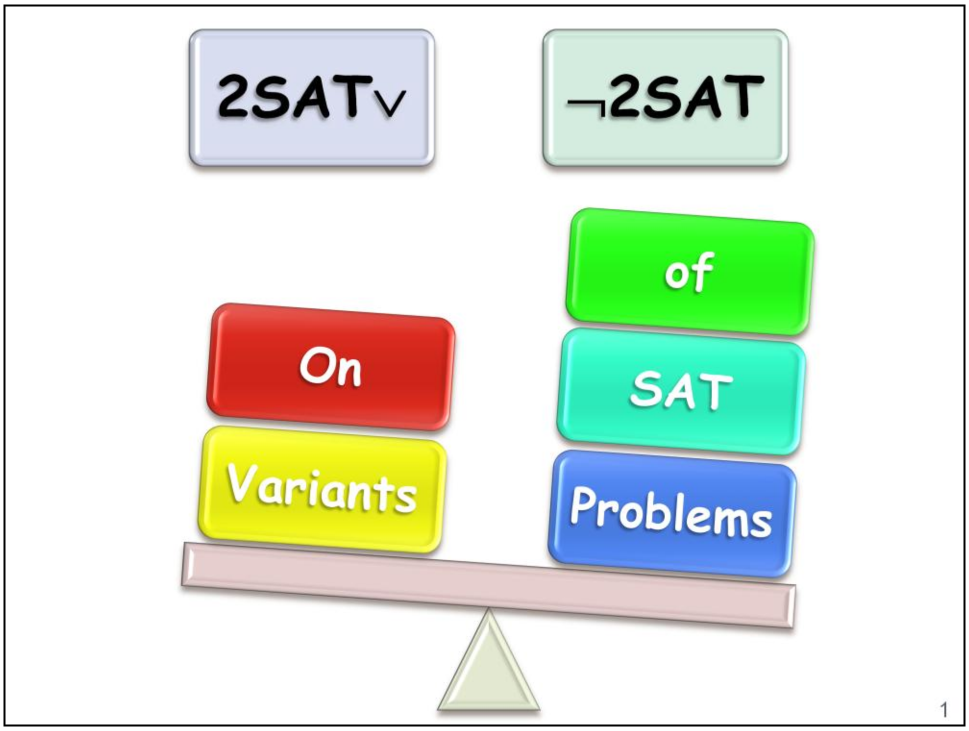
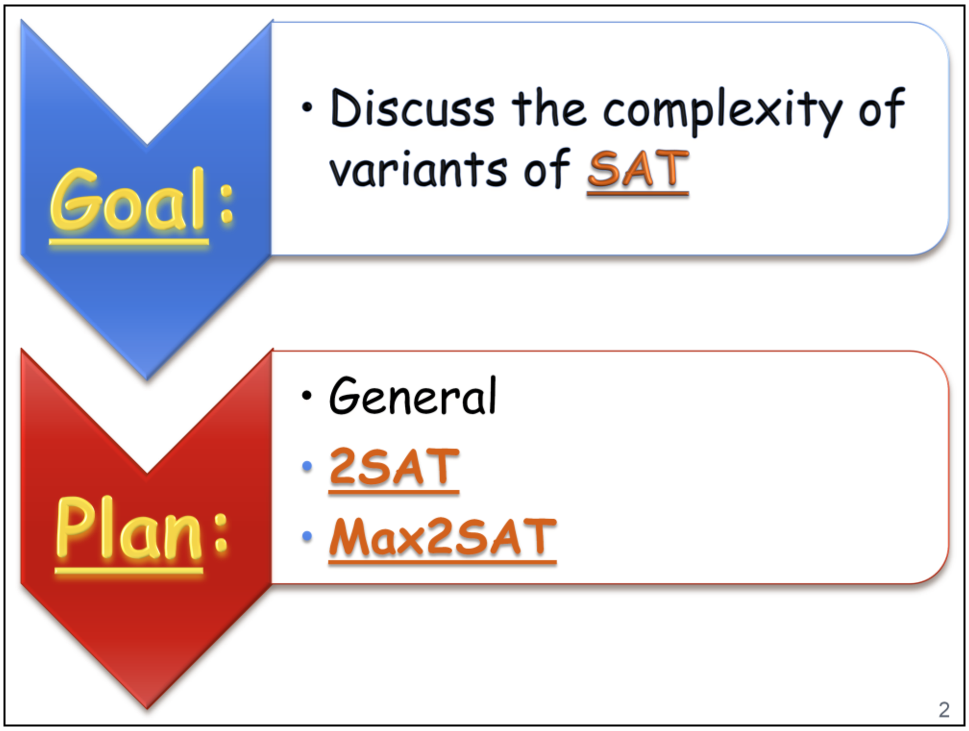


Let us now investigate some variants of the SAT problem and see how different parameters may affect the complexity of a problem.



