

# Theory of Computer Science

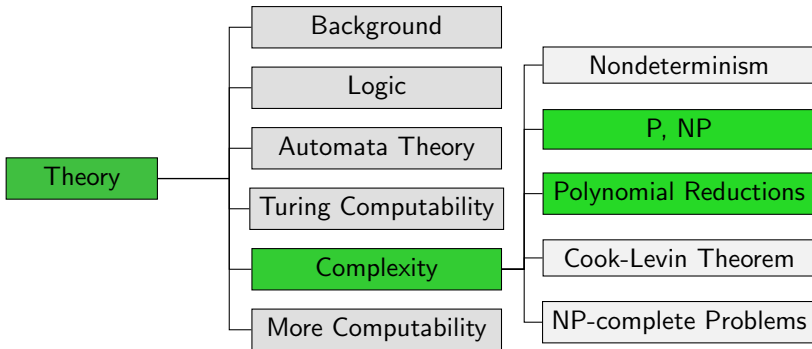
## E3. Proving NP-Completeness

Gabriele Röger

University of Basel

May 6, 2019

# Course Overview



# Overview

# Proving NP-Completeness by Reduction

- Suppose we know one NP-complete problem (we will use satisfiability of propositional logic formulas).
- With its help, we can then prove quite easily that **further problems** are NP-complete.

# Proving NP-Completeness by Reduction

- Suppose we know one NP-complete problem (we will use satisfiability of propositional logic formulas).
- With its help, we can then prove quite easily that **further problems** are NP-complete.

## Theorem (Proving NP-Completeness by Reduction)

*Let  $A$  and  $B$  be problems such that:*

- *$A$  is NP-hard, and*
- *$A \leq_p B$ .*

*Then  $B$  is also NP-hard.*

*If furthermore  $B \in NP$ , then  $B$  is NP-complete.*

# Proving NP-Completeness by Reduction: Proof

Proof.

First part: We must show  $X \leq_p B$  for all  $X \in \text{NP}$ .

# Proving NP-Completeness by Reduction: Proof

## Proof.

**First part:** We must show  $X \leq_p B$  for all  $X \in \text{NP}$ .

From  $X \leq_p A$  (because  $A$  is NP-hard) and  $A \leq_p B$  (by prerequisite), this follows due to the transitivity of  $\leq_p$ .

# Proving NP-Completeness by Reduction: Proof

## Proof.

**First part:** We must show  $X \leq_p B$  for all  $X \in \text{NP}$ .

From  $X \leq_p A$  (because  $A$  is NP-hard) and  $A \leq_p B$  (by prerequisite), this follows due to the transitivity of  $\leq_p$ .

**Second part:** follows directly by definition of NP-completeness.  $\square$



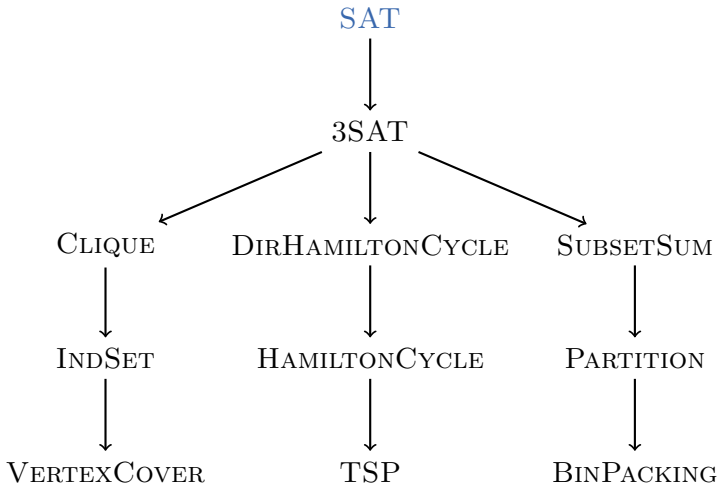
# NP-Complete Problems

- There are thousands of known NP-complete problems.
- An extensive catalog of NP-complete problems from many areas of computer science is contained in:

*Michael R. Garey and David S. Johnson:  
Computers and Intractability —  
A Guide to the Theory of NP-Completeness  
W. H. Freeman, 1979.*

- In the remaining chapters, we get to know some of these problems.

