# Recall
## Undecidability

**Undecidability** : There is no algorithm that can accurately return "yes" or "no" for a given input in a finite amount of time.

**the diagonalization method** : Used by Cantor to prove that some sets are (un)countable.

**Correspondence** We have 2 sets: A and B. We say that a function is one-to-one (injective) if $(\forall) a \in A, b \in B$ $f(a) \neq f(b) \iff a \neq b$. The function is onto (surjective) if $(\forall) b \in B, (\exists) a \in A$ such that $f(a) = b$. If both conditions are met, we say that the sets are the same size and there is a correspondence (bijective) between them.

**Countable** : A set is countable if it has a finite amount of elements or it has the same size as $\mathbb{N}$.

**Example** Let's say we have $\mathbb{N} = \{1, 2, 3, \dots\}$ and $\mathcal{E} = \{2, 4, 6, \dots\}$. There is a simple correspondence between these 2 sets given by $f: \mathbb{N} \to \mathcal{E}$ ; $f(n) = 2n$

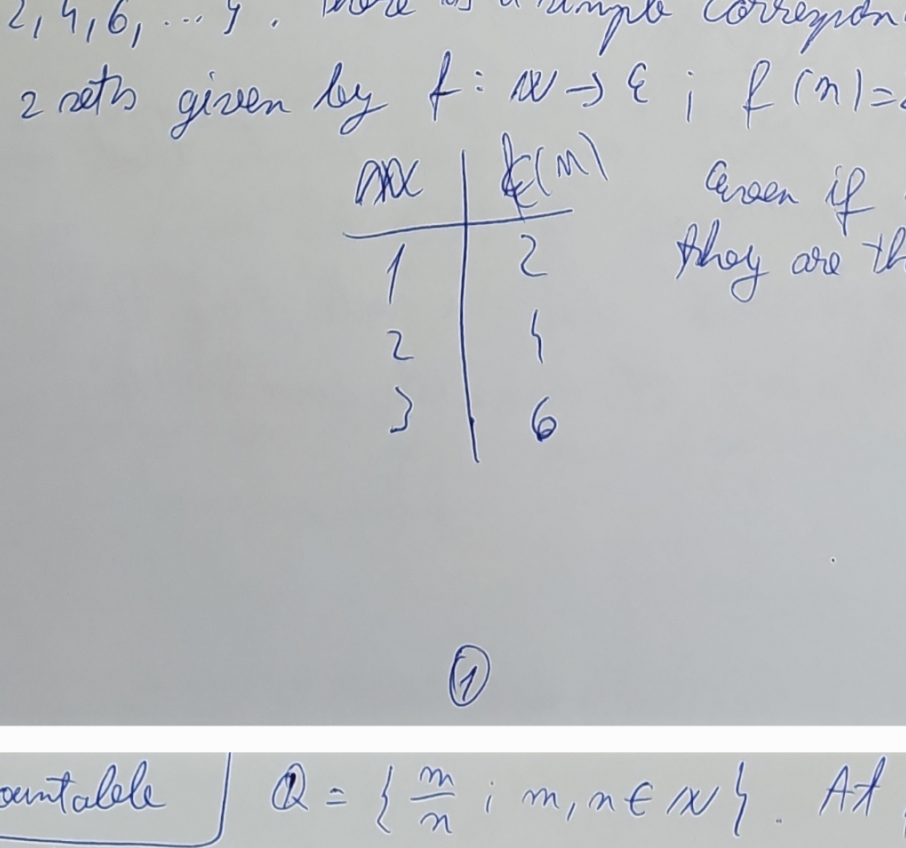
| $\mathbb{N}$ | $f(n)$ |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |

Even if $\mathbb{N}$ seems bigger they are the same size.