Goal:

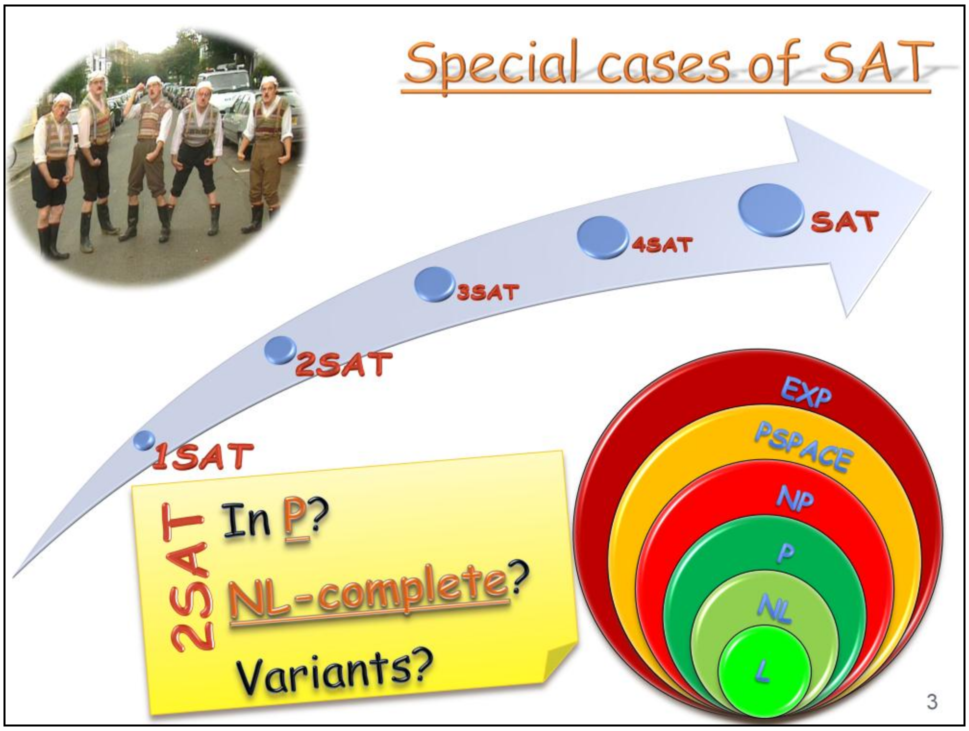
- Discuss the complexity of variants of SAT

Plan:

- General
- 2SAT
- Max2SAT

2

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



Special cases of SAT

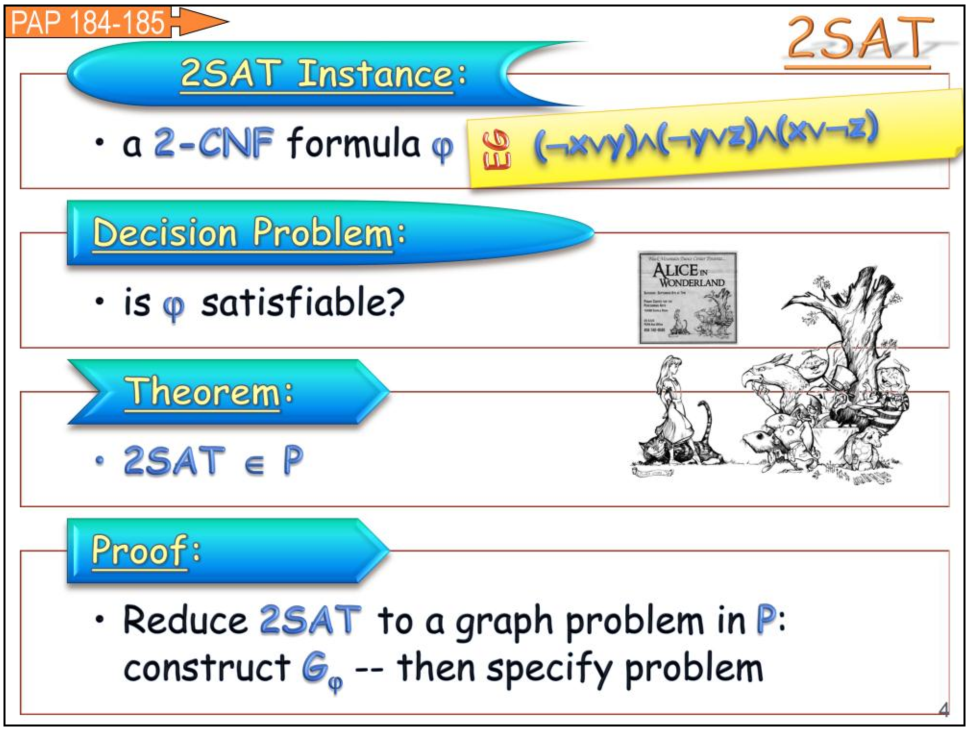
1SAT
2SAT
3SAT
4SAT
SAT
EXP
PSPACE
NP
P
NL
L

2SAT

- In P?
- NL-complete?
- Variants?

