Graphs: 2-Satisfiability

SAT Notation

1. Satisfiability: SAT problem
   - Boolean formula: n variables $x_1, x_2, \ldots, x_n$
   - 2n literals x_1, !x_1, x_2, !$x_2 \ldots$, x_n, !x_n
   - v = AND, ˆ = OR
   - CNF = conjunctive normal form
   - Clause: OR of several literals (x_3 v !x_5 v $_x$1~ v x_2)
   - f in CNF is AND of m clauses
     - (x_2) ˆ (!x_3 v x_4) ˆ (x_3 v !x_5 v !x_1 v x_2) ˆ (!x_2 v !x_1)
     - x_1 = F, x_2 = T, x_3 = F
   - Any formula can be converted to CNF form, but the formula might be large

SAT Problem

1. SAT:
   - Input: formula f in CNF with n variables and m clauses
   - Output: Assignment (T or F to each variable) satisfying if one exists
     - NO if none exists

SAT Problem Quiz

1. f = (!x_1 v !x_2 v x_3) ˆ (x_2 v x_3) ˆ (!x_3 v !x_1) ˆ (!x_3)
   - x1 = False, x2 = True, x3 = False

k-SAT

1. k-SAT:
   - Input: formula f in CNF with n variables and m clauses each of size <= k
     - Size of clause = number of literals
   - Output: Assignment (T or F to each variable) satisfying if one exists
     - NO if none exists
2. SAT is NP-complete
   - k-SAT is NP-complete for all k >= 3
   - Poly-time algorithm for 2-SAT

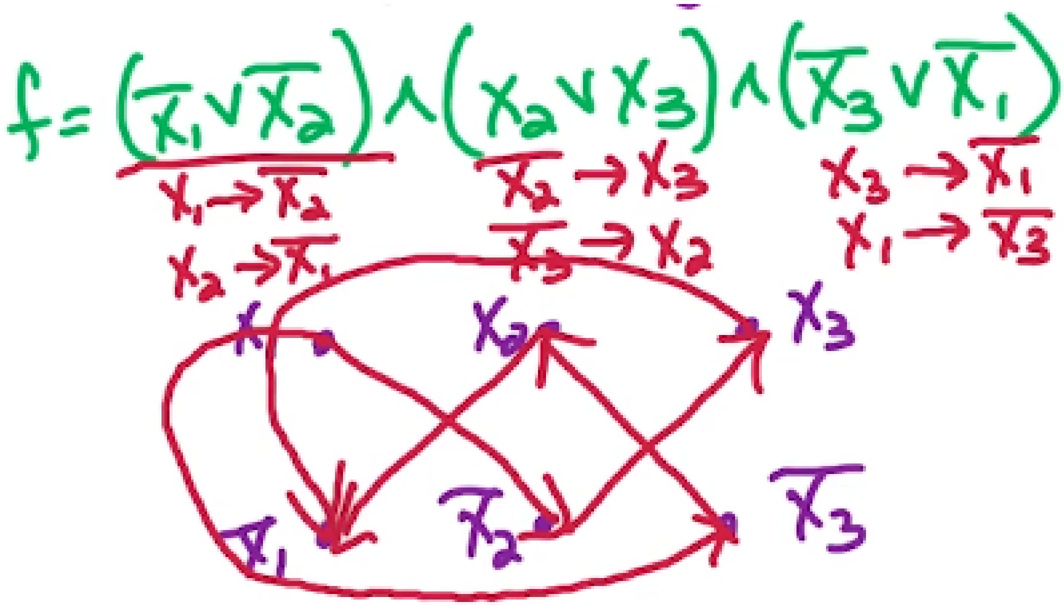
Simplifying Input

1. Consider input f for 2-SAT
   - Simplify: Unit clause = clause with 1 literal
     - Take a unit clause, say literal a_i
     - Satisfy it (set a_i = T)
     - Remove clauses containing a_i
     - Let f' be the resulting formula
   - Observation: f is satisfiable <=> f' is satisfiable
     - Can assume all clauses of size = 2