# Overview of the Reductions

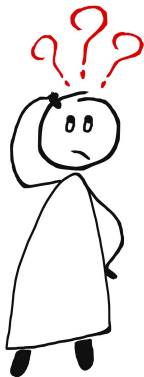


# What Do We Have to Do?

- We want to show the NP-completeness of these 11 problems.
- We first show that SAT is NP-complete.
- Then it is sufficient to show
  - that **polynomial reductions** exist for all edges in the figure (and thus all problems are NP-hard)
  - and that the problems are all in NP.

(It would be sufficient to show membership in NP only for the leaves in the figure. But membership is so easy to show that this would not save any work.)

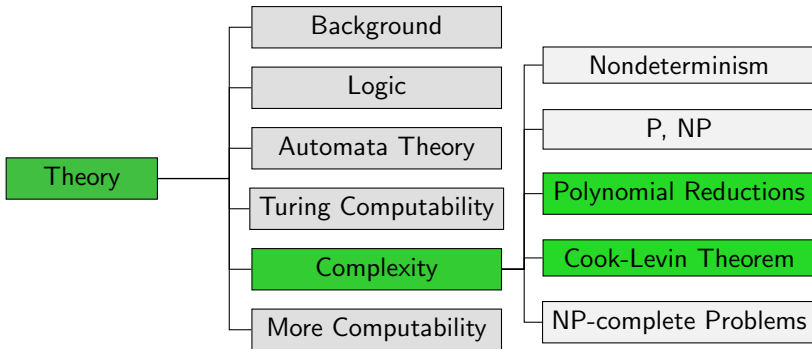
# Questions



Questions?

# Cook-Levin Theorem

# Course Overview



# SAT is NP-complete

## Definition (SAT)

The problem **SAT** (satisfiability) is defined as follows:

**Given:** a propositional logic formula  $\varphi$

**Question:** Is  $\varphi$  satisfiable?

# SAT is NP-complete

## Definition (SAT)

The problem **SAT** (satisfiability) is defined as follows:

**Given:** a propositional logic formula  $\varphi$

**Question:** Is  $\varphi$  satisfiable?

## Theorem (Cook, 1971; Levin, 1973)

*SAT is NP-complete.*



# SAT is NP-complete

## Definition (SAT)

The problem **SAT** (satisfiability) is defined as follows:

**Given:** a propositional logic formula  $\varphi$

**Question:** Is  $\varphi$  satisfiable?

## Theorem (Cook, 1971; Levin, 1973)

*SAT is NP-complete.*

## Proof.

**SAT  $\in$  NP:** guess and check.

**SAT is NP-hard:** somewhat more complicated (to be continued)

# NP-hardness of SAT (1)

Proof (continued).

We must show:  $A \leq_p \text{SAT}$  for all  $A \in \text{NP}$ .

# NP-hardness of SAT (1)

## Proof (continued).

We must show:  $A \leq_p \text{SAT}$  for all  $A \in \text{NP}$ .

Let  $A$  be an arbitrary problem in NP.

We have to find a polynomial reduction of  $A$  to SAT, i. e., a function  $f$  computable in polynomial time such that for every input word  $w$  over the alphabet of  $A$ :  
 $w \in A$  iff  $f(w)$  is a satisfiable propositional formula. ...

## NP-hardness of SAT (2)

### Proof (continued).

Because  $A \in \text{NP}$ , there is an NTM  $M$  and a polynomial  $p$  such that  $M$  accepts the problem  $A$  in time  $p$ .

**Idea:** construct a formula that encodes **the possible configurations** which  $M$  can reach in time  $p(|w|)$  on input  $w$  and that is **satisfiable if and only if an end configuration can be reached** in this time. ...

## NP-hardness of SAT (3)

### Proof (continued).

Let  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, E \rangle$  be an NTM for  $A$ , and let  $p$  be a polynomial bounding the computation time of  $M$ . Without loss of generality,  $p(n) \geq n$  for all  $n$ .

Let  $w = w_1 \dots w_n \in \Sigma^*$  be the input for  $M$ .

## NP-hardness of SAT (3)

### Proof (continued).

Let  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, E \rangle$  be an NTM for  $A$ , and let  $p$  be a polynomial bounding the computation time of  $M$ . Without loss of generality,  $p(n) \geq n$  for all  $n$ .

Let  $w = w_1 \dots w_n \in \Sigma^*$  be the input for  $M$ .

We number the tape positions with integers (positive and negative) such that the TM head initially is on position 1.

