Computational Complexity Theory

Lecture 16: Class AC and NC; P-completeness

Department of Computer Science,
Indian Institute of Science

Recap: Boolean circuits

- A Boolean circuit is a directed acyclic graph whose nodes/gates are labelled as follows:
  - A node with in-degree zero is labelled by an input variable, and it outputs the value of the variable.
  - Any other node is labelled by one of the three operations $\wedge$, $\vee$, $\neg$, and it outputs the value of the operation on its input.

  Nodes with out-degree zero are the output gates.

- **Size** of circuit is the no. of edges in it. **Depth** is the length of the longest path from an i/p to o/p node.

Recap: Boolean circuits

- A Boolean circuit is a directed acyclic graph whose nodes/gates are labelled as follows:
  - A node with in-degree zero is labelled by an input variable, and it outputs the value of the variable.
  - Any other node is labelled by one of the three operations $\wedge$, $\vee$, $\neg$, and it outputs the value of the operation on its input.

  Nodes with out-degree zero are the output gates.

- **Size** corresponds to "sequential time complexity". **Depth** corresponds to "parallel time complexity".

Recap: Class P/poly

- Let $T: \mathbb{N} \to \mathbb{N}$ be some function.
- Definition: A $T(n)$-size circuit family is a set of circuits $\{C_n\}_{n \in \mathbb{N}}$ such that C_n has n inputs and $|C_n| \leq T(n)$.

- Definition: A language L is in $SIZE(T(n))$ if there's a $T(n)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$ such that

$$x \in L \iff C_n(x) = 1, \text{ where } n = |x|.$$

- Defintion: Class $P/poly = \bigcup_{c \geq 1} SIZE(n^c)$.

Recap: Class P/poly

- Observation: $P \subseteq P/poly$.
- Is $P = P/poly$? No! $P/poly$ contains undecidable languages.

- Let $UHALT = \{1^{\#(M,y)} : (M,y) \in HALT\}$. Then, UHALT is also an undecidable language.

- Obs. Any unary language is in $P/poly$.
- Hence, $P \subsetneq P/poly$.

Recap: Class P/poly

- What makes P/poly contain undecidable languages? *Ans:* $L \in$ P/poly implies that L is decided by a circuit family $\{C_n\}$, where $|C_n| = n^{O(1)}$. We don't require that C_n is poly-time computable from 1^n.
- P/poly is a *non-uniform class* as a language in this class is allowed to have different algorithms/circuits for different input lengths.
- P is a *uniform class* as a language in this class has one algorithm for all inputs.
- Is SAT $\in$ P/poly? In other words, is NP $\subseteq$ P/poly?

Recap: Karp-Lipton theorem

- Theorem (*Karp & Lipton 1982*). If $NP \subsetneq P/poly$ then $PH = \sum_2$.

- If we can show $NP \not\subset P/poly$ assuming $P \neq NP$, then

  $NP \not\subset P/poly \iff P \neq NP$.

- Karp-Lipton theorem shows $NP \not\subset P/poly$ assuming the stronger statement $PH \neq \sum_2$.

Recap: Functions outside P/poly

- Are there Boolean functions (i.e., languages) outside P/poly? Yes! There are many. Let $\exp(m) = 2^m$.
- Theorem. $1 - \exp(-2^{n-1})$ fraction of Boolean functions on n variables **do not** have circuits of size $2^n/(22n)$.

Recap: Functions outside P/poly

- Are there Boolean functions (i.e., languages) outside P/poly? Yes! There are many. Let $\exp(m) = 2^m$.
- Theorem. $1 - \exp(-2^{n-1})$ fraction of Boolean functions on n variables **do not** have circuits of size $2^n/(22n)$.
- Is one out of so many functions outside P/poly in NP? We don't know even after ~40 yrs of research!
- Theorem. *(Iwama, Lachish, Morizumi & Raz 2002)* There is a language $L \in NP$ such that any circuit C_n that decides $L \cap \{0,1\}^n$ requires $5n - o(n)$ many $\wedge$ and $\vee$ gates.

