Fundamental Computer Science

Malin Rau and Denis Trystram
(inspired by Giorgio Lucarelli)

February, 2020

Last lecture

- Definition of time complexity classes
  - P: problems solvable in $O(n^k)$ time
  - NP: problems verifiable in $O(n^k)$ time

- Prove that a problem belongs to NP
  - give a polynomial-time *verifier*
  - (give a Non-deterministic Turing Machine)

- Reduction from problem A to problem B $\quad (A \leq_{\mathrm{P}} B)$
  1. transform an instance I_{A} of A to an instance I_{B} of B
  2. show that the reduction is of polynomial size
  3. prove that:
     "there is a solution for the problem A on the instance I_{A}
     if and only if
     there is a solution for the problem B on the instance I_{B}"

Today

- Definition of the class NP-complete
- SAT is NP-complete
- Use reductions to prove NP-completeness
- Variants of SAT

Introduction to the SAT problem

Boolean formulas

- x_i: a Boolean variable, values TRUE or FALSE
- $\bar{x}_i$: negation of x_i
- $x_i, \bar{x}_i$: literals
- $\vee$: logical OR
- $\wedge$: logical AND
- $(x_1 \vee \bar{x}_3 \vee x_4)$: clause, a set of literals in disjunction
- $\mathcal{F} = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_4) \wedge (x_1 \vee x_4)$: a Boolean formula in Conjunctive Normal Form (CNF), a set of clauses in conjunction
  - every formula can be written in CNF (focus on CNF formulas)
- *assignment*: give TRUE or FALSE value to variables
- a formula is *satisfiable* if there is an assignment evaluating to TRUE
  - i.e, $(x_1, x_2, x_3, x_4) = (\text{TRUE}, \text{TRUE}, \text{TRUE}, \text{FALSE})$ for the above formula $\mathcal{F}$

The satisfiability problem

- $X = \{x_1, x_2, \ldots, x_n\}$: set of variables
- $C = \{c_1, c_2, \ldots, c_m\}$: set of clauses
- $\mathcal{F} = c_1 \wedge c_2 \wedge \ldots \wedge c_m$

$$\text{SAT} = \{\langle \mathcal{F} \rangle \mid \mathcal{F} \text{ is a satisfiable Boolean formula } \}$$

- $k\text{SAT}$: each clause has at most k literals
  (in some definitions exactly k literals)
  - example of 2SAT: $(x_1 \vee \bar{x}_2) \wedge (x_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3)$

$2SAT \in P$

$2SAT \in P$

Preliminaries

- Assume that each clause has exactly two literals
- $x \Rightarrow y$: implication

x	y	$x \Rightarrow y$
FALSE	FALSE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	TRUE	TRUE

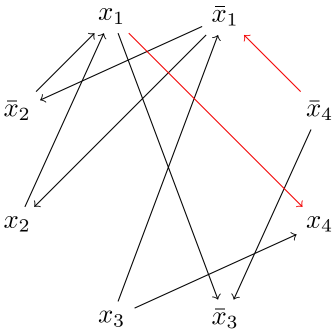
- $x \Rightarrow y = \bar{x} \vee y$

x	y	$\bar{x}$	$\bar{x} \vee y$
FALSE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE
FALSE	TRUE	TRUE	TRUE
TRUE	TRUE	FALSE	TRUE

$2SAT \in P$

- Construct a directed graph G
  - for each literal $x \in X \cup \bar{X}$, add a vertex
  - for each clause $x \vee y$, add the arcs $(\bar{x}, y)$ and $(\bar{y}, x)$
    corresponds to implications $\bar{x} \Rightarrow y$ and $\bar{y} \Rightarrow x$

$$\mathcal{F} = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_1 \vee x_2) \wedge (\bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_4)$$



We want $(\bar{x}_1 \vee x_4) = \mathrm{TRUE}$

- arc (x_1, x_4) means:
  – if $x_1 = \mathrm{T}$ then x_4 should be T
  – if $x_4 = \mathrm{F}$ then x_1 should be F
- arc $(\bar{x}_4, \bar{x}_1)$ means:
  – if $\bar{x}_4 = \mathrm{T}$ then $\bar{x}_1$ should be T
  – if $\bar{x}_1 = \mathrm{F}$ then $\bar{x}_4$ should be F

$2\text{SAT} \in \text{P}$

Lemma

If there is a path from x to y in G, then there is also a path from $\bar{y}$ to $\bar{x}$.