## NP-hardness of SAT (3)

### Proof (continued).

Let  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, E \rangle$  be an NTM for  $A$ , and let  $p$  be a polynomial bounding the computation time of  $M$ . Without loss of generality,  $p(n) \geq n$  for all  $n$ .

Let  $w = w_1 \dots w_n \in \Sigma^*$  be the input for  $M$ .

We number the tape positions with integers (positive and negative) such that the TM head initially is on position 1.

**Observation:** within  $p(n)$  computation steps the TM head can only reach positions in the set

$$Pos = \{-p(n) + 1, -p(n) + 2, \dots, -1, 0, 1, \dots, p(n) + 1\}.$$

## NP-hardness of SAT (3)

### Proof (continued).

Let  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, \square, E \rangle$  be an NTM for  $A$ , and let  $p$  be a polynomial bounding the computation time of  $M$ . Without loss of generality,  $p(n) \geq n$  for all  $n$ .

Let  $w = w_1 \dots w_n \in \Sigma^*$  be the input for  $M$ .

We number the tape positions with integers (positive and negative) such that the TM head initially is on position 1.

**Observation:** within  $p(n)$  computation steps the TM head can only reach positions in the set

$$Pos = \{-p(n) + 1, -p(n) + 2, \dots, -1, 0, 1, \dots, p(n) + 1\}.$$

Instead of infinitely many tape positions, we now only need to consider these (polynomially many!) positions.

...



# NP-hardness of SAT (4)

## Proof (continued).

We can encode configurations of  $M$  by specifying:

- what the current **state** of  $M$  is
- on which position in  $Pos$  the **TM head** is located
- which **symbols** from  $\Gamma$  the **tape** contains at positions  $Pos$

↗ can be encoded by propositional variables

# NP-hardness of SAT (4)

## Proof (continued).

We can encode configurations of  $M$  by specifying:

- what the current **state** of  $M$  is
- on which position in  $Pos$  the **TM head** is located
- which **symbols** from  $\Gamma$  the **tape** contains at positions  $Pos$

$\rightsquigarrow$  can be encoded by propositional variables

To encode a full **computation** (rather than just one configuration), we need **copies** of these variables for each computation step.

## NP-hardness of SAT (4)

### Proof (continued).

We can encode configurations of  $M$  by specifying:

- what the current **state** of  $M$  is
- on which position in  $Pos$  the **TM head** is located
- which **symbols** from  $\Gamma$  the **tape** contains at positions  $Pos$

$\rightsquigarrow$  can be encoded by propositional variables

To encode a full **computation** (rather than just one configuration), we need **copies** of these variables for each computation step.

We only need to consider the computation steps  $Steps = \{0, 1, \dots, p(n)\}$  because  $M$  should accept within  $p(n)$  steps.

...

# NP-hardness of SAT (5)

## Proof (continued).

Use the following propositional variables in formula  $f(w)$ :

- $state_{t,q}$  ( $t \in Steps, q \in Q$ )  
 $\rightsquigarrow$  encodes the state of the NTM in the  $t$ -th configuration
- $head_{t,i}$  ( $t \in Steps, i \in Pos$ )  
 $\rightsquigarrow$  encodes the head position in the  $t$ -th configuration
- $tape_{t,i,a}$  ( $t \in Steps, i \in Pos, a \in \Gamma$ )  
 $\rightsquigarrow$  encodes the tape content in the  $t$ -th configuration

Construct  $f(w)$  such that every satisfying interpretation

- describes a **sequence of TM configurations**
- that **begins with the start configuration**,
- **reaches an accepting configuration**
- and **follows the TM rules in  $\delta$**

# NP-hardness of SAT (6)

Proof (continued).

Auxiliary formula:

$$\text{oneof } X := \left( \bigvee_{x \in X} x \right) \wedge \neg \left( \bigvee_{x \in X} \bigvee_{y \in X \setminus \{x\}} (x \wedge y) \right)$$

Auxiliary notation:

The symbol  $\perp$  stands for an arbitrary unsatisfiable formula (e.g.,  $(A \wedge \neg A)$ , where  $A$  is an arbitrary proposition).

...

# NP-hardness of SAT (7)

Proof (continued).

