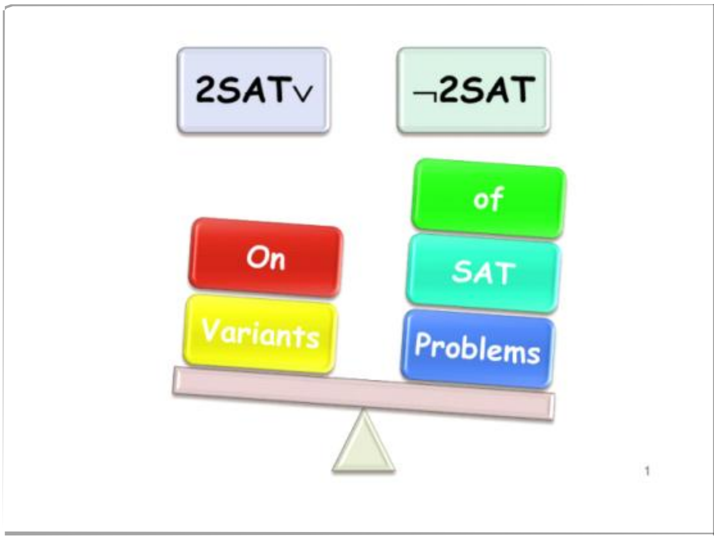
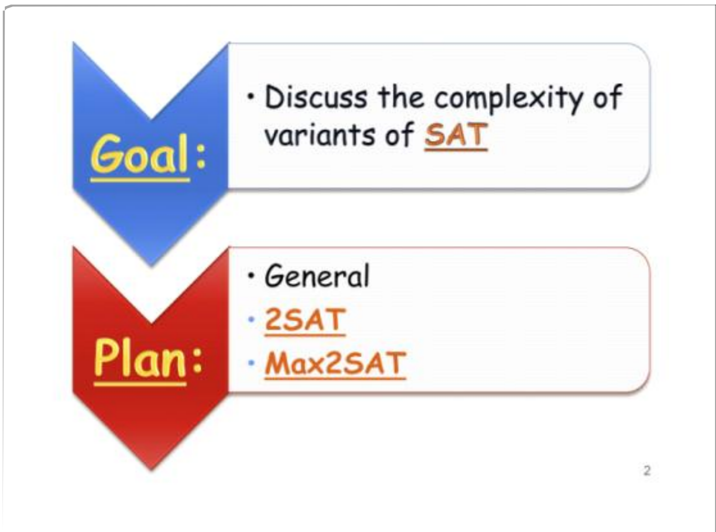
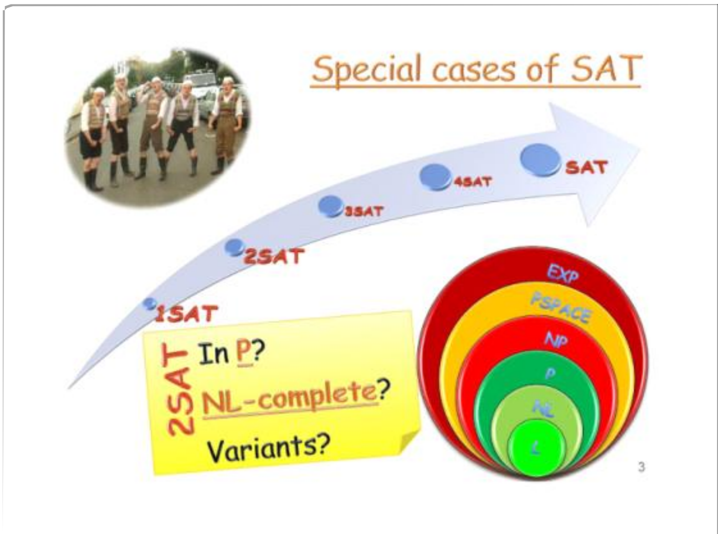
2SAT Notes



Let us now investigate some variants of the SAT problem and see how various *parameters* may *affect* the *complexity* of a problem.



In particular, we study the *2SAT problem* and its optimization version.

