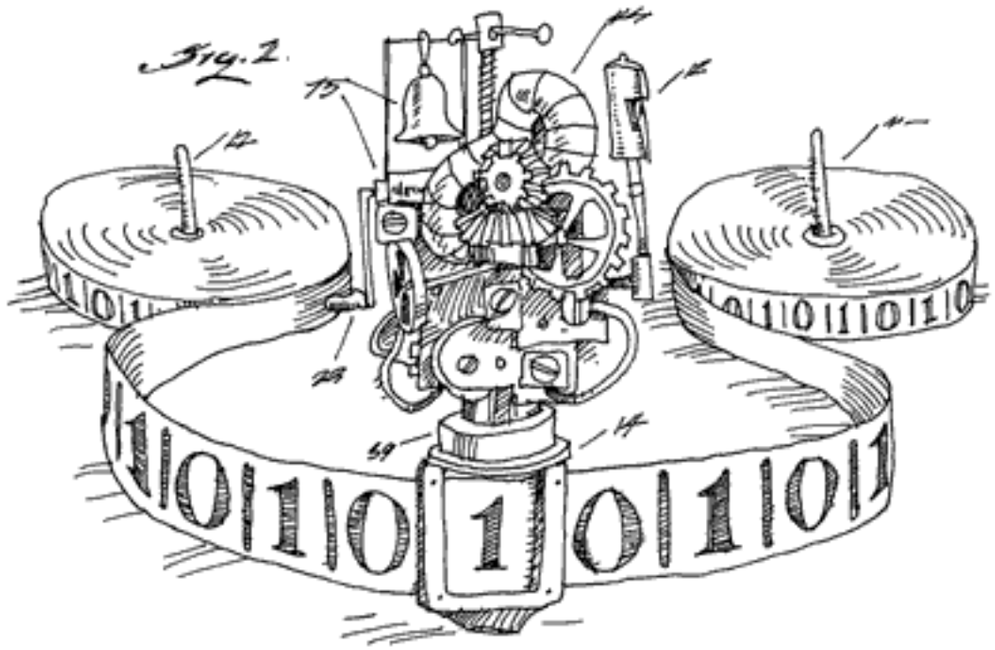
CSC4010: Introduction to Theoretical Computer Science



Lecture 14

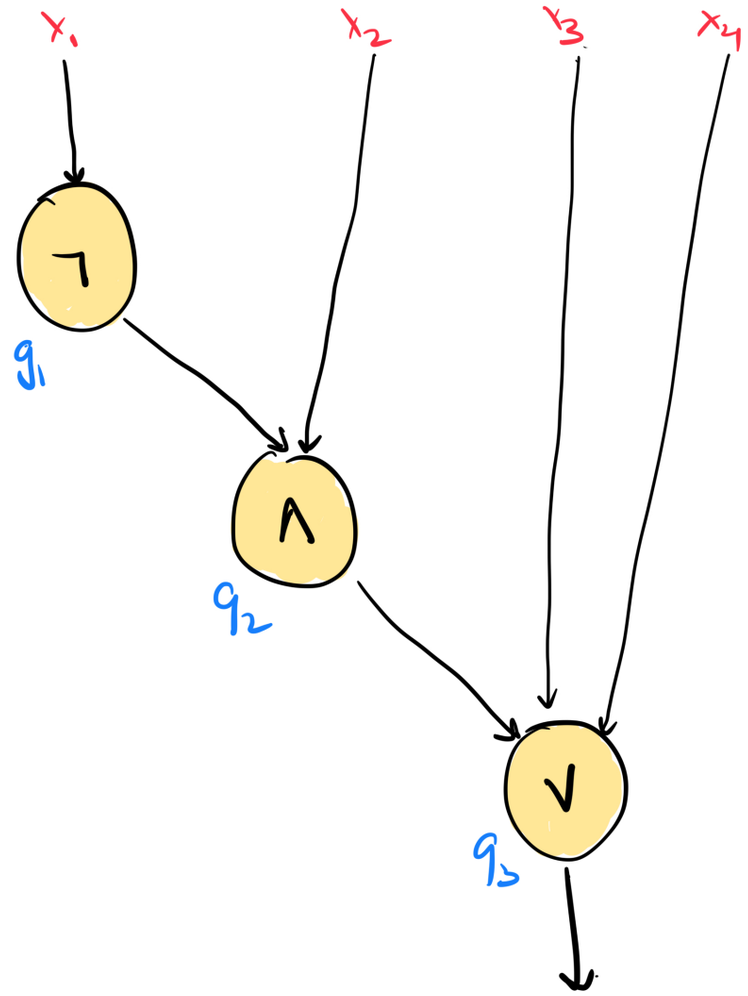
Boolean Circuits and Formulas

Boolean Circuits

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.