1. describe the configurations of the TM:

$$\begin{aligned}
 \text{Valid} := & \bigwedge_{t \in \text{Steps}} \left( \text{oneof} \{ \text{state}_{t,q} \mid q \in Q \} \wedge \right. \\
 & \text{oneof} \{ \text{head}_{t,i} \mid i \in \text{Pos} \} \wedge \\
 & \left. \bigwedge_{i \in \text{Pos}} \text{oneof} \{ \text{tape}_{t,i,a} \mid a \in \Gamma \} \right)
 \end{aligned}$$

...

# NP-hardness of SAT (8)

Proof (continued).

2. begin in the start configuration

$$Init := state_{0,q_0} \wedge head_{0,1} \wedge \bigwedge_{i=1}^n tape_{0,i,w_i} \wedge \bigwedge_{i \in Pos \setminus \{1, \dots, n\}} tape_{0,i,\square}$$

...

# NP-hardness of SAT (9)

Proof (continued).

3. reach an accepting configuration

$$\text{Accept} := \bigvee_{t \in \text{Steps}} \bigvee_{q_e \in E} \text{state}_{t, q_e}$$

...



# NP-hardness of SAT (10)

Proof (continued).

4. follow the rules in  $\delta$ :

$$Trans := \bigwedge_{t \in Steps} \left( \bigvee_{q_e \in E} state_{t,q_e} \vee \bigvee_{R \in \delta} Rule_{t,R} \right)$$

where...

...

# NP-hardness of SAT (11)

## Proof (continued).

4. follow the rules in  $\delta$  (continued):

$Rule_{t, \langle \langle q, a \rangle, \langle q', a' \rangle, D \rangle} :=$

$state_{t,q} \wedge state_{t+1,q'} \wedge$

$\bigwedge_{i \in Pos} (head_{t,i} \rightarrow tape_{t,i,a} \wedge head_{t+1,i+D} \wedge tape_{t+1,i,a'}) \wedge$

$\bigwedge_{i \in Pos} \bigwedge_{a'' \in \Gamma} (\neg head_{t,i} \wedge tape_{t,i,a''} \rightarrow tape_{t+1,i,a''})$

- For  $D$ , interpret L  $\rightsquigarrow -1$ , N  $\rightsquigarrow 0$ , R  $\rightsquigarrow +1$ .
- **special case:**  $tape$  and  $head$  variables with a tape index  $i + D$  outside of  $Pos$  are replaced by  $\perp$ ; likewise all variables with a time index outside of  $Steps$ .

# NP-hardness of SAT (12)

Proof (continued).

Putting the pieces together:

Set  $f(w) := \text{Valid} \wedge \text{Init} \wedge \text{Accept} \wedge \text{Trans}$ .

# NP-hardness of SAT (12)

## Proof (continued).

Putting the pieces together:

Set  $f(w) := \text{Valid} \wedge \text{Init} \wedge \text{Accept} \wedge \text{Trans}$ .

- $f(w)$  can be constructed in time polynomial in  $|w|$ .
- $w \in A$  iff  $M$  accepts  $w$  in  $p(|w|)$  steps
  - iff  $f(w)$  is satisfiable
  - iff  $f(w) \in \text{SAT}$

$\rightsquigarrow A \leq_p \text{SAT}$

# NP-hardness of SAT (12)

## Proof (continued).

Putting the pieces together:

Set  $f(w) := \text{Valid} \wedge \text{Init} \wedge \text{Accept} \wedge \text{Trans}$ .

- $f(w)$  can be constructed in time polynomial in  $|w|$ .
- $w \in A$  iff  $M$  accepts  $w$  in  $p(|w|)$  steps
  - iff  $f(w)$  is satisfiable
  - iff  $f(w) \in \text{SAT}$

$\rightsquigarrow A \leq_p \text{SAT}$

Since  $A \in \text{NP}$  was arbitrary, this is true for every  $A \in \text{NP}$ .

# NP-hardness of SAT (12)

## Proof (continued).

Putting the pieces together:

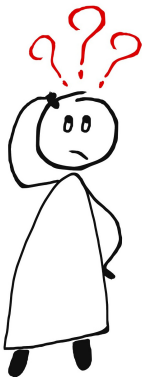
Set  $f(w) := \text{Valid} \wedge \text{Init} \wedge \text{Accept} \wedge \text{Trans}$ .

