CMSC 28100

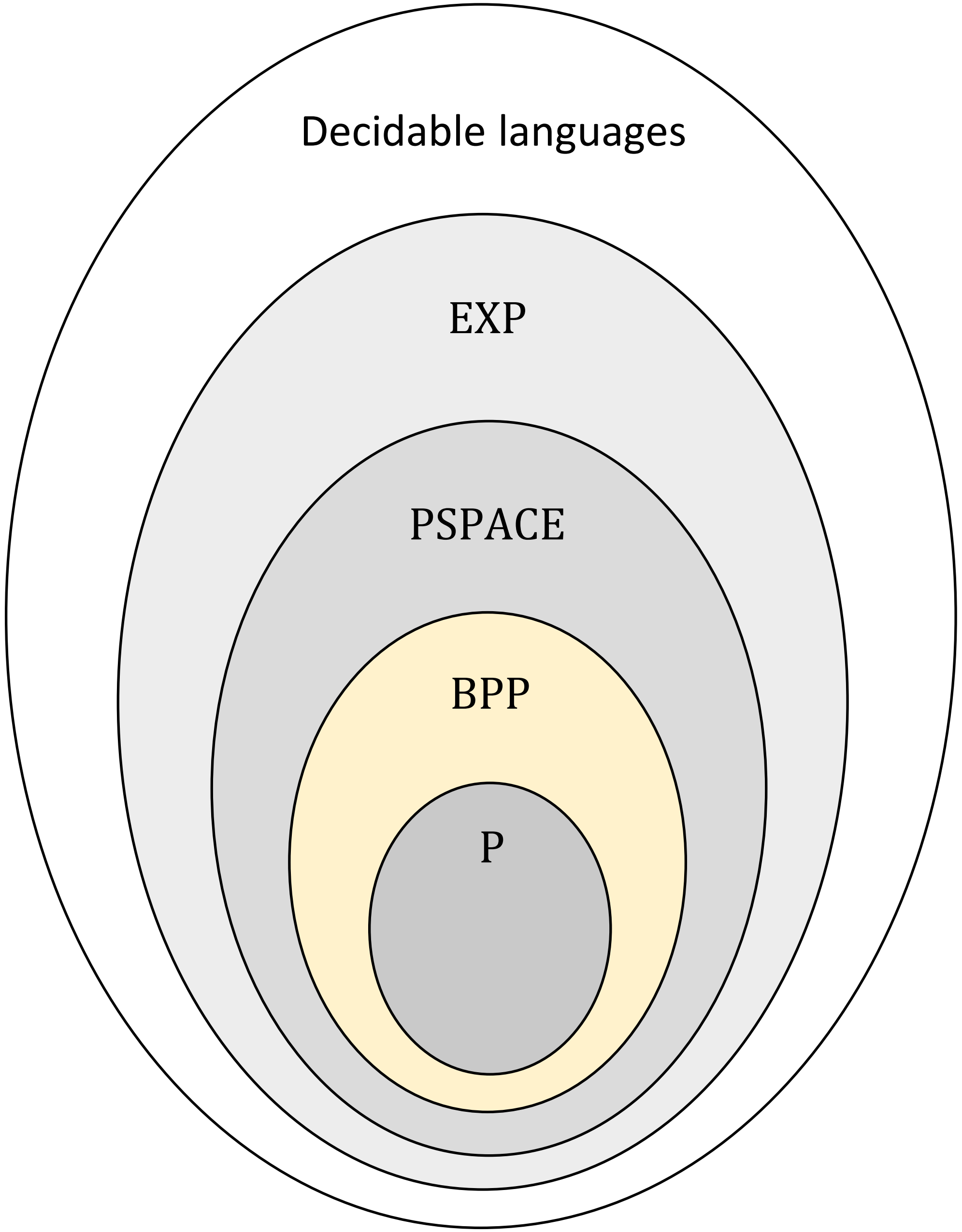
Introduction to Complexity Theory

Spring 2024

Instructor: William Hoza

The complexity class BPP

- Let $L \subseteq \Sigma^*$ be a language
- **Definition:** $L \in \mathrm{BPP}$ if there exists a randomized polynomial-time Turing machine M such that for every $w \in \Sigma^*$:
  - If $w \in L$, then $\Pr[M \text{ accepts } w] \geq 2/3$
  - If $w \notin L$, then $\Pr[M \text{ accepts } w] \leq 1/3$