Graph of Implications

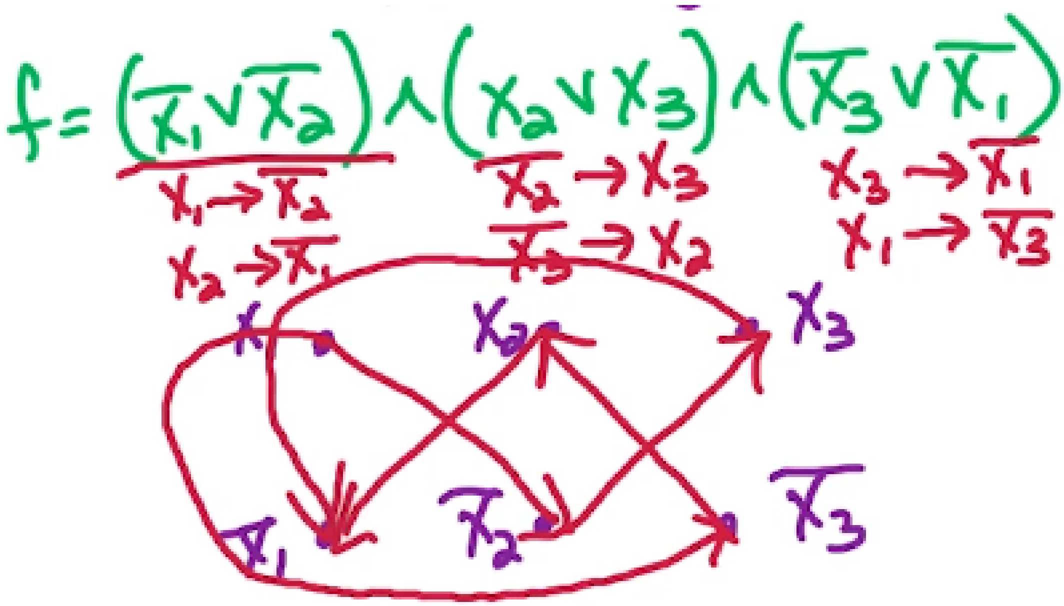
1. Take f with all clauses of size = 2, n variables m clauses
2. Create a directed graph:
   - 2n vertices corresponding to x_1, !x_1, $\ldots$, x_n, !x_n
   - 2m edges corresponding to 2 "implications" per clause

3. Example
    - f = (!x_1 v x_2) ^ (x_2 v x_3) ^ (x_3 v x_1)
        - x_1 = T -> x_2 = F
        - x_2 = T -> x_1 = F
    - (A v B)
        - !A -> B
        - !B -> A



Graph of Implications

Graph Properties



Graph of Implications

1. Path x_1 -> !x_2 -> x_3 -> !x_1
    - If x_1 = T then x_1 = F (contradiction)
    - If x_1 = F, so it might be okay?
2. If paths x_1 -> !x_1 and !x_1 -> x_1, then f is not satisfiable
    - This means x_1 and !x_1 are in the same SCC

SCC's

1. Lemma: If for some i, x_i and !x_i are in the same SCC, then f is not satisfiable
2. Lemma: If for all i, x_i and !x_i are in different SCC's, then f is satisfiable
    - Prove f is satisfiable by finding a satisfying assignment

Algorithm Idea

1. Approach 1:
    - Take sink SCC S
    - Set S = T (satisfy all literals in S)
    - Remove S and repeat
        - By satisfying the literals in S, we are not satisfying their complements
        - If the complement set is a source SCC, we set it to False

Algorithm Idea 2

1. Approach 2:
    - Take source SCC S'
        - Set S' = F
    - Complement of S' is a sink SCC
        - By setting S' = F, !S' = T
2. Can take sink SCC and set it to true or a source SCC and set it to F
    - These approaches are equivalent

2SAT Algorithm

1. Key fact: If for all i, x_i and $!x_i$ are in different SCC's
   - S is a sink SCC <=> !S is a source SCC
2. Runtime: O(n+m)

```
2SAT(f)
    Construct graph G for f
    Take a sink SCC S
        set S = T (and !S = F)
        remove S and !S
        repeat until empty
```

Proof of Key Fact

1. Key fact: If for all i, x_i and $!x_i$ are in different SCC's
   - S is a sink SCC <=> !S is a source SCC
2. Simpler claim: A -> B <=> !B -> !A
3. Proof of key fact:
   - Take sink SCC S
   - For A, B in S, we have paths A -> B and B -> A
     - Also have paths !B -> !A and !A -> B
   - S is a SCC <=> !S is a SCC

Rest of Proof

1. For A in S, no edges A -> B => No edges !B -> !A
   - No outgoing from A => no incoming to !A
   - Therefore !S is a source
   - S is a SCC <=> !S is a SCC

Proof of Claim (Satisfiability)

1. Take path A -> B, say:
   - g_0 -> g_1 -> g_2 -> . . . -> g_l
   - g_1 -> g_2 comes from ($!g_1$ v g_2)