Warm-Up Problem

Rewrite

$$(r \wedge (p \to r) \wedge (p \vee (q \wedge r)))$$

in CNF.

Have you started to write a list of definitions? Why not start now?

Propositional Logic: Resolution

Carmen Bruni

Lecture 7

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

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,

$$\Sigma \vdash_{S} \varphi \text{ implies } \Sigma \vDash \varphi \ .$$

A Minor Result

Claim: If $\Sigma \cup \{(\neg\varphi)\}$ is not satisfiable then $\Sigma \vDash \varphi$.

Proof: Let t be a valuation and suppose that $\Sigma^t = \texttt{T}$. Then since $(\Sigma \cup \{(\neg\varphi)\})^t = \texttt{F}$ we must have that $(\neg\varphi)^t = \texttt{F}$ and so $\varphi^t = \texttt{T}$ which is to say that $\Sigma \vDash \varphi$.

Resolution Is Sound

For resolution to be meaningful, we need the following.

Theorem. Suppose that $\{\alpha_1, \ldots, \alpha_m\} \vdash_{\mathit{Res}} \bot$; that is, there is a resolution refutation with premises as [CNF] clauses $\alpha_1, \ldots, \alpha_m$ and conclusion $\bot$.
Then the set $\{\alpha_1, \ldots, \alpha_m\}$ is not satisfiable.

That is, if $\Sigma \cup \{(\neg\varphi)\} \vdash_{\mathit{Res}} \bot$, then $\Sigma \cup \{(\neg\varphi)\}$ is not satisfiable.
Therefore, $\Sigma \vDash \varphi$.

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

Resolution Is Sound

Theorem. Suppose that $\{\alpha_1, \dots, \alpha_m\} \vdash_{\mathit{Res}} \bot$; that is, there is a resolution refutation with premises as [CNF] clauses $\alpha_1, \dots, \alpha_m$ and conclusion $\bot$. Then the set $\{\alpha_1, \dots, \alpha_m\}$ is not satisfiable.

We will prove this from first using a lemma and looking at the contrapositive of the statement.

Soundness: The central argument

Lemma: Suppose that a set $\Gamma = \{\beta_1, \ldots, \beta_k\}$ of [CNF] clauses is satisfiable. Let β_{k+1} be a formula [clause] obtained from Γ by one use of the resolution inference rule. Then the set $\Gamma \cup \{\beta_{k+1}\}$ is satisfiable. Further, the valuation satisfying Γ also satisfies $\Gamma \cup \{\beta_{k+1}\}$.

Soundness: The central argument

Lemma: Suppose that a set $\Gamma = \{\beta_1, \dots, \beta_k\}$ of [CNF] clauses is satisfiable. Let β_{k+1} be a formula [clause] obtained from Γ by one use of the resolution inference rule. Then the set $\Gamma \cup \{\beta_{k+1}\}$ is satisfiable. Further, the valuation satisfying Γ also satisfies $\Gamma \cup \{\beta_{k+1}\}$.

Proof: Let valuation t satisfy Γ; that is, $\beta_i^t = \mathtt{T}$ 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: $t(p) = \mathtt{F}$. Since $\beta_i^t = \mathtt{T}$, we must have $\gamma_1^t = \mathtt{T}$. Thus $\beta_{k+1}^t = \mathtt{T}$.
Case II: $t(p) = \mathtt{T}$. Since $\beta_j^t = \mathtt{T}$, we must have $\gamma_2^t = \mathtt{T}$. Thus $\beta_{k+1}^t = \mathtt{T}$.

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

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \dots, \alpha_m\}$ is a set of [CNF] clauses that is satisfiable, then $\Gamma \nvdash_{\mathsf{Res}} \bot$.

We proceed by induction...

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \dots, \alpha_m\}$ is a set of [CNF] clauses that is satisfiable, then $\Gamma \nvdash_{\mathsf{Res}} \perp$.

We proceed by induction... on the number of uses of the resolution rule!

Let $P(n)$ be the statement that...

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \ldots, \alpha_m\}$ is a set of [CNF] clauses that is satisfiable, then $\Gamma \nvdash_{\mathsf{Res}} \perp$.

We proceed by induction... on the number of uses of the resolution rule!

