CS 810: Introduction to Complexity Theory 02/18/03

Lecture 09: Sparse sets and Polynomial-Size Circuits

Instructor: Jin-Yi Cai Scribe: Christopher Hudzik, Sarah Knoop, Louis Kruger

In this lecture, we discuss sparse sets, polynomial size circuits and the connection between the two.

1 Nonuniform Complexity

Uniform complexity deals with one finite machine working on inputs of any length. Nonuniform complexity deals with a family of circuits $\{C_i\}_{i\in\mathbb{N}}$, where each C_i computes only on inputs of size i. Note that, given a length n, we may not know how to construct the circuit C_n. This gives rise to the term "nonuniform". We first formalize the notion of a circuit.

Definition 1 (Boolean Circuit). *A* boolean circuit *is an acyclic digraph with some input nodes and a unique output node. The input nodes have indegree of zero and are labelled by a variable or a Boolean constant, 0, 1. Each non-input node (also called a* gate *is labelled by a $\wedge$, for conjunction, or $\vee$, for disjunction, or by a $\neg$, for negation. The $\neg$-gates have indegree zero, whereas, other gates can have arbitrary indegree. The unique output node has zero outdegree. A* degree n boolean circuit *is a boolean circuit with the indegree of each conjunction and disjunction node bounded by n. The* size *of a boolean circuit is the number of nodes.*

By definition, any circuit computes a boolean function. Conversely, any boolean function is computed by some circuit. Given a boolean function $f : \{0,1\}^n \to \{0,1\}$ over n variables, we first write it in disjunctive normal form (DNF) circuit, as follows Define A by $\{\sigma | f(\sigma) = 1\}$, the set of satisfying assignments of f. For any $\sigma \in A$ and any variable x_i of f, define $l_i^\sigma = x_i$ if x_i is assigned 1 in σ, and $l_i^\sigma = \neg x_i$ if x_i is assigned 0 in σ. Then, f is given by the formula

$$f = \bigvee_{\sigma\in A} \bigwedge_{i=1}^{n} l_i^\sigma$$

We construct a circuit C to compute the right hand side of above equation.

Definition 2 (Polynomial-Size Circuit Family). *A language L is said to have a* polynomial-size circuit family *if there is some constant c such that, for every $n \in \mathbb{N}$, there exists a boolean circuit C_n such that*

1. *$|C_n| \leq n^c$, and*
2. *$\forall x \in \{0,1\}^n, x \in L \Leftrightarrow C_n(x) = 1$*

Example: For a set of n bits, $b_1, b_2, \ldots, b_n$, compute the parity of the bits, i.e. compute $b_1 \oplus b_2 \oplus \ldots \oplus b_n$. Here $\oplus$ refers to the XOR function. A circuit to compute the parity of n variables is shown in Figure 1. (We can implement each 2-input $\oplus$ gate using a constant amount of primitive gates $(\vee, \wedge, \neg)$.) The circuit uses $O(n)$ gates. ⊠