Proof:

$$x \longrightarrow \cdots \longrightarrow a \longrightarrow b \longrightarrow \cdots \longrightarrow y$$

- By construction:
  - we add an arc (a, b) if $(\bar{a} \vee b)$ exists in $\mathcal{F}$
  - but if $(\bar{a} \vee b)$ exists in $\mathcal{F}$, then we add also the arc $(\bar{b}, \bar{a})$
- Apply the argument for all arcs in the path from x to y

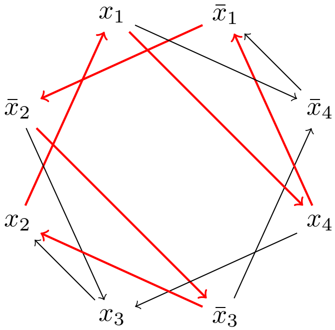
$$\bar{x} \longleftarrow \cdots \longleftarrow \bar{a} \longleftarrow \bar{b} \longleftarrow \cdots \longleftarrow \bar{y}$$

2SAT $\in$ P

Lemma

If there is a variable x such that G has both a path from x to $\bar{x}$ and a path from $\bar{x}$ to x, then $\mathcal{F}$ is not satisfiable.

$$\mathcal{F} = (x_1 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_3) \wedge (x_3 \vee \bar{x}_4) \wedge (x_4 \vee \bar{x}_1) \wedge (\bar{x}_4 \vee \bar{x}_1) \wedge (x_2 \vee x_3)$$



If $x_1 = \mathrm{TRUE}$, then
x_4 should be TRUE, and then
$(\bar{x}_4 \vee \bar{x}_1)$ is not satisfiable

If $x_1 = \mathrm{FALSE}$, then
x_2 should be FALSE, and then
$\bar{x}_3$ should be FALSE, and then
$(x_2 \vee x_3)$ is not satisfiable

$2SAT \in P$

Lemma

If there is a variable x such that G has both a path from x to $\bar{x}$ and a path from $\bar{x}$ to x, then $\mathcal{F}$ is not satisfiable.

Proof:

- assume that $\mathcal{F}$ is satisfiable (for contradiction)
- case 1: $x = \mathrm{TRUE}$

$$\begin{array}{ccccccccc} x & \longrightarrow & \cdots & \longrightarrow & a & \longrightarrow & b & \longrightarrow & \cdots & \longrightarrow & \bar{x} \\ \mathrm{T} & & & & \mathrm{T} & & \mathrm{F} & & & & \mathrm{F} \end{array}$$

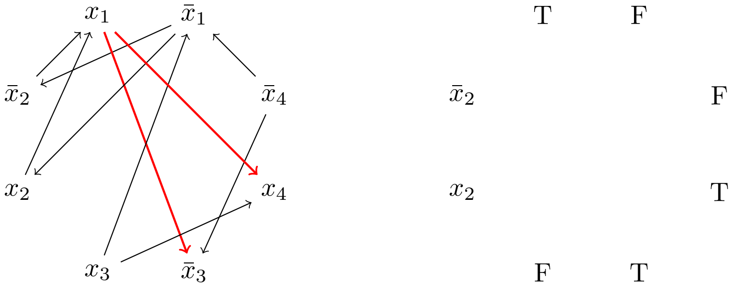
  There should be an arc (a, b) with $a = \mathrm{T}$ and $b = \mathrm{F}$.
  That is, $(\bar{a} \vee b)$ is not satisfiable.
  Hence, x cannot be TRUE.
- case 2: $x = \mathrm{FALSE}$
  Same arguments give that x cannot be FALSE on path from $\bar{x}$ to x.
- Then, $\mathcal{F}$ is not satisfiable, a contradiction.

