- Proof system:

**Defn** A proof system for a language \( L \subseteq \Sigma^* \) is a polynomial-time algorithm \( \mathcal{V} \) s.t.

\[
\forall x \in \Sigma^* : x \in L \iff \exists p \in \Sigma^* : \mathcal{V}(x, p) \text{ accepts}
\]

\( \mathcal{V} \) is polynomially bounded if \( 1p1 \leq \text{poly}(1x1) \)

ex) \( \text{SAT} := \{ F : F \text{ is a satisfiable boolean formula} \} \subseteq \Sigma^* \)

"proof": \( p \) is a satisfying assignment!

\( F = (x_1 \lor \overline{x}_2) \land (\overline{x}_3 \lor x_4) \quad p = 1011 \)

Might have seen "proof systems" called "verifiers".

**Defn** The complexity class

\( \text{NP} := \{ L \subseteq \Sigma^* : L \text{ has a poly-bounded proof system} \} \)

\( \text{SAT} := \{ \text{satisfiable formulas} \} \)

\( \text{UNSAT} := \{ \text{unsatisfiable boolean formulas} \} \)

\( F \) is unsatisfiable \( \iff \neg F \) is a tautology

**Defn** A propositional proof system is a proof system for \( \text{UNSAT} \) (or \( \text{TAUT} \)).
There is a polynomially-bounded propositional proof system iff
\[ \text{NP} = \text{coNP}. \]
\[ \text{coNP} = \{ \neg L : \text{LENP} \} \]

Proof: \((\Leftarrow)\) If \(\text{NP} = \text{coNP}\) then \(\text{UNSAT} \in \text{coNP} = \text{NP}\), so by defn \(\text{UNSAT}\) has a prop. proof system.

\((\Rightarrow)\) If \(\text{UNSAT}\) has a poly-bounded P.P.S. then
\(\text{UNSAT}\) is \(\text{coNP}\)-complete, so every language \(\text{LECONP}\) reduces to \(\text{UNSAT}\), and so \(\text{LENP}\).

\(\text{coNP} \leq \text{NP} \uparrow \text{NP} \leq \text{coNP}\) is symmetric. \(\square\)

[Cook-Reckhow 79] Research program for separating \(\text{NP}\) and \(\text{coNP}\): 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)

\[ F \text{ is an AND of ORs of LITERALS (inputs or negs)} \]

\[ \text{ex} F = x_1 \land (\neg x_1 \lor x_2) \land (\neg x_2 \lor x_3) \land \neg x_3 \quad (\text{definitely unsat}) \]

clause
**Defn** A resolution refutation of \( F = C_1 \land \ldots \land C_m \) is a sequence of clauses

\[
D_1, D_2 \ldots, D_s = \bot
\]

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

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

\[ \Rightarrow \text{If } F \text{ has a resolution refutation then } F \text{ is unsatisfiable!} \]

**ex)** \( C_1, C_2, C_3, C_4 \)

\( F = x_1 \land (\overline{x_1} \lor x_2) \land (\overline{x_2} \lor x_3) \land \overline{x_3} \)

\( \frac{C_1}{C_2} \)
\( \frac{C_2}{C_3} \)
\( \frac{C_3}{C_4} \)
\( \frac{x_2 = \text{Res}(C_1, C_2)}{\overline{x_2}} \)
\( \frac{x_2 = \text{Res}(C_3, C_4)}{\overline{x_2}} \)
\( \frac{\bot = \text{Res}(x_2, \overline{x_2})}{\bot} \)

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

A. Yes!
Resolution is complete.

Proof Let \( F \) be an unsatisfiable CNF Formula.

Idea: simulate a truth table proof

\[
\begin{array}{c}
X_1 \quad \overline{X_1} \lor X_2 \\
\overline{X_2} \lor X_3 \\
\overline{X_3} \\
\end{array}
\]

Decision tree!

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

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.

Tree-like resolution proofs of \( F \).

Lower bounds? Truth tables — All proofs are exponentially long. Resolution -
Proving lower bounds on resolution was long-standing open problem

[Tseitin 1968] Proposed lower bounds on resolution as an open question, proved lower bounds on "regular" resolution.

[Haken 1985] Any resolution refutation of the pigeonhole principle requires exponential length.

\[ \text{PHP} \] = variables \( x_{ij} \), \( i \in [n+1], j \in [n] \)

\[ x_{ij} = 1 \iff \text{pigeon } i \text{ mapped to hole } j \]

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

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

\[ \forall k \in [n] \]

Other proof systems? (See next page)
Boolean Systems

ZFC

→

Extended Frege

→

Frege

→

AC^0-Frege

Truth Tables

Semi-Algebraic

Cone Proof System

→

Dynamic sos

→

Sum-of-Squares

→

Cutting Planes

→

Sherali-Adams

→

Nullstellensatz

Algebraic

Ideal Proof System

→

PCR

→

Polynomial Calculus

Truth Tables

→

A → B : Proof system B simulates proof system A

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