**$\mathbb{Q}$ is countable** $\mathbb{Q} = \{\frac{m}{n} ; m, n \in \mathbb{N}\}$. At first, it might seem that it is much bigger than $\mathbb{N}$, thus uncountable. That is not the case. We create a matrix, where $M[i,j] = \frac{i}{j}$. A bad approach would be to start and continue mapping on the first line / column, since they are infinite and we'll never reach the next one.

A solution is to use the diagonalization method, starting from the first cell, going to the next diagonal and so on. We skip elements already mapped ($\frac{1}{1}$ and $\frac{2}{2}$)




Using this proof, one might think that any two sets could be proven to have the same size, but that is not the case. Next, we will prove that $\mathbb{R}$ is uncountable using the same diagonalization method.

/ R is uncountable / We suppose a correspondence function exists $f: \mathbb{N} \to \mathbb{R}$. We will prove the opposite using ~~contra~~ contradiction. Suppose we have a mapping to some random numbers.

| n | f(n) |
|---|---|
| 1 | 3,14159... |
| 2 | 55,55555... |
| 3 | 0,12345... |
| 4 | 123,4567 89... |

Construct a new number, x by the following rules:

1. $x \in (0,1)$
2. $d(x,i) \neq d(f(i),i)$ ; $i \in \mathbb{N}$

$d(x,i)$ → returns the i-th digit after the decimal of x.

$x = 0,2345...$ Now, we have created a number which is not mapped. This can be done infinitely many times, thus we have a contradiction. $\mathbb{R}$ is uncountable since there is no mapping between $\mathbb{N}$ and $\mathbb{R}$.

Having proved this, we can translate it into the realm of languages and Turing machines. Each TM can recognize only a single language. ~~There~~ The set of TM's is finite and there are infinitely many languages. Thus, some languages are not Turing-recognizable.



## Some languages are not Turing-recognizable

Each TM M is encoded into a string $\langle M \rangle$. If we take out all of the illegal encodings, we remain with a finite list of all TM's.

Over the alphabet $\Sigma$, the set of all strings $\Sigma^*$ is countable. Write down all strings of length 0, 1, 2, etc.

The set of all binary sequences is uncountable. A binary seq is made up from 0s and 1s. Name the set of all of these B. B can be shown is uncountable similarly to the previous proof ($\mathbb{R}$ is uncountable).

Let $\mathcal{L}$ be the set of all languages over the alphabet $\Sigma$. We show that $\mathcal{L}$ is uncountable by creating a correspondence between $\mathcal{L}$ and B. $\Sigma^* = \{s_1, s_2, s_3, \dots\}$ each sequence in A has a unique sequence in B. $\chi_A[i] = 1$ if $s_i \in A$, else 0. $\chi_A$ = characteristic sequence. Say that A is a lang over $\{0,1\}$, all strings starting with 0.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $\Sigma^*$ | $\varepsilon$ | 0 | 1 | 00 | 01 | 10 | 11 | 000 ... |
| A | | 0 | | 00 | 01 | | | 000 ... |
| $\chi_A$ | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 ... |

$f: \mathcal{L} \to B$
$f(A) = \chi_A$
B uncountable

$\Rightarrow \mathcal{L}$ uncountable $\Rightarrow$ Some langs are not Turing-recognizable

## $A_{TM}$ is undecidable

$A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \}$

Suppose $A_{TM}$ is decidable. We will prove the contrary by contradiction. Define a new TM H, a decider, following the rule

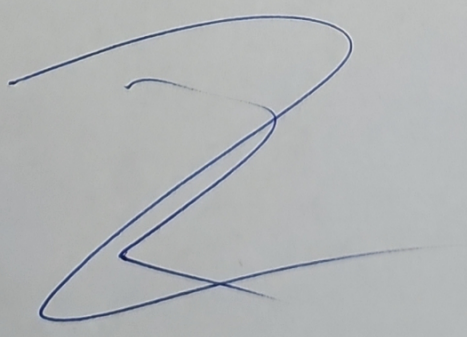
$$H(\langle M, w \rangle) = \begin{cases} \text{accept, if } M \text{ accepts } w \\ \text{reject, if } M \text{ does it accept } w \end{cases}$$

