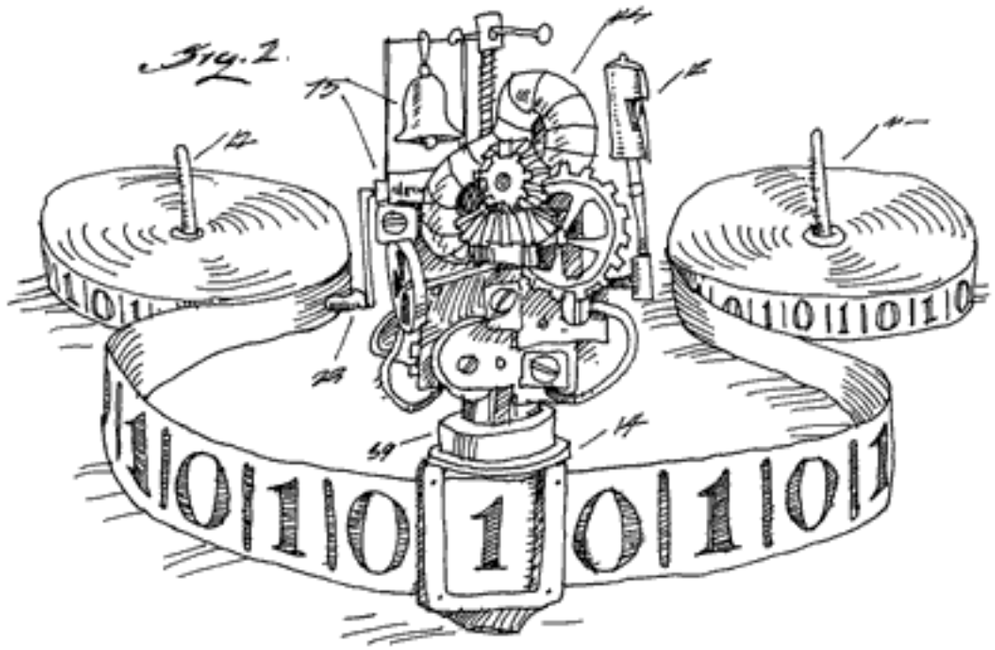
CSC4010: Introduction to Theoretical Computer Science



Lecture 13

P = class of problems for which for the most has worst-case time complexity of polynomial-time.

Recap

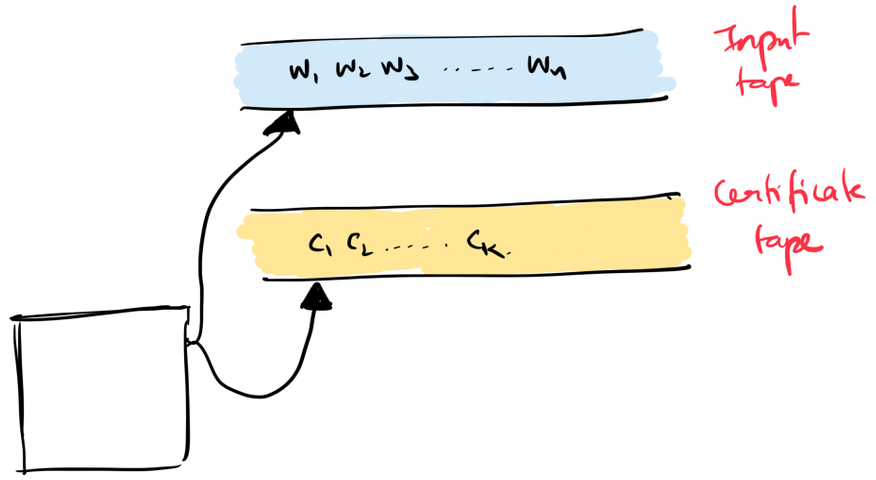
- A Verifier is a Turing machine with a read-only certificate tape, in addition to an input tape. The input to the verifier is a pair (x, c), with x, the input string written on the input tape, and c, the certificate written on the certificate tape.

V(x, c) .

- **Definition:** The language $L(V)$ recognised by the verifier V is

$$L(V) = \{x \in \{0,1\}^* : \exists c \in \{0,1\}^* \text{ s.t. } V \text{ accepts } (x, c)\}$$

The verifier decides $L(V)$ if it halts on all inputs (x, c).

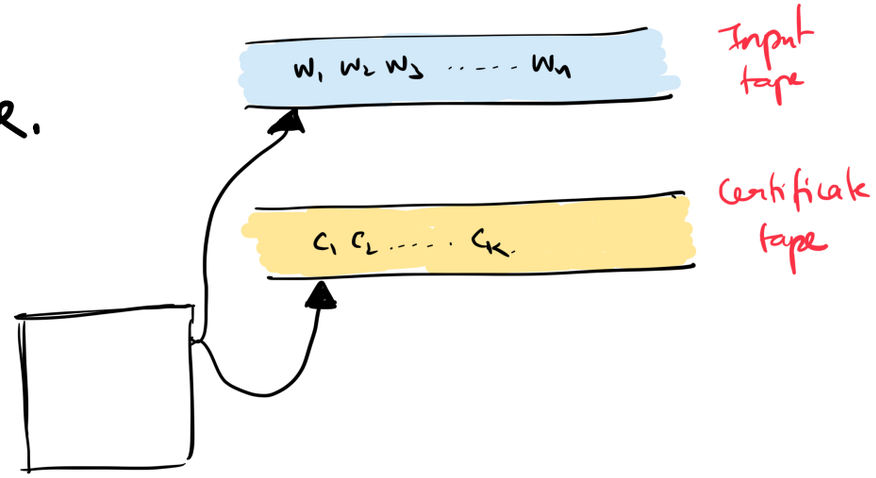


Verifier: As a Turing Machine

Definition: A polynomial-time verifier is a verifier that decides its language and has time cost $O(n^c)$ for some constant $c \geq 0$.

→ decides $L(V)$ in poly-time.

$L(V)$ is the language recognized by the verifier.



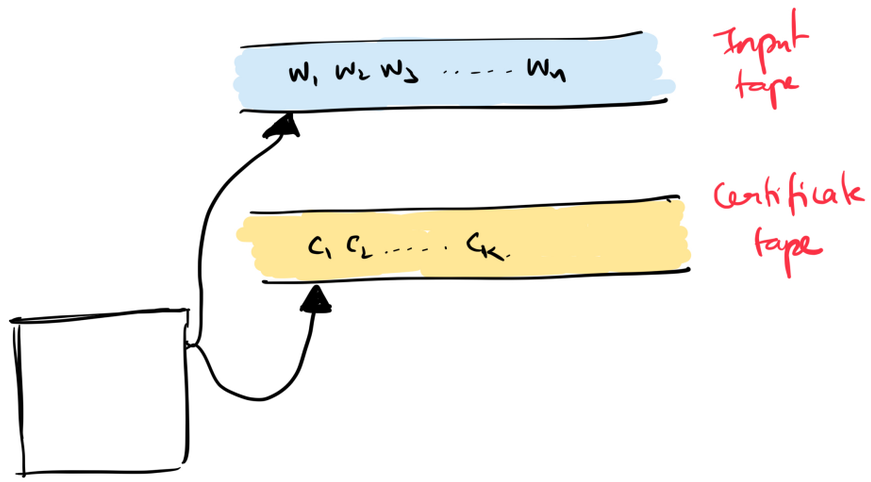
Thus, for any polynomial-time verifier V and $x \in L(V)$, there must be a certificate of length polynomial in $|x|$ that causes V to accept.

(Completeness: $\forall x \in L(V)$, $\exists c \in \{0,1\}^{p(|x|)}$ that causes $V(x,c)$ accept in time $\leq (|x|)$.

Soundness: $\forall x \notin L(V)$ there is no such $c \in \{0,1\}^{p(|x|)}$)

Verifier: As a Turing Machine

Definition: A polynomial-time verifier is a verifier that decides its language and has time cost $O(n^c)$ for some constant $c \geq 0$.



Thus, for any polynomial-time verifier V and $x \in L(V)$, there must be a certificate of length polynomial in $|x|$ that causes V to accept.

Definition: NP is the is the set of all languages that can be decided by polynomial-time verifiers.
NP stands for non-deterministic polynomial time.

Polynomial-time Reducibility

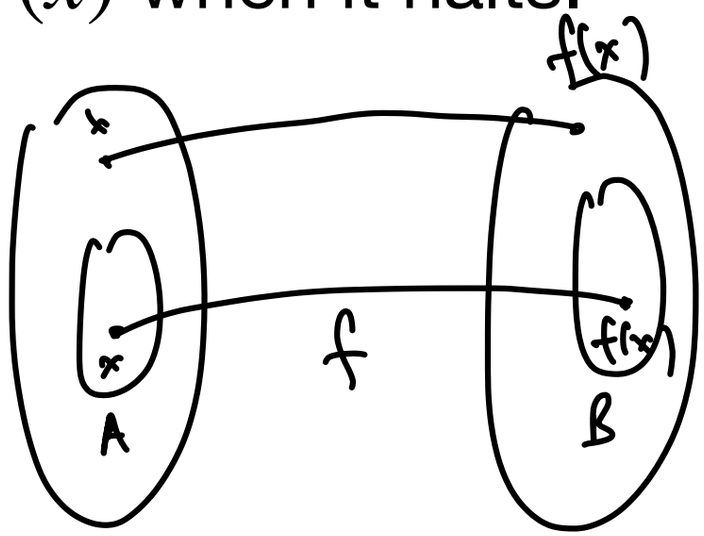
Definition: A language $A \subseteq \{0,1\}^*$ is polynomial-time reducible to the language B, denoted $A \leq_P B$ if and only if there exists a function $f : \{0,1\}^* \to \{0,1\}^*$ such that:

- For every $x \in \{0,1\}^*$, we have $x \in A \iff f(x) \in B$,

mapping reducibility.

