negated conclusion)
4.	$(\neg r)$	premise (from negated conclusion)
5.	q	1, 3 (variable p)
6.	r	2, 5 (variable q)

Example, cont'd

Now we can make inferences, starting from our premises.

1.	$((\neg p) \vee q)$	premise
2.	$((\neg q) \vee r)$	premise
3.	p	premise (from negated conclusion)
4.	$(\neg r)$	premise (from negated conclusion)
5.	q	1, 3 (variable p)
6.	r	2, 5 (variable q)
7.	$\perp$	4, 6 (variable r)

Refutation complete!

Questions About Proofs

Given a sequence of formulas, is it a proof?

Determined by examining the sequence, formula by formula.
If the sequence always follows the rules, it is a proof; if it ever does not, then it is not a proof.

Why might we want a proof?

For some (carefully constructed) proof systems, the existence of a proof implies that the conclusion is a logical consequence of the premises.
Such a system is called *sound*. If S is a sound proof system,

$$\Sigma \vdash_S \varphi \text{ implies } \Sigma \vDash \varphi \ .$$

Resolution Is Sound

For resolution to be meaningful, we need the following.

Theorem. Suppose that $\{\alpha_1, \ldots, \alpha_n\} \vdash_{Res} \bot$; that is, there is a resolution refutation with premises $\alpha_1, \ldots, \alpha_n$ and conclusion $\bot$. Then the set $\{\alpha_1, \ldots, \alpha_n\}$ is unsatisfiable (contradictory).

That is, if $\Sigma \cup \{(\neg\varphi)\} \vdash_{Res} \bot$, then $\Sigma \cup \{(\neg\varphi)\}$ is a contradiction. Therefore, $\Sigma \vDash \varphi$.

In other words, the Resolution proof system is sound.
(If we prove something, it is true.)

We prove the theorem by induction on the length of the refutation.

Soundness: The central argument

Claim: Suppose that a set $\Gamma = \{\beta_1, \ldots, \beta_k\}$ is satisfiable. Let β_{k+1} be a formula obtained from Γ by one use of the resolution inference rule. Then the set $\Gamma \cup \{\beta_{k+1}\}$ is satisfiable.

Proof: Let valuation v satisfy Γ; that is, $\beta_i^v = \mathtt{T}$ for each i.

Let β_{k+1} be $(\gamma_1 \vee \gamma_2)$, obtained by resolving $\beta_i = (p \vee \gamma_1)$ and $\beta_j = ((\neg p) \vee \gamma_2)$.

Case I: $v(p) = \mathtt{F}$. Since $\beta_i^v = \mathtt{T}$, we must have $\gamma_1^v = \mathtt{T}$. Thus $\beta_{k+1}^v = \mathtt{T}$.

Case II: $v(p) = \mathtt{T}$. Since $\beta_j^v = \mathtt{T}$, we must have $\gamma_2^v = \mathtt{T}$. Thus $\beta_{k+1}^v = \mathtt{T}$.

In either of the two possible cases, we have $\beta_{k+1}^v = \mathtt{T}$, as claimed.

The Claim Implies the Theorem

Using induction on n, the previous claim implies

> *Claim II*: Suppose that the set $\Gamma = \{\beta_1, \ldots, \beta_k\}$ is satisfiable. Let α be a formula obtained from Γ by n uses of the resolution inference rule. Then the set $\Gamma \cup \{\alpha\}$ is satisfiable.

(The previous claim is the inductive step of this one.)

Therefore, if a set of premises leads to $\perp$ after any number n of resolution steps, the set must be unsatisfiable—since any set containing $\perp$ is unsatisfiable.

Thus Resolution is a sound refutation system, as required.

Can Resolution Fail?

In some cases, there may be no way to obtain $\perp$, using any number of resolution steps. What then?

Definition. A proof system S is *complete* if every entailment has a proof; that is, if

$$\Sigma \models \alpha \quad \text{implies} \quad \Sigma \vdash_S \alpha \ .$$

Theorem. Resolution is a complete refutation system for CNF formulas. That is, if there is no proof of $\perp$ from a finite set Σ of premises in CNF, then Σ is satisfiable.