3

We can limit the number of literals in each clause of the SAT problem. We have already seen that with 3 literals the problem becomes NP-complete. This is clearly still the case when the number of literals is even larger. What about 2?



PAP 184-185

2SAT

2SAT Instance:

- a 2-CNF formula φ

EG $(\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z)$

Decision Problem:

- is φ satisfiable?

Theorem:

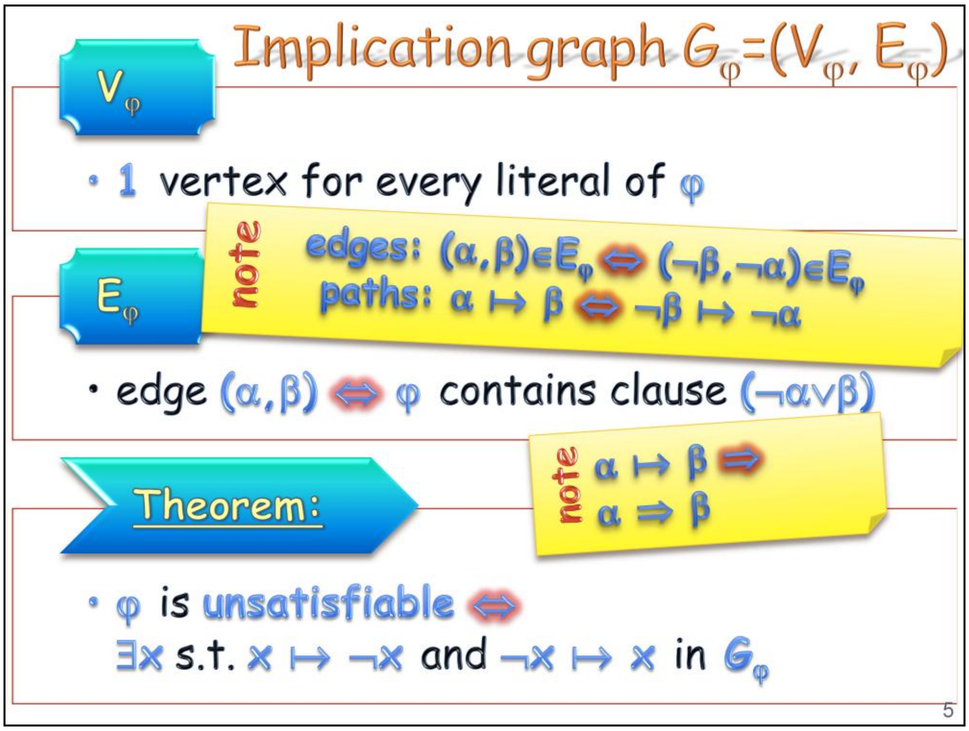
- 2SAT $\in$ P

Proof:

- Reduce 2SAT to a graph problem in P: construct G_φ -- then specify problem

4

At 2SAT instance is a 2-CNF formula, which is good if it can be satisfied. We’ll now show how it is in P by reducing it to a graph problem in P.



Implication graph $G_\varphi=(V_\varphi, E_\varphi)$

V_φ

- 1 vertex for every literal of φ

note: edges: $(\alpha,\beta)\in E_\varphi \Leftrightarrow (\neg\beta,\neg\alpha)\in E_\varphi$
paths: $\alpha \mapsto \beta \Leftrightarrow \neg\beta \mapsto \neg\alpha$

E_φ

- edge $(\alpha,\beta) \Leftrightarrow \varphi$ contains clause $(\neg\alpha\vee\beta)$

