Lecture 17

October 14, 2005

1 Introduction

In the past few lectures, we have discussed the Hilbert-Ackerman Deductive system. Unfortunately, this system is very good for automated deduction. As an example, it is not very easy to prove that $\vdash (p \to p)$. So we realize that in Hilbert-Ackerman's System checking a proof is easy but 'finding' a proof is difficult. This makes this system (together with Meredith's system which was mentioned in one of the homeworks) hard to use 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 another technique which is called **Resolution**.

We will start with a formal review of deduction.

2 Deduction

A Deduction is a formulation of a proof. Deductions contain sequences of formulas and they have a relation established on the elements of the sequence. Formally, the set of deductions Γ is described as follows:

Definition 1 (Deduction) *A deduction* $\Gamma \subseteq Form^+$ *where* $Form^+$ *is the set of sequences of formulas. Every element* $\gamma \in \Gamma$ *is of the form* $\gamma : \langle p, p \to q, q \rangle$ *denoted as:* $\frac{p \;\; p \to q}{q}$

We will continue with this notion in the **Proof Complexity** section.

3 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 literals. 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 reduce the search space from infinite to finite, albeit exponential.

Resolution has only one rule, which is as follows:

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

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.

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 $\square$.

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 unsatisfiable. The process of finding a contradiction is called finding a refutation.

Definition 2 (Γ-refutation) *A* Γ-refutation *of a set C of clauses is a sequence $\kappa_1, \ldots, \kappa_n$ of clauses such 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 $\kappa_n = \square$. If there is a Γ-refutation of C, then we say that C is* Γ-refutable.

This naturally leads to the next definition.

Definition 3 (Γ-refutable formulas) *C is Γ-refutable if there is a Γ-refutation of C.*

Definition 3 gives us the theoretical framework on which we do proofs by contradiction.

1. Convert: φ to $(\neg\varphi)$
2. Convert: $(\neg\varphi)$ to CNF
3. Find a refutation of $(\neg\varphi)$
4. If $\neg\varphi$ is refutable, then φ must be valid.

As an example 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 $\neg\varphi$, that is,

$$(p \to (q \to r)), (p \to q), p, \neg r.$$

We break these into CNF clauses and give them names (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, thereby 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.

3.1 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 *A 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 a 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 $\{\Box\}$. 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

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

  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 satisfies $R_p(C)$. 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 $\tau_2 \not\models \kappa_2$. 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.