Lecture 2

Sep 8

- Proof system:

Defn A proof system for a language $L \subseteq \{0,1\}^*$ is a polynomial-time algorithm V s.t.

$$\forall x \in \{0,1\}^* : x \in L \iff \exists p \in \{0,1\}^* : V(x,p) \text{ accepts}$$

V is polynomially bounded if $|p| \leq \text{poly}(|x|)$

ex) SAT $:= \{ F : F \text{ is a satisfiable boolean formula} \} \subseteq$ "$\{0,1\}^*$"

"proof": p is a satisfying assignment!

$$F = (x_1 \vee \overline{x_2}) \wedge (\overline{x_3} \vee x_4) \qquad p = 1011$$

- Might have seen "proof systems" called "verifier".

Defn The complexity class

$$NP := \{ L \subseteq \{0,1\}^* : L \text{ has a poly-bounded proof system} \}$$

$$SAT := \{ \text{satisfiable formulas } F \}$$

$$UNSAT := \{ \text{unsatisfiable boolean formulas } F \}$$

F is unsatisfiable $\iff \neg F$ is a tautology

Defn A propositional proof system is a proof system for UNSAT (or TAUT).

Thm There is a polynomially-bounded propositional proof system iff $NP = coNP$.

$coNP = \{ \overline{L} : L \in NP \}$

Proof ($\Leftarrow$) $NP = coNP$ then $UNSAT \in coNP = NP$, so by defn UNSAT has a prop. proof system.

($\Rightarrow$) If UNSAT has a poly-bounded P.P.S. then

UNSAT is coNP-complete, so every language $L \in coNP$ reduces to UNSAT, and so $L \in NP$.

$coNP \subseteq NP$ $\quad$ $NP \subseteq coNP$ is symmetric. $\square$

[Cook-Reckhow 79] Research program for separating NP and coNP: Prove lower bounds against stronger and stronger proof systems.

Examples of PPS

- Truth table! Just evaluate the boolean formula on every input.

Resolution Restrict F in conjunctive normal form (CNF)

$\hookrightarrow$ F is an AND of ORs of LITERALS (inputs or negations)

ex) $F = x_1 \wedge \underbrace{(\overline{x}_1 \vee x_2)}_{\text{clause}} \wedge (\overline{x}_2 \vee x_3) \wedge \overline{x}_3$ (definitely unsat)

Defn A resolution refutation of $F = C_1 \wedge \cdots \wedge C_m$ is a sequence of clauses

$$D_1, D_2, \cdots, D_s = \perp$$

Resolution Rule: $\dfrac{A \vee x \quad B \vee \overline{x}}{A \vee B}$

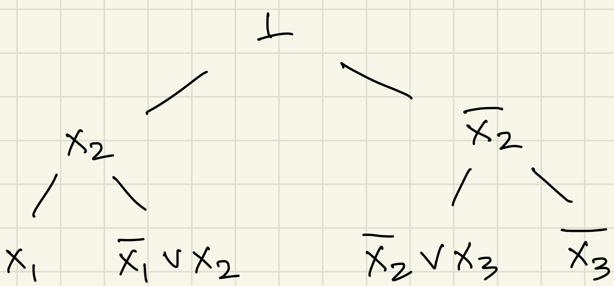
- $\forall i = 1 \cdots m$, $D_i = C_i$
- $D_s = \perp$ (empty clause, false)
- For all $i > m$, D_i is deduced from D_j, D_k with $j, k < i$ by the resolution rule.

Fact Resolution rule is sound: any assignment satisfying the input clauses also satisfies the output.

$\Rightarrow$ If F has a resolution refutation then F is unsatisfiable!

ex)

$$F = \underset{C_1}{x_1} \wedge \underset{C_2}{(\overline{x}_1 \vee x_2)} \wedge \underset{C_3}{(\overline{x}_2 \vee x_3)} \wedge \underset{C_4}{\overline{x}_3}$$



C_1
C_2
C_3
C_4
$x_2 = \mathrm{Res}(C_1, C_2)$
$\overline{x}_2 = \mathrm{Res}(C_3, C_4)$
$\perp = \mathrm{Res}(x_2, \overline{x}_2)$

Q. Is resolution complete? (i.e. does every unsat CNF formula have a resolution refutation?)

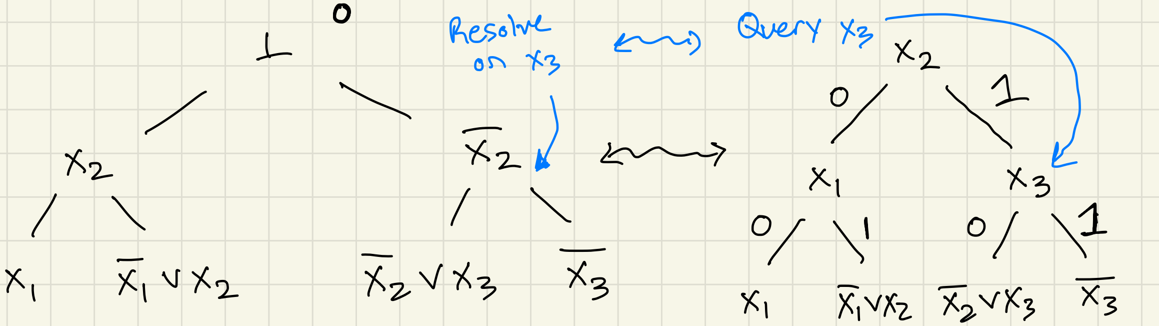
A. Yes!