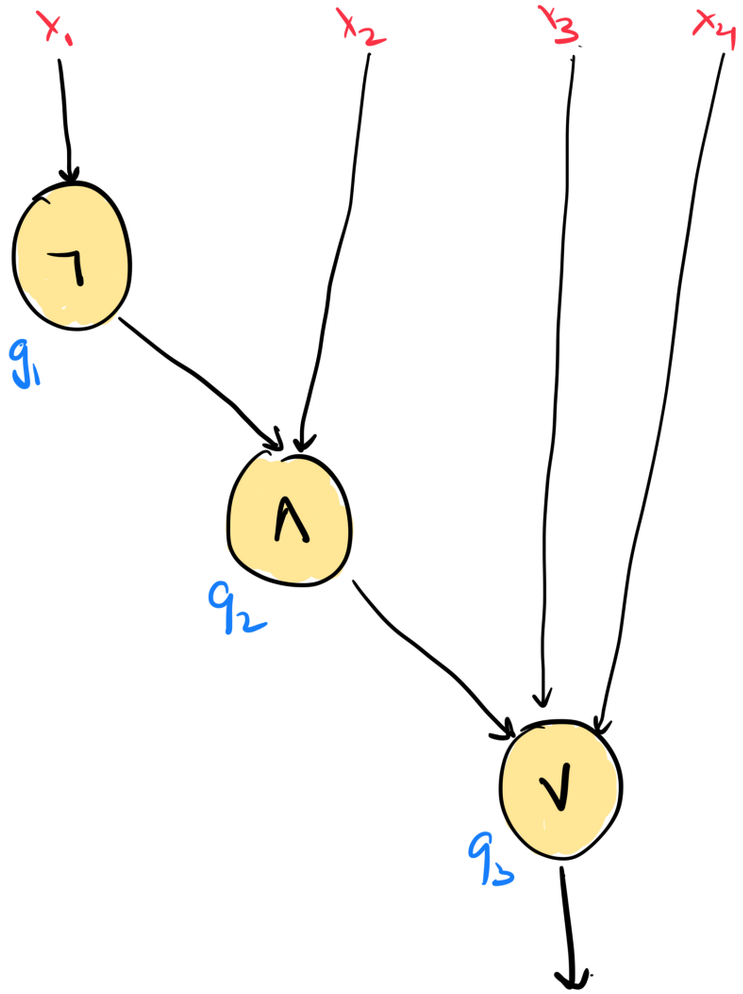


Boolean Circuit as a DAG

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

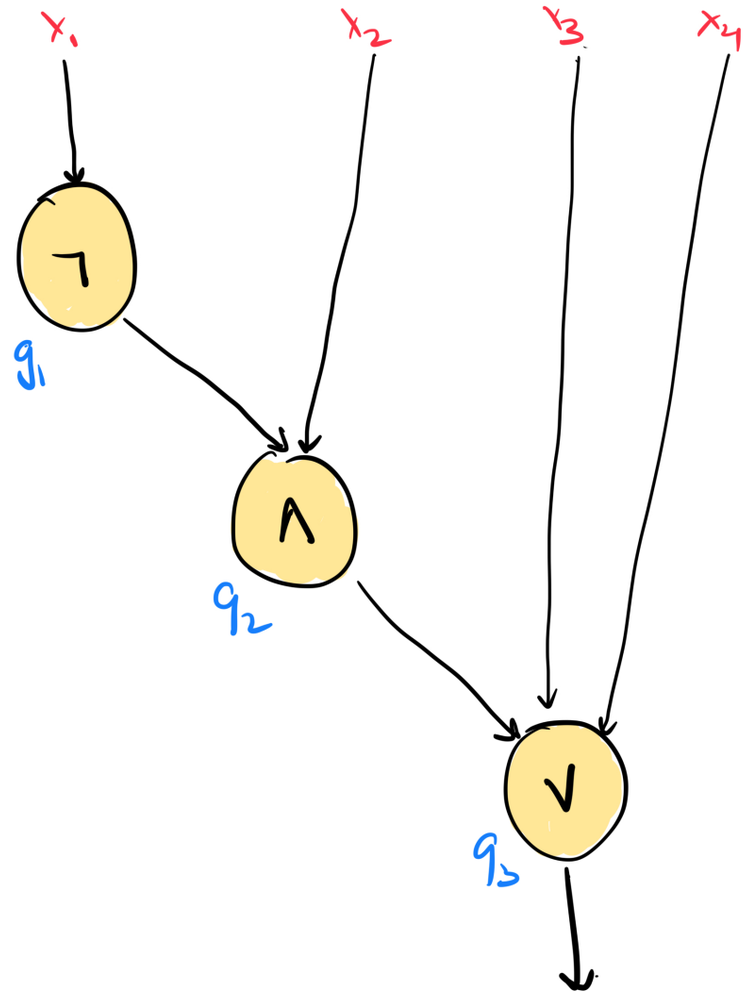
Languages and Boolean Circuits

- 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
  $L = \cup_{i \geq 1} f^{-1}(1)$.

- 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.