- $f(w)$  can be constructed in time polynomial in  $|w|$ .
- $w \in A$  iff  $M$  accepts  $w$  in  $p(|w|)$  steps  
iff  $f(w)$  is satisfiable  
iff  $f(w) \in \text{SAT}$

$\rightsquigarrow A \leq_p \text{SAT}$

Since  $A \in \text{NP}$  was arbitrary, this is true for every  $A \in \text{NP}$ .  
Hence SAT is NP-hard and thus also NP-complete. □

# Questions

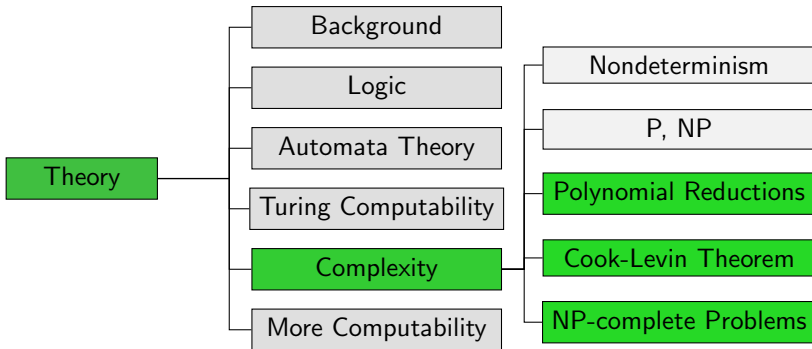


Questions?

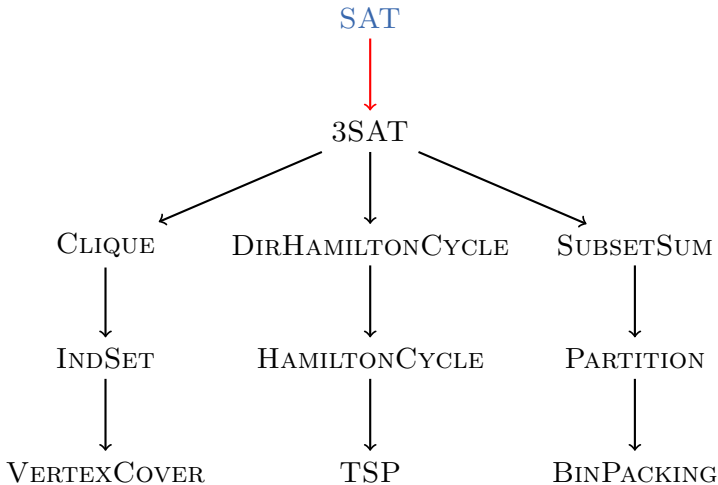
# 3SAT



# Course Overview



# SAT $\leq_p$ 3SAT



# SAT and 3SAT

## Definition (Reminder: SAT)

The problem **SAT** (satisfiability) is defined as follows:

**Given:** a propositional logic formula  $\varphi$

**Question:** Is  $\varphi$  satisfiable?

## Definition (3SAT)

The problem **3SAT** is defined as follows:

**Given:** a propositional logic formula  $\varphi$  in conjunctive normal form with at most three literals per clause

**Question:** Is  $\varphi$  satisfiable?

# 3SAT is NP-Complete (1)

Theorem (3SAT is NP-Complete)

*3SAT is NP-complete.*

## 3SAT is NP-Complete (2)

### Proof.

3SAT  $\in$  NP: guess and check.

3SAT is NP-hard: We show  $\text{SAT} \leq_p 3\text{SAT}$ .

- Let  $\varphi$  be the given input for SAT. Let  $\text{Sub}(\varphi)$  denote the set of subformulas of  $\varphi$ , including  $\varphi$  itself.

## 3SAT is NP-Complete (2)

### Proof.

3SAT  $\in$  NP: guess and check.

3SAT is NP-hard: We show  $\text{SAT} \leq_p 3\text{SAT}$ .

- Let  $\varphi$  be the given input for SAT. Let  $\text{Sub}(\varphi)$  denote the set of subformulas of  $\varphi$ , including  $\varphi$  itself.
- For all  $\psi \in \text{Sub}(\varphi)$ , we introduce a new proposition  $X_\psi$ .

## 3SAT is NP-Complete (2)

### Proof.

3SAT  $\in$  NP: guess and check.

3SAT is NP-hard: We show SAT  $\leq_p$  3SAT.