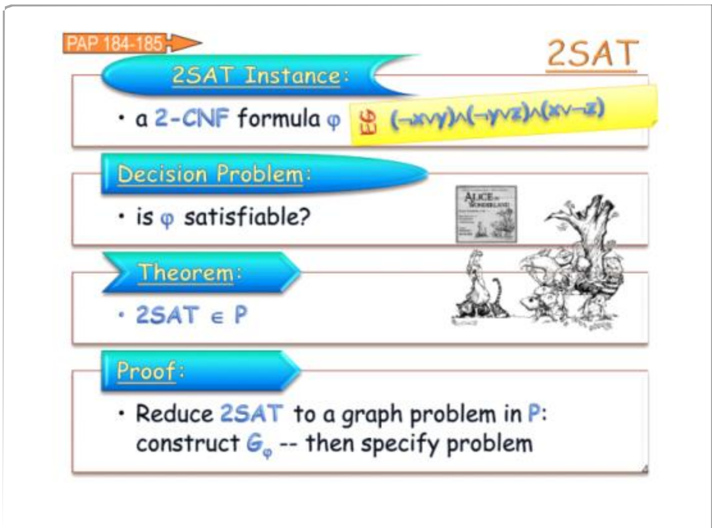


We can limit the *number* of *literals* in each *clause* of the SAT problem.

We have already seen that with 3 literals the problem is NP-complete.

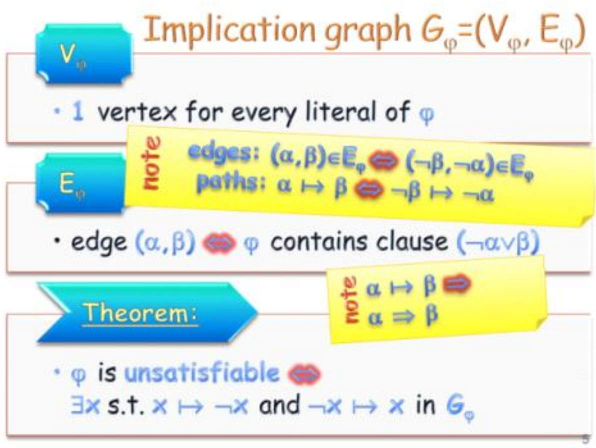
This is clearly still the case when the number of literals is even larger.

What about 2?



A 2SAT instance is a *2-CNF formula*, which is good (accepted) if it can be satisfied by some assignment to its variables.

We'll now show *it is in P* by *reducing* it to a graph problem in P.



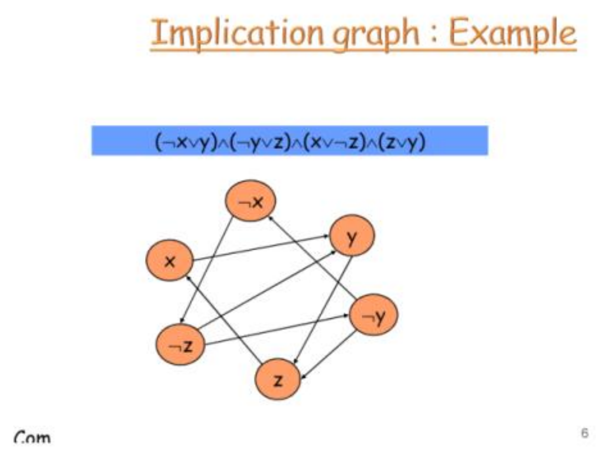
A problematic cycle is a cycle that contains both x and ¬x for some variable x.

We'll now prove that the formula can be satisfied if and only if its implication graph has no problematic cycle.

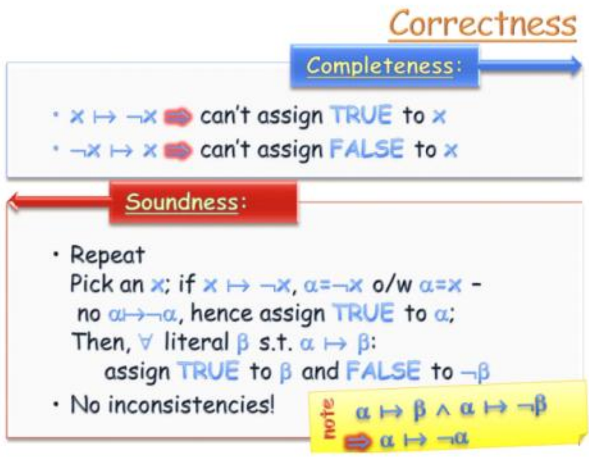
An *implication graph* for a given formula has a vertex for every literal of the formula. Namely, 2 for each variable.

For a clause to be satisfied, if one of its literals is FALSE, the other one must be TRUE. Edges in the graph correspond to such restrictions, i.e., 2 for each clause.