Recap: Lower bound for Boolean formulas

- Nevertheless, the clean combinatorial structure of a circuit has been used to prove lower bounds for some natural classes of circuits.
- The proofs of these lower bounds introduced and developed some highly interesting techniques.

- Fact. PARITY(x_1, x_2, …, x_n) can be computed by a circuit of size $O(n)$ and a formula of size $O(n^2)$.

Recap: Lower bound for Boolean formulas

- Nevertheless, the clean combinatorial structure of a circuit has been used to prove lower bounds for some natural classes of circuits.
- The proofs of these lower bounds introduced and developed some highly interesting techniques.

- Theorem. *(Khrapchenko 1971)* Any formula computing PARITY($x_1, x_2, \ldots, x_n$) has size $\Omega(n^2)$.

Recap: Lower bound for Boolean formulas

- Nevertheless, the clean combinatorial structure of a circuit has been used to prove lower bounds for some natural classes of circuits.
- The proofs of these lower bounds introduced and developed some highly interesting techniques.

- Theorem. (*Andreev 1987, Hastad 1998*) There's a f that can be computed by a $O(n)$-size circuit such that any formula computing f has size $\Omega(n^{3-o(1)})$.

Recap: Lower bound for Boolean formulas

- Nevertheless, the clean combinatorial structure of a circuit has been used to prove lower bounds for some natural classes of circuits.
- The proofs of these lower bounds introduced and developed some highly interesting techniques.

- Conjecture. *(Circuits more powerful than formulas)* There's a f that can be computed by a $O(n)$-size circuit such that any formula computing f has size $n^{\omega(1)}$.

Recap: LB for constant depth circuits

- Nevertheless, the clean combinatorial structure of a circuit has been used to prove lower bounds for some natural classes of circuits.
- The proofs of these lower bounds introduced and developed some highly interesting techniques.

- We'll discuss a lower bound for constant depth circuits in the next lecture.

Recap: Non-uniform size hierarchy

- Shanon's result. There's a constant $c \geq 1$ such that every Boolean function in n variables has a circuit of size at most $c.(2^n/n)$.

- Theorem. There's a constant $d \geq 1$ s.t. if $T_1: N \to N$ & $T_2: N \to N$ and $T_1(n) \leq d^{-1}.T_2(n) \leq T_2(n) \leq c.(2^n/n)$ then

$$SIZE(T_1(n)) \subsetneq SIZE(T_2(n)).$$

Class NC^i and AC^i

Class NC

- NC stands for Nick's Class – named after Nick Pippenger.
- Definition. For $i \in \mathbb{N}$, a language L is in NC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Definition. $NC = \bigcup_{i \in \mathbb{N}} NC^i$.

Class NC

- NC stands for Nick's Class – named after Nick Pippenger.
- Definition. For $i \in \mathbb{N}$, a language L is in NC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Definition. $NC = \bigcup_{i \in \mathbb{N}} NC^i$.

- *Homework:* PARITY is in NC^1.

Class NC

- NC stands for Nick's Class – named after Nick Pippenger.
- Definition. For $i \in \mathbb{N}$, a language L is in NC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Definition. $NC = \bigcup_{i \in \mathbb{N}} NC^i$.

- NC^1 = *poly(n)*-size Boolean formulas. (*Assignment*)

Class NC

- NC stands for Nick's Class – named after Nick Pippenger.
- Definition. For $i \in \mathbb{N}$, a language L is in NC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Further, L is in ***log-space uniform*** NC^i if C_n is implicitly log-space computable from 1^n.

Note: Sometimes NC^i is defined as log-space uniform NC^i.