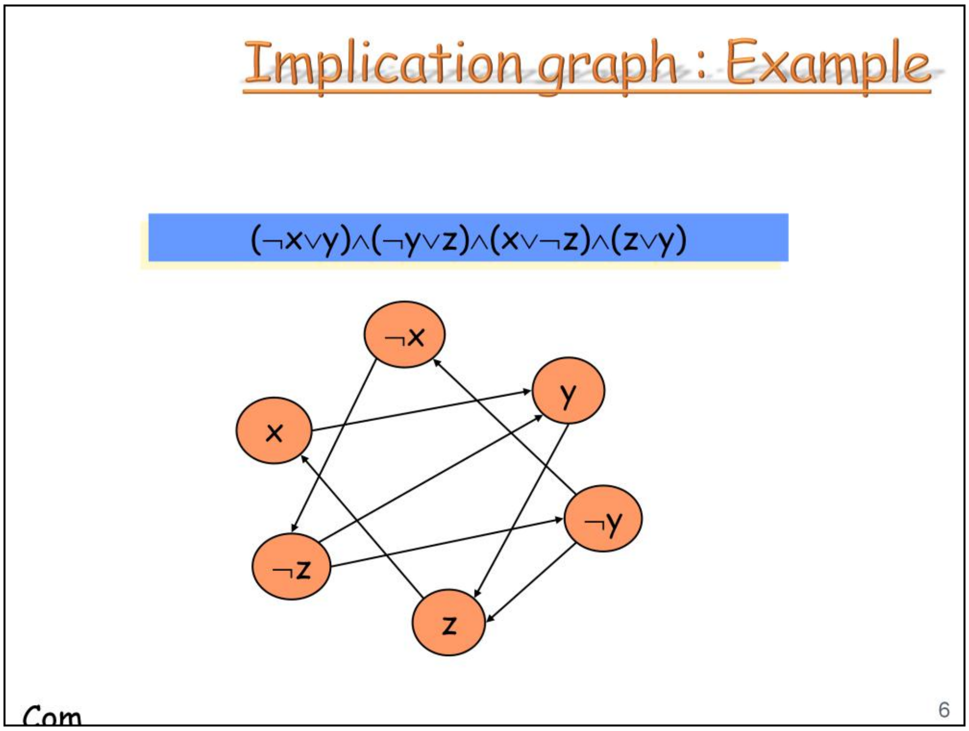
note: $\alpha \mapsto \beta \Rightarrow$
$\alpha \Rightarrow \beta$

Theorem:

- φ is unsatisfiable $\Leftrightarrow$
  $\exists x$ s.t. $x \mapsto \neg x$ and $\neg x \mapsto x$ in G_φ

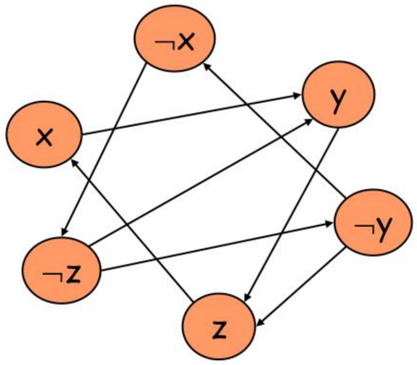
5

An *implication graph* for a formula has 1 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. That is, 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. The variable is therefore *problematic* if there is a path from it to its negation, and then from its negation back to it. We'll now prove that the formula can be satisfied if and only if its implication graph has no problematic variables.



Implication graph : Example

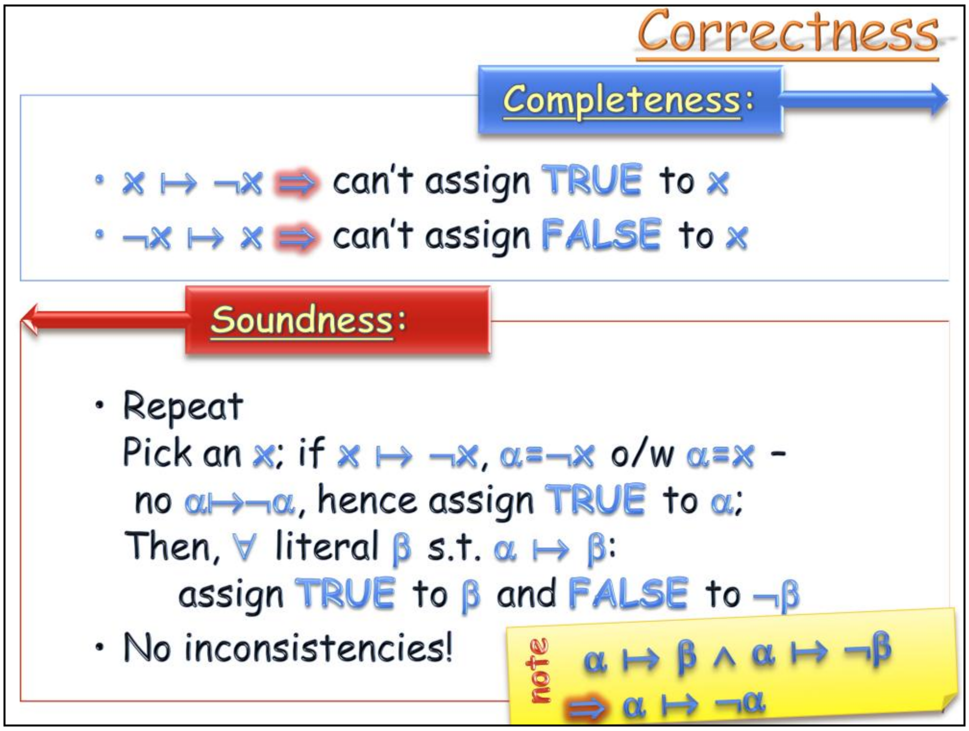
$(\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z) \wedge (z \vee y)$