$2SAT \in P$

Algorithm

1. **while** there are non-assigned variables **do**
2. Select a literal a for which there is not a path from a to $\bar{a}$.
3. Set $a = \text{TRUE}$.
4. Assign TRUE to all reachable literals from a.
5. Eliminate all assigned variables from G.

$$\mathcal{F} = (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_3) \wedge (x_1 \vee x_2) \wedge (\bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee x_4)$$



$2\mathrm{SAT} \in \mathrm{P}$

Lemma (Correctness of the algorithm)

Consider a literal a selected in Line 2 of the algorithm. There is no path from a to both b and $\bar{b}$.

Proof:

- Assume there are paths from a to b and from a to $\bar{b}$.
- Then, there are paths from $\bar{b}$ to $\bar{a}$ and from b to $\bar{a}$ (by the first lemma)
- Thus, there are paths from a to $\bar{a}$ (passing through b or $\bar{b}$)
- a cannot be selected by the algorithm because we only select a if there is not a path from a to $\bar{a}$, a contradiction.

Exercise

A Horn formula has at most one positive literal per clause.
Prove that $\textsc{Horn}\text{-}\mathrm{SAT} \in \mathrm{P}$, where

$$\textsc{Horn}\text{-}\mathrm{SAT} = \{\langle \mathcal{F} \rangle \mid \mathcal{F} \text{ is a satisfiable Horn formula}\}$$

Example:

$$\mathcal{F} = (x_1 \vee \bar{x}_2 \vee \bar{x}_5 \vee \bar{x}_3) \wedge (x_2 \vee \bar{x_3} \vee \bar{x}_4) \wedge (\bar{x}_1 \vee \bar{x}_5) \wedge (x_3 \vee \bar{x}_4) \wedge (x_4)$$

- negative literal $\bar{x}_i$, $i \in \mathbb{N}$
- positive literal x_i, $i \in \mathbb{N}$

Tipp:

- What has to happen to clauses that contain only one single literal?
- Consider the case that each clause contains a negative literal.

Solution

A Horn formula has at most one positive literal per clause.
Prove that $\textsc{Horn-SAT} \in \mathrm{P}$, where

$$\textsc{Horn-SAT} = \{\langle \mathcal{F} \rangle \mid \mathcal{F} \text{ is a satisfiable Horn formula}\}$$

Algorithm:

1. **while** there are clauses with only one literal
   1.1 pic a clause c with only one literal
   1.2 set the corresponding variable to TRUE or FALSE such that the clause is satisfied
   1.3 delete all clauses that are satisfied by the assignment and remove the variable from all the other clauses
2. set all non assigned variables to FALSE

After step 1 all the clauses contain at least one negative literal. Therefore, after setting all variables to FALSE in step 2 every clause will contain at least one literal that is TRUE. Hence all the clauses are satisfied. The algorithm has a time complexity of at most $\mathcal{O}((mn)^2)$

NP-completeness

NP-completeness

Definition

A language B is NP-complete if

- B is in NP, and
- every language A in NP is polynomially reducible to B.

Theorem

If B is NP-complete and $B \in \mathrm{P}$, then $\mathrm{P} = \mathrm{NP}$.

Proof:

- direct from the definition of reducibility

NP-completeness

Definition

A language B is NP-complete if

- B is in NP, and
- every language A in NP is polynomially reducible to B.

Theorem

If B is NP-complete *and $B \leq_{\mathrm{P}} C$ for $C \in \mathrm{NP}$, then C is* NP-complete

Proof:

- initially, $C \in \mathrm{NP}$
- we need to show: "every $A \in \mathrm{NP}$ polynomially reduces to C"
  - every language in NP polynomially reduces to B
  - B polynomially reduces to C

SAT $\in$ NP-COMPLETE

Cook-Levin theorem

Theorem

$\mathrm{SAT} \in \mathrm{P}$ *if and only if* $\mathrm{P} = \mathrm{NP}$.

equivalently: SAT is NP-COMPLETE.