- There is a polynomial time Turing machine F, that on any input $x \in \{0,1\}^*$, replaces it with $f(x)$ when it halts.

in polynomial time.



f: computable in polynomial time.

Polynomial-time Reducibility

Theorem: For every two languages $A, B \subseteq \{0,1\}^*$ that satisfy $A \leq_P B$:

- If $B \in P$, then $A \in P$ as well, and
- If $B \in NP$, then $A \in NP$ as well.

Proof: (1) Given a TM deciding B in polynomial time: M_B

(2) $F: x \in A \Leftrightarrow f(x) \in B$ and it computes f in polynomial time.

M_A: on input x:

(a) Run $F(x)$ to compute $f(x)$

(b) Input $f(x)$ to M_B and output the answer.

$M_A(x) = 1 \Leftrightarrow M_B(f(x)) = 1 \Leftrightarrow f(x) \in B \Leftrightarrow x \in A$. ($M_A$ is a decider for A).

(2)

M_A runs in polynomial time $\Rightarrow$ ~~Acc~~.

Runtime of M_A:

(1) Runtime of $F(x)$: $O(|x|^k)$ for constant k

(2) $|f(x)| \leq O(|x|)^k$.

(3) Runtime of M_B was $O(n^{k_1}) \Rightarrow M_B(f(x)) = O(|x|^{k \cdot k_1})$

Polynomial-time Reducibility

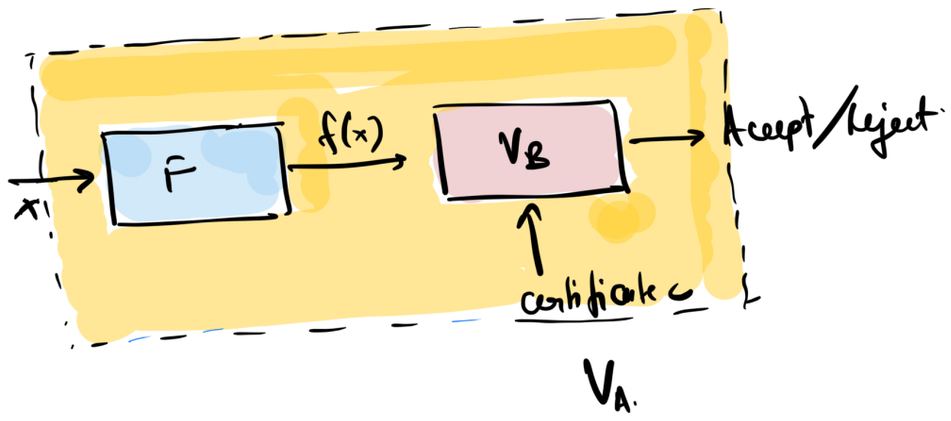
Theorem: For every two languages $A, B \subseteq \{0,1\}^*$ that satisfy $A \leq_P B$:

- If $B \in P$, then $A \in P$ as well, and
- If $B \in NP$, then $A \in NP$ as well.

→ then, $\exists$ a unified for B, V_B that decides B in polynomial time:

$\forall y \in B$, $\exists c$ s.t. $|c| \leq |y|^k$ and $V_B(y,c)$ to accept in time $|x|^\ell$.

$\forall y \notin B$, $\nexists$ no such c of $|c| \leq |y|^4$.



V_A on i/p x certificate c :

(1) Compute $f(x)$

(2) Run $V_B(f(x), c)$

(3) Accept iff $V_B(f(x), c)$ accepts.

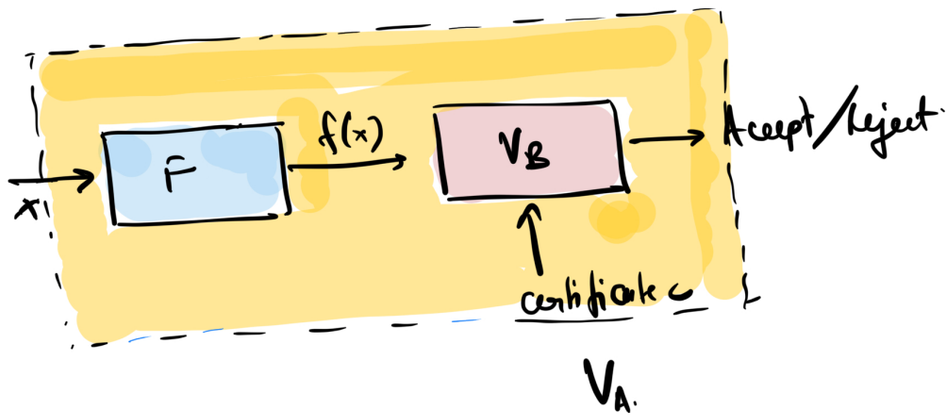
Goal: 1) Soundness & Completeness for A is satisfied by V_A.

2) Certificate and runtime are polynomial in $|x|$.

Polynomial-time Reducibility

Theorem: For every two languages $A, B \subseteq \{0,1\}^*$ that satisfy $A \leq_P B$:

- If $B \in P$, then $A \in P$ as well, and
- If $B \in NP$, then $A \in NP$ as well.



V_A on i/p x certificate c :

(1) Compute $f(x)$

(2) Run $V_B(f(x), c)$

(3) Accept iff $V_B(f(x), c)$ accepts.

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

$\Leftrightarrow \exists c$ s.t. $V_B(f(x), c)$ accepts

$\Leftrightarrow \exists c$ s.t. $V_A(x, c)$ accepts

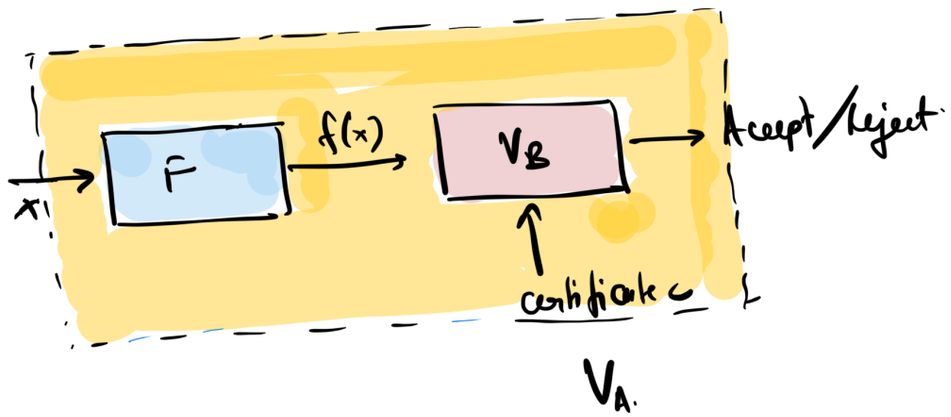
(Completeness & Soundness)

(1) If $x \in A$, then $\exists c$ for x, which is the same as the certificate for $f(x) \in B$.

(2) If $x \notin A$, then there is no such certificate as there is no such $f(x) \notin B$.

Polynomial-time Reducibility

Theorem: For every two languages $A, B \subseteq \{0,1\}^*$ that satisfy $A \leq_P B$:
- If $B \in P$, then $A \in P$ as well, and
- If $B \in NP$, then $A \in NP$ as well.



V_A on i/p x certificate c :
(1) Compute f(x)
(2) Run $V_B(f(x), c)$
(3) Accept iff $V_B(f(x), c)$ accepts.

$A \in NP$. $\Leftarrow$

Claim: (1) Running time of V_B is polynomial.
(2) $|c| \leq p(|x|)$ for some polynomial p.

Proof: (1) F runs in polynomial time: $O(|x|^k)$
(2) $|f(x)| \leq O(|x|^k)$
(3) $|c| \leq |f(x)|^\ell \leq |x|^{k\ell}$
(4) Running time of V_B: $|f(x)|^m$

c is a certificate for $f(x) \in B$.

$|x|^{k+m}$.

NP-Hardness

Lis unentscheidbar, A ist unentscheidbar $\Rightarrow A \leq_m L$.

Definition: The language L is NP-hard if every language $A \in NP$ satisfies $A \leq_P L$. The language L is NP-complete if $L \in NP$ and L is NP-hard.

Proposition: If any NP-hard language L is in P, then $P = NP$.
Equivalently, if $P \neq NP$ and L is NP-hard, then $L \notin P$.

(?) If L is NP-hard, and $L \in P$, then,

$\Downarrow$

$\forall A \in NP,\ A \leq_P L.$

$\Downarrow$

If $L \in P \Rightarrow A \in P$.

$\Rightarrow$ Every language in NP is in $P \Rightarrow P = NP$.

