Notes for Lecture 4

This lecture is on boolean circuit complexity. We first define circuits and the function they compute. Then we consider families of circuits and the language they define. In Section 2, we see how circuits relate to other complexity classes by a series of results, culminating with the Karp-Lipton theorem which states that if **NP** problems can be decided with polynomial-size circuits, then **PH** $=$ Σ_2.

1 Circuits

A circuit C has n inputs, m outputs, and is constructed with AND gates, OR gates and NOT gates. Each gate has in-degree 2 except the NOT gate which has in-degree 1. The out-degree can be any number. A circuit must have no cycle. See Figure 1.

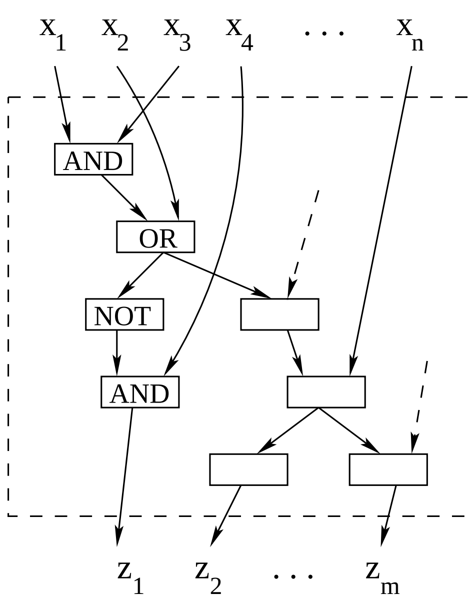

Figure 1: A Boolean circuit.

A circuit C with n inputs and m outputs computes a function $f_C : \{0,1\}^n \to \{0,1\}^m$. See Figure 2 for an example.

Define $\mathbf{SIZE}(C) = \#$ of AND and OR gates of C. By convention, we do *not* count the NOT gates.

To be compatible with other complexity classes, we need to extend the model to arbitrary input sizes:

Definition 1 *A language L is solved by a family of circuits $\{C_1, C_2, \ldots, C_n, \ldots\}$ if for every $n \geq 1$ and for every x s.t. $|x| = n$,*

$$x \in L \;\Leftrightarrow\; f_{C_n}(x) = 1.$$

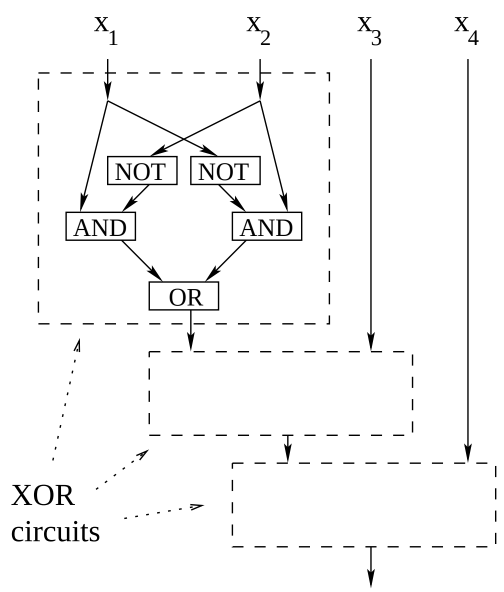

Figure 2: A circuit computing the boolean function $f_C(x_1x_2x_3x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

Definition 2 *Say $L \in \mathbf{SIZE}(s(n))$ if L is solved by a family $\{C_1, C_2, \ldots, C_n, \ldots\}$ of circuits, where C_i has at most $s(i)$ gates.*

2 Relation to other complexity classes

Proposition 1 *For every language L, $L \in \mathbf{SIZE}(O(2^n))$. In other words, exponential size circuits contain all languages.*

Proof: We need to show that for every 1-output function $f : \{0,1\}^n \to \{0,1\}$, f has circuit size $O(2^n)$.

Use the identity $f(x_1x_2\ldots x_n) = (x_1 \wedge f(1x_2\ldots x_n)) \vee (\overline{x}_1 \wedge f(0x_2\ldots x_n))$ to recursively construct a circuit for f, as shown in Figure 3.