Class NC

- NC stands for Nick's Class – named after Nick Pippenger.
- Definition. For $i \in \mathbb{N}$, a language L is in NC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Further, L is in ***log-space uniform*** NC^i if C_n is implicitly log-space computable from 1^n.

log-space uniform $NC \subseteq P$.

NC ≡ Efficient parallel computation

- Definition. A language L can be decided *efficiently in parallel* if there's a polynomial function q(.) and constants c & i s.t. $L \cap \{0,1\}^n$ can be decided using q(n) many processors in $c.(\log n)^i$ time.

NC ≡ Efficient parallel computation

- Definition. A language L can be decided *efficiently in parallel* if there's a polynomial function q(.) and constants c & i s.t. $L \cap \{0,1\}^n$ can be decided using q(n) many processors in $c.(\log n)^i$ time.
- Model: **PRAM** (has a central shared memory)
  - A processor can "deliver" a message to any other processor in O(log n) time.
  - A processor has O(log n) bits of memory and performs a poly-time computation at every step.
  - Processor computation steps are synchronized.

NC ≡ Efficient parallel computation

- Definition. A language L can be decided *efficiently in parallel* if there's a polynomial function q(.) and constants c & i s.t. $L \cap \{0,1\}^n$ can be decided using q(n) many processors in $c.(\log n)^i$ time.

- Observation. A language L is in NC if and only if L can be decided efficiently in parallel.
- Proof. Almost immediate from the assumptions on the parallel computation model.

Class AC

- Definition. For $i \in \mathbb{N} \cup \{0\}$, a language L is in AC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size **unbounded fan-in** circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Definition. $AC = \bigcup_{i \geq 0} AC^i$. (stands for *Alternating Class*)

Class AC

- Definition. For $i \in \mathbb{N} \cup \{0\}$, a language L is in AC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size **unbounded fan-in** circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Definition. $AC = \bigcup_{i \geq 0} AC^i$.

- Observation. $AC^i \subseteq NC^{i+1} \subseteq AC^{i+1}$ for all $i \geq 0$.

Replace an unbounded fan-in gate by a binary tree of bounded fan-in gates.

Class AC

- Definition. For $i \in \mathbb{N} \cup \{0\}$, a language L is in AC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size **unbounded fan-in** circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Definition. $AC = \bigcup_{i \geq 0} AC^i$.

- Observation. $NC = AC$.

Class AC

- Definition. For $i \in \mathbb{N} \cup \{0\}$, a language L is in AC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size **unbounded fan-in** circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Definition. $AC = \bigcup_{i \geq 0} AC^i$.

- In the next lecture, we'll show that PARITY is not in AC^0, i.e., $AC^0 \subsetneq NC^1$.

Class AC

- Definition. For $i \in \mathbb{N} \cup \{0\}$, a language L is in AC^i if there is a polynomial function $q(.)$ and a constant c s.t. L is decided by a $q(n)$-size **unbounded fan-in** circuit family $\{C_n\}_{n \in \mathbb{N}}$, where depth of C_n is at most $c.(\log n)^i$ for every $n \in \mathbb{N}$.

- Definition. $AC = \bigcup_{i \geq 0} AC^i$.

- Further, L is in ***log-space uniform*** AC^i if C_n is implicitly log-space computable from 1^n.

log-space uniform $AC \subseteq P$.

P-completeness

P-completeness

- Recall, to define completeness of a complexity class, we need an appropriate notion of a *reduction*.
- What kind of reductions will be suitable is guided by *a complexity question*, like a comparison between the complexity class under consideration & another class.
- Is P = (uniform) NC? Is P = L?…use log-space reduction!

- Definition. A language $L \in P$ is *P-complete* if for every L' in P, $L' \leq_l L$.

P-complete problems

- Circuit value problem. Given a circuit and an input, compute the output of the circuit. (The reduction in the Cook-Levin theorem can be made a log-space reduction.)