A First NP Complete Language

Theorem: There exists an NP-complete language.

A First NP Complete Language

Theorem: There exists an NP-complete language.

Proof: $A_{TM}^{\text{cert}} = \{\langle M \rangle x 0^m 1^t \mid \exists c \text{ with } |c| \leq m \text{ s.t. } M \text{ accepts } (x, c) \text{ in } t \text{ steps}\}$

(1) A verifier: (a) runs in polynomial time
✓ (b) $\forall \langle M \rangle x 0^m 1^t \in A_{TM}^{cert}$, $\exists c$ s.t.
V accepts this.
$|c| \leq |\langle M \rangle x 0^n 1^t|$

A First NP Complete Language

Theorem: There exists an NP-complete language.

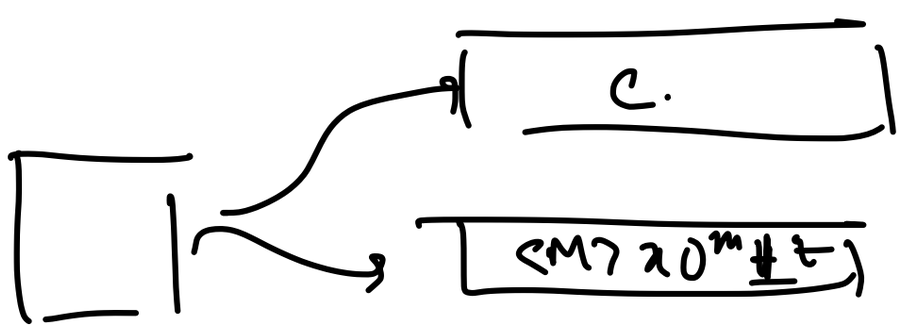
Proof: $A_{TM}^{\text{cert}} = \{\langle M \rangle x 0^m 1^t \mid \exists c \text{ with } |c| \leq m \text{ s.t. } M \text{ accepts } (x, c) \text{ in } t \text{ steps}\}$

$A_{TM}^{\text{cert}} \in NP$:

c is also a certificate for membership.

(1) Consider $\langle M \rangle x 0^m 1^t \in A_{TM}^{\text{cert}}$. There exists c of length m such that $M(x, c) = 1$ in t steps.

(2) The verifier V for A_{TM}^{cert} simulates M on (x, c) for t steps.



$V(\langle M \rangle x 0^m 1^t, c)$ accepts $\Leftrightarrow$ $M(x,c)$ accepts.

(1) Soundness & completeness:
$\forall \langle M \rangle x 0^m 1^t \in \{0,1\}^*$, $\exists c$ $|c| \leq m$ s.t. $M(x,c)$ accepts $\Leftrightarrow$ $\langle M \rangle x 0^m 1^t \in A_{TM}^{\text{cert}}$.

A First NP Complete Language

Theorem: There exists an NP-complete language.

Proof: $A_{TM}^{\text{cert}} = \{\langle M\rangle x 0^m 1^t \mid \exists c \text{ with } |c| \leq m \text{ s.t. } M \text{ accepts } (x, c) \text{ in } t \text{ steps}\}$

$A_{TM}^{\text{cert}} \in NP$:

(1) Consider $\langle M\rangle x 0^m 1^t \in A_{TM}^{\text{cert}}$. There exists c of length m such that $M(x, c) = 1$ in t steps.

(2) The verifier V for A_{TM}^{cert} simulates M on (x, c) for t steps.

Completeness: For every $\langle M\rangle x 0^m 1^t \in A_{TM}^{\text{cert}}$ there exists such a certificate c.

Soundness: For every $\langle M\rangle x 0^m 1^t \notin A_{TM}^{\text{cert}}$ there is no such certificate c .

Polynomial Time: $|c| \leq m$, each step of M can be be simulated in polynomial time.

|| at most polynomial in i/p length.

Used the total # steps ≤ t

⇒ Runtime is polynomial in |<M>x 0^m 1^t|

A First NP Complete Language

Theorem: There exists an NP-complete language.

Proof: $A_{TM}^{\text{cert}} = \{\langle M \rangle x 0^m 1^t \mid \exists c \text{ with } |c| \leq m \text{ s.t. } M \text{ accepts } (x, c) \text{ in } t \text{ steps}\}$

$L \leq_p A_{TM}^{cert}$.

A_{TM}^{cert} **is NP-hard:** Consider any $L \in NP$. There exists polynomials p, q and a verifier V such that for every $x \in \{0,1\}^n$:

(1) If $x \in L$, there exists $u \in \{0,1\}^{p(n)}$ that causes $V(x, u)$ to accept.

(2) If $x \notin L$, every possible $u \in \{0,1\}^{p(n)}$ causes $V(x, u)$ to reject.

(3) V has time cost $q(n)$.

$x \in \{0,1\}^n \longrightarrow$ an instance of A_{TM}^{cert}.

A First NP Complete Language

Theorem: There exists an NP-complete language.

Proof: $A_{TM}^{\text{cert}} = \{\langle M \rangle x 0^m 1^t \mid \exists c \text{ with } |c| \leq m \text{ s.t. } M \text{ accepts } (x, c) \text{ in } t \text{ steps}\}$

A_{TM}^{cert} is NP-hard: Consider any $L \in NP$. There exists polynomials p, q and a verifier V such that for every $x \in \{0,1\}^n$:

(1) If $x \in L$, there exists $u \in \{0,1\}^{p(n)}$ that causes $V(x, u)$ to accept.

(2) If $x \notin L$, every possible $u \in \{0,1\}^{p(n)}$ causes $V(x, u)$ to reject.

(3) V has time cost $q(n)$.

is f polynomial computable?

$$f(x) = \langle V \rangle x 0^{p(|x|)} 1^{q(|x|)}$$

$f(x) \in A_{TM}^{cert} \Leftrightarrow \exists c$ s.t. V accepts x in time $q(|x|)$ and $|c| \leq p(n)$

$\Leftrightarrow x \in L$.

A First NP Complete Language

Theorem: There exists an NP-complete language.

Proof: $A_{TM}^{\text{cert}} = \{\langle M \rangle x 0^m 1^t \mid \exists c \text{ with } |c| \leq m \text{ s.t. } M \text{ accepts } (x, c) \text{ in } t \text{ steps}\}$

A_{TM}^{cert} **is NP-hard:** Consider any $L \in NP$. There exists polynomials p, q and a verifier V such that for every $x \in \{0,1\}^n$:

(1) If $x \in L$, there exists $u \in \{0,1\}^{p(n)}$ that causes $V(x, u)$ to accept.

(2) If $x \notin L$, every possible $u \in \{0,1\}^{p(n)}$ causes $V(x, u)$ to reject.

(3) V has time cost $q(n)$.

$$f(x) = \langle V \rangle x 0^{p(|x|)} 1^{q(|x|)}$$

f: is polytime computable:
$|x|^k$ for some k.
(a) $|\langle V \rangle|$ is a constant.
(b) appending x, $0^{p(|x|)} 1^{q(|x|)}$

$\forall L \in NP$,
$L \leq_p A_{TM}^{cert}$.
$\Rightarrow A_{TM}^{cert}$ is NP-hard.

$\Rightarrow$ total time is $2(|x|)$ where
$2(|x|) = p(|x|) + q(|x|) + |x| + |\langle V \rangle|$.

A First NP Complete Language

Theorem: There exists an NP-complete language.

Proof: $A_{TM}^{\text{cert}} = \{\langle M \rangle x 0^m 1^t \mid \exists c \text{ with } |c| \leq m \text{ s.t. } M \text{ accepts } (x, c) \text{ in } t \text{ steps}\}$

A_{TM}^{cert} is NP-hard: Consider any $L \in NP$. There exists polynomials p, q and a verifier V such that for every $x \in \{0,1\}^n$:

(1) If $x \in L$, there exists $u \in \{0,1\}^{p(n)}$ that causes $V(x, u)$ to accept.

(2) If $x \notin L$, every possible $u \in \{0,1\}^{p(n)}$ causes $V(x, u)$ to reject.

(3) V has time cost $q(n)$.

$$f(x) = \langle V \rangle x 0^{p(|x|)} 1^{q(|x|)}$$

A First NP Complete Language

Theorem: There exists an NP-complete language.

Proof: $A_{TM}^{\text{cert}} = \{\langle M \rangle x 0^m 1^t \mid \exists c \text{ with } |c| \leq m \text{ s.t. } M \text{ accepts } (x, c) \text{ in } t \text{ steps}\}$

A_{TM}^{cert} is NP-hard: Consider any $L \in NP$. There exists polynomials p, q and a verifier V such that for every $x \in \{0,1\}^n$:

(1) If $x \in L$, there exists $u \in \{0,1\}^{p(n)}$ that causes $V(x, u)$ to accept.

