# Logic and Computation I

## Chapter 2. Propositional logic and computational complexity

Kazuyuki Tanaka

BIMSA

October 22, 2024

北京雁栖湖
应用数学研究院
Beijing Institute
of Mathematical
Sciences and Applications

## Logic and Computation I

- **Part 1. Introduction to Theory of Computation**
- **Part 2. Propositional Logic and Computational Complexity**
- **Part 3. First Order Logic and Decision Problems**
- **Part 4. Modal logic**

## Part 2. Schedule

- Oct.10, (1) Tautologies and proofs
- Oct.15, (2) The completeness theorem of propositional logic
- Oct.17, (3) SAT and NP-complete problems
- Oct.22, (4) NP-complete problems about graphs
- Oct.24, (5) Time-bound and space-bound complexity classes
- Oct.29, (6) PSPACE-completeness and TQBF

# Recap

- A Yes/No problem belongs to **P** if there exists a **deterministic** TM and a polynomial $p(x)$ s.t. for any input string of length $n$, it returns the correct answer within $p(n)$ steps.
- A problem belongs to **NP** if there is a **nondet.** TM and a polynomial $p(x)$ s.t. for any input string of length $n$, it stops within $p(n)$ steps.
  - ▷ The answer is Yes, if at least one accepting computation process admits it;
  - ▷ The answer is No, if all the computation processes reject.
- $\mathrm{Q}_1$ is poly(nomial)-time reducible to $\mathrm{Q}_2$, denote $\mathrm{Q}_1 \leq_{\mathbf{p}} \mathrm{Q}_2$, if there exists a polytime algorithm $A$ which solves a problem $q_1$ in $\mathrm{Q}_1$ as problem $A(q_1)$ in $\mathrm{Q}_2$.
- Q is **NP-hard** if for any NP problem Q′, $\mathrm{Q}' \leq_p \mathrm{Q}$.
- An NP-hard NP problem is said to be **NP-complete**.

**Theorem 2.20**

The Cook-Levin theorem: SAT is NP-complete.

We also showed the satifiablity problem SAT restricted to some special Boolean formulas remains NP-complete.

- A variable $x$ and its negation $\neg x$ are called **literals**. A disjunction ($\vee$) of literals is called a **clause**. A conjunction ($\wedge$) of clauses is called a **CNF** (conjunctive normal form).
- **CNF-SAT** is the satisfiability problem for conjunctive normal forms.

### Theorem 2.23

CNF-SAT is NP-complete.

- A CNF with exactly 3 literals in each clause is called a **3-CNF**. **3-SAT** is the satisfiability problem for 3-CNF.

### Theorem 2.24

3-SAT is NP-complete.

**Proof.**

- To show $\mathrm{CNF\text{-}SAT} \leq_p$ 3-$\mathrm{SAT}$, let $\phi$ be a $\mathrm{CNF}$ formula.
- If $\phi$ has a clause $l_1 \vee \cdots \vee l_k (k \geq 4)$, replace it with the following:

  $(l_1 \vee l_2 \vee x_1) \wedge (l_3 \vee \bar{x}_1 \vee x_2) \wedge (l_4 \vee \bar{x}_2 \vee x_3) \wedge \cdots \wedge (l_{k-2} \vee \bar{x}_{k-4} \vee x_{k-3}) \wedge (l_{k-1} \vee l_k \vee \bar{x}_{k-3})$

  where $\bar{x}$ represents $\neg x$.
- For a clause with only one literal $l_1$, replace it with

$$(l_1 \vee x_1 \vee x_2) \wedge (l_1 \vee x_1 \vee \bar{x}_2) \wedge (l_1 \vee \bar{x}_1 \vee x_2) \wedge (l_1 \vee \bar{x}_1 \vee \bar{x}_2).$$

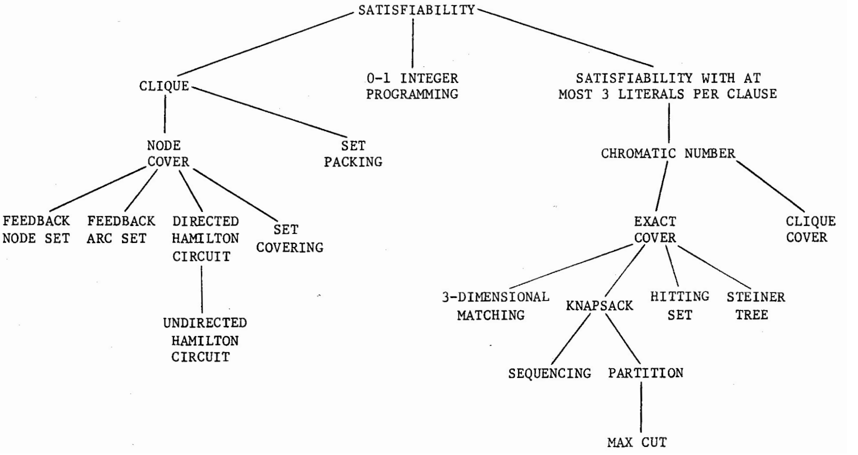
- For a clause with only two literals $l_1 \vee l_2$, replace it with

