COMP 409, Lecture 16

March 18, 2003

1 Introduction

In the past few lectures, we have discussed the Hilbert-Ackerman and Meredith Deductive systems. Unfortunately, neither of these systems is very good for automated deduction. As an example, try to prove that $\vdash (p \to p)$. So we realize that in Hilbert-Ackerman's System and Meredith's System, checking a proof is easy but 'finding' a proof is difficult. This makes these two systems useless for automated reasoning since searching for a proof in these systems is difficult. This is precisely because of the fact that in these systems, there is no bound on the length of proofs. To alleviate this problem, we change our basis and go to clausal formulas. We resort to what is called **Resolution**.

2 Resolution

The technique of resolution was developed by John Robinson in 1965 at Rice University. It was considered a breakthrough, due to its fundamental break with the traditional structure of Hilbert-style proof systems.

Resolution changes the rules of the game by doing away with axioms and dealing exclusively with clauses; that is, disjunctions of primitives. We know that if we have n propositions, we have at most $2n$ literals and so at most $2^{2n} = 4^n$ clauses. If we omit the trivial clauses(like $(p \vee \neg p)$), we are left with at most 3^n non trivial clauses. This gives us at most 2^{3^n} clausal formulas. So by dealing with clausal formulas we have reduced our search space from infinity to something exponential.
Resolution also has only one rule, which is as follows:

$$\frac{p_0 \vee p_1 \vee \cdots \vee p_m \vee p, q_0 \vee q_1 \vee \cdots q_n \vee \neg p}{p_0 \vee p_1 \vee \cdots \vee p_m \vee q_0 \vee q_1 \vee \cdots \vee q_m}$$

Note that in this rule, p_i and q_i can be either propositions or their negations. This rule is used to take two clauses from the pool of proven ones and combine them (if one contains a proposition and the other one its negation) into a new clause which is added to the pool. The rule is clearly sound. Since |Clauses|=3^n, we can always find a proof by simply performing all possible resolutions. (but maybe not very soon)

Note that if we have two clauses of the form $\{p\}$ and $\{\neg p\}$, the resolution rule produces the empty clause, a disjunction of no terms. Since we define a disjunction to be a formula which is true only if one or more of its literals are true, the empty clause is unsatisfiable. We call this empty clause 'box', written as $\Box$.

To turn this into a proof technique, we take a set of clauses and try to show that they are inconsistent and hence lead to a contradiction. If there is no contradiction then the clauses must be satisfiable. If there is a contradiction then the formula made up of the clauses must be invalid. The process of finding a contradiction is called finding a refutation.

Definition 1 *Formally, a* Γ-refutation *of a set C of clauses is a sequence $\kappa_1, \ldots, \kappa_n$ of clauses that every κ_i is either in C or follows by resolution (i.e., it is a* resolvent*) of prior clauses in the sequence. In addition we, require that $\kappa_n = \Box$. If there is a Γ-refutation of C, then we say that C is* Γ-refutable.

Definition 2 *C is Γ-refutable if there is a Γ-refutation of C.*

This is the basis on which we do proofs by contradiction.

1. Convert: φ to $(\neg\varphi)$
2. Convert: $(\neg\varphi)$ to CNF
3. Find a refutation of $(\neg\varphi)$

For an example on how we prove things in this system, suppose we want to prove $\models \varphi$ where $\varphi = ((p \to (q \to r)) \to ((p \to q) \to (p \to r)))$
We assume $\neg\varphi$:
Then we start with
$(p \to (q \to r)), (p \to q), p, \neg r$
or (a): $\neg p \vee \neg q \vee r$, (b): $\neg p \vee q$, (c): p, (d) $\neg r$
Then we start resolving.

1. q (from (b) and (c))
2. $\neg q \vee r$ (from (a) and (c))
3. r (from (1) and (2))
4. $\Box$ (from (3) and (d))

So $\neg\varphi$ is refuted proving $\models \varphi$.

Note that the full resolution process is completed in exponential time, because the set of all possible clauses over a fixed set of propositions is $O(3^n)$, where n is the number of propositions.

2.1 Completeness of Resolution

Theorem 1 *C is refutable iff C is unsatisfiable.*

This theorem will prove that **Resolution** is both sound and complete. The forward direction is 'easy' and we will only prove the other direction. The intuition being resolution allows you to eliminate propositions (as we eliminate variables in simultaneous equations).

Lemma 1 *An unsatisfiable finite set of clauses is refutable.*

Proof (by induction of the number of propositions):

- **Proviso**: Note that we ignore *trivial clauses*, i.e., clauses that contain complementary literals. This proof can be extended to the trivial clause containing set by using the function $NonTriv(C) = \{k \in C : k \text{ is non-trivial}\}$, which eliminates trivial clauses.

Let C be an unsatisfiable finite not-trivial set of clauses.

- **Base case**: $|AP(C)| = 0$. There are two cases to consider:
  - C is $\emptyset$. This case is not possible because $\emptyset$ is satisfiable.
  - C is $\{\square\}$. This case already contains a refutation.

- **Inductive step**: Recall that for a set C of clauses and a proposition $p \in AP(C)$, we define the set $R_p(C)$ (C resolved with p) to consist of the clauses

  $\{\alpha : \alpha \in C$ and p does not occur in $\alpha\} \cup \{\alpha \vee \beta : p \vee \alpha \in C$ and $\neg p \vee \beta \in C\}$.