Define another TM D, that has H as its subroutine. D takes another TM as input and it will return the opposite of what H returns.

$$D(\langle M \rangle) = \begin{cases} \text{accept; if } H(M, \langle M \rangle) \text{ rejects} \\ \text{reject; if } H(M, \langle M \rangle) \text{ accepts} \end{cases}$$

$$D(\langle D \rangle) = \begin{cases} \text{accept; if } D \text{ rejects } D \\ \text{reject; if } D \text{ accepts } D \end{cases}$$

clear contradiction. Neither D nor H can exist





Recall
NP-completeness

1. Satisfiable boolean formula
2. Polynomial time reducibility
3. (3) CNF-formula
4. NP-complete
5. Cook-Levin theorem
6. 3SAT is NP-complete

The first problem proved to be NP-complete is the satisfiability problem. Variables can be true or false

True = 1
False = 0

AND = $\wedge$
OR = $\vee$
NOT = $\neg$ / $\bar{x}$

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

**Def** A boolean formula is satisfiable if there is a variable assignment such that $\phi$ is True.

$SAT = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable boolean formula} \}$

SAT $\in$ P iff P = NP

---

**Def** A function $f: \Sigma^* \to \Sigma^*$ is a polynomial time computable function if there is a polynomial time TM, M that halts with just $f(w)$ on its tape when w is given as an input.

**Def** Language $A$ is polynomial time reducible to $B$ ($A \leq_p B$) if there exists a function $f: \Sigma^* \to \Sigma^*$ such that $\forall w \in A$ ; $w \in A \Leftrightarrow f(w) \in B$

$f$ is a polynomial time reduction from $A$ to $B$

**Theorem** If $A \leq_p B$ and $B \in P$, then $A \in P$

$M$ = polynomial time algorithm deciding $B$
$N$ = polynomial time algorithm deciding $A$
$f$ = polynomial time reduction from $A$ to $B$

$N$ = "on input $w$
1. Compute $f(w)$
2. Run $M$ on $f(w)$ and output whatever $M$ outputs"

$w \in A$, whenever $f(w) \in B$ since $f$ is a reduction from $A$ to $B$. Thus, $M$ accepts $f(w)$ whenever $w \in A$. $N$ runs in polynomial time since it consists of two operation running in polynomial time.

---

3SAT = a special case of the satisfiability problem where all formulas are in a special form.

Literals = (negation) Boolean variables
clause = Literals + $\vee$s
A formula is in conjunctive normal form (cnf-formula) if clauses + $\wedge$s



3 cnf-formula = cnf-formula where all clauses have 3 literals

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x_4} \vee x_5 \vee \bar{x_6})$$

3SAT = $\{\langle \phi \rangle$ $\phi$ is a satisfiable 3 cnf-formula $\}$

Now, we prove that 3SAT is polynomial time reducible to CLIQUE

The ~~function~~ polynomial time reduction function that we demonstrate from 3SAT to CLIQUE will convert formulas to graphs.

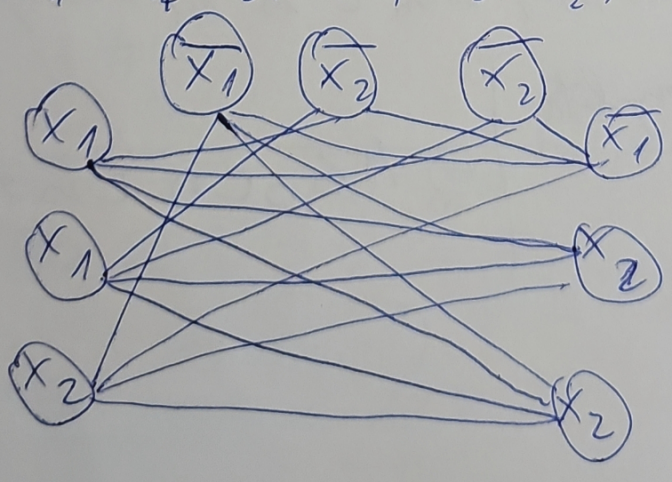
**Proof** Let $\phi$ a formula with $k$ clauses.

