COL:750/7250

Foundations of Automatic Verification

Instructor: Priyanka Golia

Course Webpage

https://priyanka-golia.github.io/teaching/COL-750-COL7250/index.html

Boolean /propositional formulas ——> SAT Solvers

If formula is SATisfiable, gives an satisfying assignment

UNSAT

Equisatisfiable Formulas

Boolean (propositional) formulas F and G are equisatisfiable if the following holds:

$Vars(G) \subseteq Vars(F)$

- Every satisfying assignment of G can be extended to the satisfying assignment of F.
  - For every $\tau \vDash G$, there is a τ' such that τ' extends τ to $Vars(F/G)$, and $\tau' \vDash F$
- Every satisfying assignment of F can be projected on $Vars(G)$ to get the satisfying assignment of G.
  - For every $\tau' \vDash F$, there is a τ such that $\tau = \tau'_{\downarrow Vars(G)}$ and $\tau \vDash G$

Equisatisfiable Formulas

$$F = (p \vee \alpha) \wedge (\neg p \vee \beta) \quad \text{and} \quad G = (\alpha \vee \beta)$$

$$Models(F) := \{(p \mapsto 1, \alpha \mapsto 0, \beta \mapsto 1), (p \mapsto 1, \alpha \mapsto 1, \beta \mapsto 1), (p \mapsto 0, \alpha \mapsto 1, \beta \mapsto 0), (p \mapsto 0, \alpha \mapsto 1, \beta \mapsto 1)\}$$

$$Models(F)_{\downarrow Vars(G)} := \{(\alpha \mapsto 0, \beta \mapsto 1), (\alpha \mapsto 1, \beta \mapsto 1), (\alpha \mapsto 1, \beta \mapsto 0)\}$$

$$Models(F)_{\downarrow Vars(G)} := Models(G)$$

For every $\tau \vDash G$, there is a τ' such that τ' extends τ to $Vars(F/G)$, and $\tau' \vDash F$

For every $\tau' \vDash F$, there is a τ such that $\tau = \tau'_{\downarrow Vars(G)}$ and $\tau \vDash G$

Equisatisfiable Formulas (modified)

$G = p \vee (q \wedge r)$

$F = (p \vee t) \wedge (t \leftrightarrow q \wedge r)$

$F' = (p \vee t) \wedge (t \rightarrow q \wedge r)$

Is F and G equisatisfiable?

Is F’ and G equisatisfiable?

Exercise:

$G = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$

F $= (t_1 \vee t_2) \wedge (t_1 \leftrightarrow (x_1 \wedge y_1) \wedge (t_2 \leftrightarrow (x_2 \wedge y_2)$

F’ $= (t_1 \vee t_2) \wedge (t_1 \rightarrow (x_1 \wedge y_1) \wedge (t_2 \rightarrow (x_2 \wedge y_2)$

Is F and G equisatisfiable?

Is F’ and G equisatisfiable?

Every formula F can be represented in CNF form, say F_{CNF} in polynomial time such that F is satisfiable if and only if F_{CNF} is satisfiable.

K-SAT

CNF: $F = C_1 \wedge C_2 \wedge C_3 \ldots \wedge C_m$

where $C_i = (l_1 \vee l_2 \vee \ldots \vee l_k)$

where $l_j = p; l_j = \neg p$

Where p is propositional variable

If $K = 2$, then $2 - SAT$. $F = (x_1 \vee \neg x_2) \wedge (x_3 \vee x_4)$

If $K = 3$, then $3 - SAT$. $F = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4)$