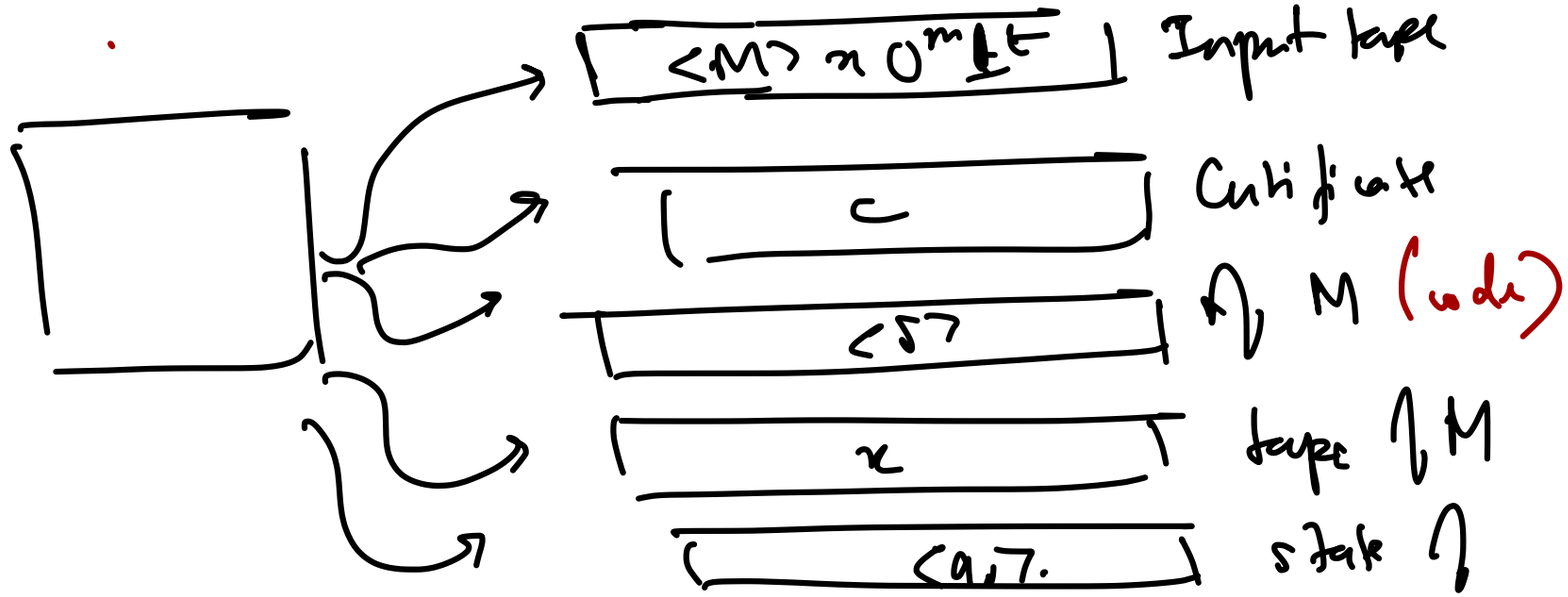
(2) If $x \notin L$, every possible $u \in \{0,1\}^{p(n)}$ causes $V(x, u)$ to reject.

(3) V has time cost $q(n)$.

$$f(x) = \langle V \rangle x 0^{p(|x|)} 1^{q(|x|)}$$

↳ A TM which on i/p n output $p(n)$ in polynomial time. $O(n^c)$ for c constant.

V_A : $A^{\text{cut}}_{\text{TM}}$ $\Rightarrow \langle M \rangle x 0^m 1^t$ and a certificate c.



Boolean Circuits and Formulas

goal: Simulate $M(x,c)$:

(1) In each step what does V_A do:

1) State tape update
   $q_0 \to q_1$
2) $x_i \to x_i'$
3) move the heads.

$\delta(q_0, x_i, c_i)$
$\Vert$
on the code tape, find entry corresp to $\delta(q_0, x_i, c_i)$
$= (q_1, x_i', c_i, p_1, p_2)$

Boolean Circuits and Formulas

- We have seen several models of computation.
- Next: Computation based on networks of logic gates.
  - Closely related to practical electronics.
  - Extremely important in theory, too!

Binary Logical Operations

- AND: $a \wedge b$ → $a \wedge 1 = 1$ iff $a = 1, 1 = 1$.
- OR: $a \vee b$ → $a \vee 1 = 1$ iff ($a = 1, 1 = 0$ or $a = 0, 1 = 1$, or $a = 1, 1 = 1$)
- XOR: $a \oplus b = 0 \Rightarrow$ $a \oplus 1 = 1$ iff exactly one of a or $1 = 1$.
  ($a = 0$, $b = 0$)
- Equality: $a == b$ if (1) a and 1 are 1 or (2) $a, 1$ are 0.
- NOT: $\bar{a}$ or $\neg a$

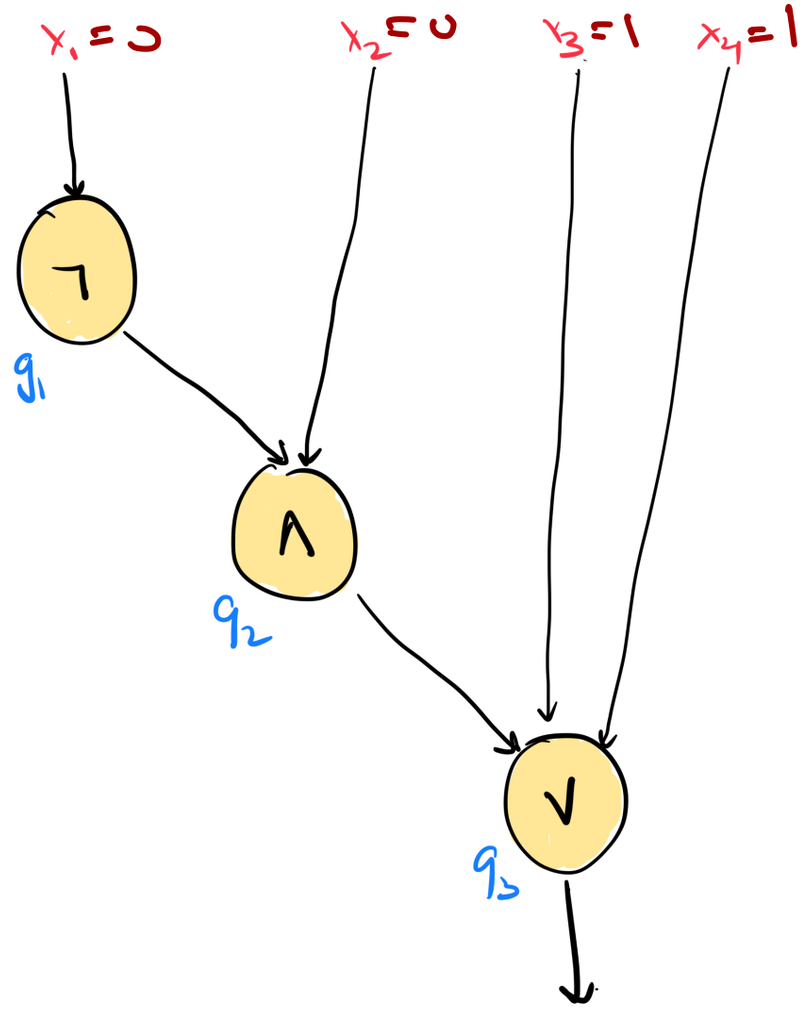
Boolean Circuits

$f = (\neg x_1 \wedge x_2) \vee x_3 \vee x_4$.

A boolean circuit consists of n input bits labelled $x_1, x_2, \cdots, x_n$ and a sequence of gates $g_1, g_2, \cdots, g_m$. There are three types of gates:

- NOT gates
- AND gates
- OR gates

The input to the gate g_k are either one of the input bits $x_1, \cdots, x_n$ or one of the gates $g_1, \cdots, g_{k-1}$. The output of the circuit is the output of gate g_m.



Boolean Circuit as a DAG

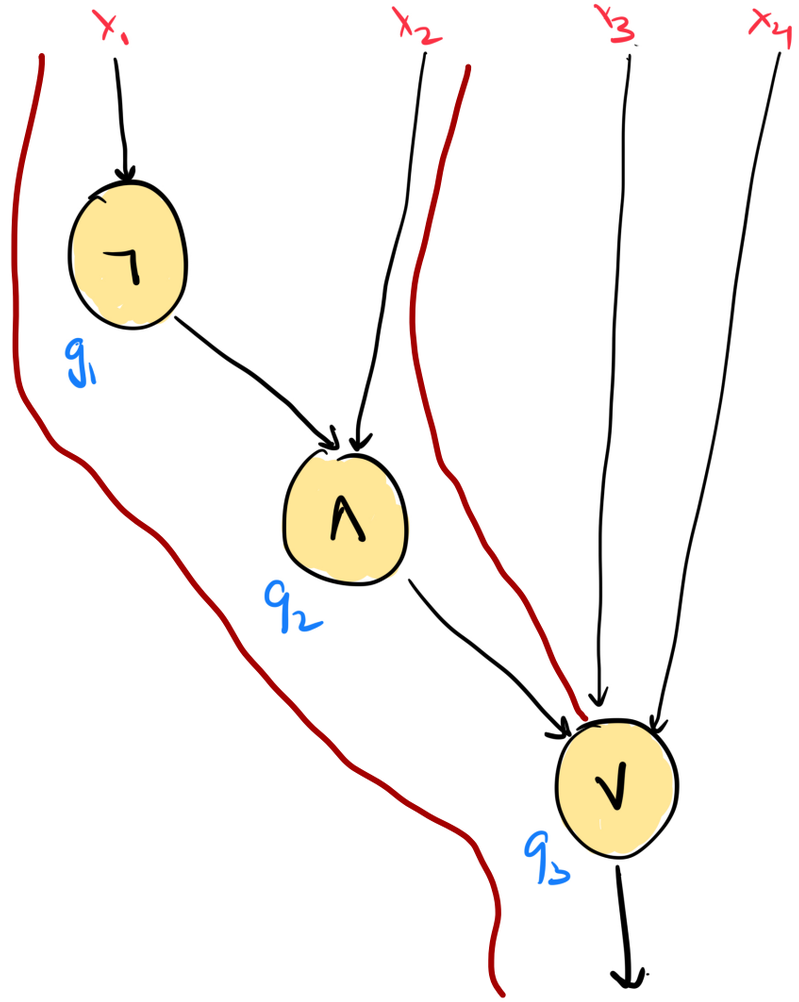
↳ this condition implies that there are no directed cycles in the circuit.

computes a boolean function

Complexity Measure

Size: The total number of AND and OR gates it contains.

