

7.3 The Class NP

- The class NP [Theorem 7.20, Corollary 7.22]:

$$\begin{aligned} \text{NP} &= \bigcup \{ \text{NTIME}(t) \mid t \text{ is a polynomial} \} \\ &= \bigcup_{k \geq 0} \text{NTIME}(n^k) \end{aligned}$$

- In other words, the set of languages decidable in *polynomial* time by a nondeterministic Turing machine
- Obviously, $P \subseteq \text{NP}$
- When we talk about problems in NP, we usually mean those problems that do not belong to P: $\text{NP} \setminus P$
- For example, one does not know how to find a Hamiltonian path in a directed graph in polynomial time, instead one has to resort to the brute-force algorithm

- The following nondeterministic Turing machine decides the Hamiltonian path problem in polynomial time
- On input $\langle G, s, t \rangle$, where G is a directed graph with nodes s and t , the number of nodes in G is m :
 - Write a list of m numbers, p_1, \dots, p_m , where each p_i is non-deterministically selected to be between 1 and m
 - If there are repetitions in the list p_1, \dots, p_m , *reject*
 - If $p_1 \neq s$ or $p_m \neq t$, *reject*
 - For each $i \in \{1, \dots, m\}$ check whether (p_i, p_{i+1}) is an edge of G . If any are not, *reject*. Otherwise, we have verified the nondeterministic selection, so *accept*

CLIQUE is in NP

- A **clique** in an undirected graph is a subgraph, wherein every two nodes are connected by an edge
- A k -clique is a clique that contains k nodes

$\text{CLIQUE} = \{ \langle G, k \rangle \mid G \text{ is an undirected graph with a } k\text{-clique} \}$

- We can solve clique nondeterministically by first selecting (guessing) k nodes of the graph G
- Thereafter we deterministically verify that all selected nodes really are connected by an edge in G
- If any are not, *reject*. Otherwise, *accept*.

SUBSET-SUM is in NP

- Given a multiset of numbers $S = \{x_1, \dots, x_k\}$ and a target number t , does S contain a subcollection $\{y_1, \dots, y_l\}$ s.t. $\sum_i y_i = t$
- E.g., the pair $\langle \{4, 11, 16, 21, 27\}, 25 \rangle$ belongs to the language of this problem, because $4 + 21 = 25$
- Exhaustive search: examine all 2^k subcollections and determine whether the sum of the elements is t
- Nondeterministically select (guess) a subcollection of numbers from the given multiset S
- Deterministically verify that the chosen numbers sum to t

P =?= NP

- P is the class of languages for which membership can be **decided** quickly
- NP is the class of languages for which membership can be **verified** quickly
- It would seem clear that NP is significantly larger class than P, but it has not been proved for a single language in $NP \setminus P$ that it would not have a polynomial-time decider algorithm
- Hence, in principle it is possible that $P = NP$
- For problems in $NP \setminus P$ one is only aware of deterministic algorithms requiring an exponential time

$$NP \subseteq EXPTIME = \bigcup_k DTIME(2^{n^k})$$



7.4 NP-Completeness

- A function $f: \Sigma^* \rightarrow \Gamma^*$ is a polynomial time computable if there exists a Turing machine M and a polynomial p for which
 - $f = f_M$ and
 - $\text{time}_M(n) \leq p(n)$ for all n
- Let $A \subseteq \Sigma^*$, $B \subseteq \Gamma^*$ be two formal languages

Definition 7.29 Language A is **polynomial time reducible** to language B , written

$$A \leq_m^p B,$$

if a polynomial time computable function $f: \Sigma^* \rightarrow \Gamma^*$ exists, where for every $x \in \Sigma^*$,

$$x \in A \Leftrightarrow f(x) \in B$$





Theorem 7.31 (Extended)

For all languages A, B, C it holds

- i. $A \leq_m^p A$, (reflexive)
- ii. if $A \leq_m^p B$ and $B \leq_m^p C$, then $A \leq_m^p C$ (transitive),
- iii. if $A \leq_m^p B$ and $B \in \text{NP}$, then $A \in \text{NP}$, and
- iv. if $A \leq_m^p B$ and $B \in \text{P}$, then $A \in \text{P}$.

Note: for the part of mapping reducibility this theorem is exactly the same as Lemma K (Theorem 5.22). The difference is the polynomial time computability of the reduction.