Com

6

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



Correctness

Completeness:

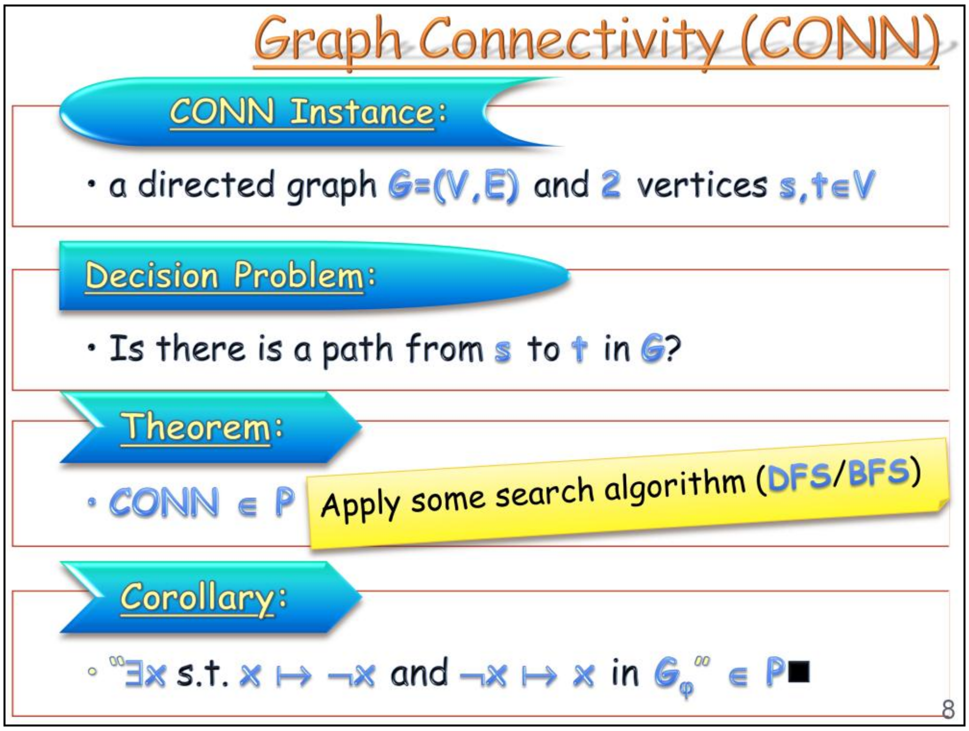
- $x \mapsto \neg x \Rightarrow$ can't assign TRUE to x
- $\neg x \mapsto x \Rightarrow$ can't assign FALSE to x

Soundness:

- Repeat
  Pick an x; if $x \mapsto \neg x$, $\alpha = \neg x$ o/w $\alpha = x$ -
  no $\alpha \mapsto \neg\alpha$, hence assign TRUE to α;
  Then, $\forall$ literal β s.t. $\alpha \mapsto \beta$:
  assign TRUE to β and FALSE to $\neg\beta$
- No inconsistencies!

note $\alpha \mapsto \beta \wedge \alpha \mapsto \neg\beta$
$\Rightarrow \alpha \mapsto \neg\alpha$

The completeness proof is straightforward: if a variable is problematic, it can be assigned neither TRUE nor FALSE. Hence a formula that can be satisfied results in a graph with no problematic variables. As to soundness, with no problematic variables, one can construct a satisfying assignment: Assign an arbitrary variable with a value that is not contradicted by a path from it to its negation or vice versa. Now assign all values that are implied by this. With no problematic variables there can be no contradictions in this process. If not done, pick another variable and proceed in the same manner.



Graph Connectivity (CONN)

CONN Instance:

- a directed graph $G=(V,E)$ and 2 vertices $s,t \in V$

Decision Problem:

- Is there is a path from s to t in G?