  By construction, $AP(R_p(C)) \subseteq AP(C)$. It remains to be shown that $R_p(C)$ is still unsatisfiable.

  Lemma 2 *If C is unsatisfiable, then $R_p(C)$ is also unsatisfiable.*

  Proof: Assume that C is unsatisfiable, and suppose that τ is a truth assignment that satisfy $R_p(C)$. We can assume that τ is not defined on p, since $p \notin AP(R_p(C))$. Let $\tau_1 = \tau[p \mapsto 0]$ and $\tau_2 = \tau[p \mapsto 1]$. We know that $\tau_1 \not\models C$ and $\tau_2 \not\models C$. So there are clauses $\kappa_1, \kappa_2 \in C$ such that $\tau_1 \not\models \kappa_1$ and $\tau_1 \not\models \kappa_1$. It follows that $\kappa_1, \kappa_2 \notin R_p(C)$, so p occur in both κ_1 and κ_2. That is, we must have that $\kappa_1 = p \vee \alpha$ and $\kappa_2 = \neg p \vee \beta$. It follows that $\alpha \vee \beta \in R_p(C)$, so $\tau \models \alpha \vee \beta$. Thus, $\tau \models \alpha$ or $\tau \models \beta$. But in the former case we get $\tau_1 \models \kappa_1$, and in the latter case we get $\tau_2 \models \kappa_2$. Contradiction. QED

  This completes the proof of the completeness of resolution.

3 Proof Complexity

What we expect from any proof is that we should be able to check the steps of the proof 'easily'. We now formalize what it means to be able to check a proof 'easily'. We define $Deductions(\Gamma)$ as follows.

Definition 3 *A* Γ-deduction *is a sequence* $\varphi_1, ..., \varphi_n$ *such that for every* φ_i*, there are* $i_1 < i_2 < ... < i_{k-1} < i$ *such that* $\varphi_{i_1}\varphi_{i_2}...\varphi_{i_{k-1}}\varphi_i \in \Gamma$.
$Deductions(\Gamma) = \{\varphi_1, ..., \varphi_n | \varphi_1, ..., \varphi_n \text{ is a } \Gamma - deduction\}$

Now we define *polynomial feasibility* of Γ as follows.

Definition 4 Γ *is polynomially feasible if* $Deductions(\Gamma) \in PTIME$.

Definition 5 Γ *is a polynomially bounded system iff there is polynomial* ρ *s.t. whenever* $\vdash_\Gamma \varphi$*, there is a deduction* $\varphi_1, ..., \varphi_n \vdash \varphi$ *in* Γ*, such that*

- $n \leq \rho(|\varphi|)$
- $|\varphi_i| \leq \rho(|\varphi|)$

A proof is polynomially bounded iff it has polynomial lines and each line has polynomial length.

Theorem 2 *If* Γ *is a sound and complete deductive system which is polynomially feasible and bounded,* $VALID \in NP$.

Since VALID is co-NP-complete, it'd follow that NP=co-NP, which is widely believed not to be the case.

Corollary 1 *We don't think we can find* Γ*, that is sound, complete, polynomially feasible and bounded.*

Theorem Cook+Reckhow, 1974: *For a more general notion of deductive system, NP=co-NP iff there is a sound, complete, polynomially feasible and bounded deduction system.*

Resolution is sound and complete, but it is not polynomial bounded. Recall that, in Resolution Theorem Proving, we iteratively construct increasingly larger sets of formulas until a contradiction is reached.

$$\begin{aligned} C_0 &= \varphi \\ C_1 &= R_{p_1}(C_0) \\ C_2 &= R_{p_2}(C_1) \\ &\vdots \\ C_n &= R_{p_n}(C_{n-1}) \end{aligned}$$

While each set C_i is only quadratically bigger than C_{i-1}, the overall growth can be exponential. Put otherwise, the length of a resolution refutation can be exponential. We now focus on the length of the refutations.

Recall the *Pigeon-Hole Principle*: If we have $f : \{1, \ldots, n\} \mapsto \{1, \ldots, n-1\}$, then for some $1 \leq i \leq j \leq n$ we have $f(i) = f(j)$.

It can be write as: $R \subseteq \{1, ..., n\} \times \{1, ..., n-1\}$, for every pigeon i there is some hole ' k s.t. $R(i, k)$. There exists $1 \leq i < j \leq n$ and $1 \leq k \leq n-1$, such that both $R(i, k)$ and $R(j, k)$ hold.

We formulate this (for a fixed $n \in \mathbb{N}$) in propositional logic. To do so, we will use the following atomic propositions, for each i, j such that $1 \leq i \leq j \leq n$:

$$P_{ij} \stackrel{\text{def}}{=} R(i, j)$$

PHP_n

$$\left(\bigwedge_{1 \leq i \leq n} \bigvee_{1 \leq k \leq n-1} P_{ik} \right) \rightarrow \left(\bigvee_{1 \leq i < j \leq n} \bigvee_{1 \leq k < n} (P_{ik} \wedge P_{jk}) \right)$$

So, this formula is valid iff the Pigeon-Hole Principle (for a fixed n) holds.

Theorem Haken, 1975: *Every resolution refutation of* $\neg PHP_n$ *has length* $2^{\Omega(n)}$.

So, Resolution Theorem Proving is not *polynomially-bounded*: there is no polynomial function of the length of a theorem that places an upper bound on the length of the smallest proof of that theorem.

Conceivably, resolution have long refutations because it is such a weak system. So an active research area is establishing lower bounds for more powerful proof systems.