- Linear programming. Check the feasibility of a system of linear inequality constraints over rationals. (Assignment problem)

- CFG membership. Given a context-free grammar and a string, decide if the string can be generated by the grammar.

No log-space algo for PC problems

- Theorem. Let L be a P-complete language. Then,

  L is in L $\Leftrightarrow$ P = L.

- Proof. Easy.

- Can't hope to get a log-space algorithm for a P-complete problem unless P = L.

No parallel algo for PC problems

- Theorem. Let L be a P-complete language. Then,

  L is in NC $\Leftrightarrow$ P $\subseteq$ NC.

- Proof. $\Leftarrow$ direction is straightforward.

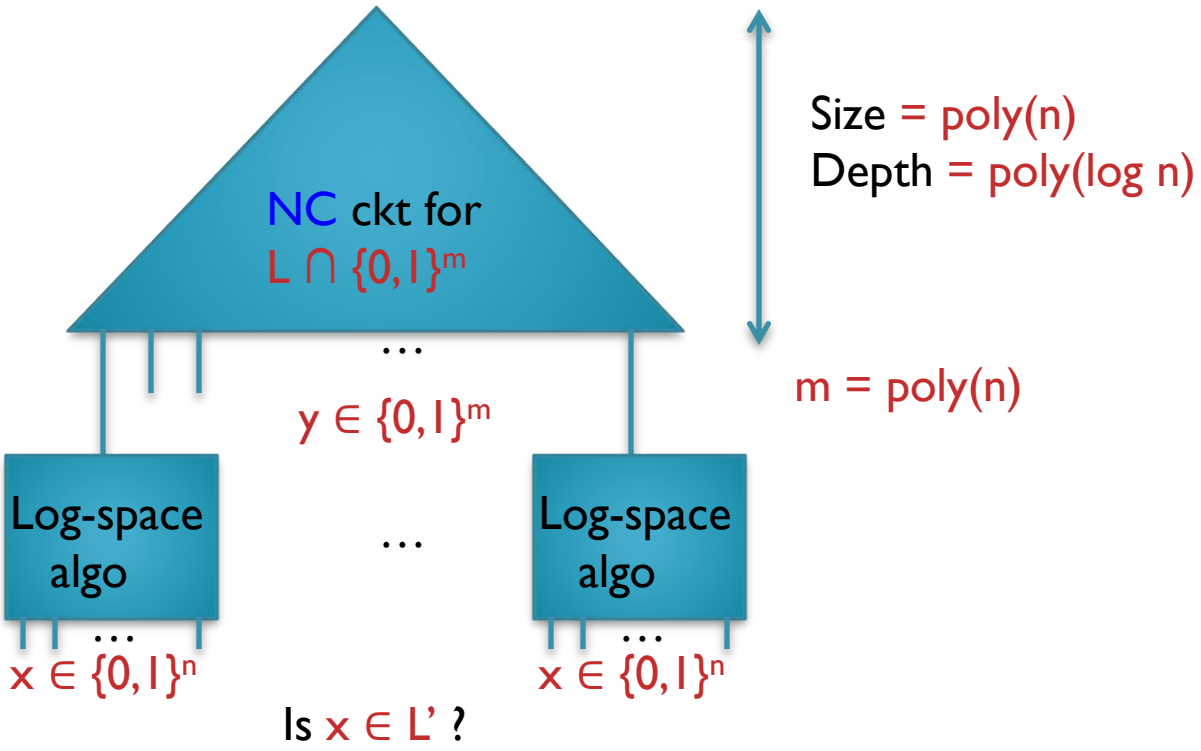
- Can't hope to get an efficient parallel algorithm for a P-complete problem unless P $\subseteq$ NC.

No parallel algo for PC problems

- Theorem. Let L be a P-complete language. Then,

  L is in NC $\iff$ P $\subseteq$ NC.

- Proof.($\Rightarrow$) Let $L' \in P$. As L is P-complete, $L' \leq_l L$.