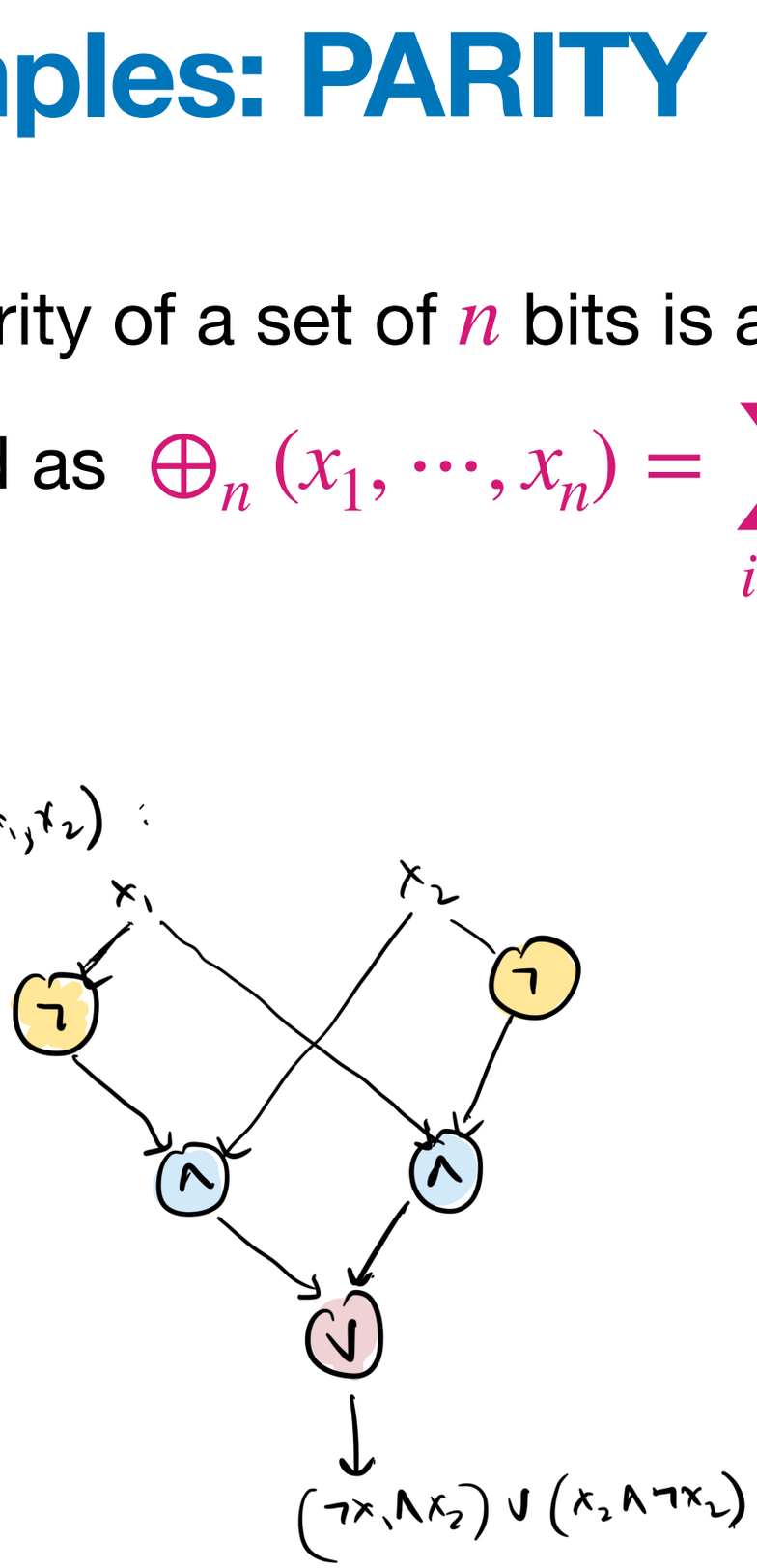
Boolean Circuit as a DAG

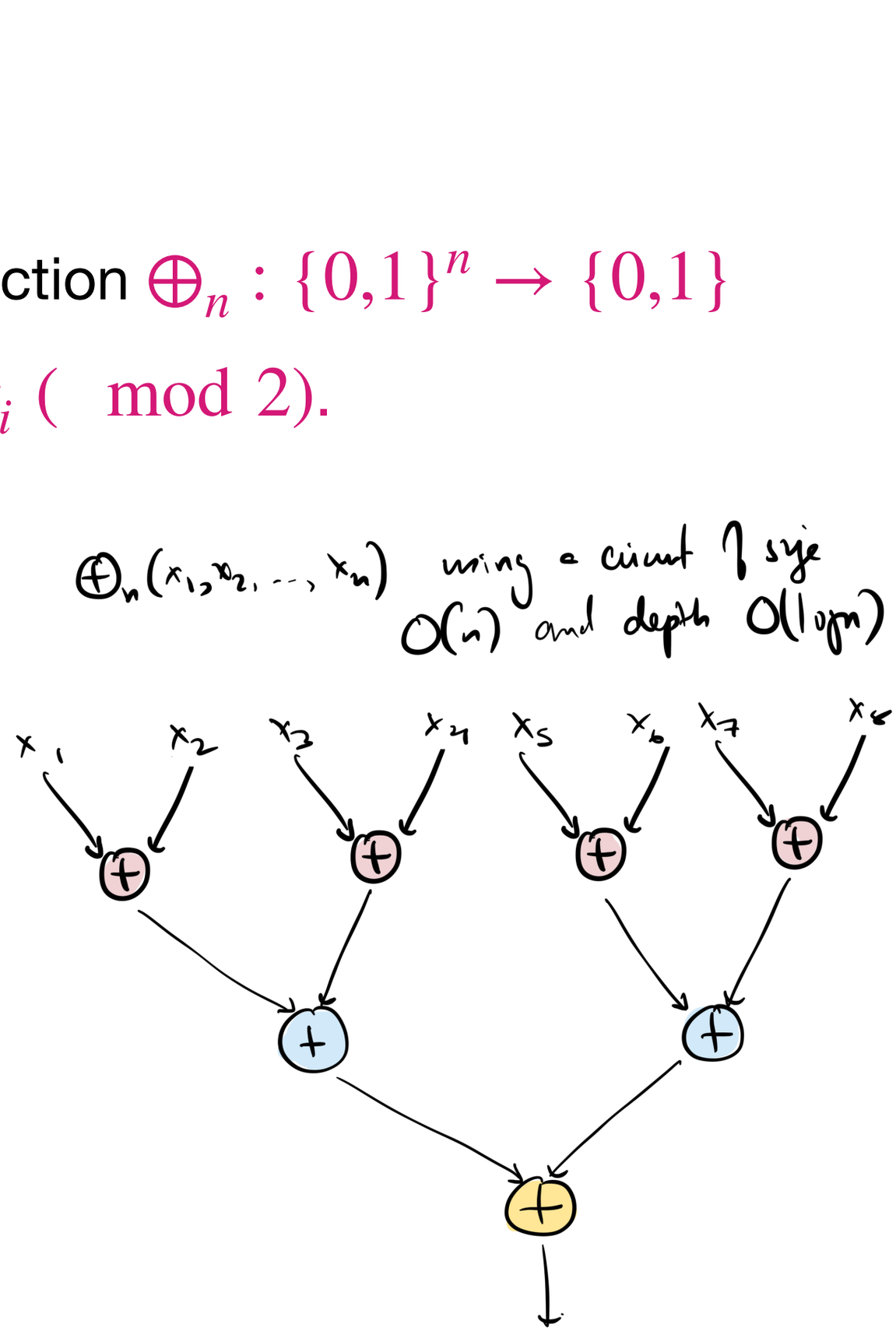