Thm Resolution is complete. Sep 8

Proof Let F be an unsatisfiable CNF Formula.

Idea: simulate a truth table proof

Decision tree!



Defn If F is an unsatisfiable CNF formula, then

Search(F)

is the following (algorithmic) problem: given an assignment z to the inputs of F, find a clause falsified by z.

- Every path in this tree is consistent with some boolean assignment.
- Clauses at leaves are falsified by the assignments on the path to the leaf.

Obs Decision trees solving Search(F)

$\equiv$

Tree-like resolution proofs of F. $\square$

Lower bounds?

Truth tables — All proofs are exponentially long.

Resolution —

Proving lower bounds on resolution was long-standing open problem Sep 8

[Tseitin early 60s] Proposed lower bounds on resolution as an open question, proved lower bounds on "regular" resolution.

[Haken 85] Any resolution refutation of the pigeonhole principle requires exponential length.

PHP_n^{n+1} = variables x_{ij} $i \in [n+1], j \in [n]$

$$x_{ij} = 1 \Leftrightarrow \text{pigeon } i \text{ mapped to hole } j$$

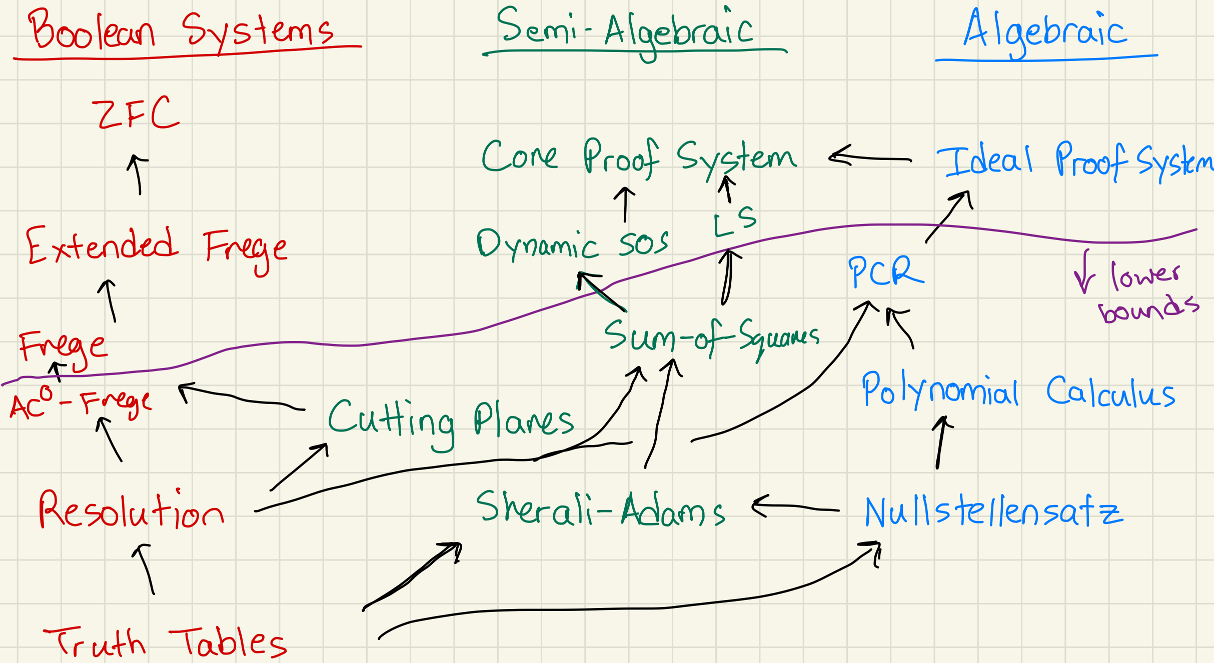
$\forall i \in [n+1]$: $\bigvee_{j=1}^{n} x_{ij}$ (all pigeons in a hole)

$\forall i \neq j \in [n+1]$ $\overline{x_{ik}} \vee \overline{x_{jk}}$ (no 2 pigeons in one hole)

$\forall k \in [n]$

Other proof systems? (See next page)

Sep 8



$A \longrightarrow B$: Proof system B simulates proof system A

i.e. B is at least as efficient as A.