Decidable languages
EXP
PSPACE
BPP
P

P vs. BPP

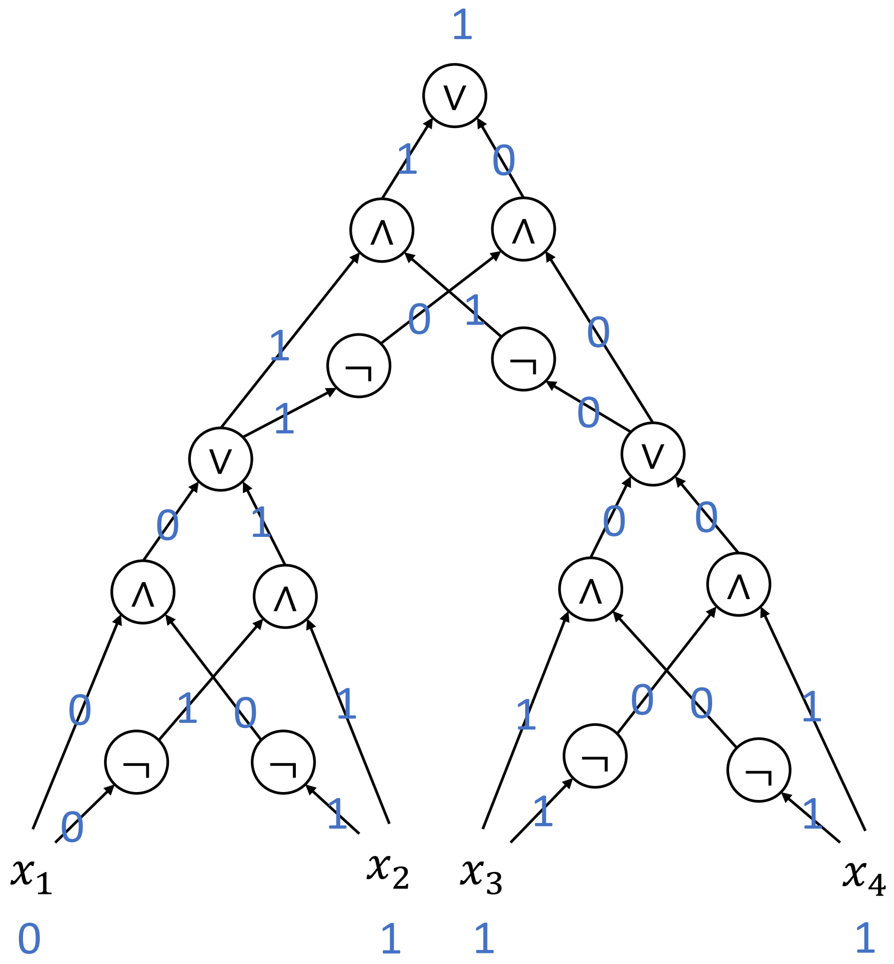
- $\mathrm{P} \subseteq \mathrm{BPP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$

Conjecture: $\mathrm{P} = \mathrm{BPP}$

- We will show that languages in BPP can be decided by “circuits” consisting of a polynomial number of logic gates
- This is tantalizingly similar to the statement “$\mathrm{P} = \mathrm{BPP}$”

Boolean circuits

- For us, a "circuit" is a network of logic gates applied to Boolean variables



- ∨ means OR
- ∧ means AND
- ¬ means NOT
- Each x_i can be either 0 or 1

  (FALSE or TRUE)

Boolean circuits

- **Definition:** An n-input m-output circuit is a directed acyclic graph with the following types of nodes:
  - Nodes with zero incoming edges (“wires”). Each such node is labeled with a variable x_i ($1 \leq i \leq n$) or a constant (0 or 1)
  - Nodes with one incoming wire, labeled $\neg$
  - Nodes with two incoming wires, labeled $\wedge$ or $\vee$

  “gates”
- Among the nodes with zero outgoing wires, m of them are additionally labeled as “output 1”, “output 2”, …, “output m”

Boolean circuits

- Each node g computes a function $g: \{0,1\}^n \to \{0,1\}$ defined inductively:
  - If g is labeled x_i, then $g(x) =$ the i-th bit of x
  - If g is labeled $\neg$ and its incoming wire comes from f, then $g(x) = \neg f(x)$
  - If g is labeled $\wedge$ and its incoming wires come from f and h, then $g(x) = f(x) \wedge h(x)$
  - If g is labeled $\vee$ and its incoming wires come from f and h, then $g(x) = f(x) \vee h(x)$