Examples: PARITY

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

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

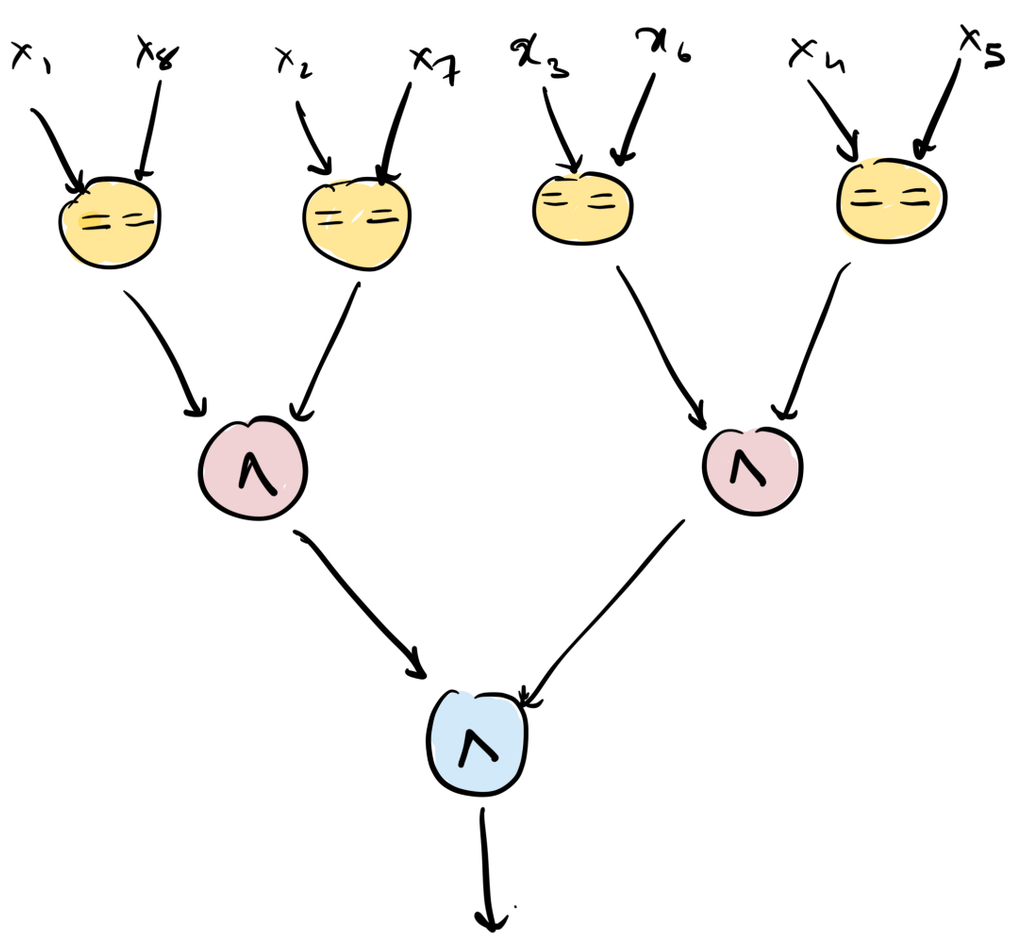
$\oplus_2(x_1, x_2)$:

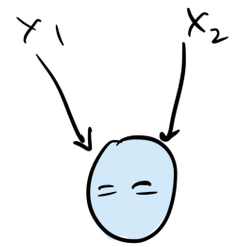


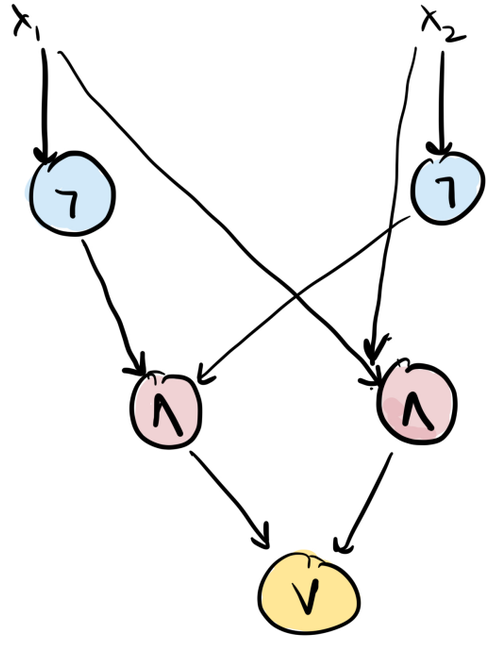


Examples: PALINDROME

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



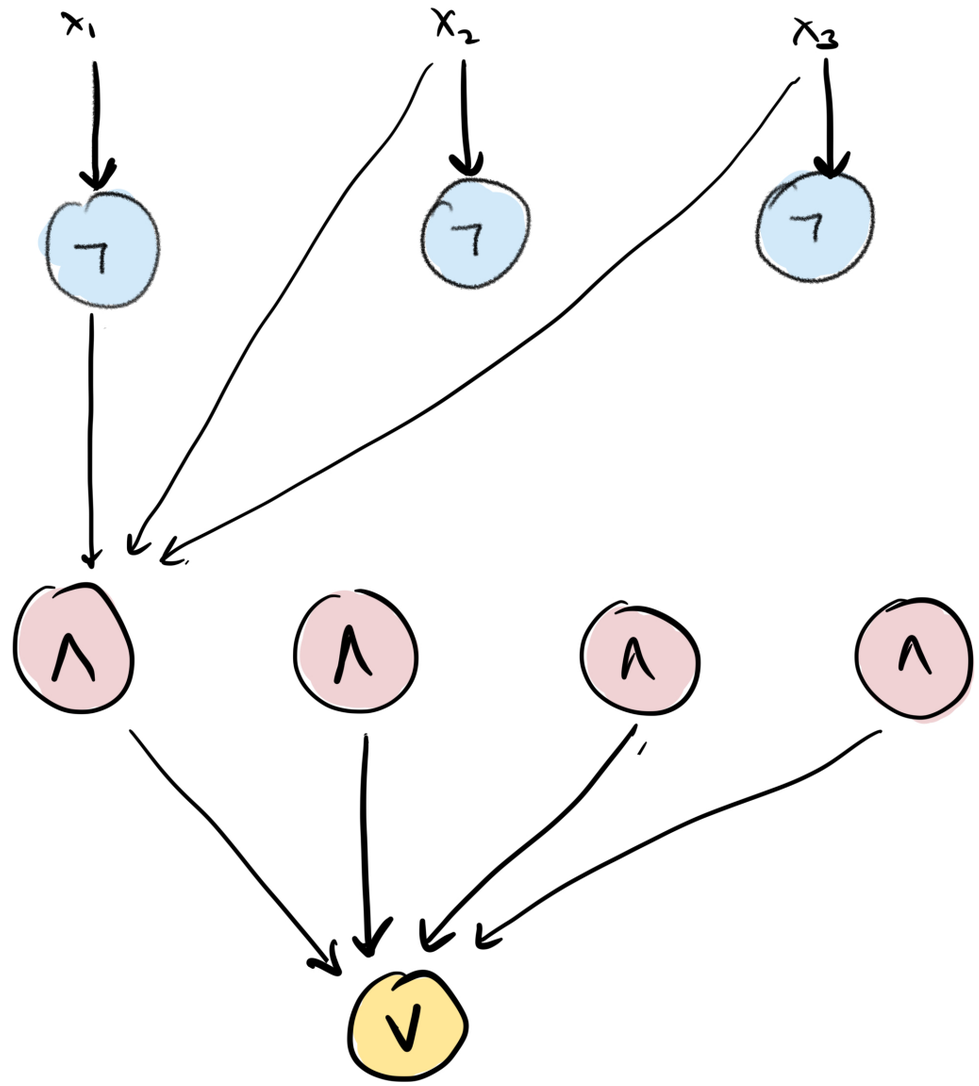




Shannon's Theorem

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.