Theorem:

- CONN $\in$ P

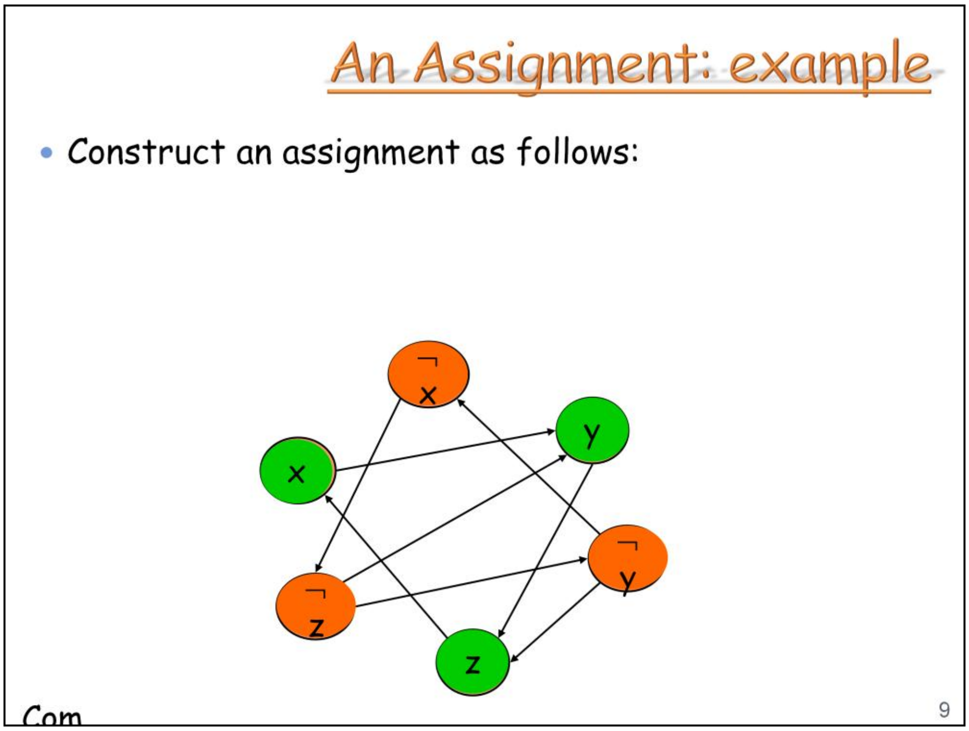
Apply some search algorithm (DFS/BFS)

Corollary:

- "$\exists x$ s.t. $x \mapsto \neg x$ and $\neg x \mapsto x$ in G_φ" $\in$ P■

8

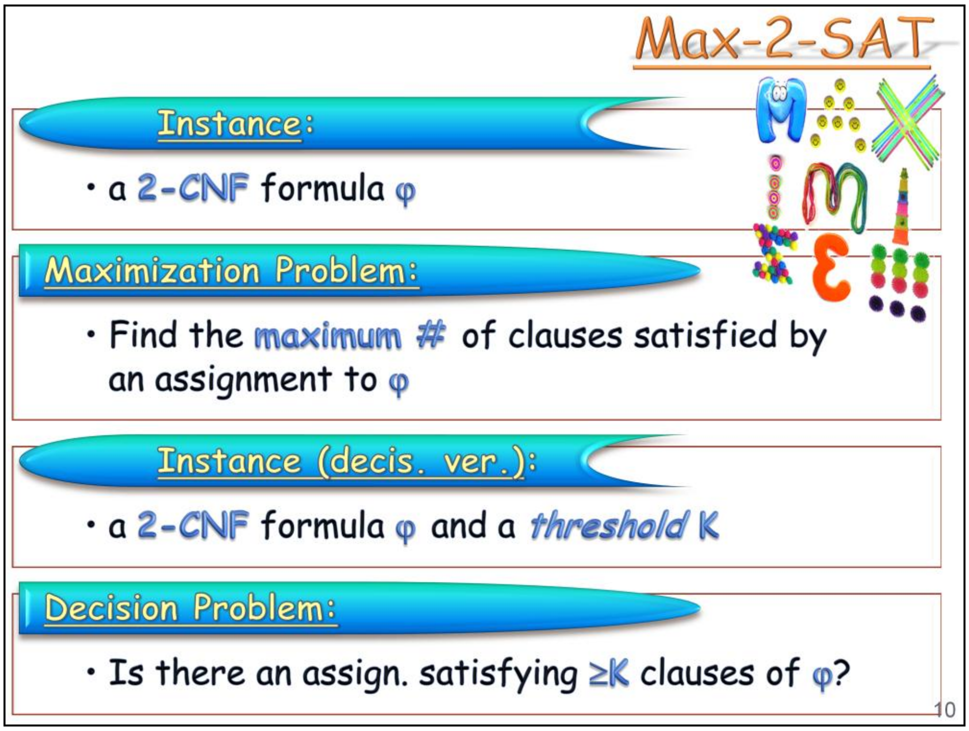
The property we have reduced our problem to, constitutes of a set of connectivity problems. Therefore, it is in P.



An Assignment: example

- Construct an assignment as follows:

Here is an example of their assignment being constructed.