Proof.

- i. We choose $f(x) = x$ to be the reduction.
- ii. The composite function $h(x) = g(f(x))$ is a reduction from A to C ,
 $h: A \leq_m C$ (see Lemma K, Theorem 5.22).

h can be computed in polynomial time:

Let $M_f(M_g)$ be the Turing machine computing function $f(g)$ in time bounded by polynomial $p(q)$.

We can assume that p and q are everywhere non-descending.

Let M_g , M_{REW} , and M_f work as in the proof of Lemma K.



By combining the TMs as previously, we get a TM M_h that computes the function h , and uses the following time on input x :

$$\begin{aligned} \text{time}_{M_f}(x) + \text{time}_{M_{\text{REW}}}(f(x)v) + \text{time}_{M_g}(f(x)) \\ \leq p(|x|) + 2p(|x|) + q(|f(x)|) \\ \leq 3p(|x|) + q(p(|x|)) \\ = O(q(p(|x|))), \end{aligned}$$

which is polynomial in the length of x .

iii. (and iv.) By combining

- the TM M_f which computes the reduction $f: A \leq_m^p B$ in time bounded by the polynomial p ,
- the TM M_B , which decides the language B in time bounded by q , and
- M_{REW}

similarly as in the proof of Lemma K, we get the TM M_A , which decides the language A in time $O(q(p(|x|)))$. It is deterministic whenever M_B is. \square



Satisfiability of Boolean formulas, SAT

Given a Boolean formula φ , which consists of

- Boolean variables x_1, \dots, x_n ,
- Constant values 0 (false) and 1 (true), and
- Boolean operations \vee , \wedge , and \neg .

Is φ satisfiable? Is there an assignment of values 0 and 1 to the variables $t: \{x_1, \dots, x_n\} \rightarrow \{0, 1\}$, such that

$$\varphi(t(x_1), \dots, t(x_n)) = 1$$

Let us guess the assignment t of values for the variables and verify that $\varphi(t) = 1$.

If φ contains n Boolean variables, then t can be represented as a binary string of n bits and it can be verified in polynomial time





- Stephen Cook and Leonid Levin discovered in the early 1970s that there exists the class of *NP-complete* problems
- The individual complexity of an *NP-complete* problem is related to that of the entire class of *NP*
- If a polynomial-time algorithm exists for any of the *NP-complete* problems, all problems in *NP* would be polynomial-time solvable
- *NP-complete* problems help to study the question $P \stackrel{?}{=} NP$ and to recognize difficult practical problems

Theorem 7.27 (Cook-Levin theorem)

$$SAT \in P \Leftrightarrow P = NP$$



- The satisfiability problem for many special forms of Boolean formulas is also *NP-complete*
- A formula φ is in *conjunctive normal form* (cnf), if it comprises several conjuncts

$$\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_m$$

where each *clause* C_i is a disjunction

$$C_i = \alpha_{i1} \vee \alpha_{i2} \vee \dots \vee \alpha_{ir}$$

- Terms α_{ij} are *literals*: Boolean variables or their negations
- *CSAT* is the satisfiability problem for cnf-formulas:

$$\{ \varphi \mid \varphi \text{ is a satisfiable cnf-formula} \}$$

- Obviously, $CSAT \in NP$. An arbitrary Boolean formula can be converted to a cnf-formula in polynomial time





- By restricting the number of terms in a clause of a cnf-formula to be exactly k literals, we get the k -conjunctive normal form (k -cnf)
- A family of languages:

$$kSAT = \{ \varphi \mid \varphi \text{ is a satisfiable } k\text{-cnf-formula} \}$$
- Language 2SAT belongs to P

Theorem $CSAT \leq_m^P 3SAT$

Proof.

- The given cnf-formula φ can be converted in polynomial time into an equivalent 3-cnf-formula φ'
- Let $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_m$



- Each clause $C_k = \alpha_1 \vee \alpha_2 \vee \dots \vee \alpha_r$, $r \geq 3$, is replaced by a 3-cnf-formula

$$C_k' = (\alpha_1 \vee \alpha_2 \vee t_1) \wedge (\neg t_1 \vee \alpha_3 \vee t_2) \wedge \dots \wedge (\neg t_{r-3} \vee \alpha_{r-1} \vee \alpha_r),$$
 where t_1, \dots, t_{r-3} are new variables. The formula C_k' can clearly be obtained from clause C_k in polynomial time.