SAT $\in$ NP-complete

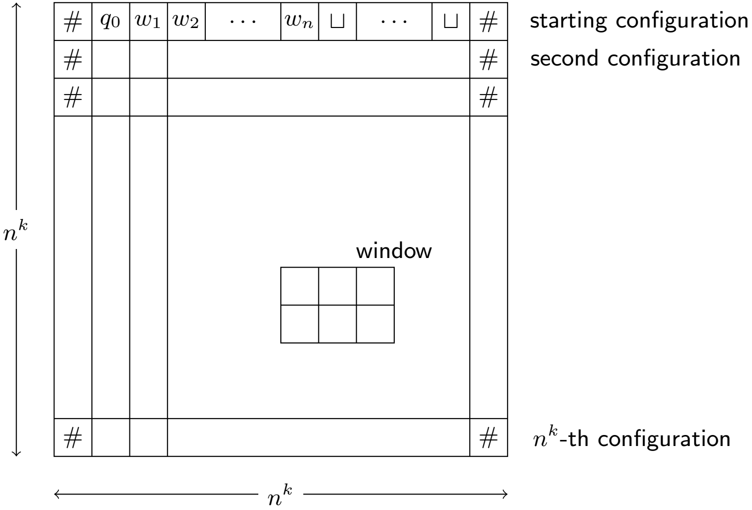
SAT is in NP

- given an assignment of variables, scan all clauses to check if they evaluate to TRUE

$A \leq_P$ SAT for every language $A \in$ NP

- M: a Non-Deterministic Turing Machine that *decides* A in n^k time
- create a table of size $n^k \times n^k$
  - each row i corresponds to a configuration $c_i = \#w_1 w_2 \ldots w_{\ell-1} q w_\ell \ldots w_r \#$
  - the head is on w_ℓ
  - $c_i \vdash_M c_{i+1}$
  - describes a branch of computation of M
- a table is **accepting** if any row is an accepting configuration

SAT ∈ NP-complete



SAT $\in$ NP-complete

For each i, j, s, where $1 \leq i, j \leq n^k$ and $s \in \Gamma \cup K$, define a variable

$$x_{i,j,s} = \begin{cases} \text{TRUE} & \text{if the cell in row } i \text{ and column } j \text{ contains the symbol } s \\ \text{FALSE} & \text{otherwise} \end{cases}$$

Define clauses to guarantee the calculation of M

- there is exactly one symbol in each cell

$$\phi_{\mathsf{cell}} = \bigwedge_{1 \leq i,j \leq n^k} \left[\left(\bigvee_{s \in \Gamma \cup K} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in \Gamma \cup K \\ s \neq t}} (\bar{x}_{i,j,s} \vee \bar{x}_{i,j,t}) \right) \right]$$

SAT $\in$ NP-complete

- the first row corresponds to the starting configuration

$$\begin{aligned}\phi_{\mathsf{start}} = \quad & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+2,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\end{aligned}$$

- there is an accepting state

$$\phi_{\mathsf{accept}} = \bigvee_{1 \le i,j \le n^k} x_{i,j,yes}$$

SAT $\in$ NP-complete

- every window is legal
  - example: legal configurations for $\Delta(q_1, a) = \{(q_1, b, \rightarrow)\}$ and $\Delta(q_1, b) = \{(q_2, c, \leftarrow), (q_2, a, \rightarrow)\}$

(a)

a	q_1	b
q_2	a	c

(b)

a	q_1	b
a	a	q_2

(c)

a	a	q_1
a	a	b

(d)

$\#$	b	a
$\#$	b	a

(e)

a	b	a
a	b	q_2

(f)

b	b	b
c	b	b

  - then,

$$\phi_{\text{legal}}^{i,j} = \bigvee_{\substack{a_1,\ldots,a_6 \\ \text{is a legal window}}} \left(x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}\right)$$

$$\phi_{\text{move}} = \bigwedge_{1 \leq i,j \leq n^k} \phi_{\text{legal}}^{i,j}$$

SAT $\in$ NP-complete

Construct $\mathcal{F} = \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{accept}} \wedge \phi_{\text{move}}$

- $\mathcal{F}$ has $n^{O(k)}$ variables and clauses

Theorem: $\mathcal{F}$ is satisfiable if and only if A is decided by M

3SAT $\in$ NP-COMPLETE

$3SAT \in$ NP-COMPLETE

3SAT Problem

- as SAT but each clause has at most 3 literals

How can we prove a problem A is NP-COMPLETE?