$$(l_1 \vee l_2 \vee x_1) \wedge (l_1 \vee l_2 \vee \bar{x}_1).$$

- It is easy to see that the satisfiability condition does not change by these transformations.
- Since the above transformations can be performed by polynomial-time algorithms, $\mathrm{CNF\text{-}SAT}$ is polynomial-time reducible to 3-$\mathrm{SAT}$.

# §2.4. NP-complete problems about graphs

- Following Cook's work on SAT, in 1972 R. Karp published a list of 21 NP-complete problems. In addition to 3-SAT, there are Hamiltonian cycle, graph coloring, knapsack problem.



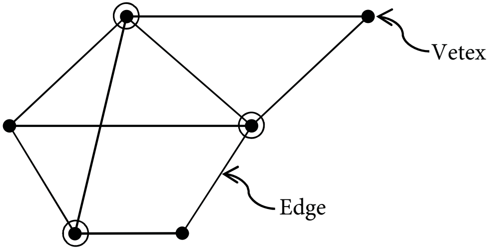
## Definition 2.25

A **directed graph** $G = (V, E)$ consists of a set of vertices $V$ and a set of edges $E \subseteq V \times V$. A graph s.t. $(u, v) \in E \Leftrightarrow (v, u) \in E$ is called an **undirected graph**.

Here, we only consider finite graphs.

## Definition 2.26

A **vertex cover** of an undirected graph $G = (V, E)$ is a subset $S \subseteq V$ of vertices such that for any edge $(u, v) \in E$ of $G$, $u \in S$ or $v \in S$.




The set of ⊚ is a vertex cover, in which every edge has at least one endpoint.

## Definition 2.27

- The **vertex cover problem VC**: Given an undirected graph $G$ and a natural number $k$, decide whether there exists a vertex cover $S$ of $G$ consisting of $k$ vertices.
- The problem of finding the minimum size $k$ of a vertex cover for an undirected graph $G$ is called the **minimum vertex cover problem**.

## Theorem 2.28

The vertex cover problem for undirected graphs is NP-complete.

**Consider how to input $G = (V, E)$ to the TM**

- If the cardinality of $V$ is $n$, $E$ can be represented by an $n \times n$ matrix with components $0, 1$, which is called a **adjacency matrix**. Then, a graph $G$ can be represented by a $0, 1$ sequence of length $n^2$.
- So, a polynomial size of a graph $G$ can be regarded as a polynomial of $V$.

**Proof.**

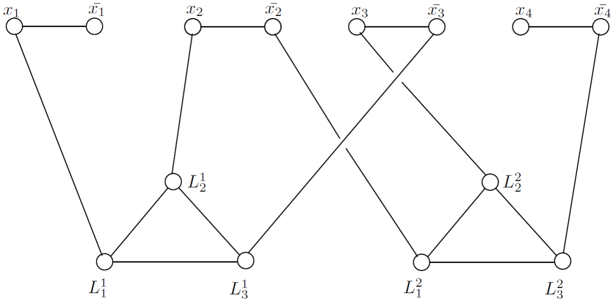
VC is an NP problem

- Choose a arbitrary set $S$ of $k$ vertices and check if it is a vertex cover.
- It is easy to decide (in poly-time) whether $S$ is a vertex cover, since we only need to check that for each edge, one of its endpoints belongs to $S$.

To show it is NP-hard, prove 3-SAT $\leq_p$ VC

- Consider a 3-CNF formula $\varphi = \bigwedge_{j \leq m} (l_1^j \vee l_2^j \vee l_3^j)$. Let $\{x_1, \ldots, x_n\}$ be the variables in $\varphi$. That is, $l_s^j$ $(j \leq m, s \leq 3)$ is $x_i$ or $\overline{x_i}$.
- Then construct the graph $G = (V, E)$ such that
  - $V = \{x_i, \bar{x}_i : i \leq n\} \cup \{L_1^j, L_2^j, L_3^j : j \leq m\}$,
  - $E = \{(x_i, \bar{x}_i) : i \leq n\} \cup \{(L_1^j, L_2^j), (L_2^j, L_3^j), (L_3^j, L_1^j) : j \leq m\}$ $\cup \{(l_s^j, L_s^j) : j \leq m, s \leq 3\}$.