To satisfy all clauses, if a literal is assigned TRUE, all the outgoing edges from its corresponding vertex, enforce the adjacent literals to be TRUE as well.



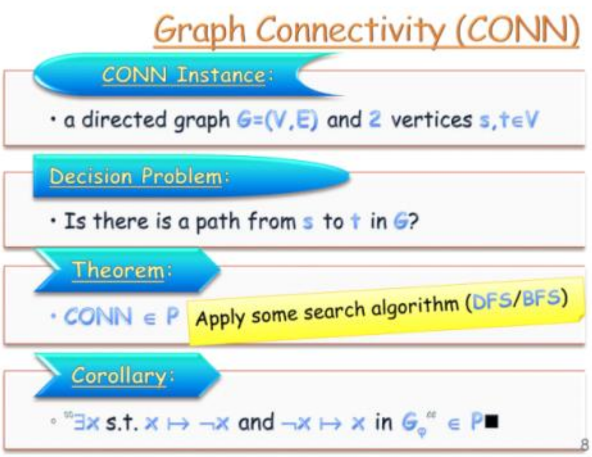
Here is an example of the implication graph of a simple formula.



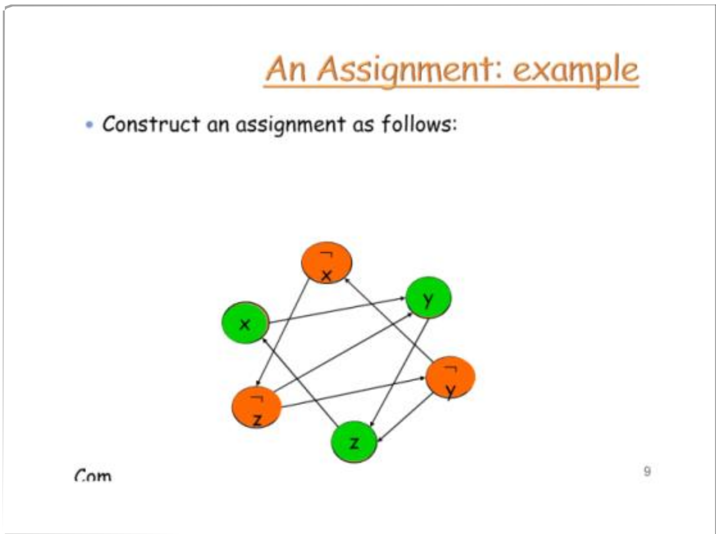
The completeness proof is straightforward: if a problematic cycle contains x and ¬x, then x can be assigned neither TRUE nor FALSE. Hence a formula that can be satisfied results in a graph with no problematic cycles.

As to soundness, with no problematic cycles, one can construct a satisfying assignment: Assign an arbitrary x with a value that is not contradicted by a path from x to ¬x or vice versa.
Now assign all values that are implied by this.
With no problematic cycles there can be no contradictions in this process.

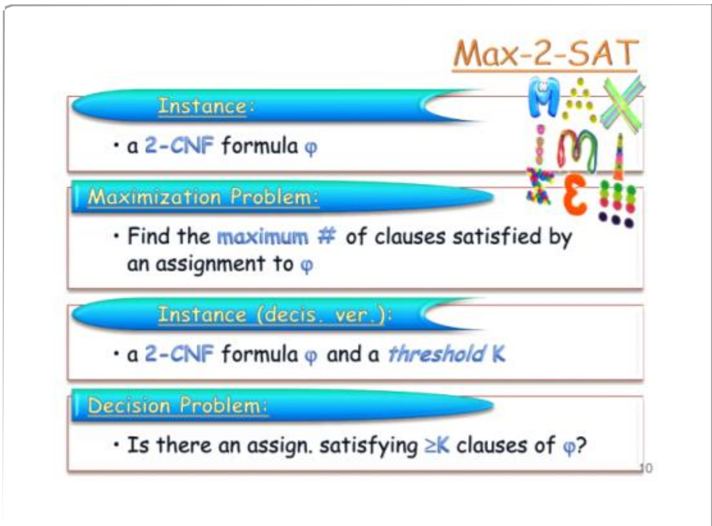
If not done, pick another variable and proceed in the same manner.



The property we have reduced our problem to, constitutes of a set of *connectivity problems*.
Therefore, it is *in P*.



