Boolean formula:

Variables: $x_1, x_2, \ldots, x_n$ taking values True or False

literals: $x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}$

$\wedge$ = AND
$\vee$ = OR

CNF = conjunctive normal form

clause is an OR of several literals

example: $(\overline{x_3} \vee x_5 \vee \overline{x_2} \vee x_1)$

formula f is the AND of m such clauses:

example: $(x_2) \wedge (\overline{x_3} \vee x_5 \vee \overline{x_2} \vee x_1) \wedge (\overline{x_2} \vee \overline{x_1})$

SAT:

input: Given formula f in CNF with n variables & m clauses

output: assignment satisfying f if one exists
NO if no satisfying assignment exists

k-SAT: Same as SAT but the input f has clauses with $\leq k$ literals in each.

We'll see: SAT is NP-complete
k-SAT is NP-complete for each $k \geq 3$.

Today: 2-SAT has a poly-time algorithm.

Take input f for 2-SAT.
We'll assume all clauses have size exactly 2.
What about "unit clauses" = clauses of size 1?

For f, take a unit clause say literal (a_i)
- Satisfy it (so set $a_i = T$)
- remove all clauses containing a_i (these are all satisfied)
- drop any occurrences of $\overline{a_i}$

Let f' be the resulting formula.

Observation: f is satisfiable $\Leftrightarrow$ f' is satisfiable.

Thus we can repeat the above procedure until no unit clauses remain.

Take f with clauses of size 2, n variables, m clauses

Create a directed graph:

$2n$ vertices corresponding to: $x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n$.

$2m$ edges corresponding to 2 "implications" per clause

example: $(x_i \vee \overline{x}_j)$

if $x_i = F$ then we need to set $x_j = F$ to satisfy this clause.

if $x_j = T$ then we need $x_i = T$.

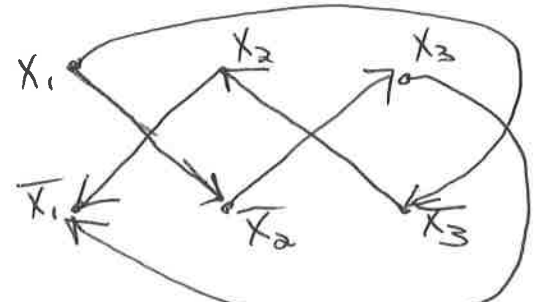
So include edges: $\overline{x}_i \rightarrow \overline{x}_j$

& $x_j \rightarrow x_i$

In general for clause $(\alpha \vee \beta)$

add edges $\overline{\alpha} \rightarrow \beta$ & $\overline{\beta} \rightarrow \alpha$

<u>Example</u>: $f = (\overline{x}_1 \vee \overline{x}_2) \wedge (x_2 \vee x_3) \wedge (\overline{x}_3 \vee \overline{x}_1)$



Note path $x_1 \rightarrow \overline{x}_2 \rightarrow x_3 \rightarrow \overline{x}_1$

So if $x_1 = T$ then need $\overline{x}_1 = T$ (i.e., $x_1 = F$)

$\Rightarrow\Leftarrow$

but if $x_1 = F$ then we can set $x_2 = T$ & $x_3 =$ anything to satisfy f.

Note: a path $x_i \rightsquigarrow x_j$ means setting $x_i = T$
we must also set $x_j = T$

Thus if there's a path $x_i \rightsquigarrow \overline{x_i}$ then we can't
set $x_i = T$ and satisfy f.
Similarly if there's a path $\overline{x_i} \rightsquigarrow x_i$ then we can't
set $x_i = F$.
Therefore if x_i & $\overline{x_i}$ are in the same SCC
then f is unsatisfiable.

Lemma: f is satisfiable $\iff$ $\forall i$, x_i & $\overline{x_i}$ are in
different SCC's.
We just argued that if $\exists i$ where x_i & $\overline{x_i}$ are in the same SCC,
then f is unsatisfiable.
Thus we proved $\Rightarrow$.
We now need to show $\Leftarrow$ which we'll do
by giving an algorithm.

Key idea:
Take a sink SCC S
– Set S to T (by satisfying all literals in S)
Note: S has no outgoing edges so no implications
from setting $S = T$.
But we've set $\overline{S} = F$ (does $\overline{S}$ have incoming
edges?)

Take a source SCC S'

- Set S' to F

Note: S' has no incoming edges so we'll never be forced to set S' to T.

But: we've set $\overline{S'} = T$ (Does $\overline{S'}$ have outgoing edges?)

Lemma: S is a sink SCC $\Leftrightarrow$ $\overline{S}$ is a source SCC.

So we take a sink SCC S:

- Set $S = T$ (and thus $\overline{S} = F$)
- Remove S & $\overline{S}$
- Repeat until empty graph.

This is valid because no variable x_i has x_i & $\overline{x_i}$ in the same SCC.

Just need to prove the lemma.

Claim: Path $\alpha \rightsquigarrow \beta \iff$ Path $\overline{\beta} \rightsquigarrow \overline{\alpha}$

Proof: Let's prove $\Rightarrow$. Take path $\alpha \rightsquigarrow \beta$ say it's:

$$\gamma_0 \to \gamma_1 \to \cdots \to \gamma_\ell \text{ where } \begin{array}{l}\gamma_0 = \alpha \\ \gamma_\ell = \beta\end{array}$$

edge $\gamma_i \to \gamma_{i+1}$ comes from

clause $(\overline{\gamma_i} \vee \gamma_{i+1})$ and the

otheredge is $\overline{\gamma_{i+1}} \to \overline{\gamma_i}$

Thus we have the path

$$\overline{\gamma_\ell} \to \overline{\gamma_{\ell-1}} \to \cdots \to \overline{\gamma_0} \text{ where } \begin{array}{l}\overline{\gamma_\ell} = \overline{\beta} \\ \overline{\gamma_0} = \overline{\alpha}\end{array}$$

For $\Leftarrow$, Similarly from a path $\overline{\beta} \rightsquigarrow \overline{\alpha}$ we can construct a path $\alpha \rightsquigarrow \beta$. $\blacksquare$

Take SCC S.

For $\alpha, \beta \in S$, there's paths $\alpha \rightsquigarrow \beta$ & $\beta \rightsquigarrow \alpha$

Thus there's also paths $\overline{\beta} \rightsquigarrow \overline{\alpha}$ & $\overline{\alpha} \rightsquigarrow \overline{\beta}$

and so $\overline{\alpha}$ & $\overline{\beta}$ are in the same SCC $\overline{S}$.

Therefore $\overline{S}$ is a SCC.

So S is a SCC $\iff$ $\overline{S}$ is a SCC.

And $\overline{S}$ has no incoming edges (so source SCC)

iff S has no outgoing edges (sink SCC).

This proves the lemma.