Depth: The number of AND and OR gates in the longest path from any input bit to the output of the circuit.

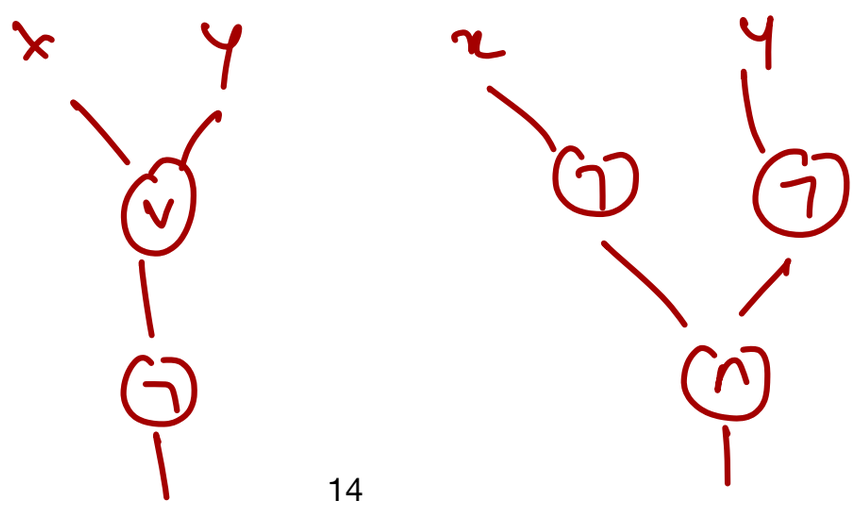


Boolean Circuit as a DAG

De Morgan's law:

$\neg(x \vee y) = (\neg x \wedge \neg y)$

$n \longrightarrow C_n$



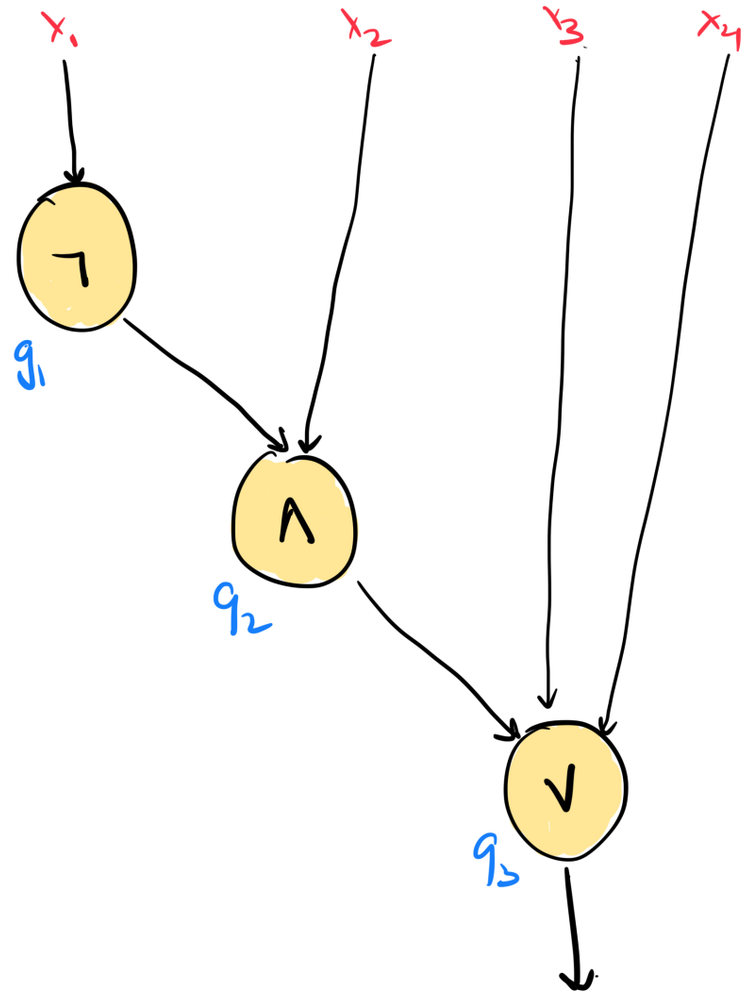
Languages and Boolean Circuits

- Number of inputs to a Boolean circuit is fixed. It computes a Boolean function:
  $f : \{0,1\}^n \to \{0,1\}$.

- A language $L \subseteq \{0,1\}^*$ can be expressed as family of Boolean functions:
  $\{f_n : \{0,1\}^n \to \{0,1\}$, where
  $L = \cup_{i \geq 1} f_i^{-1}(1)$.

- A single circuit does not compute a language, but a family of circuits $\{C_n\}_{n \geq 0}$ does.

- In contrast with a TM. Non-uniform model of computation.



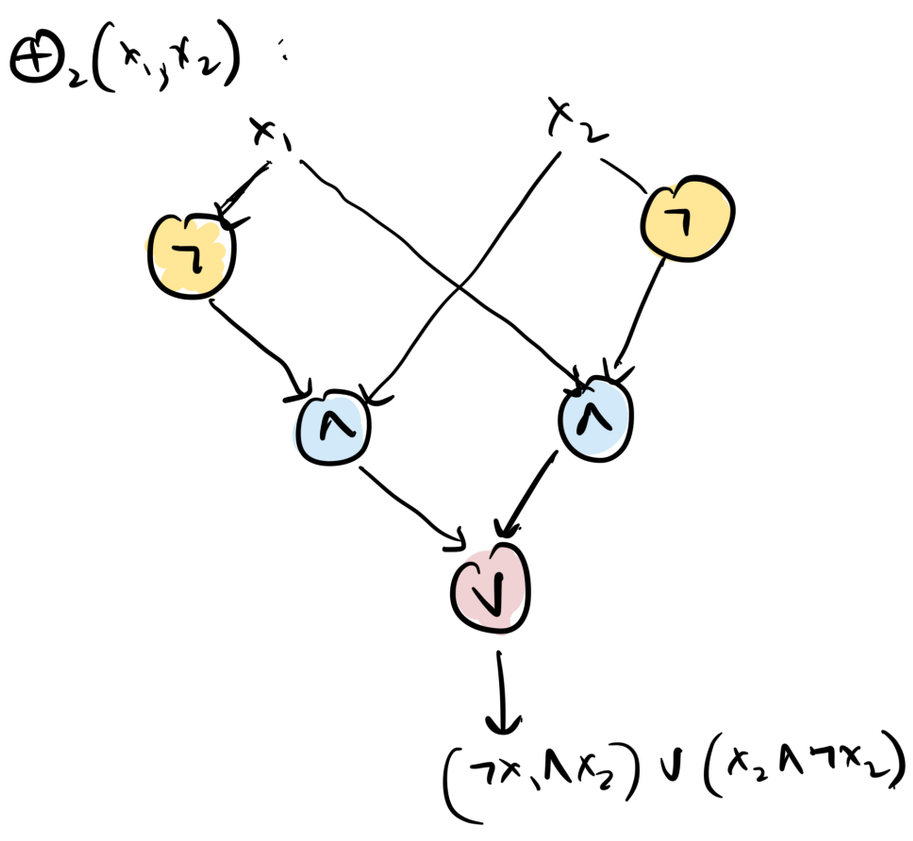
Boolean Circuit as a DAG

Examples: PARITY

- The parity of a set of n bits is a function $\oplus_n : \{0,1\}^n \to \{0,1\}$ defined as $\oplus_n (x_1, \cdots, x_n) = \sum_{i=1}^{n} x_i \pmod 2$.