The recurrence relation for the size of the circuit is: $s(n) = 3 + 2s(n-1)$ with base case $s(1) = 1$, which solves to $s(n) = 2 \cdot 2^n - 3 = O(2^n)$. □

Proposition 2 *If $L \in \mathbf{DTIME}(t(n))$, then $L \in \mathbf{SIZE}(O(t^2(n)))$.*

Proof: Let L be a decision problem solved by a machine M in time $t(n)$. Fix n and x s.t. $|x| = n$, and consider the $t(n) \times t(n)$ tableau of the computation of $M(x)$. See Figure 4.

Assume that each entry (a, q) of the tableau is encoded using k bits. By Proposition 1, the transition function $\{0,1\}^{3k} \to \{0,1\}^k$ used by the machine can be implemented by a "next state circuit" of size $k \cdot O(2^{3k})$, which is exponential in k but constant in n. This building block can be used to create a circuit of size $O(t^2(n))$ that computes the complete tableau, thus also computes the answer to the decision problem. This is shown in Figure 5.
□

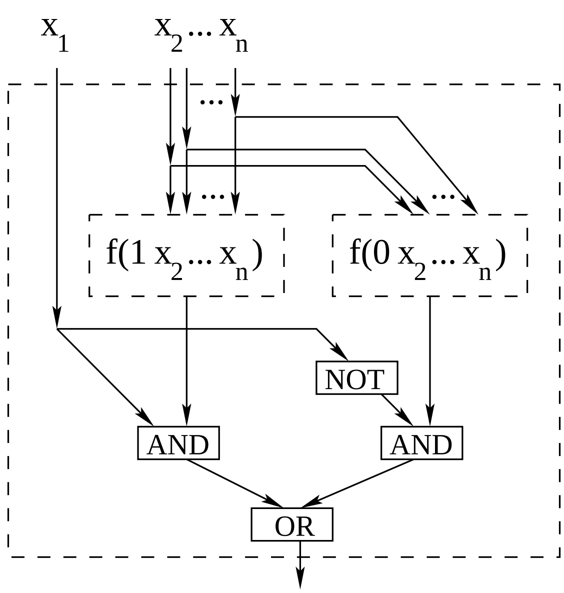

Figure 3: A circuit computing any function $f(x_1x_2\dots x_n)$ of n variables assuming circuits for two functions of $n-1$ variables.

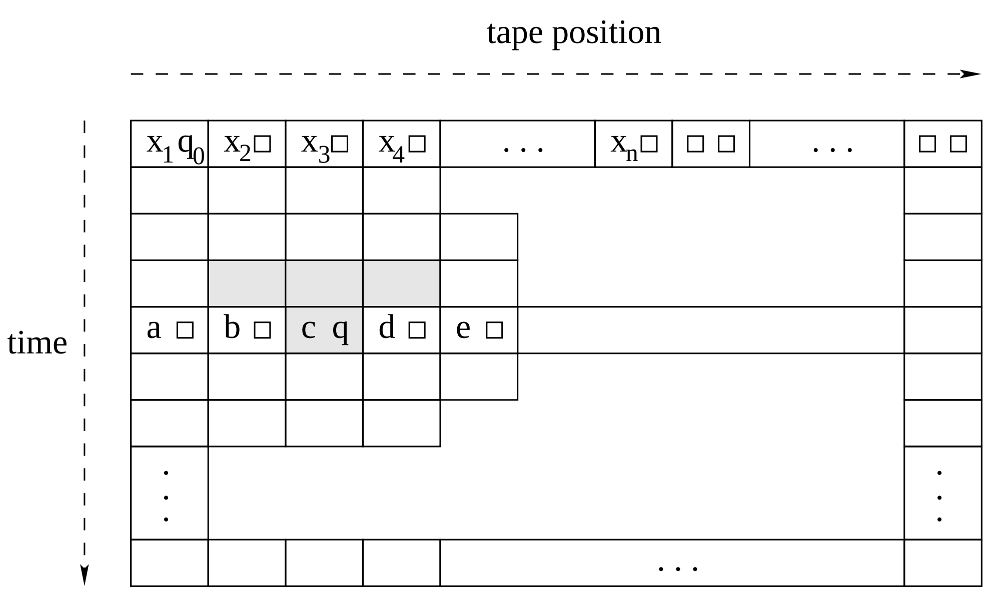

Figure 4: $t(n)\times t(n)$ tableau of computation. The left entry of each cell is the tape symbol at that position and time. The right entry is the machine state or a blank symbol, depending on the position of the machine head.