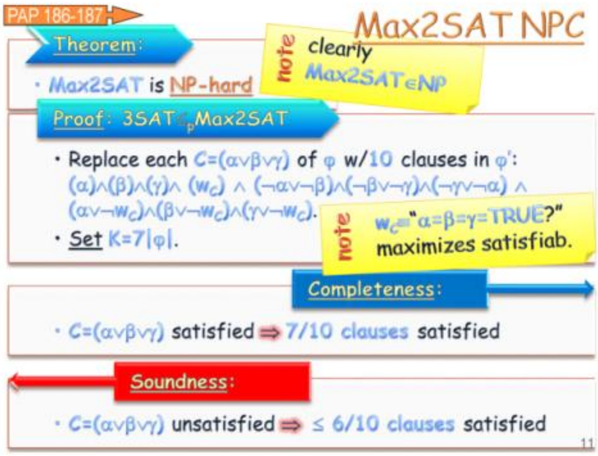
Here is an example of the assignment being constructed.



Now consider the problem of, given a 2-CNF formula, *maximize* the number of clauses satisfied.

To better analyze this problem we translate it into a decision problem. We add to the input a threshold K, and simply ask whether there exists an assignment that satisfies at least K of the clauses.

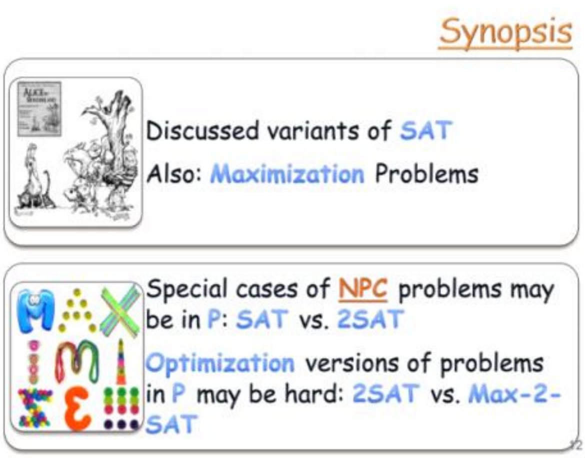
We call this problem *Max-2-SAT*.



This problem is shown to be *NP-hard* by a simple reduction from 3SAT.

Let us replace every clause by 10 clauses (plus an extra variable), so that at the most 7 of the 10 can be satisfied. And so that, in case the original clause is satisfied, exactly 7 of the 10 can be satisfied. While, in case the original clause is not satisfied, exactly 6 clauses can be satisfied.

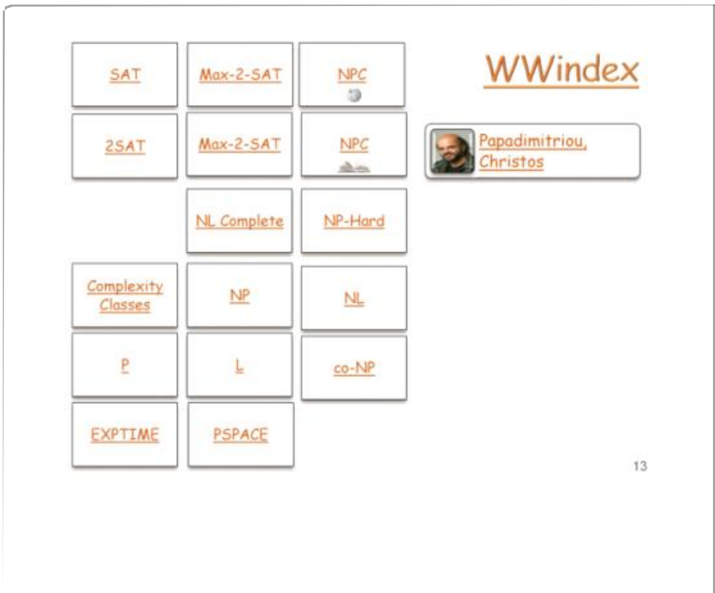
To be convinced this is indeed the case, note that setting the auxiliary variable to TRUE only if all 3 literals are TRUE maximized the number of satisfied clauses. Now consider cases according to the number of literals TRUE in the original clause.



We have discussed some SAT problems.

More importantly, we have introduced the notion of an *optimization* problem, and have investigated the complexity of the optimization version of 2SAT.

We'll come back to this notion later and study it extensively.



- SAT
- Max-2-SAT
- NPC
- 2SAT
- NL Complete
- Complexity Classes
- P
- NP
- co-NP
- EXPTIME
- L
- NL
- PSPACE
- NPC
- NP-Hard
- Max-2-SAT