**Proof.** For example, let $G$ be the graph for $\varphi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_2 \vee x_3 \vee \bar{x}_4)$.




- Check whether there is a v.c. $S$ of size $k = n + 2m (= 8)$ for $G$.
- A v.c. must contain $x_i$ or $\bar{x}_i$ for each $i \leq n$, and at least two of $L_1^j, L_2^j, L_3^j$ for each $j \leq m$. Hence, a v.c. of size $k = n + 2m$ will contain exactly one of $x_i$ or $\bar{x}_i$ and exactly two of $L_1^j, L_2^j, L_3^j$.
- So, one of $L_1^j, L_2^j, L_3^j$ is not in $S$, and it must connect with either $x_i$ or $\bar{x}_i$ in $S$.
- Now, put $x_i = \mathrm{T}$ if $x_i \in S$, and $x_i = \mathrm{F}$ if $\bar{x}_i \in S$. Then for each $j$, one of $l_1^j, l_2^j, l_3^j$ is $\mathrm{T}$, thus $l_1^j \vee l_2^j \vee l_3^j = \mathrm{T}$. Therefore, $\varphi$ is satisfiable.
- Conversely, suppose there is a truth value function $f$ satisfying $\varphi$. First, put $x_i$ (or $\bar{x}_i$) into $S$ if $x_i = \mathrm{T}$ (or $\bar{x}_i = \mathrm{T}$) by $f$. Then for each $j$, at least one of $L_1^j, L_2^j, L_3^j$ connects to $x_i$ or $\bar{x}_i$ in $S$. Except one of such, put the other two into $S$, which makes a v.c. $S$ of size $n + 2m$.

# Hamiltonian cycles

Only finite connected graphs are considered here.

**Recall: Eulerian cycles in an undirected graph**

- An **Eulerian path** passes through every edge exactly once. An **Eulerian cycle** is a Eulerian path whose start and end points coincide.
- A graph $G$ has an Eulerian cycle iff the degree of each vertex of $G$ is even.

- A **Hamiltonian cycle** is a cycle passing through every vertex exactly once.
- There is no known simple criterion for the existence of Hamiltonian path.
- R. Karp showed that this problem is NP-complete, which makes it clear that in principle it is difficult to find such a criterion.

## Definition 2.29

**Directed Hamiltonian cycle problem** (**dHAMCYCLE**): for a directed connected graph, decide whether there is a Hamiltonian cycle (passing every vertex exactly once, following the direction of the edges)

## Theorem 2.30

$\mathrm{dHAMCYCLE}$ is NP-complete.

**Proof.** To show $\mathrm{dHAMCYCLE} \in$ NP, choose an arbitrary path of a given directed graph and check whether it is a Hamiltonian cycle.

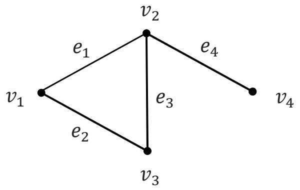
For its NP-completeness, we prove that $\mathrm{VC}$ is poly-time reducible to it.

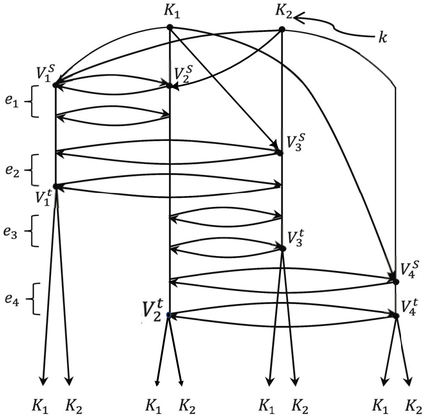
- An undirected graphs $G = (V, E)$ and $k$ are given to check VC.
- We construct a directed graph $G^* = (V^*, E^*)$ s.t. the following are equivalent
  - ▷ $G^*$ has a Hamiltonian cycle.
  - ▷ $G$ has a vertex cover of size $k$.

**Proof.** (VC$\leq_p$dHAMCYCLE, continued)
Consider the graph $G$ in the right figure with $k = 2$. We construct $G^*$ as follows:

- For $k = 2$, two points $K_1, K_2$ are placed at the top.
- Since $G$ consists of 4 vertices $v_i (i \leq 4)$, draw 4 lines $\overrightarrow{V_i^s V_i^t}$ downward. If there is an edge between $v_i$ and $v_j$ in $G$, then a pair of two-way bridges are built so that it can go back and forth between $\overrightarrow{V_i^s V_i^t}$ and $\overrightarrow{V_j^s V_j^t}$.
- Finally, draw a line from each $K_l$ to each $V_i^s$ and from each $V_i^t$ to each $K_l$.