- show that the problem is NP
- find a suitable problem B that is NP-COMPLETE
- show that $B \leq_P A$
  - find a polynomial transformation that transforms each instance I_B of B to an instance I_A of A
  - prove that there is a solution for the problem B on the instance I_B if and only if there is a solution for the problem A on the instance I_A.

3SAT $\in$ NP-COMPLETE

3SAT is in NP

- given an assignment of variables, scan all clauses to check if they evaluate to TRUE

SAT $\leq_{\text{P}}$ 3SAT

Transformation: given any formula $\mathcal{F}$ of SAT in CNF with m clauses and n variables, we construct a formula $\mathcal{F}'$ of 3SAT:

- replace each clause $(a_1 \vee a_2 \vee \ldots \vee a_\ell)$ in $\mathcal{F}$ with $\ell - 2$ clauses

  $$(a_1 \vee a_2 \vee z_1) \wedge (\bar{z}_1 \vee a_3 \vee z_2) \wedge (\bar{z}_2 \vee a_4 \vee z_3) \wedge \ldots \wedge (\bar{z}_{\ell-3} \vee a_{\ell-1} \vee a_\ell)$$

1. Polynomiality: $\mathcal{F}'$ has $O(nm)$ variables and clauses
2. $\mathcal{F}$ is satisfiable iff $\mathcal{F}'$ is satisfiable

$3\mathrm{SAT} \in \mathrm{NP}$-COMPLETE

$\mathrm{SAT} \leq_{\mathrm{P}} 3\mathrm{SAT}$

Transformation: given any formula $\mathcal{F}$ of SAT in CNF with m clauses and n variables, we construct a formula $\mathcal{F}'$ of $3\mathrm{SAT}$:

- replace each clause $(a_1 \vee a_2 \vee \ldots \vee a_\ell)$ in $\mathcal{F}$ with $\ell - 2$ clauses

$$(a_1 \vee a_2 \vee z_1) \wedge (\bar{z}_1 \vee a_3 \vee z_2) \wedge (\bar{z}_2 \vee a_4 \vee z_3) \wedge \ldots \wedge (\bar{z}_{\ell-3} \vee a_{\ell-1} \vee a_\ell)$$

Proving $\mathcal{F}$ is satisfiable iff $\mathcal{F}'$ is satisfiable

1. $\mathcal{F}'$ is satisfiable if $\mathcal{F}$ is satisfiable
    - assume that $\mathcal{F}$ is satisfiable
    - then some a_i is TRUE for all clauses
    - use the same assignment for the common variables of $\mathcal{F}$ and $\mathcal{F}'$
    - set $z_j = \mathrm{TRUE}$ for $1 \leq j \leq i - 2$
    - set $z_j = \mathrm{FALSE}$ for $i - 1 \leq j \leq \ell - 3$
    - all clauses of $\mathcal{F}'$ are satisfied

Example

$$(a_1 \vee a_2 \vee z_1) \wedge (\bar{z}_1 \vee a_3 \vee z_2) \wedge (\bar{z}_2 \vee a_4 \vee z_3) \wedge (\bar{z}_3 \vee a_5 \vee a_6)$$
$$(F \vee F \vee z_1) \wedge (\bar{z}_1 \vee T \vee z_2) \wedge (\bar{z}_2 \vee F \vee z_3) \wedge (\bar{z}_3 \vee F \vee F)$$
$$(F \vee F \vee T) \wedge (F \vee T \vee F) \wedge (T \vee F \vee F) \wedge (T \vee F \vee F)$$

3SAT $\in$ NP-complete

SAT $\leq_P$ 3SAT

Transformation: given any formula $\mathcal{F}$ of SAT in CNF with m clauses and n variables, we construct a formula $\mathcal{F}'$ of 3SAT:

- replace each clause $(a_1 \vee a_2 \vee \ldots \vee a_\ell)$ in $\mathcal{F}$ with $\ell - 2$ clauses

$$(a_1 \vee a_2 \vee z_1) \wedge (\bar{z}_1 \vee a_3 \vee z_2) \wedge (\bar{z}_2 \vee a_4 \vee z_3) \wedge \ldots \wedge (\bar{z}_{\ell-3} \vee a_{\ell-1} \vee a_\ell)$$

Proving $\mathcal{F}$ is satisfiable iff $\mathcal{F}'$ is satisfiable

1. $\mathcal{F}'$ is satisfiable if $\mathcal{F}$ is satisfiable ✓
2. $\mathcal{F}$ is satisfiable if $\mathcal{F}'$ is satisfiable
   - assume that $\mathcal{F}'$ is satisfiable
   - at least one of the literals a_i should be TRUE for each clause
   - if not, then z_1 should be TRUE which implies that z_2 should be TRUE, etc
   - hence, the clause $(\bar{z}_{\ell-3} \vee a_{\ell-1} \vee a_\ell)$ is not satisfiable, contradiction
   - then there is an assignment that satisfies $\mathcal{F}$

3SAT $\in$ NP-complete

- 3SAT is in NP ✓
- give a transformation form SAT to 3SAT ✓
- it is polynomial ✓
- $\mathcal{F} \in$ SAT is satisfiable iff $\mathcal{F}' \in$ 3SAT is satisfiable ✓

$\Rightarrow$ 3SAT $\in$ NP-complete

MAX-2SAT $\in$ NP-complete

MAX-2SAT $\in$ NP-complete

$\text{MAX-2SAT} = \{\langle \mathcal{F}, k \rangle \mid \mathcal{F}$ is a formula with k TRUE clauses$\}$

MAX-2SAT is in NP

- given an assignment of variables, scan all clauses to check if there are at least k of them evaluated to TRUE

$\text{3SAT} \leq_{\text{P}} \text{MAX-2SAT}$

1. given any formula $\mathcal{F}$ of 3SAT, we construct a formula $\mathcal{F}'$ of MAX-2SAT
   - replace each clause $(x \vee y \vee z)$ with

     $$(x)\wedge(y)\wedge(z)\wedge(\bar{x}\vee\bar{y})\wedge(\bar{y}\vee\bar{z})\wedge(\bar{z}\vee\bar{x})\wedge(w)\wedge(\bar{w}\vee x)\wedge(\bar{w}\vee y)\wedge(\bar{w}\vee z)$$

   - $k = 7m$ (m is the number of clauses)
2. $\mathcal{F}'$ has $O(n+m)$ variables and $O(m)$ clauses

MAX-2SAT $\in$ NP-COMPLETE

3SAT $\leq_P$ MAX-2SAT

1. recall: replace each clause $(x \vee y \vee z)$ with
   $(x) \wedge (y) \wedge (z) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{y} \vee \bar{z}) \wedge (\bar{z} \vee \bar{x}) \wedge (w) \wedge (\bar{w} \vee x) \wedge (\bar{w} \vee y) \wedge (\bar{w} \vee z)$

3. $\mathcal{F}$ is satisfiable iff $\mathcal{F}'$ has at least k satisfied clauses
   - assume that $\mathcal{F}$ is satisfiable
   - if $x = \mathrm{T}$, $y = \mathrm{F}$ and $z = \mathrm{F}$, then set $w = \mathrm{F}$: 7 satisfied clauses
   - if $x = \mathrm{T}$, $y = \mathrm{T}$ and $z = \mathrm{F}$, then set $w = \mathrm{F}$: 7 satisfied clauses
   - if $x = \mathrm{T}$, $y = \mathrm{T}$ and $z = \mathrm{T}$, then set $w = \mathrm{T}$: 7 satisfied clauses
   - in all cases, there are 7 satisfied clauses in $\mathcal{F}'$ for each clause of $\mathcal{F}$