We still need to check that the transformation satisfies reducibility

$$\varphi \in CSAT \Leftrightarrow \varphi' \in 3SAT:$$

1. φ satisfiable $\Rightarrow \varphi'$ satisfiable:

For all clauses C_k the assignment satisfying φ must set $\alpha_i = 1$ for some $\alpha_i \in C_k$.

C_k' gets satisfied when we set the values of literals as in the assignment satisfying C_k and the new variables get values as





follows

$$t_j = \begin{cases} 1, & \text{if } j \leq i-2 \\ 0, & \text{if } j > i-2 \end{cases}$$

2. φ is *satisfiable* $\Leftrightarrow \varphi'$ is *satisfiable*:

Also the subformulas C_k' corresponding to the clauses C_k of φ must be satisfied. Then either

- a) Some literal $\alpha_i = 1$, $\alpha_i \in C_k$, and C_k gets satisfied by it, or
- b) For some $i < r-3$:

$$t_i = 1 \wedge t_{i+1} = 0,$$

and it must be that $\alpha_{i+2} = 1$, and again C_k gets satisfied.



If $r \leq 3$, then

$$C_k = \alpha_1 \vee \alpha_2 \vee \alpha_3 \Rightarrow$$

$$C_k' = C_k$$

$$C_k = \alpha_1 \vee \alpha_2 \Rightarrow$$

$$C_k' = (\alpha_1 \vee \alpha_2 \vee t) \wedge (\alpha_1 \vee \alpha_2 \vee \neg t)$$

$$C_k = \alpha \Rightarrow$$

$$C_k' = (\alpha \vee t_1 \vee t_2) \wedge (\alpha \vee t_1 \vee \neg t_2) \\ \wedge (\alpha \vee \neg t_1 \vee t_2) \wedge (\alpha \vee \neg t_1 \vee \neg t_2)$$

The equivalence of satisfiability of the formulas is maintained. \square



Vertex Cover, VC

Given an undirected graph G and a natural number k .

Does G contain a subset of k nodes that cover every edge of G ?

- A node covers an edge if the edge touches the node.

To represent VC as a formal language we need to encode graphs as strings. Similar encoding techniques as those used with Turing machines apply.

We can guess the given number k of nodes from the given graph G , and then verify in time polynomial in the size of the graph that the chosen k nodes cover all edges of G



Theorem 7.44 $3SAT \leq_m^p VC$

Proof. Let $\varphi = C_1 \wedge C_2 \wedge \dots \wedge C_m$ be a 3-cnf-formula with variables x_1, \dots, x_n .

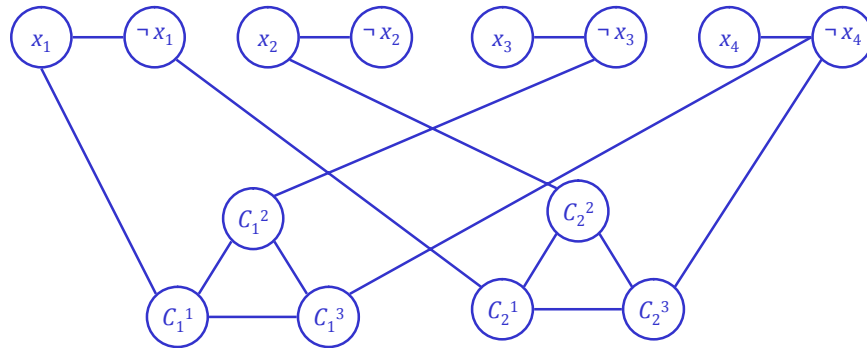
The corresponding instance of vertex cover $\langle G, k \rangle$ is as follows:

- G has a node corresponding to each literal
- G has 3 nodes C_j^1, C_j^2, C_j^3 corresponding to each clause C_j of φ
- G has edges:
 - $(x_p, \neg x_p)$,
 - $(C_j^1, C_j^2), (C_j^2, C_j^3), (C_j^3, C_j^1)$, and
 - If $C_j = \alpha_1 \vee \alpha_2 \vee \alpha_3$, then $(C_j^1, \alpha_1), (C_j^2, \alpha_2), (C_j^3, \alpha_3)$
- $k = n + 2m$

Clearly G can be composed in polynomial time from formula φ .