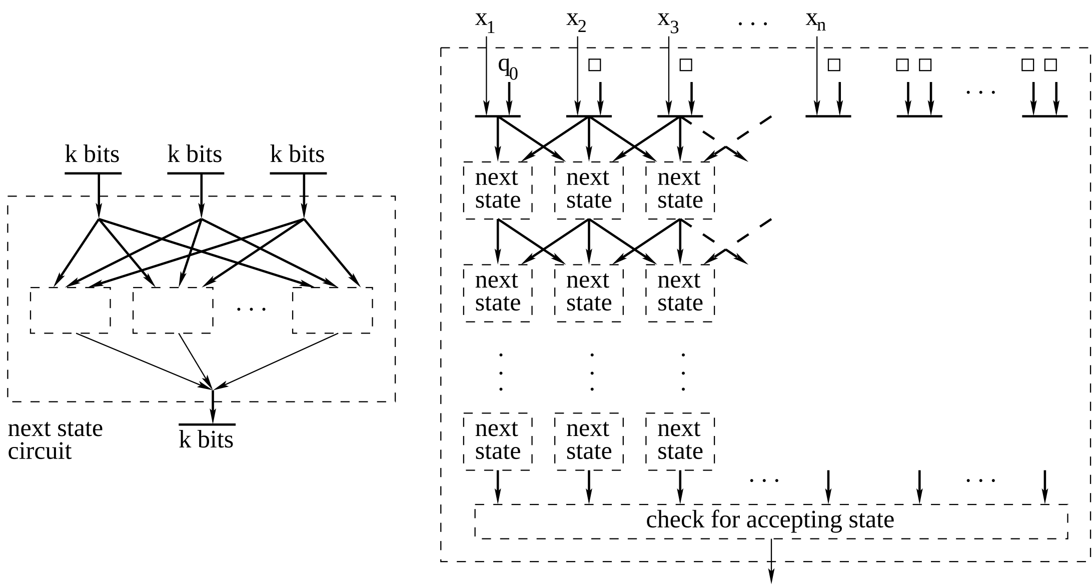

Figure 5: Circuit to simulate a Turing machine computation by constructing the tableau.

Corollary 3 $\mathbf{P} \subseteq \mathbf{SIZE}(n^{O(1)})$.

On the other hand, it's easy to show that $\mathbf{P} \neq \mathbf{SIZE}(n^{O(1)})$.

Proposition 4 *There are languages L such that $L \notin \mathbf{SIZE}(2^{o(n)})$. In other words, for every n, there exists $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by a circuit of size $2^{o(n)}$.*

Proof: This is a counting argument. There are 2^{2^n} functions $f : \{0,1\}^n \to \{0,1\}$, and we claim that the number of circuits of size s is at most $2^{O(s \log s)}$, assuming $s \geq n$. To bound the number of circuits of size s we create a compact binary encoding of such circuits. Identify gates with numbers $1, \ldots, s$. For each gate, specify where the two inputs are coming from, whether they are complemented, and the type of gate. The total number of bits required to represent the circuit is

$$s(2\log(n+s)+3) \leq s(2\log 2s+3) = s(2\log 2s+5).$$

So the number of circuits of size s is at most $2^{2s\log s+5s}$, and this is not sufficient to compute all possible functions if

$$2^{2s\log s+5s} < 2^{2^n}.$$

This is satisfied if $s = 2^{o(n)}$. $\square$

Theorem 5 (Karp-Lipton-Sipser) *If $\mathbf{NP} \subseteq \mathbf{SIZE}(n^{O(1)})$ then $\mathbf{PH} = \Sigma_2$. In other words, the polynomial hierarchy would collapse to its second level.*

Before proving the above theorem, we first show a result that contains some of the ideas in the proof of the Karp-Lipton-Sipser theorem.