Let $P(n)$ be the statement that... if α is obtained from n uses of the resolution rule of formulas in any set of clauses Γ with a valuation t satisfying $\Gamma^t = \mathtt{T}$, then $(\Gamma \cup \{\alpha\})^t = \mathtt{T}$.

Base Case: If $n = 0$ then...

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \ldots, \alpha_m\}$ is a set of [CNF] clauses that is satisfiable, then $\Gamma \nvdash_{\text{Res}} \bot$.

We proceed by induction... on the number of uses of the resolution rule!

Let $P(n)$ be the statement that... if α is obtained from n uses of the resolution rule of formulas in any set of clauses Γ with a valuation t satisfying $\Gamma^t = \mathtt{T}$, then $(\Gamma \cup \{\alpha\})^t = \mathtt{T}$.

Base Case: If $n = 0$ then...$\alpha \in \Gamma$ to begin with and so $\Gamma \cup \{\alpha\} = \Gamma$ is satisfiable by assumption.

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \ldots, \alpha_m\}$ is a set of [CNF] clauses that is satisfiable, then $\Gamma \nvdash_{\text{Res}} \perp$.

We proceed by induction... on the number of uses of the resolution rule!

Let $P(n)$ be the statement that... if α is obtained from n uses of the resolution rule of formulas in any set of clauses Γ with a valuation t satisfying $\Gamma^t = \mathtt{T}$, then $(\Gamma \cup \{\alpha\})^t = \mathtt{T}$.

Base Case: If $n = 0$ then...$\alpha \in \Gamma$ to begin with and so $\Gamma \cup \{\alpha\} = \Gamma$ is satisfiable by assumption.

Induction Hypothesis: Assume that $P(i)$ is true for all $0 \leq i \leq k$ for some $k \in \mathbb{N}$.

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \dots, \alpha_m\}$ is a set of well-formed formulas in CNF that is satisfiable, then $\Gamma \nvdash_{\mathsf{Res}} \bot$.

Inductive Conclusion: Assume that α is formed by using $k+1$ instances of the resolution rule for formulas starting in the set Γ and suppose that t is a valuation satisfying $\Gamma^t = \mathtt{T}$.

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \ldots, \alpha_m\}$ is a set of well-formed formulas in CNF that is satisfiable, then $\Gamma \nvdash_{\mathsf{Res}} \perp$.

Inductive Conclusion: Assume that α is formed by using $k + 1$ instances of the resolution rule for formulas starting in the set Γ and suppose that t is a valuation satisfying $\Gamma^t = \mathtt{T}$.

Now, let $\{\beta_1, \ldots, \beta_k\}$ be the set of formulas in Γ and all of the clauses β_j created by using the resolution rule the k times before arriving at α. By the induction hypothesis, we have that $\beta_j^t = \mathtt{T}$ holds for all $1 \leq j \leq k$.

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \ldots, \alpha_m\}$ is a set of well-formed formulas in CNF that is satisfiable, then $\Gamma \nvdash_{\text{Res}} \bot$.

Inductive Conclusion: Assume that α is formed by using $k+1$ instances of the resolution rule for formulas starting in the set Γ and suppose that t is a valuation satisfying $\Gamma^t = \mathtt{T}$.

Now, let $\{\beta_1, \ldots, \beta_k\}$ be the set of formulas in Γ and all of the clauses β_j created by using the resolution rule the k times before arriving at α. By the induction hypothesis, we have that $\beta_j^t = \mathtt{T}$ holds for all $1 \leq j \leq k$.

This still leaves the question on how do we show that α is satisfiable?

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \dots, \alpha_m\}$ is a set of well-formed formulas in CNF that is satisfiable, then $\Gamma \nvdash_{\text{Res}} \perp$.

Inductive Conclusion: Assume that α is formed by using $k+1$ instances of the resolution rule for formulas starting in the set Γ and suppose that t is a valuation satisfying $\Gamma^t = \mathtt{T}$.

Now, let $\{\beta_1, ..., \beta_k\}$ be the set of formulas in Γ and all of the clauses β_j created by using the resolution rule the k times before arriving at α. By the induction hypothesis, we have that $\beta_j^t = \mathtt{T}$ holds for all $1 \leq j \leq k$.