Boolean circuits

- Let the output nodes be $g_1, \ldots, g_m$
- As a whole, the circuit computes $C: \{0,1\}^n \to \{0,1\}^m$ defined by

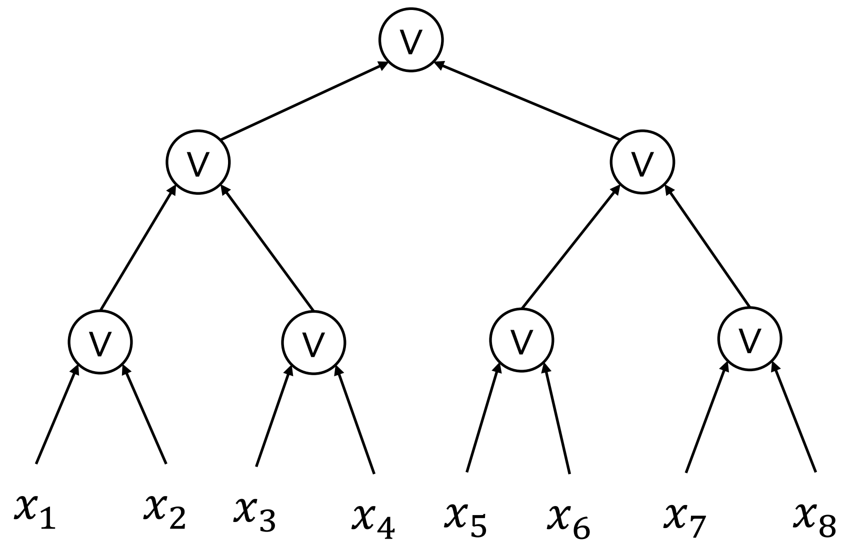
$$C(x) = \left(g_1(x), \ldots, g_m(x)\right)$$

Circuit size

- The size of the circuit is the total number of AND/OR/NOT gates
- Size is a measure of the total amount of "effort" that the circuit exerts
- If $f : \{0,1\}^n \to \{0,1\}^m$ is a function, the circuit complexity of f is the size of the smallest circuit that computes f
- Circuit complexity is a measure of how much effort is required to compute f

Circuit complexity example 1

- Let $f(x) = x_1 \vee x_2 \vee \cdots \vee x_n$



What is the circuit complexity of f?

A: $\Theta(n^2)$

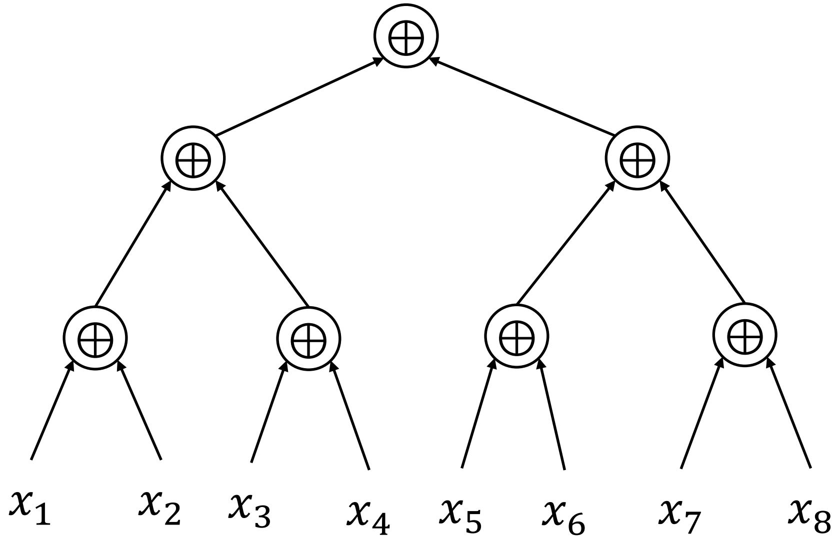
B: $O(1)$

C: $\Theta(n)$

D: $\Theta(2^n)$

Respond at PollEv.com/whoza or text “whoza” to 22333

Circuit complexity example 2



- Let $f(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$
- What is the circuit complexity of f?
- Answer: $\Theta(n)$
- Each "$\oplus$ gate" can be implemented using $O(1)$ many $\wedge$, $\vee$, $\neg$ gates

Every function has a circuit

- Are there functions with infinite circuit complexity?
- Recall: Some languages cannot be decided by algorithms
- Are there functions that cannot be computed by circuits?