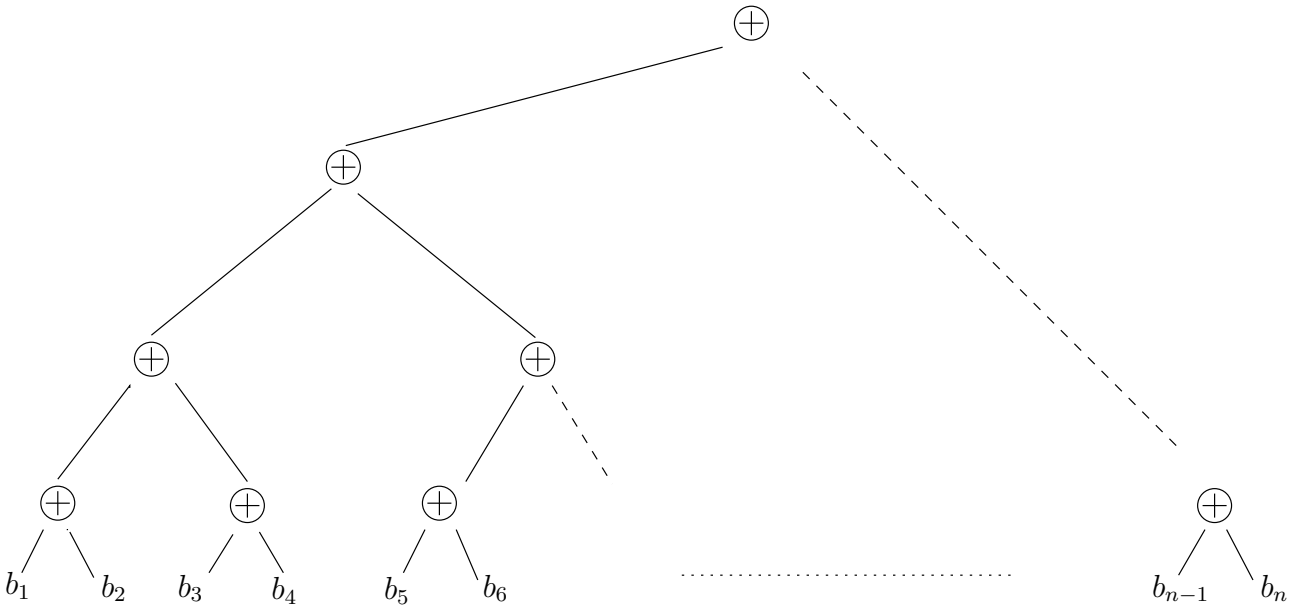

Figure 1: Boolean circuit of $O(log\, n)$ depth and $O(n)$ size computing parity.

Definition 3 (Circuit Complexity). *For a language L and integer n let $C_L(n)$ be the size of the smallest circuit computing L at length n, i.e. , i $C_L(n) \doteq \min\{|C_n| \mid C_n(x) = L(x), \forall x \in \{0,1\}^n\}$. The function $C_L(n)$ is called the* circuit complexity *of L.*

A language L is said to have polynomial-size circuits if $C_L(n) \leq n^c$, for some constant c. We define P/*poly* by the class of languages with polynomial-size circuits.

2 $\mathbf{P} \subseteq \mathbf{P}/poly$

Theorem 1. *P⊆P/poly (i.e. any polynomial time computation can be reduced to a polynomial size circuit)*

Proof. The idea behind this proof is to model the accepting computation of the TM as a circuit, and show this circuit is of polynomial size. This proof is reminiscent of the proof of Cook's theorem.

Let M be a polynomial time deterministic Turing machine. Let its running time be $T(n)$. Represent the computation as a tableaux. The machine can use at most $T(n)$ tape cells. To represent a configuration (state of the machine + position of tape head + contents of the tape) we need only $b = c * T(n)$ bits. where, c is some constant that depends only on the size of the alphabet and number of states. Note that b is still a polynomial in n. Now construct a circuit that has $T(n) + 1$ levels. Each level takes as input a configuration and outputs the next configuration (according to the computation of the machine). The new state depends on the current state, the current symbol (determined by current tape head position and tape contents) and transition function of the machine. Computing the new state takes only polynomial number of gates. Similarly, computing new tape head position takes only polynomial number of gates. To compute the new tape contents: content of the cell currently under the tape head may need to be changed; contents of other cells do not change. It can be seen that, using polynomial number of gates the new configuration can

be computed from the current configuration. We use $T(n)$ such levels in the circuit to simulate the computation of the machine through $T(n)$ steps. Finally, we check if the state output by the last level is accepting. □

We have shown that P $\subseteq$ P/*poly*. It is relevant to ask whether NP $\subseteq$ P/*poly*. It is a challenging open question. But, there is some evidence to show that NP may not be contained in P/*poly*. The Karp–Lipton theorem says that if NP is contained in P/*poly* the the polynomial hierarchy collapses to the second level, i.e.

$$\text{NP} \subseteq \text{P}/poly \Longrightarrow \text{PH} = \Sigma_2^p = \Pi_2^p.$$

We will prove the theorem in next lecture. How about the reverse containment? Is P/poly contained in P? Is P/poly contained in NP? Answer to either question is negative. In fact, P/poly contains some undecidable languages! Here, we shall show that a version of the undecidable halting problem is in P/poly.

Consider the halting problem represented in unary: $L = \{1^n | M_n \text{ on input } n \text{ halts}\}$. Clearly, L is undecidable as it is nothing but the familiar halting problem written in unary. The main property of L is that it has at most one string of given length! We will now construct a family of polynomial size circuits $\{C_n\}$ that decide L. Let A_n be a circuit, that given an input string x of length n, accepts x iff $x = 1^n$. It is clear that such a circuit needs only $O(n)$ 2-input AND gates. Let B_n be a circuit that rejects all its inputs. Clearly, either A_n or B_n is a correct circuit at length n. Let C_n be the correct circuit! Now, the family $\{C_n\}$ decides L.