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

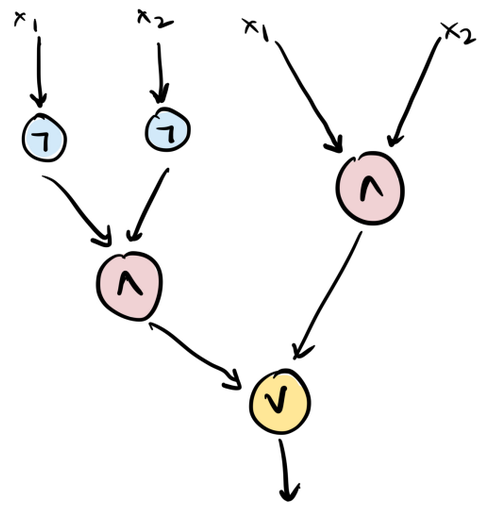
Corollary: For every language $L \subseteq \{0,1\}^*$, there exists a family of circuits $\{C_n\}_{n \geq 0}$ that computes L.

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).

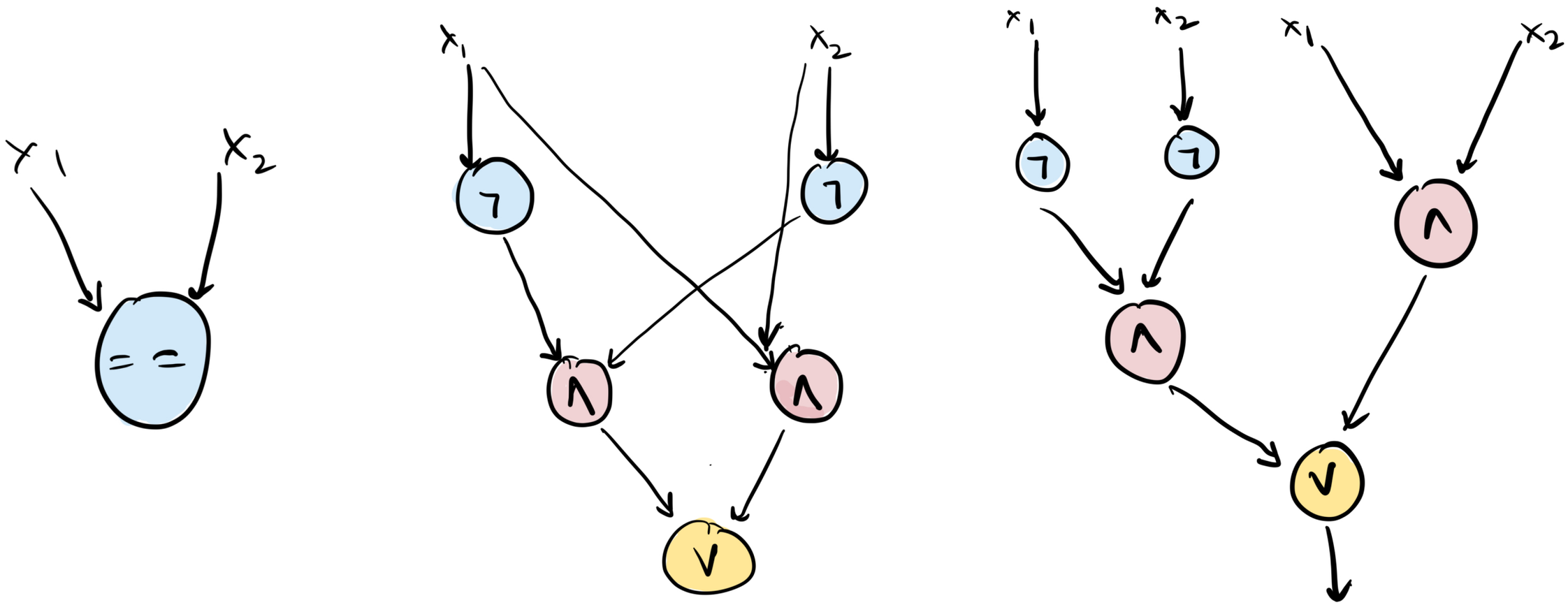
- A boolean formula has bounded fan-in when each of $\wedge$, $\vee$ gates have exactly two inputs.