The graph for formula $\varphi = (x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_4)$



1. φ is satisfiable \Leftrightarrow

G has a vertex cover of at most $k = n + 2m$ nodes:

- Let us include into the vertex cover corresponding to the value assignment, the node representing the literal which obtains value 1 (n nodes)
- For each clause C_j , one edge (C_j^i, α_i) of the corresponding triangle is now covered
- We take to the vertex cover the two remaining corners of the triangle (altogether $2m$ nodes)

2. G has a vertex cover of at most k nodes $\Leftrightarrow \varphi$ is satisfiable

- Let V' , $|V'| \leq k$, be a vertex cover of G
- For V' to be able to cover all edges of G , it must contain one node per each variable and at least two nodes from each C_j -triangle
- Hence, $|V'| = k$
- Let us set

$$t(x_i) = \begin{cases} 1, & \text{if } x_i \in V' \\ 0, & \text{if } \neg x_i \in V' \end{cases}$$

- One of the edges starting from the corners of each C_j -triangle is covered by a literal node $\alpha \in V'$
- Then $t(\alpha) = t(C_j) = 1$ □



- Hence, $SAT \leq_m^p CSAT \leq_m^p 3SAT \leq_m^p VC$

Definition 7.34 A language B is *NP-complete* if it satisfies:

1. $B \in NP$, and
 2. $A \leq_m^p B$ for every $A \in NP$
- A *NP-complete* language can be decided deterministically in polynomial time if and only if all other languages in *NP* can also be decided deterministically in polynomial time

Theorem 7.35 If B is *NP-complete* and $B \in P$, then $P = NP$.





Theorem 7.36 If B is NP-complete and $B \leq_m^p C$ for $C \in NP$, then C is NP-complete.

Proof. Because B is NP-complete, by definition $A \leq_m^p B$ for every language $A \in NP$. On the other hand, $B \leq_m^p C$, and by the transitivity of polynomial time reductions (Theorem 7.31) it must hold that $A \leq_m^p C$ for all $A \in NP$. By assumption $C \in NP$, and the claim holds. \square

- Hence, to show that language C is NP-complete, it suffices to reduce in polynomial time some language B known to be NP-complete to C and in addition verify that $C \in NP$
- However, we should find the first NP-complete language



Theorem (Cook-Levin) Language

$SAT = \{ \varphi \mid \varphi \text{ is a satisfiable Boolean formula} \}$

is NP-complete.

- We need to show that $A \leq_m^p SAT$ for any $A \in NP$
- All that we know about A is that it has a polynomial time nondeterministic decider N
- The reduction for A takes a string w and produces a Boolean formula φ_w that simulates N on input w
- φ_w is satisfiable iff $w \in L(N) = A$
- For each possible computation of N we have one truth value assignment of the variables in φ_w
- The formula φ_w is composed to give those conditions by which the given assignment corresponds to an accepting computation of N



Corollary 7.42 *CSAT, 3SAT, and VC are NP-complete.*

Independent Set, IS: Given an undirected graph G and a natural number k . Does G have at least k nodes which have no edges with each other?

By the following lemma it is easy to compose reductions

$VC \leq_m^p IS$ and $IS \leq_m^p CLIQUE$

Lemma Let $G = (V, E)$ be an undirected graph and $V' \subseteq V$. Then the following conditions are equivalent:

1. V' is a vertex cover in G ,
2. $V \setminus V'$ is an independent set, and
3. $V \setminus V'$ is a clique in the complement graph of G :
 $\check{G} = (V, (V \times V) \setminus E)$

□



$VC \leq_m^p IS$:

Let us choose the mapping f :

$$f\langle G, k \rangle = \langle G, |V| - k \rangle.$$

Clearly this transformation can be computed in polynomial time. Now, by the preceding lemma

$$\langle G, k \rangle \in VC \Leftrightarrow \langle G, |V| - k \rangle \in IS.$$

Hence, $f: VC \leq_m^p IS$.

$IS \leq_m^p CLIQUE$:

Let us now choose the mapping f :

$$f\langle G, k \rangle = \langle \check{G}, k \rangle.$$

This transformation can be computed in polynomial time and by the preceding lemma

$$\langle G, k \rangle \in IS \Leftrightarrow \langle \check{G}, k \rangle \in CLIQUE.$$

Thus, $f: IS \leq_m^p CLIQUE$.