Exercise-3: Propositional Logic

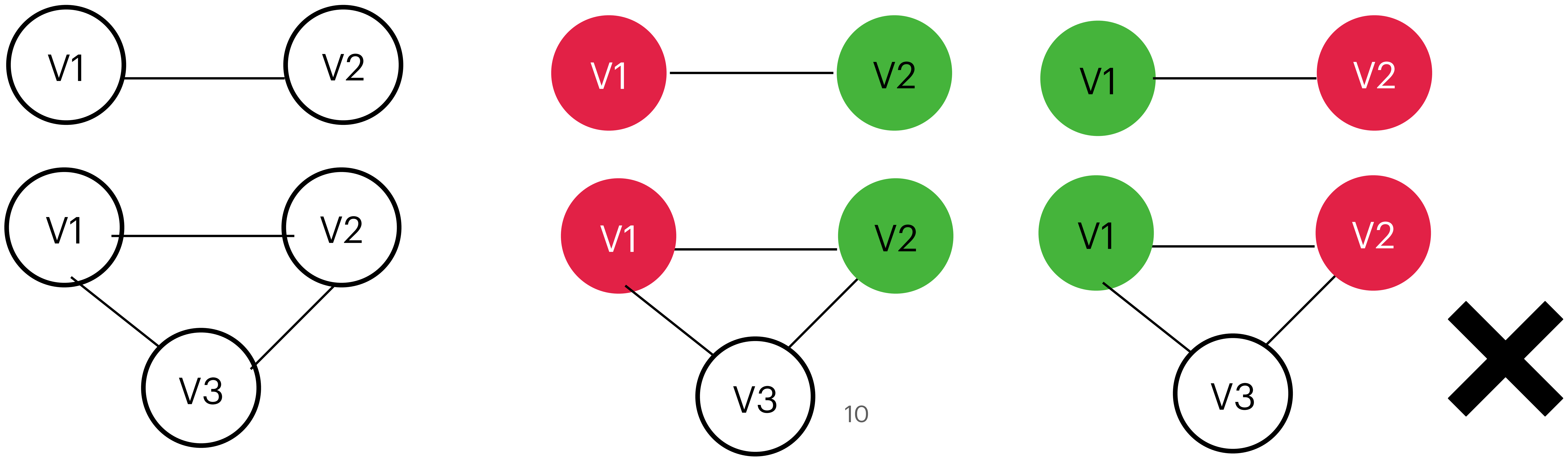
Can you convert $4 - SAT$ formula into $3 - SAT$ formula?

Can you convert $3 - SAT$ formula into $2 - SAT$ formula?

Constraint Encoding

Encoding of Graph Coloring to SAT

- Proper coloring: An assignment of colors to the vertices of a graph such that no two adjacent vertices have same color.
- K-color: A proper coloring involving a total of K colors.
- Is the following graphs 2-colorable?