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \dots \wedge (a_k \vee b_k \vee c_k)$$

The reduction generates $\langle G, k \rangle$, where $G$ is constructed as follows.

$G$ has $k$ groups of 3 nodes each, corresponding to each clauses ant each literal. There are two rules for connecting nodes. All nodes are connected apart from
1. nodes from the same triple / clause
2. nodes with contradictory labels ($x_1$ and $\bar{x_1}$)

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x_1} \vee \bar{x_2} \vee \bar{x_2}) \wedge (\bar{x_1} \vee x_2 \vee x_2)$$



**Proof** We show that $\phi$ is satisfiable iff $G$ has a $k$-clique.

Suppose $\phi$ is satisfiable. We have at least 1 true literal in each clause. Select one random true literal from each clause. The nodes selected can't be from the same clause since we selected one from each and they can't have contradictory labels, since we chose only true literals. Thus, it is a $k$-clique $\Rightarrow$ $G$ has a $k$-clique.

Suppose $G$ has a $k$-clique. Assign the value true to each of the literals in the clique. They can't be from the same triple and can't be contradictory. Thus, we created a formula $\phi$ which has at least one true variable in each of its clauses $\Rightarrow$ $\phi$ is satisfiable.

**Def** A language $B$ is NP-complete if

1. $B$ is in NP
2. Every $A$ in NP is polynomial time reducible to $B$

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

**Theorem** If $B$ is NP-complete and $B \leq_p C$ for $C$ in NP, then $C$ is NP-complete

**Proof** $B$ is NP-complete
$A \leq_p B$
$B \leq_p C$
$\Rightarrow A \leq_p C \Rightarrow C$ is NP-complete



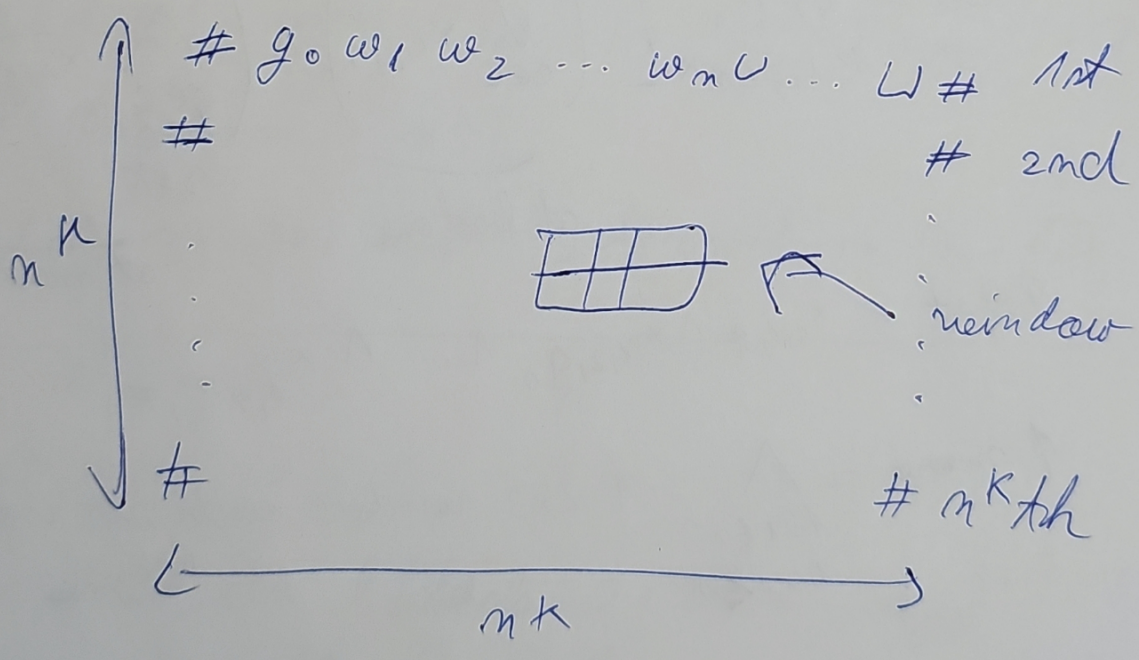
**Cook-Levin theorem** SAT is NP-complete