Max-2-SAT

Instance:

- a 2-CNF formula φ

Maximization Problem:

- Find the maximum # of clauses satisfied by an assignment to φ

Instance (decis. ver.):

- a 2-CNF formula φ and a *threshold* K

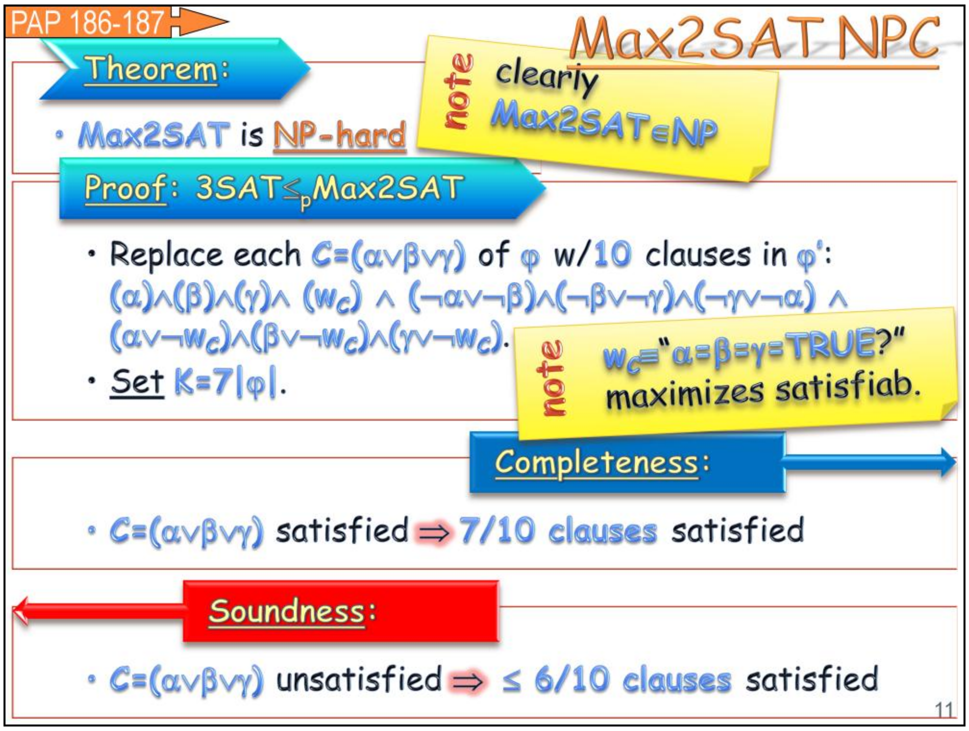
Decision Problem:

- Is there an assign. satisfying ≥K clauses of φ?

10

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.


PAP 186-187

Max2SAT NPC

Theorem:

- Max2SAT is NP-hard

note: clearly $Max2SAT \in NP$

Proof: $3SAT \leq_p Max2SAT$

- Replace each $C=(\alpha \vee \beta \vee \gamma)$ of φ w/10 clauses in φ': $(\alpha) \wedge (\beta) \wedge (\gamma) \wedge (w_C) \wedge (\neg\alpha \vee \neg\beta) \wedge (\neg\beta \vee \neg\gamma) \wedge (\neg\gamma \vee \neg\alpha) \wedge (\alpha \vee \neg w_C) \wedge (\beta \vee \neg w_C) \wedge (\gamma \vee \neg w_C)$.
- Set $K=7|\varphi|$.

note: $w_C \equiv$ "$\alpha=\beta=\gamma=$TRUE?" maximizes satisfiab.

Completeness:

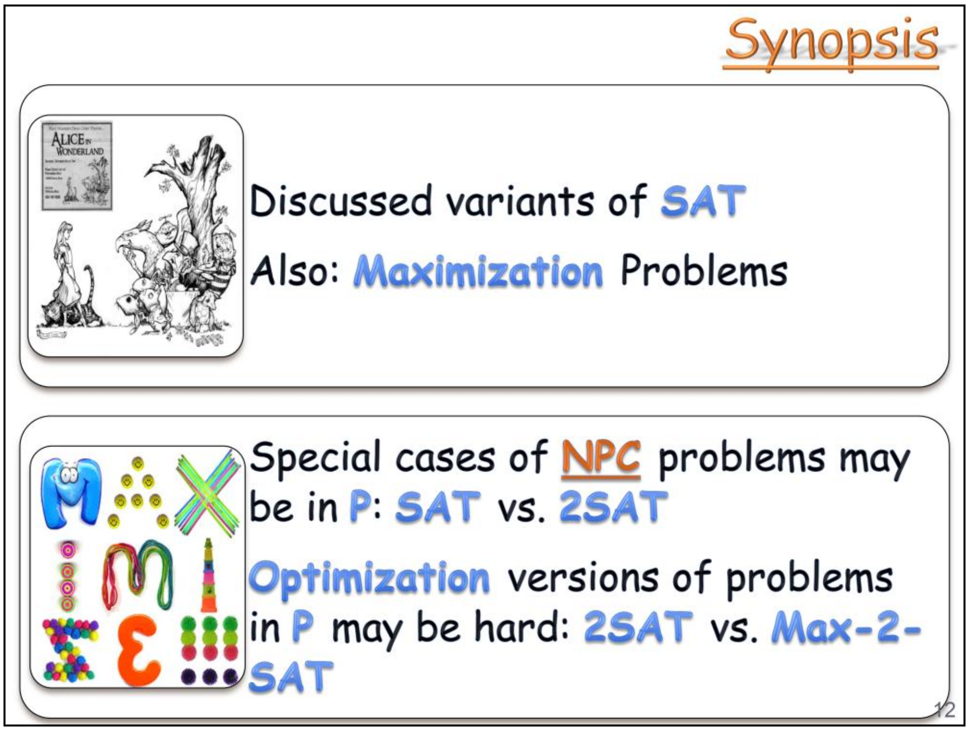
- $C=(\alpha \vee \beta \vee \gamma)$ satisfied $\Rightarrow$ 7/10 clauses satisfied

