MTAT.05.125 Theoretical Computer Science

Spring 2022

Vitaly Skachek
Estonian version by Reimo Palm
English version by Yauhen Yakimenka

Lecture 15. NP-complete languages: examples

3-SAT

Theorem. *3-SAT is* NP*-complete.*

Proof. First, 3-SAT $\in \mathsf{NP}$, since we can non-deterministically "guess" assignment and verify that it satisfies the given 3-CNF-formula.

Second, let

$$\phi = (l_{1,1} \vee l_{1,2} \vee \ldots \vee l_{1,i_1}) \wedge (l_{2,1} \vee l_{2,2} \vee \ldots \vee l_{2,i_2}) \wedge \ldots \wedge (l_{k,1} \vee l_{k,2} \vee \ldots \vee l_{k,i_k}),$$

where $l_{i,j}$ are literals (can contain "not" sign). We convert each clause into "and" of clauses with 3 literals, as follows:

$$\varphi_j = (l_{j,1} \vee l_{j,2} \vee \ldots \vee l_{j,i_j})$$

is converted into

$$\hat{\varphi}_j = (l_{j,1} \vee l_{j,2} \vee z_1) \wedge (\bar{z}_1 \vee l_{j,3} \vee z_2) \wedge (\bar{z}_2 \vee l_{j,4} \vee z_3) \wedge \ldots \wedge (\bar{z}_{i_j-3} \vee l_{i_j-1} \vee l_{i_j}),$$

where z_1, z_2, $\ldots$, z_{i_j-3} are new Boolean variables. Then $f(\phi) = \hat{\varphi}_1 \wedge \hat{\varphi}_2 \wedge \ldots \wedge \hat{\varphi}_k$.

Correctness. Prove that $\phi \in$ SAT if and only if $f(\phi) \in$ 3-SAT. In other words, ϕ is satisfiable if and only if $f(\phi)$ is satisfiable.

Direction 1) Let ϕ be satisfiable, and consider assignment to literals that satisfy ϕ. Consider clause $\varphi_j = (l_{j,1} \vee l_{j,2} \vee \ldots \vee l_{j,i_j})$. There exists $l_{j,r}$ in that assignment such that $l_{j,r} = \text{TRUE}$. Then we choose assignments to $z_1, \ldots, z_{i_j-3}$ such that all clauses with three variables are satisfied:

$$(l_{j,1} \vee l_{j,2} \vee \overset{T}{z_1}) \wedge (\overset{F}{\bar{z}_1} \vee l_{j,3} \vee \overset{T}{z_2}) \wedge (\overset{F}{\bar{z}_2} \vee l_{j,4} \vee \overset{T}{z_3}) \wedge \ldots$$

$$\ldots \wedge (\overset{F}{\bar{z}_{r-2}} \vee \overset{T}{l_{j,r}} \vee \overset{F}{z_{r-1}}) \wedge \ldots \wedge (\overset{T}{\bar{z}_{i_j-3}} \vee l_{j,i_j-1} \vee l_{i_j}).$$

Here T and F stand for TRUE and FALSE, respectively.

Direction 2) Let $f(\phi)$ be satisfiable, and consider assignment to $l_{j,r}$ and z_r that satisfy $f(\phi)$. Then, the same assignment to $l_{j,r}$ satisfies ϕ because there are $i_j - 2$ clauses with three variables, but only $i_j - 3$ variables z_r can be TRUE. So at least one of the clauses is satisfied with $l_{j,r} = \text{TRUE}$.

Polynomiality. $f(\phi)$ can be constructed from ϕ in time polynomial in length of ϕ. □

Practise session

1. **Definition:** given an undirected graph $\mathcal{G}$, *independent set* S is a subset of vertices of $\mathcal{G}$, such that for any two vertices $u, v \in S$, there is no edge between u and v in $\mathcal{G}$.

   Define the language INDEPENDENT-SET:

$$\mathsf{INDEPENDENT\text{-}SET} = \Big\{ \langle \mathcal{G}, k \rangle \mid \mathcal{G} \text{ is an undirected graph, which has an independent set with } k \text{ vertices} \Big\}.$$

   In this exercise, we will show that INDEPENDENT-SET is $\mathcal{NP}$-complete.

   (a) Show that INDEPENDENT-SET $\in \mathcal{NP}$.

   (b) Show that INDEPENDENT-SET is $\mathcal{NP}$-hard.

Solution.

   (a) We use a non-deterministic algorithm, which chooses non-deterministically a set S of k vertices. Then it checks that there is no edge between any two vertices in that set S. If all checks are satisfied – accepts, otherwise – rejects.

   The checks are performed in polynomial time, in particular to check that there are no edges between any two vertices in S requires at most

$O(n^2)$ steps, where n is the number of vertices in $\mathcal{G}$ (here, $O(n^2)$ is an upper bound on the number of pairs of vertices). Consequently, this non-deterministic algorithm is polynomial.

(b) Recall the language CLIQUE defined as follows:

$$\mathsf{CLIQUE} = \{\langle \mathcal{G}, k\rangle \mid \mathcal{G} \text{ is an undirected graph, which has a clique of size } k\} \ .$$

It was shown in the class that CLIQUE on $\mathcal{NP}$-complete.