Examples: PARITY

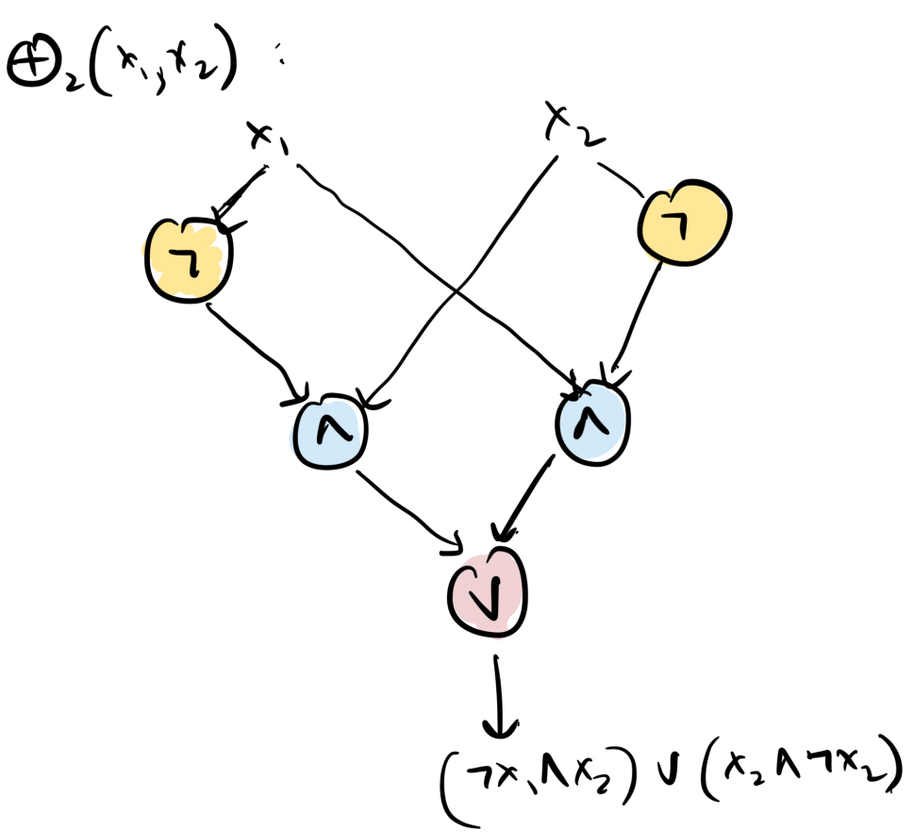
- The parity of a set of n bits is a function $\oplus_n : \{0,1\}^n \to \{0,1\}$ defined as $\oplus_n (x_1, \cdots, x_n) = \sum_{i=1}^{n} x_i \ (\mod 2)$.

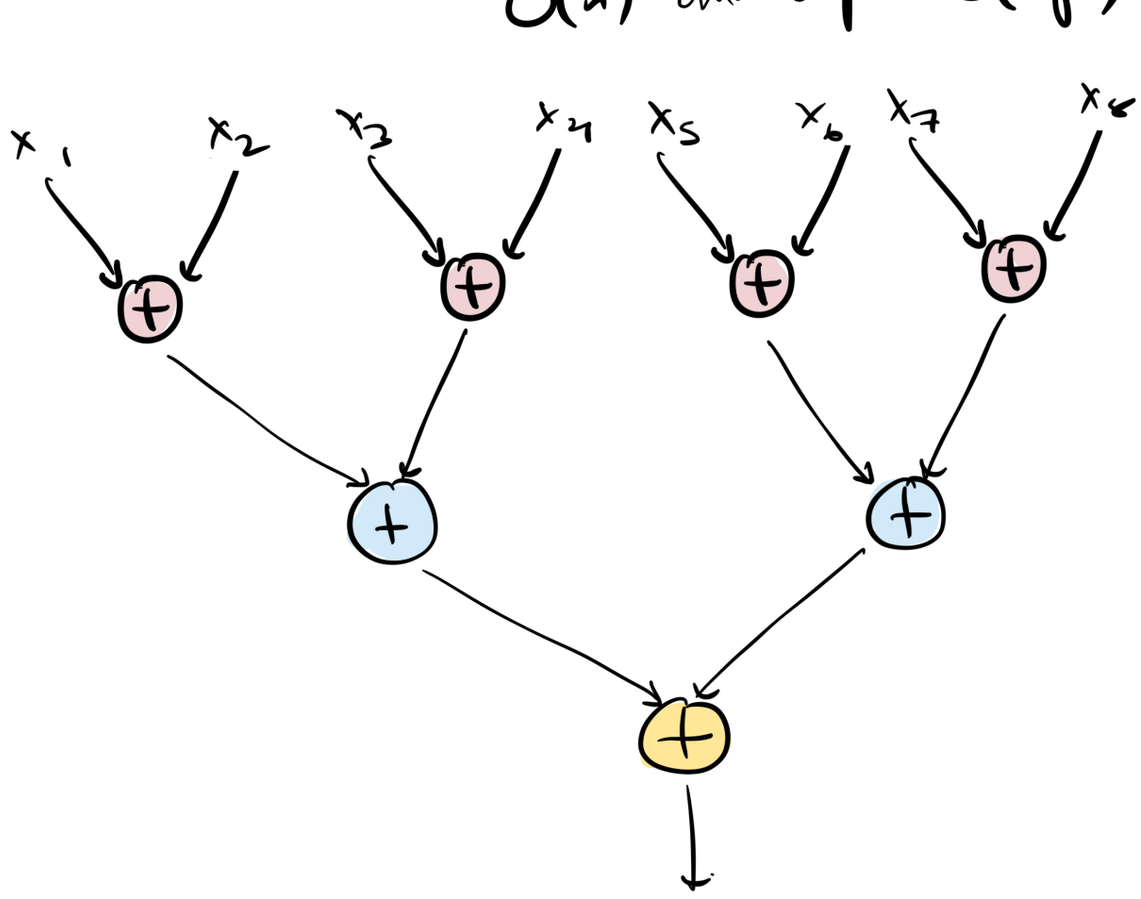


Examples: PARITY

- The parity of a set of n bits is a function $\oplus_n : \{0,1\}^n \to \{0,1\}$ defined as $\oplus_n (x_1, \cdots, x_n) = \sum_{i=1}^{n} x_i \pmod 2$.

$\oplus_n(x_1, x_2, \ldots, x_n)$ using a circuit of size $O(n)$ and depth $O(\log n)$



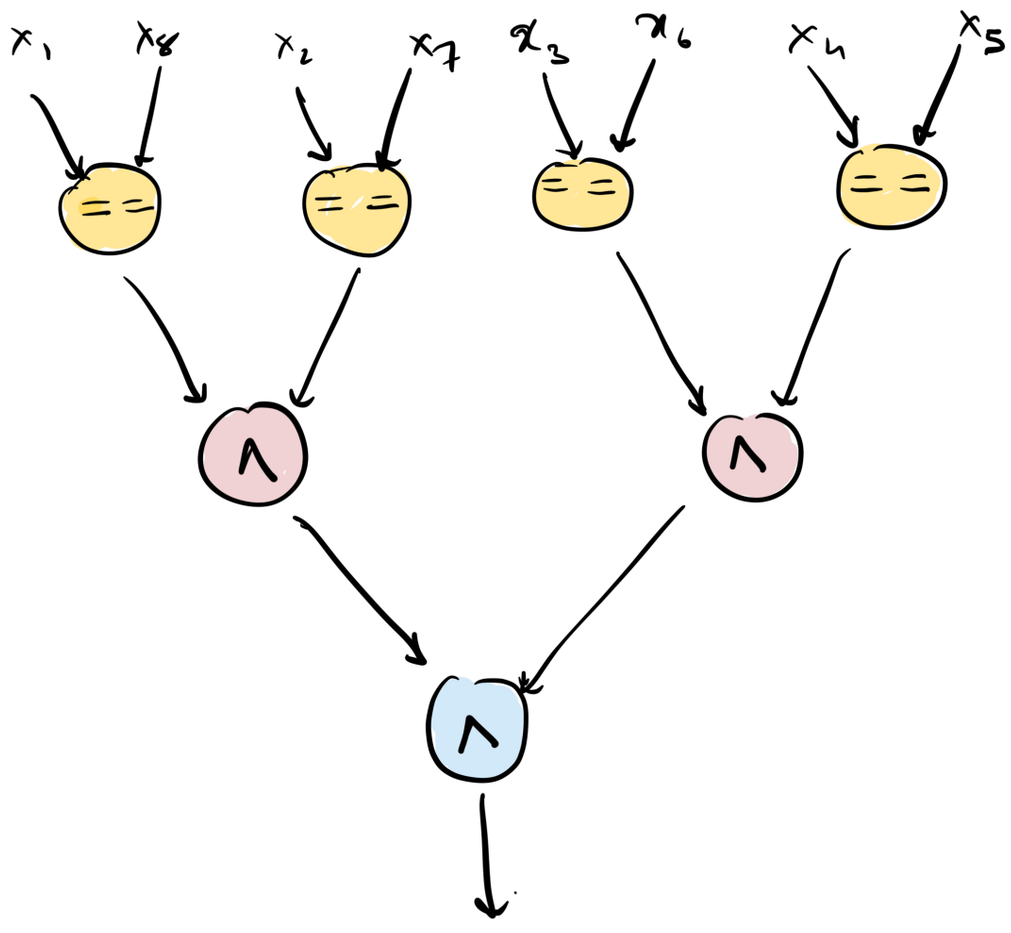


Examples: PALINDROME

- Define $f : \{0,1\}^n \to \{0,1\}$ by $f(x) = 1 \iff x$ is a palindrome.