**Proof** First, we show that SAT is in NP. A nondet polynomial time machine can guess an assignment to $\phi$ and accept if the assignment satisfies $\phi$.

Next, we take any language $A$ and show that it is polynomial time reducible to SAT.

Let $N$ be a nondet. TM that decides $A$ in $n^k$ time. For convenience assume that $N$ runs in $n^k - 3$ time.

A tableau for $N$ is a $n^k \times n^k$ table, for which each row is a config of a branch of the computation of $N$ on input $w$.



Assume every config. starts and ends with #. The first row is the starting configuration of N on w, and each next row is generated based on the transition function. A tableau is accepting if any row in the configuration is accepting

N accepts w $\iff$ accepting tableau for N on w exists.

Q = state set $\quad C = Q \cup \Gamma \cup \{\#\}$

$\Gamma$ = tape alphabet $\quad x_{i,j,s}$ = variable

cell $[i,j] = s$ if $x_{i,j,s} =$ True

$\phi = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$

$\phi_{cell} = \bigwedge_{1 \le i,j \le n^k} \left[ \left( \bigvee_{s \in C} x_{i,j,s} \right) \wedge \left( \bigwedge_{\substack{s \in C \\ t \in C \\ s \neq t}} \overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right) \right]$

(at least one; at most one)

$\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \dots \wedge x_{1,n^k,\#}$

$\phi_{move} = \bigwedge_{\substack{1 \le i < n^k \\ 1 < j < n^k}}$ ( 2×3 window with top row center in $(i,j)$ is legal )

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



$\delta(q_1, a) = \{(q_1, b, R)\}$

$\delta(q_1, b) = \{(q_2, a, R), (q_2, b, L)\}$

Legal window

| $q_1$ | a | b |
|---|---|---|
| b | $q_1$ | b |

Illegal window

| a | b | b |
|---|---|---|
| c | b | b |

$\phi_{move} = \bigwedge_{\substack{1 \le i < n^k \\ 1 < j < n^k}} \left( \bigvee_{a_1 \dots a_6 \text{ is a legal window}} (x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge \dots \wedge x_{i+1,j+1,a_6}) \right)$

---

size $(\phi) = O(n^{2k})$ ($n^{2k}$ cells, $l$ variables)

size $(\phi_{cell})$ = size $(\phi_{move})$ = size $(\phi_{accept}) = O(n^{2k})$

size $(\phi_{start}) = O(n^k)$

$\Rightarrow$ size $(\phi) = O(n^{2k}) \Rightarrow \phi$ is polynomial in n $\square$

---

3SAT is NP-complete.

Convert SAT to 3CNF. $(a_1 \vee a_2 \vee a_3 \vee a_4) = (a_1 \vee a_2 \vee z_1) \wedge (\overline{z_1} \vee a_3 \vee a_4)$

If a clause has $l$ literals $a_1 \vee a_2 \vee \dots \vee a_l$

Replace it with $l-2$ clauses

$(a_1 \vee a_2 \vee z_1) \wedge (\overline{z_1} \vee a_3 \vee a_4) \wedge \dots \wedge (\overline{z_{l-3}} \vee a_{l-1} \vee a_l)$