- Let  $\varphi$  be the given input for SAT. Let  $Sub(\varphi)$  denote the set of subformulas of  $\varphi$ , including  $\varphi$  itself.
- For all  $\psi \in Sub(\varphi)$ , we introduce a new proposition  $X_\psi$ .
- For each new proposition  $X_\psi$ , define the following auxiliary formula  $\chi_\psi$ :
  - If  $\psi = A$  for an atom  $A$ :  $\chi_\psi = (X_\psi \leftrightarrow A)$
  - If  $\psi = \neg\psi'$ :  $\chi_\psi = (X_\psi \leftrightarrow \neg X_{\psi'})$
  - If  $\psi = (\psi' \wedge \psi'')$ :  $\chi_\psi = (X_\psi \leftrightarrow (X_{\psi'} \wedge X_{\psi''}))$
  - If  $\psi = (\psi' \vee \psi'')$ :  $\chi_\psi = (X_\psi \leftrightarrow (X_{\psi'} \vee X_{\psi''}))$

## 3SAT is NP-Complete (3)

Proof (continued).

- Consider the conjunction of all these auxiliary formulas,

$$\chi_{\text{all}} := \bigwedge_{\psi \in \text{Sub}(\varphi)} \chi_{\psi}.$$



## 3SAT is NP-Complete (3)

### Proof (continued).

- Consider the conjunction of all these auxiliary formulas,  
 $\chi_{\text{all}} := \bigwedge_{\psi \in \text{Sub}(\varphi)} \chi_{\psi}$ .
- Every interpretation  $\mathcal{I}$  of the original variables can be extended to a model  $\mathcal{I}'$  of  $\chi_{\text{all}}$  in exactly one way: for each  $\psi \in \text{Sub}(\varphi)$ , set  $\mathcal{I}'(X_{\psi}) = 1$  iff  $\mathcal{I} \models \psi$ .

## 3SAT is NP-Complete (3)

### Proof (continued).

- Consider the conjunction of all these auxiliary formulas,  
 $\chi_{\text{all}} := \bigwedge_{\psi \in \text{Sub}(\varphi)} \chi_{\psi}$ .
- Every interpretation  $\mathcal{I}$  of the original variables can be extended to a model  $\mathcal{I}'$  of  $\chi_{\text{all}}$  in exactly one way:  
for each  $\psi \in \text{Sub}(\varphi)$ , set  $\mathcal{I}'(X_{\psi}) = 1$  iff  $\mathcal{I} \models \psi$ .
- It follows that  $\varphi$  is satisfiable iff  $(\chi_{\text{all}} \wedge X_{\varphi})$  is satisfiable.

## 3SAT is NP-Complete (3)

### Proof (continued).

- Consider the conjunction of all these auxiliary formulas,  
 $\chi_{\text{all}} := \bigwedge_{\psi \in \text{Sub}(\varphi)} \chi_{\psi}$ .
- Every interpretation  $\mathcal{I}$  of the original variables can be extended to a model  $\mathcal{I}'$  of  $\chi_{\text{all}}$  in exactly one way:  
for each  $\psi \in \text{Sub}(\varphi)$ , set  $\mathcal{I}'(X_{\psi}) = 1$  iff  $\mathcal{I} \models \psi$ .
- It follows that  $\varphi$  is satisfiable iff  $(\chi_{\text{all}} \wedge X_{\varphi})$  is satisfiable.
- This formula can be computed in linear time.

## 3SAT is NP-Complete (3)

### Proof (continued).

- Consider the conjunction of all these auxiliary formulas,
 
$$\chi_{\text{all}} := \bigwedge_{\psi \in \text{Sub}(\varphi)} \chi_{\psi}.$$
- Every interpretation  $\mathcal{I}$  of the original variables can be extended to a model  $\mathcal{I}'$  of  $\chi_{\text{all}}$  in exactly one way: for each  $\psi \in \text{Sub}(\varphi)$ , set  $\mathcal{I}'(X_{\psi}) = 1$  iff  $\mathcal{I} \models \psi$ .
- It follows that  $\varphi$  is satisfiable iff  $(\chi_{\text{all}} \wedge X_{\varphi})$  is satisfiable.
- This formula can be computed in linear time.
- It can also be converted to 3-CNF in linear time because it is the conjunction of constant-size parts involving at most three variables each.  
(Each part can be converted to 3-CNF independently.)