   - contrapositive: assume that $\mathcal{F}$ is not satisfiable
   - there is one clause for which $x = y = z = \mathrm{F}$
   - then, in $\mathcal{F}'$ we correspondingly have:
     – 4 satisfied clauses if $w = \mathrm{T}$
     – 6 satisfied clauses if $w = \mathrm{F}$
   - hence, in $\mathcal{F}'$ there are less than k clauses that are satisfied

CLIQUE $\in$ NP-complete

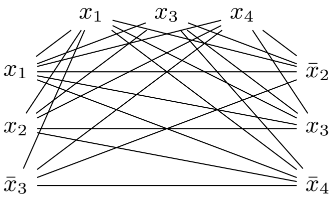
CLIQUE $\in$ NP-COMPLETE

CLIQUE is in NP

- given a set of vertices, check if there is an edge between any pair of them

3SAT $\leq_P$ CLIQUE

1. given any formula $\mathcal{F}$ of SAT, we construct an instance $I = \langle G, k \rangle$ of CLIQUE
   - add a vertex for each literal
   - add an edge between any two literals except:
     (a) literals in the same clause
     (b) a literal and its negation
   - $k = m$ (number of clauses)
   - example: $\mathcal{F} = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_4)$



CLIQUE $\in$ NP-complete

3SAT $\leq_P$ CLIQUE

2. $|V| = 3m$, $|E| = O(m^2)$
3. $\mathcal{F}$ is satisfiable iff there is a clique of size k in G
   - assume that $\mathcal{F}$ is satisfiable
   - at least one literal is TRUE in any clause
   - there is an edge between such literals (why?)
   - hence, the corresponding vertices form a k-clique

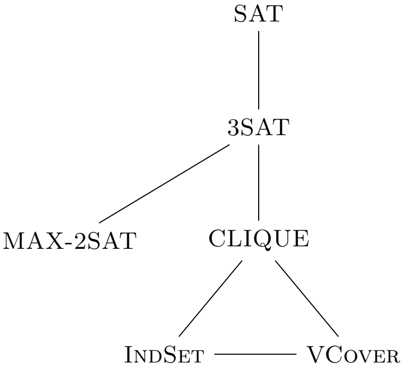
   - assume there is a k-clique in G
   - this clique contains at most one vertex from each clause
   - $k = m$, hence the clique contains exactly one vertex from each clause
   - each pair of these vertices is compatible (no a literal and its negation)
   - set the corresponding literals to TRUE
   - $\mathcal{F}$ is satisfiable

Summarize: NP-COMPLETENESS proofs

Summarize: NP-completeness proofs

1. Prove that the problem is in NP (give a verifier)
2. Give a polynomial time reduction from a known NP-complete problem
   - important: choose the correct problem

NP-complete problems



Exercises

- Show that $\textsc{Independent Set}$ is $\text{NP-}\textsc{complete}$ by a reduction from 3-SAT or $\textsc{Clique}$, where

  $\textsc{Independent Set} = \{\langle G, k\rangle \mid G = (V, E)$ is a graph with a set $A \subseteq V$ such that $|A| = k$ and for each $x, y \in A$ with $x \neq y$, it holds that $\{x, y\} \notin E\}$.

- Show that $\textsc{Vertex Cover}$ is $\text{NP-}\textsc{complete}$ by a reduction from 3-SAT, $\textsc{Clique}$ or $\textsc{Independent Set}$, where

  $\textsc{Vertex Cover} = \{\langle G, k\rangle \mid G = (V, E)$ is a graph with a set $A \subseteq V$ such that $|A| = k$ and every $e \in E$ is incident to a vertex in $A\}$

- Show that $3\text{-}\textsc{Coloring}$ is $\text{NP-}\textsc{complete}$ by a reduction from 3-SAT where

  $3\text{-}\textsc{Coloring} = \{\langle G, k\rangle \mid G = (V, E)$ is a graph and there exists a function $f : V \to \{1, 2, 3\}$ such that for every edge $\{u, v\} \in E$ we have $f(u) \neq f(v)\}$