This still leaves the question on how do we show that α is satisfiable? This is the lemma applied to $\Sigma = \Gamma \cup \{\beta_1, ..., \beta_k\}$! Thus, $\alpha^t = \mathtt{T}$.

This completes the proof of the statement of $P(n)$ by mathematical induction.

Resolution Is Sound

Theorem Restated If $\Gamma = \{\alpha_1, \ldots, \alpha_m\}$ is a set of [CNF] clauses that is satisfiable, then $\Gamma \nvdash_{\text{Res}} \bot$.

We showed that $P(n)$ is true where $P(n)$ is the statement that:

> If α is obtained from n uses of the resolution rule of formulas in any set of clauses Γ with a valuation t satisfying $\Gamma^t = \mathtt{T}$, then $(\Gamma \cup \{\alpha\})^t = \mathtt{T}$.

Thus, assume towards a contradiction that $\Gamma \vdash_{\text{Res}} \bot$ for some satisfiable set of clauses Γ, then we could form $\bot$ using the resolution rule after some number of steps. The only way to form bottom is if, after some number of resolution rules applied to Γ, we derive both p and $(\neg p)$ for some atom p. The statement above implies that $p^t = \mathtt{T} = (\neg p)^t$.

Can Resolution Fail?

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

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

$$\Sigma \models \alpha \quad \text{implies} \quad \Sigma \vdash_{\mathsf{S}} \alpha \ .$$

or equivalently (by the contrapositive):

$$\Sigma \nvdash_{\mathsf{S}} \alpha \quad \text{implies} \quad \Sigma \not\models \alpha \ .$$

Theorem. Resolution is a complete refutation system for CNF formulas. That is, by the contrapositive, if there is no proof of $\bot$ 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.

Resolution Is Complete (Outline)

Claim. Suppose that a resolution proof "reaches a dead end"—that is, no new clause can be obtained, and yet $\bot$ 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.

Let $P(n)$ be the statement that any set of formulas formed by starting with a set of formulas Γ with n Propositional variables and joining with it all of the formulas in a resolution proof that reached a dead end is satisfiable.

Resolution Is Complete, part I

Let $P(n)$ be the statement that any set of formulas formed by starting with a set of formulas Γ with n Propositional variables and joining with it all of the formulas in a resolution proof that reached a dead end is satisfiable.

Resolution Is Complete, part I

Let $P(n)$ be the statement that any set of formulas formed by starting with a set of formulas Γ with n Propositional variables and joining with it all of the formulas in a resolution proof that reached a dead end is satisfiable.

Base Case: 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.

Resolution Is Complete, part I

Let $P(n)$ be the statement that any set of formulas formed by starting with a set of formulas Γ with n Propositional variables and joining with it all of the formulas in a resolution proof that reached a dead end is satisfiable.

Base Case: 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.

Inductive Hypothesis: We have that $P(i)$ holds for all $0 \leq i \leq k$ for some fixed $k \in \mathbb{N}$.

Completeness Proof, part II

Inductive Conclusion: 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 $\perp$. Select any one variable, say p, and separate the clauses into three sets:

$S_{\boldsymbol{p}}$: the clauses that contain the literal p.
$S_{(\neg \boldsymbol{p})}$: 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 t.

Completeness Proof, part III

We have a valuation t, such that $R^t = \mathtt{T}$.

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

In this case, the set S_p is already satisfied. Define a valuation v satisfying

$$v(q) = \begin{cases} t(q) & \text{if } q \neq p \\ \mathtt{F} & \text{if } q = p. \end{cases}$$

Then, this v must satisfy $S^v_{(\neg p)} = \mathtt{T}$ by definition (as $(\neg p)^v = \mathtt{T}$) and since v and t agree outside of p, we see that $S^v_p = R^v = \mathtt{T}$.

Completeness Proof, part IV

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

In this case, define

$$v(q) = \begin{cases} t(q) & \text{if } q \neq p \\ \mathtt{T} & \text{if } q = p. \end{cases}$$

Then, this v satisfies $S_p^v = \mathtt{T}$ and $R^v = \mathtt{T}$ as before.

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 $P = NP$.