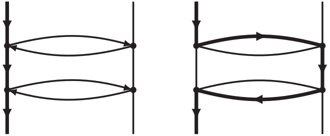
Suppose $G^*$ has a Ham cycle $C$. If $C$ has an edge from $K_l$ to $V_i^s$, then $v_i$ is put into a set $S$ of vertices of $G$. Since there are only $m = 2$ top points $K_l$, the size of $S$ is 2.

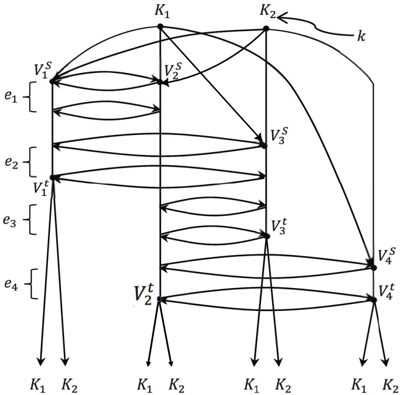


**Proof.** (VC$\leq_p$dHAMCYCLE, continued)

- First note that to go down from a vertex $V_i^s$, $C$ must go in one of the two ways shown in the right figures (straight and detour).
- Suppose $C$ enters $V_i^s$ from $K_1$. If $C$ does not enter $V_j^s (j \neq i)$ from $K_2$, then there must be a pair of double bridges between $\overline{V_i^s V_i^t}$ and $\overline{V_j^s V_j^t}$, and $C$ detours across them. Otherwise, goes down straight along $\overline{V_i^s V_i^t}$.
- Thus, if $C$ makes a detour, just one end of the corresponding edge belongs to $S$; otherwise, both endpoints are in $S$. In any case, $S$ is a vertex cover of $k$ vertices.





**Proof.** ($\mathrm{VC} \leq_p \mathrm{dHAMCYCLE}$, continued)

- Conversely, if a vertex cover $S$ of $k$ vertices is given first, we can make a Hamiltonian circuit $C$ by entering $V_i^s$ from $K_l$ and choosing appropriate detours from $\overline{V_i^s V_i^t}$ for $v_i \in S$. Therefore,
  $$G \text{ has a vertex cover of size } k \Leftrightarrow G^* \text{ has a Hamiltonian cycle.}$$
- The above argument can be generalized to any graphs. We omit this routine work.
- Although $G^*$ looks much larger than $G$, it can be obtained by a polynomial-time algorithm. In fact, the number of vertices of $G^*$ is a constant multiple of the number of edges of $G$.
- That is, $\mathrm{VC}$ is polynomial-time reducible to $\mathrm{dHAMCYCLE}$.

From this result, we can also show that the decision problem on the existence of Hamiltonian cycles for undirected graphs is NP-complete.

**Exercise 2.4.1**

Show that the decision problem of the existence of Hamiltonian cycles for undirected graphs $(\mathrm{HAMCYCLE})$ is NP-complete.

**Hint**. It is sufficient to show $\mathrm{dHAMCYCLE} \leq_{\mathsf{q}} \mathrm{HAMCYCLE}$. Given a directed graph $(V, E)$, we can constructed a undirected graph $(V', E')$.

- $V' = \{v^{\mathsf{mid}}, v^{\mathsf{in}}, v^{\mathsf{out}} \mid v \in V\}$.
- $E' = \bigcup_{v \in V} \Big\{\{v^{\mathsf{in}}, v^{\mathsf{mid}}\}, \{v^{\mathsf{mid}}, v^{\mathsf{out}}\}\Big\} \cup \bigcup_{(u,v) \in E} \Big\{\{u^{\mathsf{out}}, v^{\mathsf{in}}\}\Big\}$.

- The **Traveling Salesman Problem** (**TSP**) is a variation of the Hamiltonian cycle problem.
  - A weight (distance) assigned for each edge of an undirected graph.
  - Is it possible to traverse all the points so that the sum of the weights of the passed edges does not exceed a given limit $k$?
  - Equivalently, does there exist a Hamiltonian cycle such that the sum of edge weights is less than or equal to $k$?

It can be shown that $\mathrm{TSP}$ is also NP-complete.

- For $\mathrm{TSP}$ to be NP, choose an arbitrary path and check whether it satisfies the condition or not.
- For the reversal, the existence of a Hamiltonian cycle is the existence of a $\mathrm{TSP}$ solution with edge weight 1 and sufficiently large $k$, and so

$$\mathrm{HAMCYCLE} \leq_p \mathrm{TSP}.$$

# Summary

- We have shown that the vertex cover problem $\mathrm{VC}$ and the directed Hamiltonian cycle problem dHAMCYCLE are NP-complete.

**Further readings**

M. Sipser, Introduction to the Theory of Computation, 3rd ed., Course Technology, 2012

Thank you for your attention!