Theorem: For every $f: \{0,1\}^n \to \{0,1\}^m$,

there exists a circuit that computes f.

- For simplicity, let's only prove the case $m = 1$

Boolean expressions: Literals

- For the proof, we will reason about Boolean expressions
- A Boolean expression is a way of representing a function $f: \{0,1\}^n \to \{0,1\}$ as a string
- The simplest Boolean expression is a single variable "x_i"
- We use the notation $\bar{x}_i$ to denote the negation of x_i (also denoted $\neg x_i$)
- **Definition:** A literal is a variable or its negation (x_i or $\bar{x}_i$)

Conjunctive normal form formulas

- **Definition:** A clause is a disjunction (OR) of literals. Example:

$$x_1 \vee \bar{x}_2 \vee x_7$$

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

$$\phi = (x_1 \vee \bar{x}_2) \wedge (x_5 \vee x_1 \vee x_2) \wedge (x_3 \vee \bar{x}_5 \vee x_4)$$

- In other words, a CNF formula is an AND of ORs of literals

Every function has a CNF formula

- Let $f: \{0,1\}^n \to \{0,1\}$ be any function

Lemma: The function f can be represented by a CNF formula in which there are at most 2^n clauses and each clause has at most n literals.

- **Proof:** For each $z \in \{0,1\}^n$ such that $f(z) = 0$, we make a clause C_z asserting that $x \neq z$
- Example: $f(x_1, x_2) = x_1 \oplus x_2 = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

Every function has a circuit

Theorem: For every $f: \{0,1\}^n \rightarrow \{0,1\}$, there exists a circuit of size $O(n \cdot 2^n)$ that computes f.

- **Proof:** Using the CNF representation, we have $f(x) = \bigwedge_{i=1}^{2^n} \bigvee_{j=1}^{n} \ell_{ij}$ where each ℓ_{ij} is a literal
- Circuit: A tree of $O(2^n)$ many $\wedge$ gates. At each leaf, we have a tree of $O(n)$ many $\vee$ gates. At each leaf of that tree, we have a variable and possibly a $\neg$ gate

Polynomial-size circuits

- We showed that every function has a circuit, but the circuit we constructed has exponential size
- Which functions have polynomial circuit complexity?
- Technically, it wouldn't make sense to say that an individual function $f : \{0,1\}^n \to \{0,1\}$ has "polynomial circuit complexity," because n is fixed
- Therefore, let's switch to our familiar framework of languages

Circuit complexity of a binary language

- Let $L \subseteq \{0,1\}^*$ be a language
- For each $n \in \mathbb{N}$, we define $L_n: \{0,1\}^n \to \{0,1\}$ by the rule

$$L_n(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{if } w \notin L \end{cases}$$

- We define the circuit complexity of L to be the function $S: \mathbb{N} \to \mathbb{N}$ defined by $S(n) =$ the size of the smallest circuit that computes L_n
- Note: Each circuit only handles a single input length! Different from TMs

Circuit complexity of an arbitrary language

- More generally, let $L \subseteq \Sigma^*$ be a language where $|\Sigma| \geq 2$
- Let $r = \lceil \log |\Sigma| \rceil \geq 1$
- For each $n \in \mathbb{N}$, we define $L_n : \{0,1\}^{nr} \to \{0,1\}$ by the rule

$$L_n(\langle w \rangle) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{if } w \notin L \end{cases}$$

- We define the circuit complexity of L to be the function $S : \mathbb{N} \to \mathbb{N}$ defined by $S(n) =$ the size of the smallest circuit that computes L_n

The complexity class PSIZE

- Let $S: \mathbb{N} \to \mathbb{N}$ be a function
- **Definition:** $\mathrm{SIZE}(S)$ is the set of all languages L such that the circuit complexity of L is $O(S)$
- **Definition:** PSIZE is the set of all languages with polynomial circuit complexity:

$$\mathrm{PSIZE} = \bigcup_{k=1}^{\infty} \mathrm{SIZE}(n^k)$$