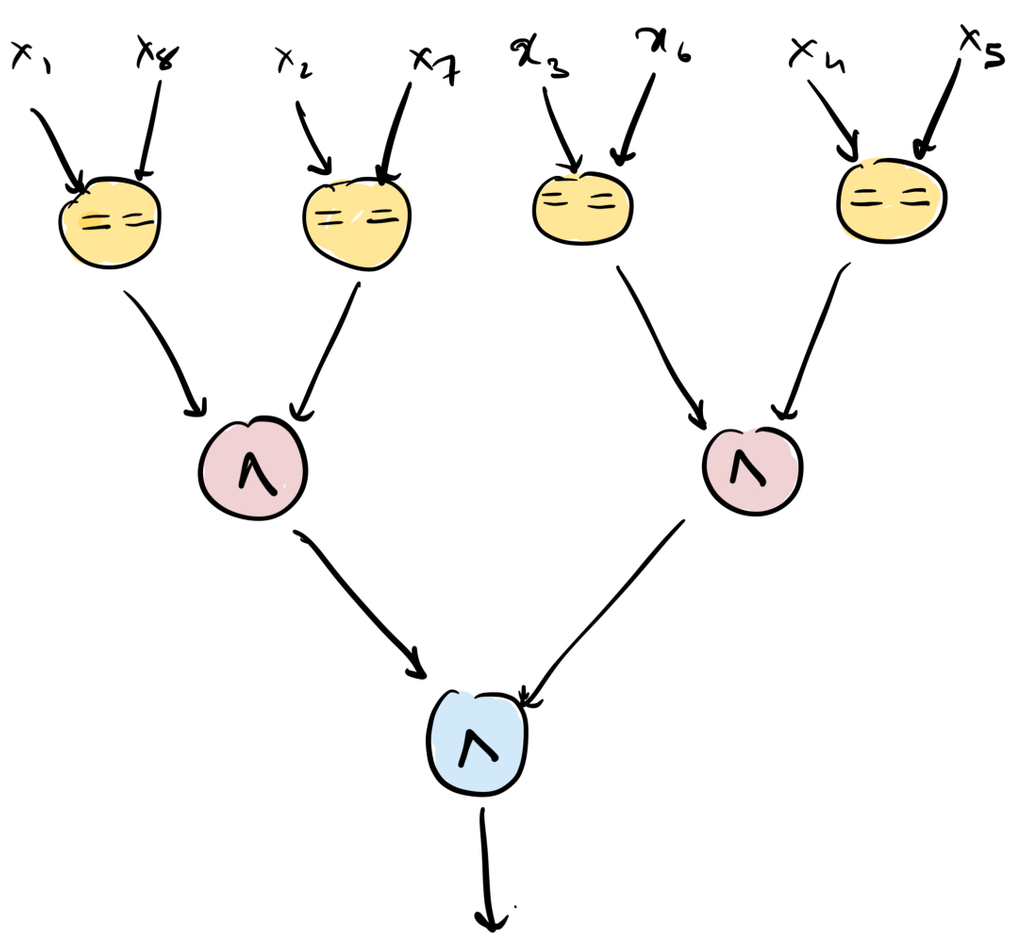
Examples: PALINDROME

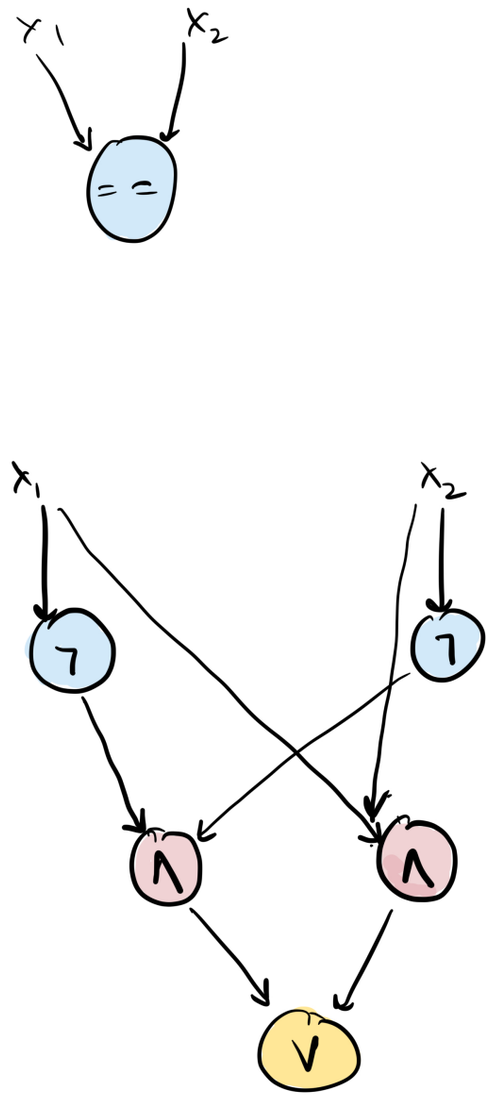
- Define $f : \{0,1\}^n \to \{0,1\}$ by $f(x) = 1 \iff x$ is a palindrome.



Examples: PALINDROME

- Define $f : \{0,1\}^n \to \{0,1\}$ by $f(x) = 1 \iff x$ is a palindrome.





Shannon's Theorem

Theorem: For every function $f : \{0,1\}^n \to \{0,1\}$, there exists a Boolean circuit C of size at most $2^n + 1$ that computes f.

Shannon's Theorem

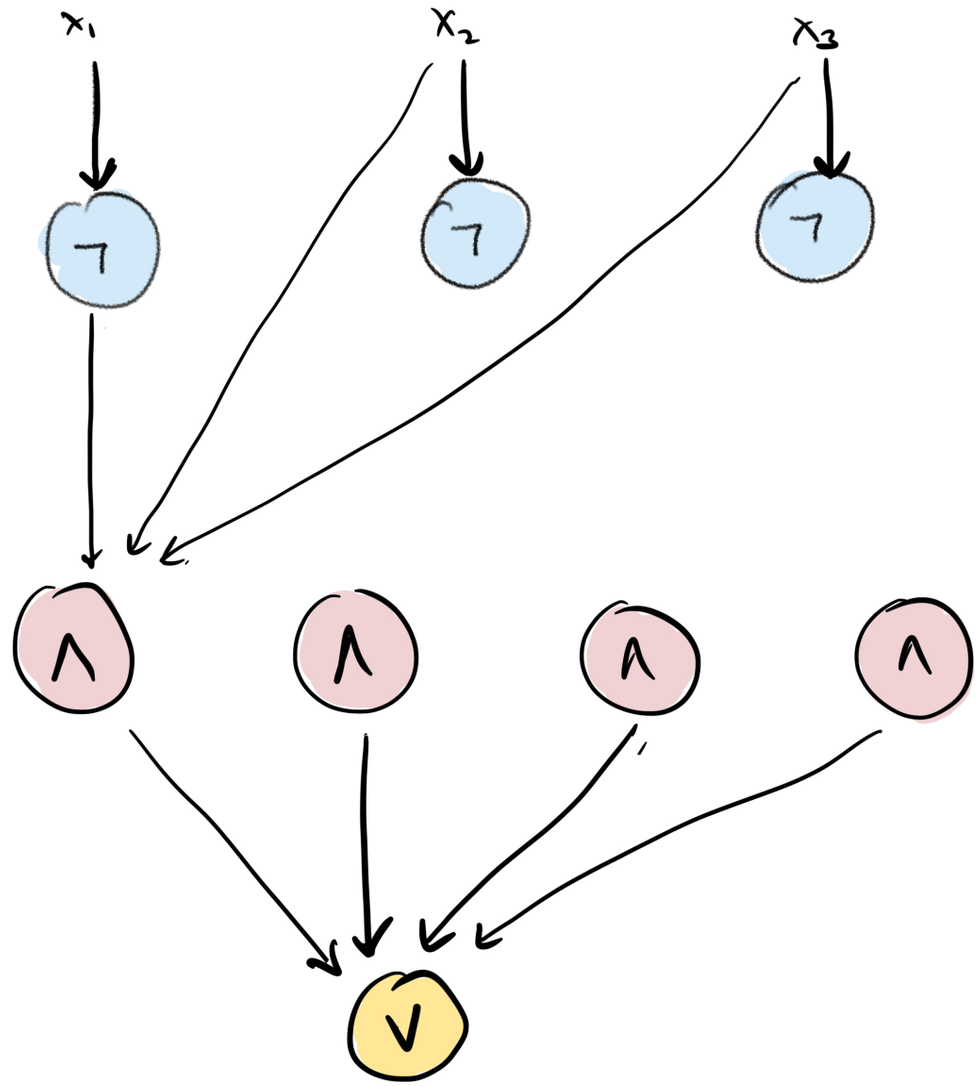
Theorem: For every function $f : \{0,1\}^n \to \{0,1\}$, there exists a Boolean circuit C of size at most $2^n + 1$ that computes f.

x_1	x_2	x_3	$Maj_3(x)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Shannon's Theorem

Theorem: For every function $f : \{0,1\}^n \to \{0,1\}$, there exists a Boolean circuit C of size at most $2^n + 1$ that computes f.

x_1	x_2	x_3	$Maj_3(x)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Shannon's Theorem

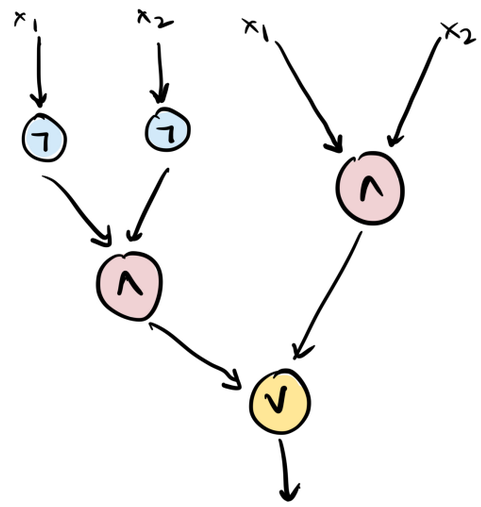
Corollary: For every language $L \subseteq \{0,1\}^*$, there exists a family of circuits $\{C_n\}_{n \geq 0}$ that computes L.