Resolution Is Complete (Outline)

Claim. Suppose that a resolution proof "reaches a dead end"—that is, no new clause can be obtained, and yet $\perp$ has not been derived. Then the entire set of formulas (including the premises!) is satisfiable.

Proof (outline): We use induction again. However, it is not an induction on the length of the proof, nor on the number of formulas. Instead, we use induction on the number of variables present in the formulas.

Basis: there are no variables at all—that is, the set of clauses is the empty set. The empty set of clauses is satisfiable, by definition.

Completeness Proof, part II

Inductive hypothesis: The claim holds for sets having at most k variables.

Consider a set of clauses using $k+1$ variables, from which no additional clause can be derived via the resolution rule. Suppose that it does not contain $\bot$. Select any one variable, say p, and separate the clauses into three sets:

S_p: the clauses that contain the literal p.
$S_{(\neg p)}$: the clauses that contain the literal $(\neg p)$.
R: the remaining clauses, which do not contain p at all.

The "remainder" set R has at most k variables.
Thus the hypothesis applies: the set R has a satisfying valuation v.

Completeness Proof, part III

We have a valuation v, on the variables other than p, that satisfies set R.

Case I: Every clause in S_p, of the form $(p \vee \alpha)$, has $\alpha^v = \mathtt{T}$.

In this case, the set S_p is already satisfied. Define $v(p) = \mathtt{F}$, which additionally makes every clause in $S_{(\neg p)}$ true.

Case II: S_p has some clause $(p \vee \alpha)$ with $\alpha^v = \mathtt{F}$.

In this case, set $v(p) = \mathtt{T}$; this satisfies every formula in S_p.

What about a clause $\big((\neg p) \vee \beta\big)$ in $S_{(\neg p)}$?
Consider the formula $(\alpha \vee \beta)$, obtained by resolution from $(p \vee \alpha)$ and $\big((\neg p) \vee \beta\big)$. It must lie in R; thus $\beta^v = \mathtt{T}$. Thus also $\big((\neg p) \vee \beta\big)^v = \mathtt{T}$, as required.

Thus the full set of clauses $S_p \cup S_{(\neg p)} \cup R$ is satisfiable.

By induction, every set that cannot produce $\perp$ is satisfiable.

Resolution Provides an Algorithm

The resolution method yields an algorithm to determine whether a given formula, or set of formulas, is satisfiable or contradictory.

- Convert to CNF. (A well-specified series of steps.)
- Form resolvents, until either $\perp$ is derived, or no more derivations are possible. (Why must this eventually stop?)
- If $\perp$ is derived, the original formula/set is contradictory. Otherwise, the preceding proof describes how to find a satisfying valuation.

The Algorithm Can Be Very Slow

The algorithm can be "souped up" in many ways.

- Choosing a good order of doing resolution steps. (It matters!)
- Sophisticated data structures, to handle large numbers of clauses.
- Additional techniques: setting variables, "learning", etc.

However, it still has limitations.

Theorem (Haken, 1985): There is a number $c > 1$ such that

> For every n, there is an unsatisfiable formula on n variables (and about $n^{1.5}$ total literals) whose smallest resolution refutation contains more than c^n steps.

Resolution is an exponential-time algorithm!
(And you thought quadratic was bad....)

Resolution in Practice: Satisfiability (SAT) solvers

Determining the satisfiability of a set of propositional formulas is a fundamental problem in computer science.

Examples:

- software and hardware verification
- automatic generation of test patterns
- planning
- scheduling

...many problems of practical importance can be formulated as determining the satisfiability of a set of formulas.

Resolution in practice: "SAT Solvers"

Modern SAT solvers can often solve hard real-world instances with over a million propositional variables and several million clauses.

Annual SAT competitions:

http://www.satcompetition.org/

Many are open source systems.

Currently, the best SAT solvers use "backtracking search" to find resolvable clauses.

Satisfiability in Theory

If a formula is satisfiable, then there is a short demonstration of that: simply give the valuation. Anyone can easily check that it is correct.

The class of problems with this property is known as NP.

The class of problems for which one can **find** a solution efficiently is known as P.

(For a precise definition, we need to define "efficiently." We won't, here.)

A Fundamental Question: Is $P = NP$?

A partial answer: If SAT is in P (by any algorithm), then $P = NP$.