Define a function f that maps the pair $\langle \mathcal{G}, k\rangle$ to the pair $\langle \overline{\mathcal{G}}, k\rangle$, where $\overline{\mathcal{G}}$ is a complement of the graph $\mathcal{G}$. Specifically, there is an edge between any two vertices u and v in $\mathcal{G}$ if and only if there is no edge between u and v in $\overline{\mathcal{G}}$. The function f can be computed in time polynomial with respect to the number of vertices and edges in the graph $\mathcal{G}$, because we need to look at all vertex pairs in the graph $\mathcal{G}$ (there are polynomial number of them) and reverse the presence/absence of an edge in each pair (can be done in polynomial time).

If $\langle \mathcal{G}, k\rangle \in \mathsf{CLIQUE}$, then the graph $\mathcal{G}$ contains a click S with k vertices. There are edges between any two vertices of S in $\mathcal{G}$. Then, there are no edges between any two vertices of S in $\overline{\mathcal{G}}$, and thus S forms an independent set with k vertices in $\overline{\mathcal{G}}$. Consequently, $\langle \overline{\mathcal{G}}, k\rangle \in \mathsf{INDEPENDENT\text{-}SET}$. Similarly, we get that if $\langle \overline{\mathcal{G}}, k\rangle \in \mathsf{INDEPENDENT\text{-}SET}$, then $\langle \mathcal{G}, k\rangle \in \mathsf{CLIQUE}$ (all logical transitions are invertible).

Polynomiality: we see that f is a polynomial-time reduction from the language CLIQUE to the language $\mathsf{INDEPENDENT\text{-}SET}$.

Since CLIQUE is $\mathcal{NP}$-complete, and $\mathsf{CLIQUE} \leq_{\mathrm{P}} \mathsf{INDEPENDENT\text{-}SET}$, we conclude that $\mathsf{INDEPENDENT\text{-}SET}$ is $\mathcal{NP}$-complete.

2. **Definition:** given an undirected graph $\mathcal{G}$, *vertex cover* C is a subset of vertices of $\mathcal{G}$ such that any edge e in $\mathcal{G}$ has at least one of its endpoints in C.

Define the language $\mathsf{VERTEX\text{-}COVER}$:

$$\mathsf{VERTEX\text{-}COVER} = \Big\{\langle \mathcal{G}, k\rangle \mid \mathcal{G} \text{ is an undirected graph,} \\ \text{which has a vertex cover on } k \text{ vertices}\Big\} \ .$$

In this exercise, we will show that $\mathsf{VERTEX\text{-}COVER}$ is $\mathcal{NP}$-complete.

(a) Show that $\mathsf{VERTEX\text{-}COVER} \in \mathcal{NP}$.

(b) Show that VERTEX-COVER is $\mathcal{NP}$-hard.

Solution.

(a) Similar to the previous exercise, we use a non-deterministic algorithm, which chooses non-deterministically a set C of k vertices. Then it checks that every edge in $\mathcal{G}$ has an endpoint in C. If all checks are satisfied – accepts, otherwise – rejects.

The checks are performed in polynomial time. This requires at most $O(m)$ steps, where m is the number of edges in $\mathcal{G}$. Consequently, this non-deterministic algorithm is polynomial.

(b) In the previous exercise we showed that INDEPENDENT-SET is $\mathcal{NP}$-complete. In this exercise, we use reduction from INDEPENDENT-SET to VERTEX-COVER.

Define a function f that maps the pair $\langle \mathcal{G}, k \rangle$ to the pair $\langle \mathcal{G}, n - k \rangle$, where n is the number of vertices in $\mathcal{G}$. The function f can be computed in time polynomial with respect to the number of vertices and edges in the graph $\mathcal{G}$, because we need only to replace k by $n - k$.

Take a set S with k vertices in $\mathcal{G}$. Denote by C a set of all vertices of $\mathcal{G}$, which are not in S. Its size is $n - k$. Next, we show that S is an independent set if and only if C is a vertex cover.

1. Let S be an independent set. Consider an arbitrary edge e connecting two vertices u and v in $\mathcal{G}$. It is impossible that both $u, v \in S$ (since S is an independent set). Thus, at least one of the vertices u, v is in C, and the edge e is "covered" by C. Thus, C is a vertex cover.
2. Let C be a vertex cover. Then, every edge in $\mathcal{G}$ is "covered" by C. Take an edge e in $\mathcal{G}$. At least one of its endpoints is in C, and therefore at most one of its endpoints is in S (since S and C complement each other). Thus, every two vertices in S are not connected by an edge, and therefore S is an independent set.

We conclude that $\langle \mathcal{G}, k \rangle \in$ INDEPENDENT-SET if and only if $\langle \mathcal{G}, n - k \rangle \in$ VERTEX-COVER.

<u>Polynomiality:</u> we see that f is a polynomial-time reduction from the language INDEPENDENT-SET to the language VERTEX-COVER.

Since INDEPENDENT-SET is $\mathcal{NP}$-complete, and INDEPENDENT-SET $\leq_{\mathrm{P}}$ VERTEX-COVER, we conclude that VERTEX-COVER is $\mathcal{NP}$-complete.