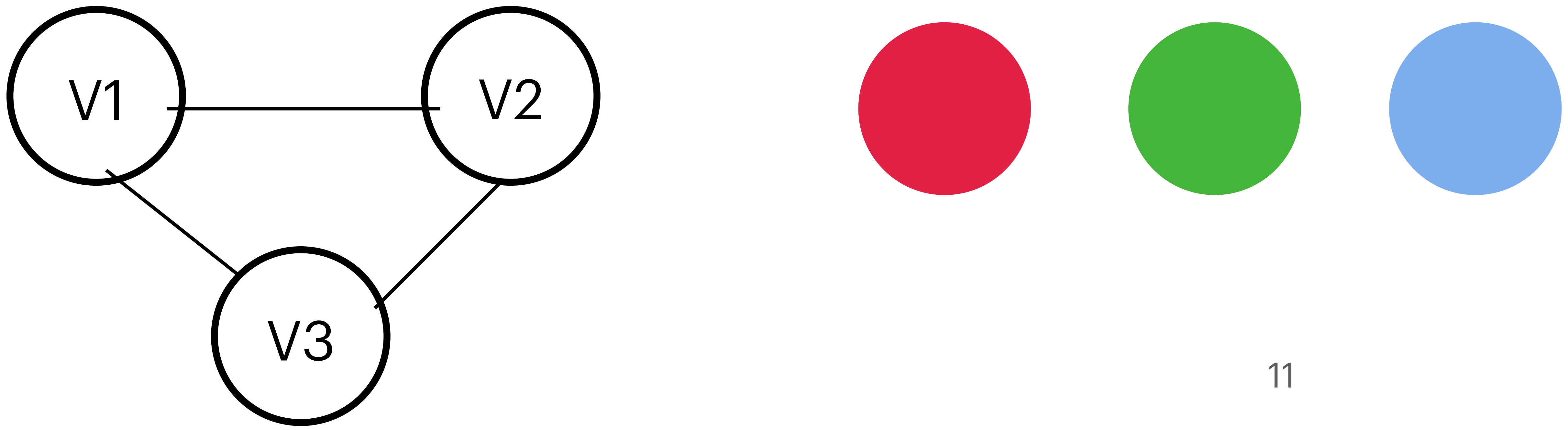


Encoding of Graph Coloring to SAT

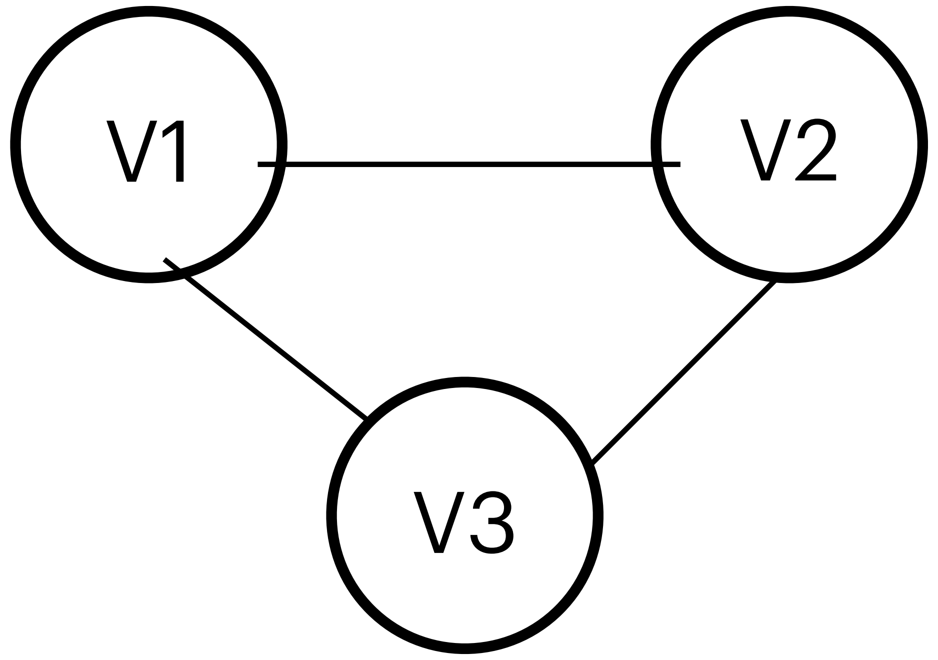
Given a graph G(V,E) with V as a set of vertices and E as a set of edges, and an integer K (representing the number of colors), can we encode the proper graph coloring into a CNF formula such that the formula is satisfiable (SAT) if and only if the graph is K-colorable.

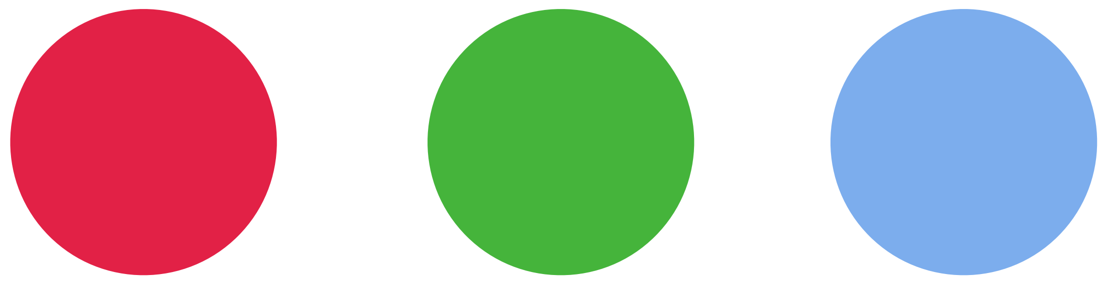
We want to encode that:

- No two adjacent vertices share the same color.
- Each vertex has exactly one color.



Step 1: Propositional Variables

- Use propositional variables $v_{i,g}$, where $i \in \{1,2,3\}, g \in \{R, G, B\}$
- $v_{i,g}$ is True, if and only if, vertex i is assigned g color.
- 



$v_{1,G}, v_{1,R}, v_{1,B}$ $v_{2,G}, v_{2,R}, v_{2,B}$ $v_{3,G}, v_{3,R}, v_{3,B}$

Step 2: Encoding Constraints

- Each vertex must have exactly one color.
    - Each vertex must have at least one color, and each vertex must have at most one color

How are we going to encode, each vertex must have at least one color:

For vertex $V_1 : v_{1,G} \vee v_{1,R} \vee v_{1,B}$ $\quad V_2 : v_{2,G} \vee v_{2,R} \vee v_{2,B}$ $\quad V_3 : v_{3,G} \vee v_{3,R} \vee v_{3,B}$