Boolean Formulas

- A boolean formula is a Boolean circuit which has an additional restriction: all of the gates have only one output wire. Each output can be used as the input of at most one gate. (Bounded fan-out).

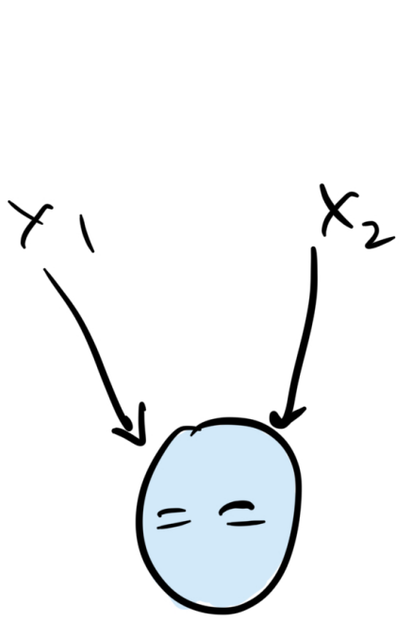
- A boolean formula has bounded fan-in when each of $\wedge$, $\vee$ gates have exactly two inputs.

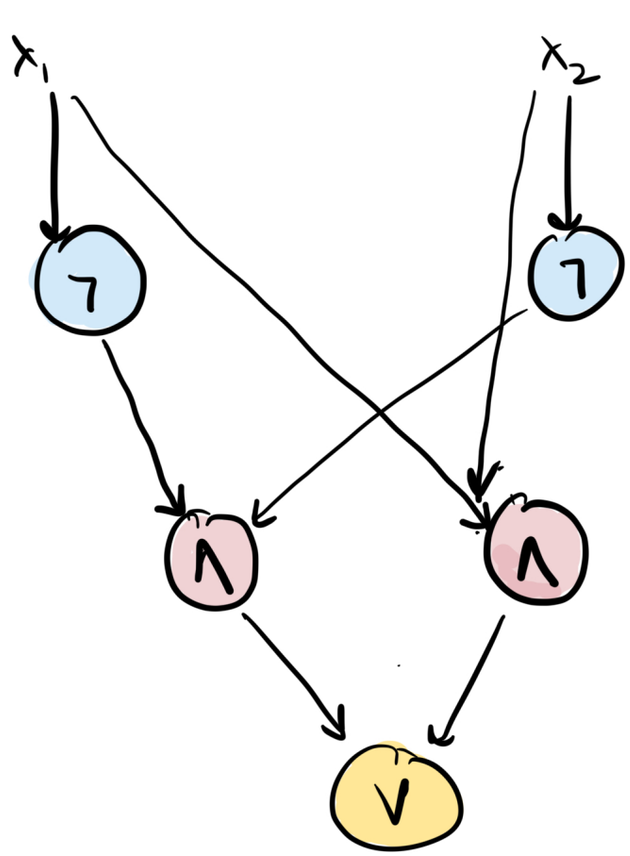
- Represented as a rooted binary tree, with leaves being multiple copies of x_i, $\neg x_i$.

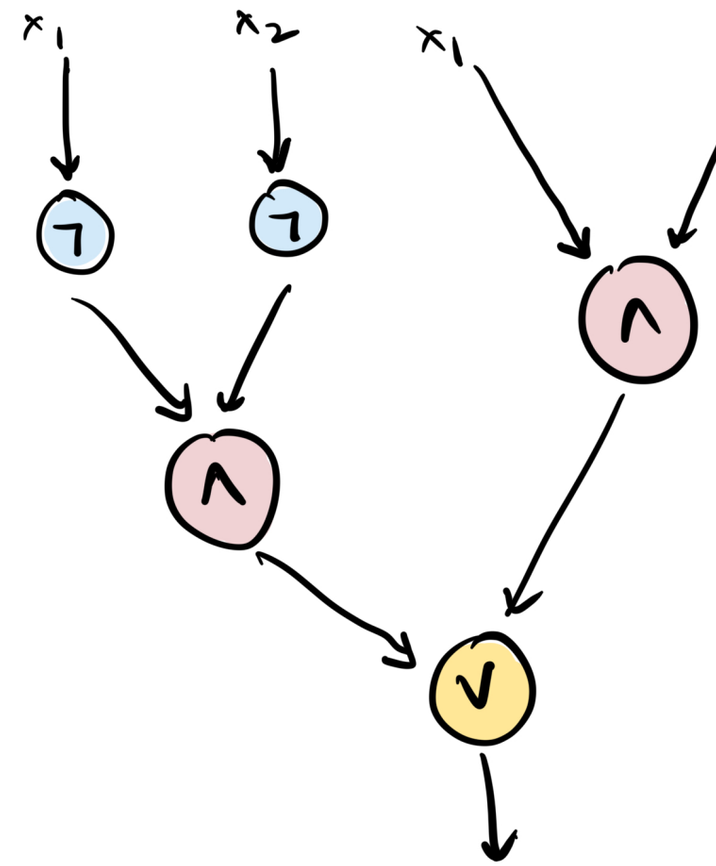


Complexity Measure: Boolean Formula

Size: The number of leaves in the tree representation of the formula.







Conjunctive Normal Form

Definition: A clause is an OR of literals. Example: $C = \bar{x}_1 \vee x_2 \vee \bar{x}_3$.

Definition: A conjunctive normal form (CNF) formula is a conjunction of clauses (AND of ORs of literals). Example:

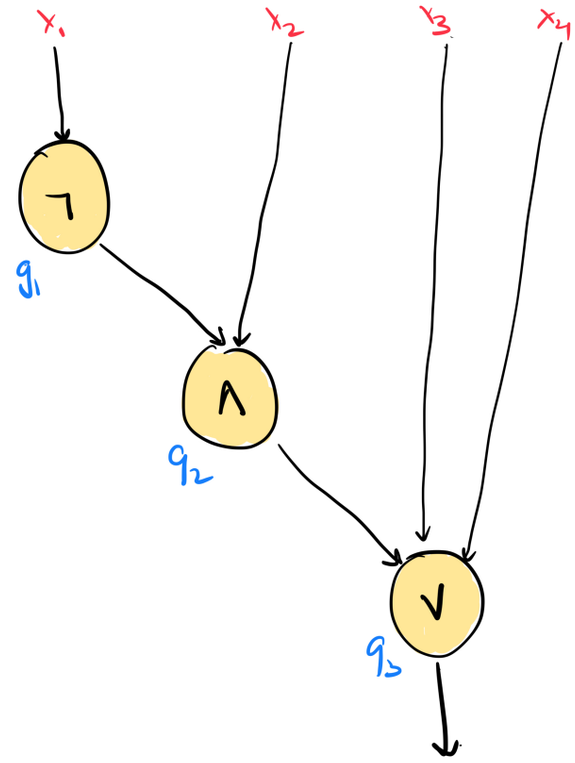
$$f(x) = (\bar{x}_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$$

Every Function has a CNF Formula

Theorem: Let $f : \{0,1\}^n \to \{0,1\}$ be any function, then there is a CNF formula that computes f, with at most 2^n clauses and n literals per clause.

x_1	x_2	x_3	$Maj_3(x)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Circuits as Code as Data



Recall: A Turing Machine can be encoded as a string $\langle M \rangle$. $\langle M \rangle$ is an algorithm, but at the same time, $\langle M \rangle$ can be an input to another algorithm.

Similar Idea: A circuit C can be encoded as a string $\langle C \rangle$.

- Assume n variables and m gates.
- Each node in the graph can be of four types: literal, AND, OR, NOT

	x_1	x_2	x_3	x_4	g_1	g_2	g_3
x_1	0	0	0	0	1	0	0
x_2	0	0	0	0	0	1	0
x_3	0	0	0	0	0	0	1
x_4	0	0	0	0	0	0	1
g_1	0	0	0	0	0	1	0
g_2	0	0	0	0	0	0	1
g_3	0	0	0	0	0	0	0

Circuit Value Problem

Let CVP $= \{\langle C, x\rangle : C$ is a circuit and $C(x) = 1\}$

Lemma: CVP $\in P$

Proof Sketch: Suppose C has m nodes. To compute $C(x)$:

(a) Mark all the input nodes with their values.

(b) While there is an unmarked node:

i) For every gate g find all nodes that feed into g. If they are marked with their values, then mark g with its value.