## 3SAT is NP-Complete (3)

### Proof (continued).

- Consider the conjunction of all these auxiliary formulas,  
 $\chi_{\text{all}} := \bigwedge_{\psi \in \text{Sub}(\varphi)} \chi_{\psi}$ .
- Every interpretation  $\mathcal{I}$  of the original variables can be extended to a model  $\mathcal{I}'$  of  $\chi_{\text{all}}$  in exactly one way: for each  $\psi \in \text{Sub}(\varphi)$ , set  $\mathcal{I}'(X_{\psi}) = 1$  iff  $\mathcal{I} \models \psi$ .
- It follows that  $\varphi$  is satisfiable iff  $(\chi_{\text{all}} \wedge X_{\varphi})$  is satisfiable.
- This formula can be computed in linear time.
- It can also be converted to 3-CNF in linear time because it is the conjunction of constant-size parts involving at most three variables each.  
(Each part can be converted to 3-CNF independently.)
- Hence, this describes a polynomial-time reduction.



# Restricted 3SAT

**Note:** 3SAT remains NP-complete if we also require that

- every clause contains **exactly** three literals and
- a clause may not contain **the same literal twice**

# Restricted 3SAT

**Note:** 3SAT remains NP-complete if we also require that

- every clause contains **exactly** three literals and
- a clause may not contain **the same literal twice**

Idea:

- remove duplicated literals from each clause.

# Restricted 3SAT

**Note:** 3SAT remains NP-complete if we also require that

- every clause contains **exactly** three literals and
- a clause may not contain **the same literal twice**

Idea:

- remove duplicated literals from each clause.
- add new variables:  $X, Y, Z$
- add new clauses:  $(X \vee Y \vee Z), (X \vee Y \vee \neg Z), (X \vee \neg Y \vee Z),$   
 $(\neg X \vee Y \vee Z), (X \vee \neg Y \vee \neg Z), (\neg X \vee Y \vee \neg Z),$   
 $(\neg X \vee \neg Y \vee Z)$



# Restricted 3SAT

**Note:** 3SAT remains NP-complete if we also require that

- every clause contains **exactly** three literals and
- a clause may not contain **the same literal twice**

Idea:

- remove duplicated literals from each clause.
- add new variables:  $X, Y, Z$
- add new clauses:  $(X \vee Y \vee Z), (X \vee Y \vee \neg Z), (X \vee \neg Y \vee Z),$   
 $(\neg X \vee Y \vee Z), (X \vee \neg Y \vee \neg Z), (\neg X \vee Y \vee \neg Z),$   
 $(\neg X \vee \neg Y \vee Z)$

↪ satisfied if and only if  $X, Y, Z$  are all true

# Restricted 3SAT

**Note:** 3SAT remains NP-complete if we also require that

- every clause contains **exactly** three literals and
- a clause may not contain **the same literal twice**

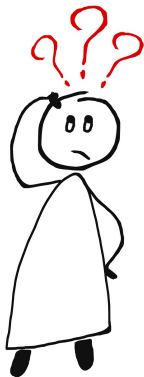
Idea:

- remove duplicated literals from each clause.
- add new variables:  $X, Y, Z$
- add new clauses:  $(X \vee Y \vee Z), (X \vee Y \vee \neg Z), (X \vee \neg Y \vee Z),$   
 $(\neg X \vee Y \vee Z), (X \vee \neg Y \vee \neg Z), (\neg X \vee Y \vee \neg Z),$   
 $(\neg X \vee \neg Y \vee Z)$

↪ satisfied if and only if  $X, Y, Z$  are all true

- fill up clauses with fewer than three literals with  $\neg X$  and if necessary additionally with  $\neg Y$

# Questions



Questions?

# Summary

# Summary

- Thousands of important problems are NP-complete.
- The satisfiability problem of propositional logic (**SAT**) is NP-complete.
- **Proof idea for NP-hardness:**
  - Every problem in NP can be solved by an NTM in polynomial time  $p(|w|)$  for input  $w$ .
  - Given a word  $w$ , construct a propositional logic formula  $\varphi$  that encodes the computation steps of the NTM on input  $w$ .
  - Construct  $\varphi$  so that it is satisfiable if and only if there is an accepting computation of length  $p(|w|)$ .
- Usually (as seen for 3SAT), the easiest way to show that another problem is NP-complete is to
  - show that it is in NP with a guess-and-check algorithm, and
  - polynomially reduce a known NP-complete to it.