Lemma 6 *If* $\mathbf{NP} \subseteq \mathbf{SIZE}(n^{O(1)})$ *then there is a family of polynomial-size circuits that on input a 3CNF formula* ϕ *outputs a satisfying assignment for* ϕ *if one such assignment exists and a sequence of zeroes otherwise.*

Proof: We define the circuits $C_n^1, \ldots, C_n^n$ as follows:

- C_n^1, on input a formula ϕ over n variables outputs 1 if and only if there is a satisfying assignment for ϕ where $x_1 = 1$,
- $\cdots$
- C_n^i, on input a formula ϕ over n variables and bits $b_1, \ldots, b_{i-1}$, outputs 1 if and only if there is a satisfying assignment for ϕ where $x_1 = b_1, \ldots, x_{i-1} = b_{i-1}, x_i = 1$
- $\cdots$
- C_n^n, on input a formula ϕ over n variables and bits $b_1, \ldots, b_{n-1}$, outputs 1 if and only if ϕ is satisfied by the assignment $x_1 = b_1, \ldots, x_{n-1} = b_{n-1}, x_n = 1$.

Also, each circuit realizes an **NP** computation, and so it can be built of polynomial size. Consider now the sequence $b_1 = C_n^1(\phi)$, $b_2 = C_n^2(b_1, \phi)$, $\ldots$, $b_n C_n^n(b_1, \ldots, b_{n-1}, \phi)$. The reader should be able to convince himself that this is a satisfying assignment for ϕ if ϕ is satisfiable, and a sequence of zeroes otherwise. □

We now prove the Karp-Lipton-Sipser theorem.

Proof: [Of Theorem 5] We will show that if $\mathbf{NP} \subseteq \mathbf{SIZE}(n^{O(1)})$ then $\Pi_2 \subseteq \Sigma_2$. By a result in a previous lecture, this implies that $\mathbf{PH} = \Sigma_2$.

Let $L \in \Pi_2$, then there is a polynomial $p()$ and a polynomial-time computable $V()$ such that

$$x \in L \text{ iff } \forall y_1.|y_1| \leq p(|x|)\exists y_2.|y_2| \leq p(|x|).V(x, y_1, y_2) = 1$$

By adapting the proof of Lemma 6 (see Figure 6), or by using the statement of the Lemma and Cook's theorem, we can show that, for every n, there is a circuit C_n of size polynomial in n such that for every x and every y_1, $|y_1| \leq p(|x|)$,

$$\exists y_2.|y_2| \leq p(|x|) \wedge V(x, y_1, y_2) = 1 \text{ iff } V(x, y_1, C_n(x, y_1)) = 1$$

Let $q(n)$ be a polynomial upper bound to the size of C_n.

So now we have that for inputs x of length n,

$$x \in L \text{ iff } \exists C_n.|C_n| \leq q(n).\forall y_1.|y_1| \leq p(n).V(x, y_1, C_n(x, y_1)) = 1$$

which shows that L is in Σ_2. □

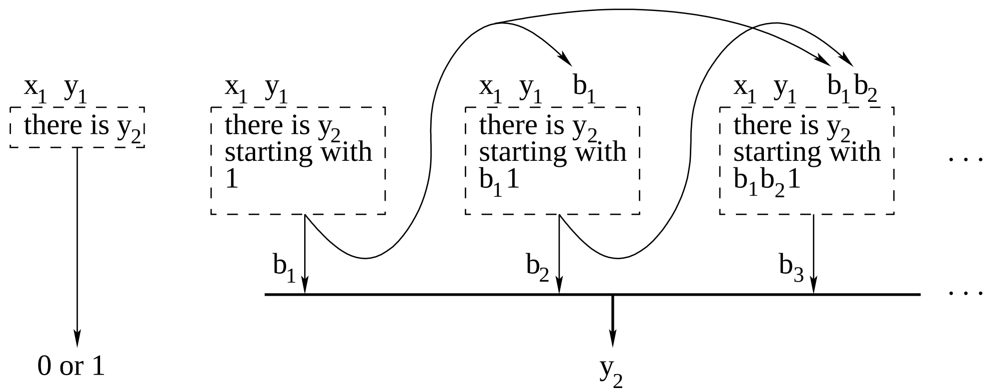

Figure 6: How to use decision problem solvers to find a witness to a search problem.

Exercises

1. Show that $\mathbf{SIZE}(n^{O(1)}) \not\subseteq \mathbf{P}$.

2. Show that there is a language in $\mathbf{SPACE}(2^{n^{O(1)}})$ that does not belong to $\mathbf{SIZE}(2^{o(n)})$.

3. Define $\mathbf{EXP} = \mathbf{DTIME}(2^{n^{O(1)}})$. Prove that if $\mathbf{EXP} \subseteq \mathbf{SIZE}(n^{O(1)})$ then $\mathbf{EXP} = \Sigma_2$.