- 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

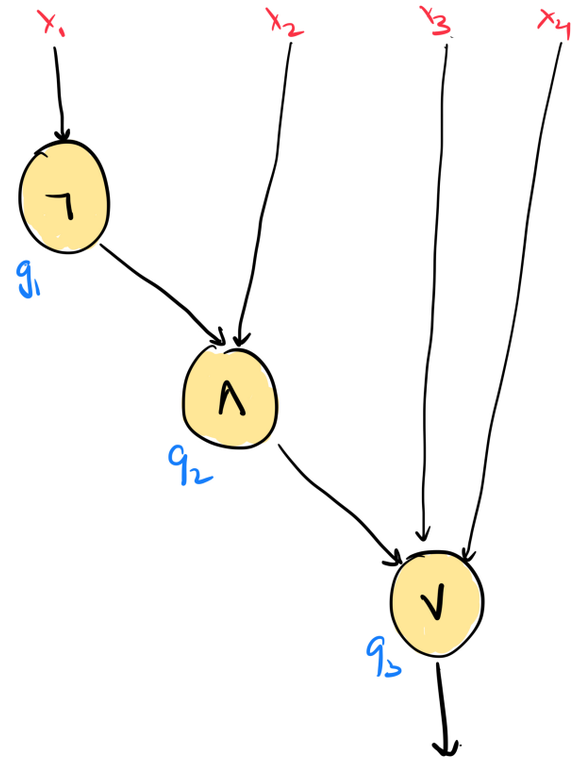
Theorem: Let $f : \{0,1\}^n \to \{0,1\}$ be any function, then there is a CNF formula that computes f, with at most 2^n clauses and n literals per clause.

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

Why are we studying circuits?

- Why are we studying circuits?
  - It will help us show that many interesting problems are NP-Complete.
- Key Idea: Code as Data

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.

Circuit Satisfiability

We say that circuit C (on n inputs) is satisfiable if there exists $x \in \{0,1\}^n$ such that $C(x) = 1$

$$\text{CIRCUIT-SAT} = \{\langle C \rangle \mid \exists x \text{ s.t. } C(x) = 1\}$$

CIRCUIT-SAT $\in NP$

Parting Thoughts

Circuit Size

Every function has a circuit (of size exponential).

Do you really need exponential size?

Theorem: For every $n \geq 4$, there exists a function $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by any Boolean circuit of size $2^{n/2}/4$.

Circuit Complexity of a Binary Language

Let $Y \subseteq \{0,1\}^*$. Let $Y_n = Y \cap \{0,1\}^n$.

Definition: The circuit complexity of Y is the function $S : \mathbb{N} \to \mathbb{N}$ defined by $S(n)$ = the size of the smallest circuit that computes Y_n.

Let $S : \mathbb{N} \to \mathbb{N}$ be a function

Definition: SIZE$(S) = \{Y \subseteq \{0,1\} * :$ the circuit complexity of Y is $O(S)\}$

Definition: PSIZE $= \{Y \subseteq \{0,1\}^* :$ circuit complexity of Y is poly$(n)\}$

Next Time

We will show P $\subseteq$ PSIZE. But is the converse true?

Theorem: There exists an undecidable language Y with circuit complexity $O(n)$.

How to interpret PSIZE?

We will prove $P \subsetneq$ PSIZE

Circuits are more powerful than Turing machines.

Does this mean something is wrong with Turing machines?

No! Something is “wrong” with the circuit model.

PSIZE is not a good model of tractable languages.

$''Y \in$ PSIZE$''$ means there is a family of polynomial-sized circuits that decide Y (one for each input size).

Each circuit performs only a polynomial work. But what about the work needed to construct these circuits?

Nonuniformity

PSIZE allows us to use different “algorithms” for different input lengths.

Computing in this “nonuniform” manner is cheating/unrealistic.

But circuits as we will see, are a valuable conceptual tool.