How are we going to encode, each vertex must have at most one color:

$$V_1 : (\neg v_{1,G} \vee \neg v_{1,R}) \wedge \quad V_2 : (\neg v_{2,G} \vee \neg v_{2,R}) \wedge \quad V_3 : (\neg v_{3,G} \vee \neg v_{3,R}) \wedge$$

$$(\neg v_{1,G} \vee \neg v_{1,B}) \wedge \quad (\neg v_{2,G} \vee \neg v_{2,B}) \wedge \quad (\neg v_{3,G} \vee \neg v_{3,B}) \wedge$$

$$(\neg v_{1,R} \vee \neg v_{1,B}) \wedge \quad (\neg v_{2,R} \vee \neg v_{2,B}) \wedge \quad (\neg v_{3,R} \vee \neg v_{3,B}) \wedge$$

Step 2: Encoding Constraints

- No two adjacent vertex have the same color.

For V_1 and V_2:

$(\neg v_{1,R} \vee \neg v_{2,R}) \wedge$

$(\neg v_{1,G} \vee \neg v_{2,G}) \wedge$

$(\neg v_{1,B} \vee \neg v_{2,B}) \wedge$

For V_1 and V_3:

$(\neg v_{1,R} \vee \neg v_{3,R}) \wedge$

$(\neg v_{1,G} \vee \neg v_{3,G}) \wedge$

$(\neg v_{1,B} \vee \neg v_{3,B}) \wedge$

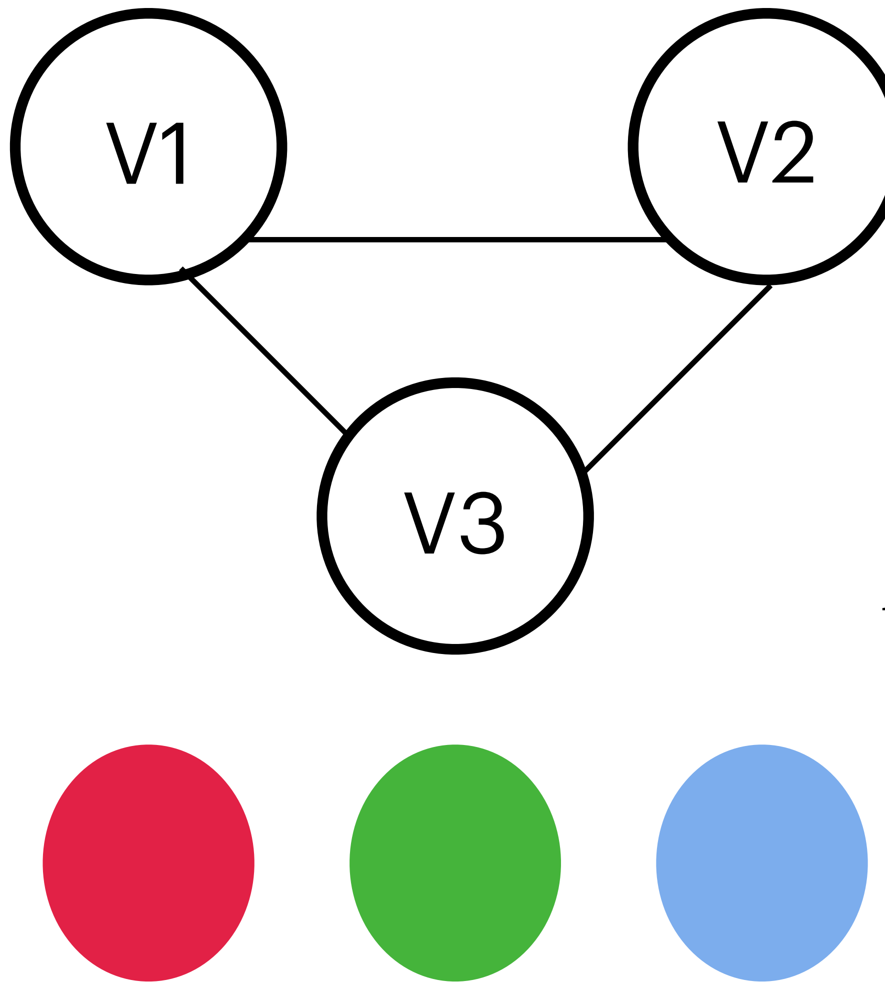
For V_2 and V_3:

$(\neg v_{2,R} \vee \neg v_{3,R}) \wedge$

$(\neg v_{2,G} \vee \neg v_{3,G}) \wedge$

$(\neg v_{2,B} \vee \neg v_{3,B})$

Proper Coloring to SAT



$$
\begin{aligned}
F_{CNF} = \; & (v_{1,G} \lor v_{1,R} \lor v_{1,B}) \land (v_{2,G} \lor v_{2,R} \lor v_{2,B}) \land (v_{3,G} \lor v_{3,R} \lor v_{3,B}) \land \\
& (\neg v_{1,G} \lor \neg v_{1,R}) \land (\neg v_{1,G} \lor \neg v_{1,B}) \land (\neg v_{1,R} \lor \neg v_{1,B}) \land \\
& (\neg v_{2,G} \lor \neg v_{2,R}) \land (\neg v_{2,G} \lor \neg v_{2,B}) \land (\neg v_{2,R} \lor \neg v_{2,B}) \land \\
& (\neg v_{3,G} \lor \neg v_{3,R}) \land (\neg v_{3,R} \lor \neg v_{3,B}) \land (\neg v_{3,R} \lor \neg v_{3,B}) \land \\
& (\neg v_{1,R} \lor \neg v_{2,R}) \land (\neg v_{1,G} \lor \neg v_{2,G}) \land (\neg v_{1,B} \lor \neg v_{2,B}) \land \\
& (\neg v_{1,R} \lor \neg v_{3,R}) \land (\neg v_{1,G} \lor \neg v_{3,G}) \land (\neg v_{1,B} \lor \neg v_{3,B}) \land \\
& (\neg v_{2,R} \lor \neg v_{3,R}) \land (\neg v_{2,G} \lor \neg v_{3,G}) \land (\neg v_{2,B} \lor \neg v_{3,B})
\end{aligned}
$$

Encoding of Pigeon Hole Principle to SAT

Theorem: If we place n+1 pigeons in n holes then there is a hole with at least 2 pigeons

Thm is true for any n; can we prove it for a fixed n using SAT solvers?

Exercise:

Encode Pigeon hole principle to a CNF formula for 3 pigeons and 2 holes.
The CNF formula should be SAT if and if 3 pigeons can fit in 2 holes, otherwise formula should be UNSAT.

Boolean /propositional formulas ——> SAT Solvers

If formula is SATisfiable, gives an satisfying assignment

UNSAT

SAT Solvers

Given a formula F, can we determine whether it is satisfiable?

Let F is over X variables, where $X = \{x_1, x_2, \dots, x_n\}$.

CheckSAT(F){

For τ in 2^n {

If $F(\tau) = 1$ then:

Return SAT, τ

}

Return UNSAT

}

Can we do better ?

We don't know!

Resolution Refutation

List of clauses $C_1, C_2, \ldots, C_t$ is a resolution refutation of formula F_{CNF} if:

1. C_t is empty $\square$
2. $C_K \in F_{CNF}$ or C_k is derived using resolution from C_i and C_j, where $i, j < k$

$Models(F) = \varnothing$
F is UNSAT

$C_i = p \vee \alpha$
$C_j = \neg p \vee \beta$
Then,
$C_k = \alpha \vee \beta$

C_k is derived from C_i, C_j

Resolution Refutation

$$F = \underbrace{(\neg p \vee \neg q \vee r)}_{C_1} \wedge \underbrace{(\neg p \vee q)}_{C_2} \wedge \underbrace{(p)}_{C_3} \wedge \underbrace{(\neg r)}_{C_4}$$

Resolution on C_1, C_3 $\quad \dfrac{(\neg p \vee \neg q \vee r) \wedge (p)}{C_5 : (\neg q \vee r)}$

Resolution on C_2, C_3 $\quad \dfrac{(\neg p \vee q) \wedge (p)}{C_6 : (q)}$

Resolution on C_5, C_4 $\quad \dfrac{(\neg q \vee r) \wedge (\neg r)}{C_7 : (\neg q)}$

Resolution on C_6, C_7 $\quad \dfrac{(q) \wedge (\neg q)}{C_8 : \square}$

List of clauses $C_1, C_2, \ldots, C_8$ is a resolution refutation of F

SAT Solving using Resolution

1. Start with F_{CNF}
2. Perform Resolution until
    1. Empty clause is derived —> return UNSAT
    2. No further resolution is possible —-> return SAT

One of these two cases will occur — resolution is sound and complete.