The above proof also shows the main difference between P and P/*poly*. We saw that P is contained in P/poly. For any language L in P, there is a polynomial size family of circuits $\{C_n\}$ that decide L. The main point is that, we can construct a polynomial time TM that takes 1^n as input as outputs the circuit C_n. In other words, the family $\{C_n\}$ is constructible *uniformly*. In contrast, consider a language L in P/poly. There is a family of circuits $\{C_n\}$ that decide L. But, there need not be a Turing machine that given 1^n outputs C_n. In other words, $\{C_n\}$ may not be uniformly constructible. Thus, $\{C_n\}$ is allowed (by definition of P/poly) to be a nonuniform family. For example, the family of circuits that we constructed for the unary halting problem is nonuniform. Given 1^n, no Turing machine can decide whether A_n or B_n is the correct circuit.

3 Sparse Sets

A set is said to be sparse if it has only a polynomial number of strings at any given length. These are sets with "low information content". Formally,

Definition 4 (Sparse Set). *A set $S \subseteq \{0,1\}^*$ is a* sparse set *if there exists a polynomial p such that $|S^{=n}| \leq p(n)$ for all $n \in \mathbb{N}$.*

Theorem 2. *$L \in P/poly \iff \exists$ a sparse set S such that $L \in P^S$*

Proof. Assume $L \in$ P/*poly*. To prove the forward implication, we will encode the circuit family in a sparse set S. Let $\{C_n\}_{n\in\mathbb{N}}$ be a family of polynomially sized circuits computing L. For each n, let $b_{n,1}b_{n,2}\dots b_{n,l_n}$ be a binary representation of C_n. Note that l_n is polynomially bounded in n. Define

$$S = \{\langle 1^n, i, b_{n,1}b_{n,2}\dots b_{n,i}\rangle | n \in \mathbb{N} \text{ and } i = 1, \dots, l_n\}$$

We claim that S is sparse: Fix any $N \in \mathbb{N}$. For any n, we added l_n strings of the form $\langle 1^n, \cdot, \cdot \rangle$. If $n > N$, all these l_n strings are of length $> N$. Thus, only for $n < N$, the l_n strings added could be of length $\leq N$. For each such n, we added only $l_n = \text{poly(n)}$ strings to S. Thus there can be at most $N * poly(n) = poly(N)$ strings of length $\leq N$. So S is sparse.

Now, we show that $L \in \mathrm{P}^S$: For a TM M to compute L on input x of length n, it will first extract the circuit C_n from S. To do this, it first queries $\langle 1^n, 1, 0 \rangle \in S$ and $\langle 1^n, 1, 1 \rangle \in S$. The answer to this query will determine b_{n1}, the first bit of C_n. Assuming that M has determined the first $i-1$ bits of C_n, the i^{th} pair of queries will be $\langle 1^n, i, b_1 b_2 \ldots b_{i-1} 0 \rangle \in S$ and $\langle 1^n, i, b_1 b_2 \ldots b_{i-1} 1 \rangle \in S$. The answer will determine the i^{th} bit, b_i, of C_n. M will make these queries until both are false, meaning that M already has the last bit of C_n. So, by induction, M can extract the binary representation C_n in polynomial time. The second stage of M's computation is simulating C_n on input x. This can be done in polynomial time, also. Hence, M's total running time is polynomial, implying that $L(M) = L \in \mathrm{P}^S$.

Assume there exists a sparse set S such that $L \in \mathrm{P}^S$. We will hardwire enough of S in C_n to cover all the queries that might be made to S by a polynomial-time TM computing L.
Suppose TM M computes L with queries to the oracle S. By Theorem 1, M can be simulated by a family of polynomial-sized circuits $\{C_n\}_{n \in \mathbb{N}}$. The queries to S can only be of polynomial-size, $p(n)$. Thus, the only part of S that M can query on input of size n is $S^{\leq p(n)}$, and, therefore, the only part of S that C_n needs is $S^{\leq p(n)}$. Because S is sparse, $S^{\leq p(n)}$ is of polynomial-size. Therefore, hardwiring $S^{\leq p(n)}$ into a circuit requires only a polynomial-size addition of size to the circuit. We now have a family of polynomial-size circuits $\{C'_n\}_{n \in \mathbb{N}}$ such that C'_n computes C_n, but also has $S^{\leq p(n)}$ hardwired. Any query that would be made to S can be computed by the part of C'_n encoding $S^{\leq p(n)}$. Therefore, $L \in \mathrm{P}/poly$. □