The most important NP-complete (logic) problem family!

**SAT = Satisfiability:**  
- given a Boolean formula (AND, OR, NOT) over \( n \) variables \( x_1, x_2, \ldots, x_n \)  
- can you set \( x_i \)'s to make formula true?

\[ \text{Circuit SAT: formula expressed as circuit of gates} \]

\[
\begin{align*}
&x_1 \\
&x_2 \\
&x_3 \\
\end{align*}
\]

\[ \text{(allows re-use)} \]

**CNF SAT:**  
formula = AND of clauses  
clause = OR of literals  
literal \( \in \{ x_i, \text{NOT } x_i \} \)

- can view as bipartite graph: variables vs. clauses, positive/negative edges

**3SAT:**  
clause = OR of 3 literals  
i.e. clause degrees = 3 (but allow repeats)

**3SAT-3:** each variable occurs in \( \leq 3 \) clauses

- E3SAT-4 but E3SAT-3 \( \notin \mathbf{P} \)
- exactly 3 distinct literals per clause

**Monotone 3SAT:**  
each clause all positive or all negative
Beware polynomial-time variants!

**2SAT:** clause = OR of 2 literals
- \( \text{polynomial} \)
- \( x \lor y \equiv \text{NOT } x \Rightarrow y \) (\( \equiv \) NOT \( y \Rightarrow x \))
- guess \( x_i \) and follow all implication chains to check ok

But...

**Max 2SAT:** set variables to maximize # true clauses
- NP-complete [Garey, Johnson, Stockmeyer 1976]

**Horn SAT:** each clause has \( \leq 1 \) positive literal
- \( \text{NOT } x \lor \text{NOT } y \lor \text{NOT } z \lor w \)
- \( \equiv \text{NOT } (x \land y \land z) \lor w \)
- \( \equiv (x \land y \land z) \Rightarrow w \)
- \( \Rightarrow \text{polynomial like 2SAT} \) [Horn 1951]

**Dual-Horn SAT:** each clause has \( \leq 1 \) negative literal
- "weakly positive satisfiability" [Schaefer 1978]
- negate all variables \( \Rightarrow \) Horn SAT
- \( \Rightarrow \text{polynomial} \)

**DNF SAT:** formula = OR of clauses
- clause = AND of literals
- \( \Rightarrow \) satisfiable \( \iff \geq 1 \) clause

\( \downarrow \) **Disjunctive Normal Form**
Alternative clauses for 3SAT:

1-in-3SAT = exactly-1 3SAT \[\text{[Schaefer 1978]}\]
- clause = exactly 1 of 3 literals is true
  \(\Rightarrow 2\) false \(\sim\) TFF, FTF, FFT

Positive 1-in-3SAT: no negations – all literals positive
But... sometimes called “monotone”

Positive not-exactly-1 3SAT: \[\text{[Schaefer 1978]}\]
- clause = 0, 2, or 3 variables are true
  i.e. \(x_i \Rightarrow (x_j \text{ or } x_k) \Rightarrow \text{Dual Horn}\)
- also require \(x_1 = \text{TRUE}\) (else set all \(x_i = \text{FALSE}\))
  & \(x_2 = \text{FALSE}\) (or allow \(|\text{clause}| \leq 3\))
- polynomial

NAE 3SAT = not-all-equal 3SAT \[\text{[Schaefer 1978]}\]
- clause = 3 literals not all the same value
  (forbid FFF \& TTT \(\Rightarrow 1\) or 2 true, 2 or 1 false
  \(\sim\) whereas 3SAT forbids just FFF)
- nice symmetry between TRUE \& FALSE

Positive NAE 3SAT: no negations – all literals positive
Schaefer’s Dichotomy:  
- formula = AND of clauses  
- general clause = relation on variables  
  - assume in CNF (unique if minimal)  
  ⇒ AND of subclauses  
⇒ SAT is polynomial if either: 
  - setting all variables true or all variables false satisfies all relations  
  - subclauses are all Horn or all Dual Horn  
  - relations are all 2-CNF (subclause sizes ≤ 2)  
  - every relation can be expressed as a system of linear equations over \( \mathbb{Z}_2 \):  
    \[
    \lor \oplus \land \, x_i \oplus x_j \oplus x_k \oplus x_l = 0 \text{ or } 1
    \]
  "XOR SAT" \( \Leftrightarrow \text{XOR} \) \( \Leftrightarrow \) Gaussian elimination  
& otherwise, SAT is NP-complete!

Another hard version of SAT – seldom used?

2-colorable perfect matching:  
- given a planar 3-regular graph  
- 2-color the vertices such that every vertex has exactly 1 same-colored neighbor  
- special case of 2-in-4-SAT (planarity & 3-regular left as exercise)
Pushing blocks:
- 1x1 robot navigating grid of blocks
- goal: get robot from start to target

- Push-k: robot can push up to k blocks at once
- Push-∞: infinite strength
- PushPush: blocks slide until they hit something
- PushPushPush: blocks slide other blocks in chain reaction, up to strength k
- Push---F: some blocks are fixed
- Push---X: robot path cannot self-intersect (tiles disappear after traversal)

- Sokoban = Push-1F but with goal of filling target squares with blocks
Push-$*$: reduction from 3SAT \[\text{[Hoffmann 2000]}\]
- variable: push right in $x_i$ or $\overline{x}_i$ row
  $\rightarrow$ fill in row of connection gadget
- connection: 1 free cell per occurrence of literal
- bridge: move up & block off leftward path
- clause: need a free spot below to traverse

\underline{Push-Push-1 in 3D}: reduction from 3SAT
\[\text{[O'Rourke & Smith Problem Solving Group 1999]}\]

\underline{(Push)Push-1 (in 2D): } reduction from 3SAT
\[\text{[Demaine, Demaine, O'Rourke 2000]}\]
- clause gadget: block other, lock gadget
- XOR crossover: $N \rightarrow S$ xor $W \rightarrow E$
- unidir. crossover: optional $N \rightarrow S$, then $W \rightarrow E