Soundness:

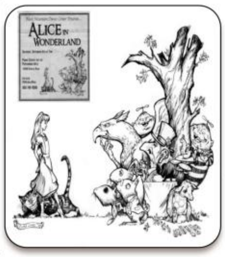
- $C=(\alpha \vee \beta \vee \gamma)$ unsatisfied $\Rightarrow$ $\leq$ 6/10 clauses satisfied

11

This problem is shown to be NP-hard by a simple reduction from 3SAT. We replace every clause by 10 clauses, so that at the most seven can be satisfied, with an extra variable for each clause. In case the original clause is satisfied, exactly seven of the ten clauses can be satisfied. In case the original clause is not satisfied, exactly six 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.



Synopsis



Discussed variants of SAT

Also: Maximization Problems

Special cases of NPC problems may be in P: SAT vs. 2SAT

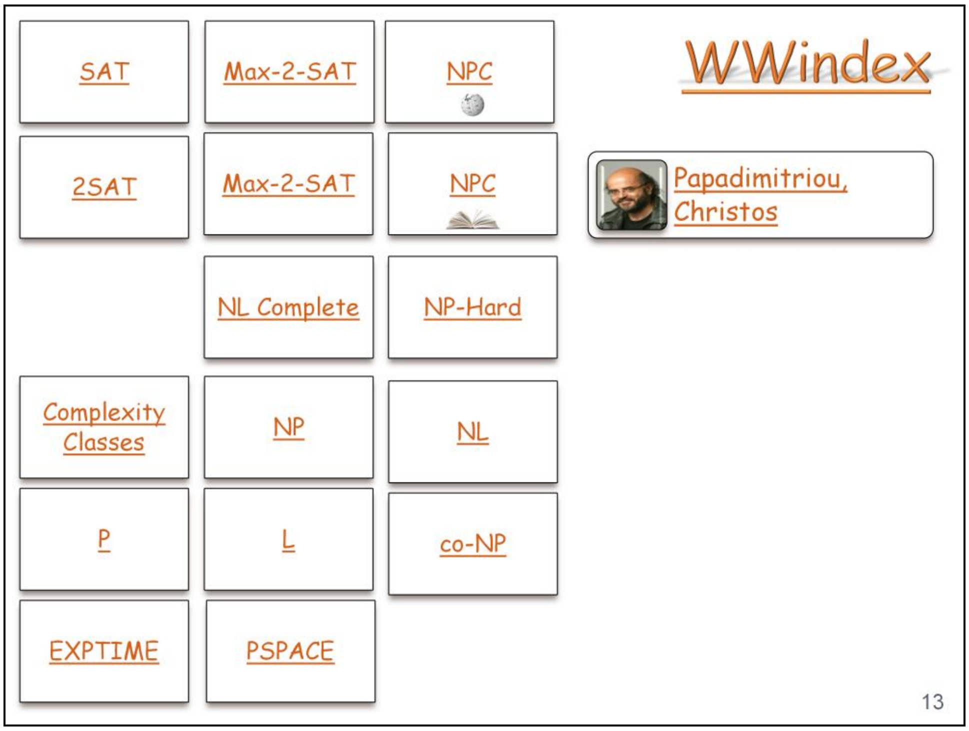
Optimization versions of problems in P may be hard: 2SAT vs. Max-2-SAT

12

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.



WWindex

SAT

Max-2-SAT

NPC

2SAT

Max-2-SAT

NPC

NL Complete

NP-Hard

Complexity Classes

NP

NL

P

L

co-NP

EXPTIME

PSPACE

Papadimitriou, Christos

13