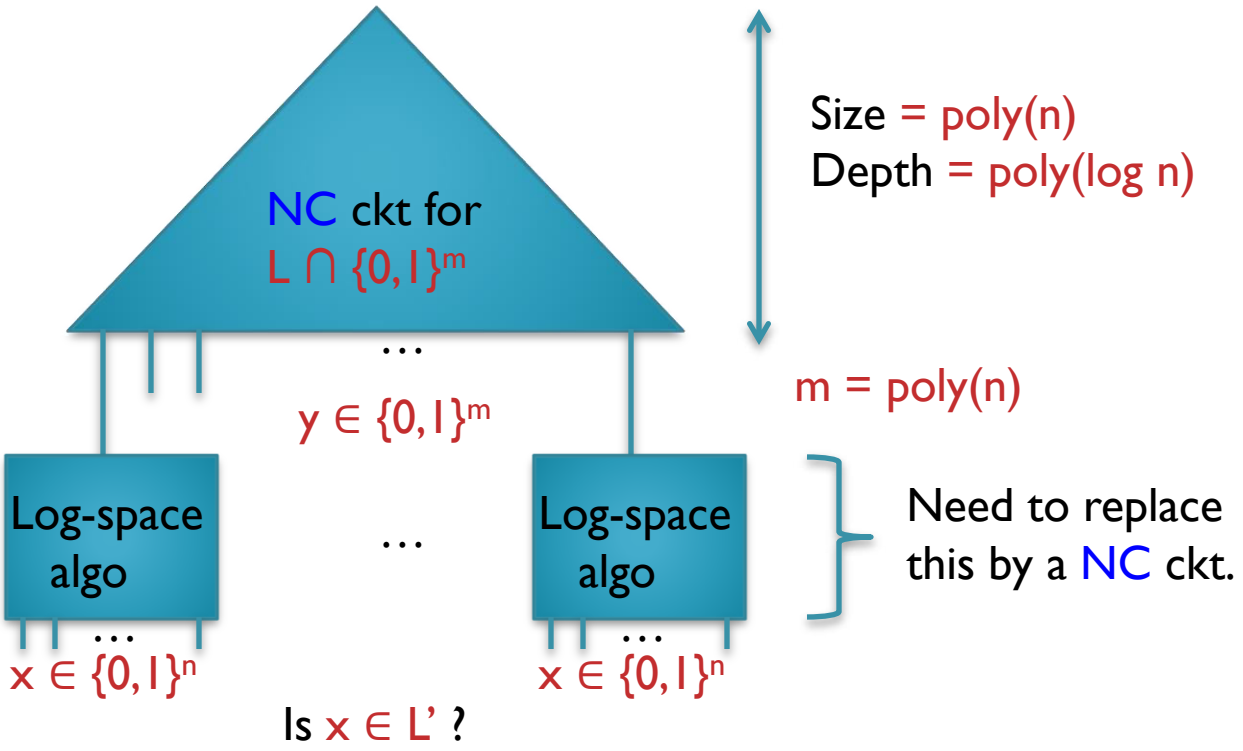


No parallel algo for PC problems

- Theorem. Let L be a P-complete language. Then,

  L is in NC $\Leftrightarrow$ P $\subseteq$ NC.

- Proof.($\Rightarrow$) Let L' $\in$ P. As L is P-complete, L' $\leq_l$ L.



Parallelization of Log-space

- Do problems in L have efficient parallel algorithms? Yes!

- Theorem. NL ⊆ (uniform) NC. *(Assignment problem)*

Parallelization of Log-space

- Do problems in L have efficient parallel algorithms? Yes!

- Theorem. NL ⊆ (uniform) NC. *(Assignment problem)*
- Proof sketch.
- 1. Construct the adjacency matrix A of the configuration graph.
- 2. Use repeated squaring of A to find out if there's a path from start to accept configurations.

Complexity zoo