U.C. Berkeley — CS278: Computational Complexity Handout N6
Professor Luca Trevisan 9/20/2002

Notes for Lecture 6

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.

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.

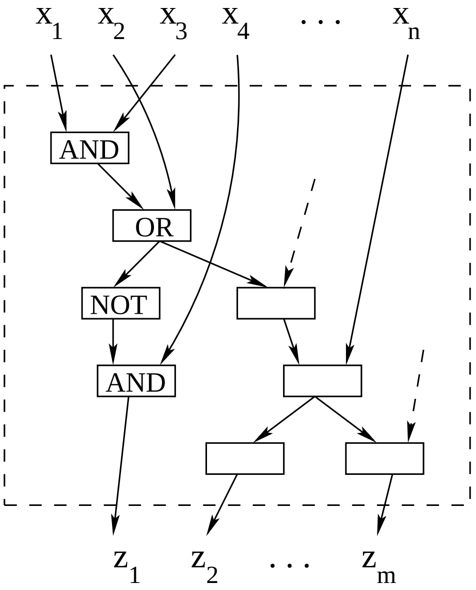

Figure 1: A Boolean circuit.

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 **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. $|x| = n$,*

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

Definition 2 *Say $L \in$ **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.*

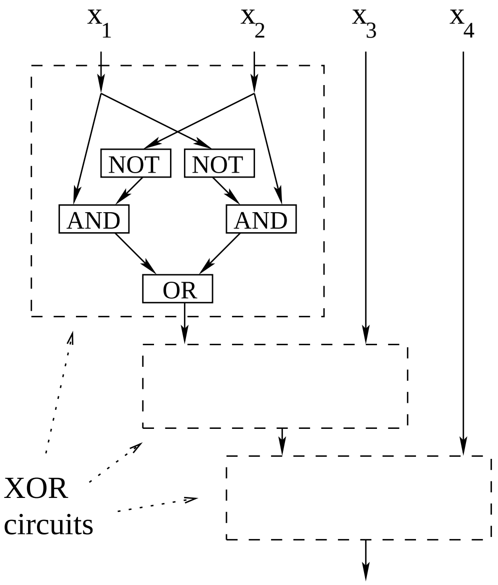

Figure 2: A circuit computing the boolean function $f_C(x_1x_2x_3x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

2 Relation to other complexity classes

Unlike other complexity measures, like time and space, for which there are languages of arbitrarily high complexity, the size complexity of a problem is always at most exponential.

Theorem 1 *For every language L, $L \in \mathbf{SIZE}(O(2^n))$.*

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 $f(x_1x_2\dots x_n) = (x_1 \wedge f(1x_2\dots x_n)) \vee (\overline{x}_1 \wedge f(0x_2\dots x_n))$ 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 $s(n) = 2 \cdot 2^n - 3 = O(2^n)$. □

The exponential bound is nearly tight.

Theorem 2 *There are languages L such that $L \notin \mathbf{SIZE}(o(2^n/n))$. In particular, for every $n \geq 11$, there exists $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by a circuit of size $2^n/4n$.*

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 $1, \dots, s$. 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

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

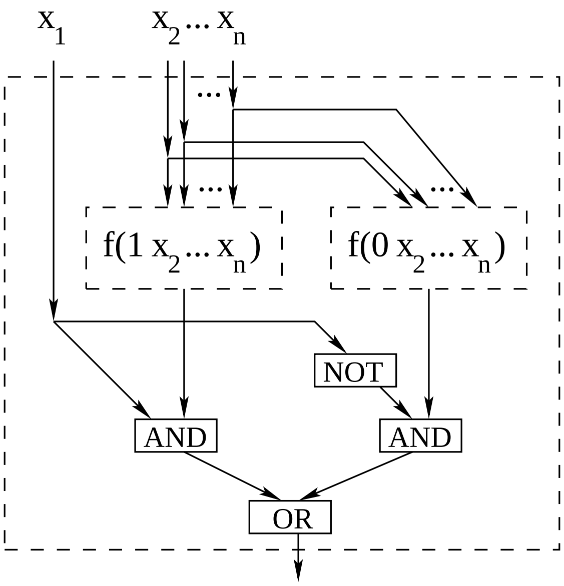

Figure 3: A circuit computing any function $f(x_1 x_2 \dots x_n)$ of n variables assuming circuits for two functions of $n-1$ variables.

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 $s \leq \frac{2^n}{4n}$ and $n \geq 11$. □

The following result shows that efficient computations can be simulated by small circuits.

Theorem 3 *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. □

Corollary 4 $\mathbf{P} \subseteq \mathbf{SIZE}(n^{O(1)})$.

On the other hand, it's easy to show that $\mathbf{P} \neq \mathbf{SIZE}(n^{O(1)})$, and, in fact, one can define languages in $\mathbf{SIZE}(O(1))$ that are undecidable.

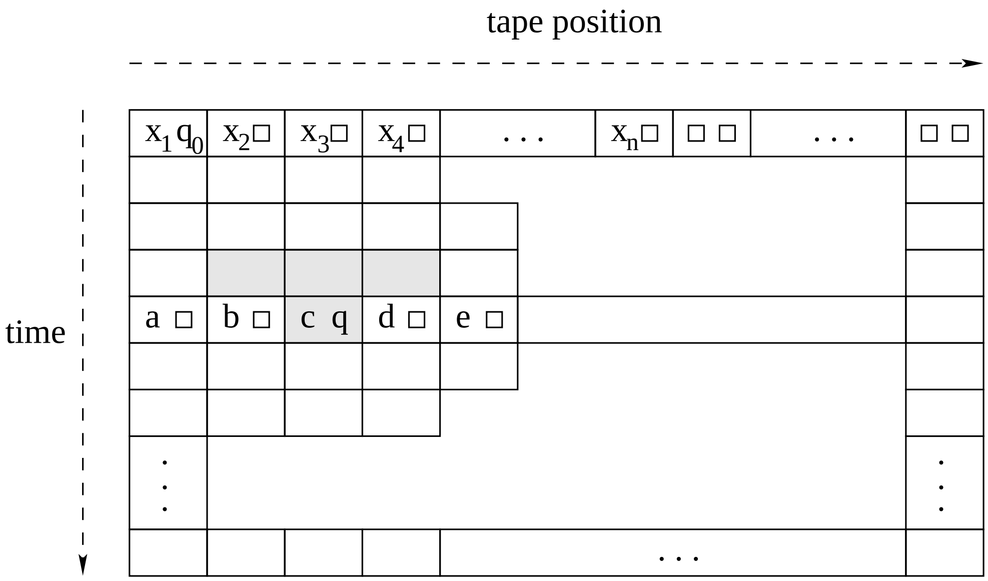

Figure 4: $t(n) \times t(n)$ tableau of computation. The left entry of each cell is the tape symbol at that position and time. The right entry is the machine state or a blank symbol, depending on the position of the machine head.

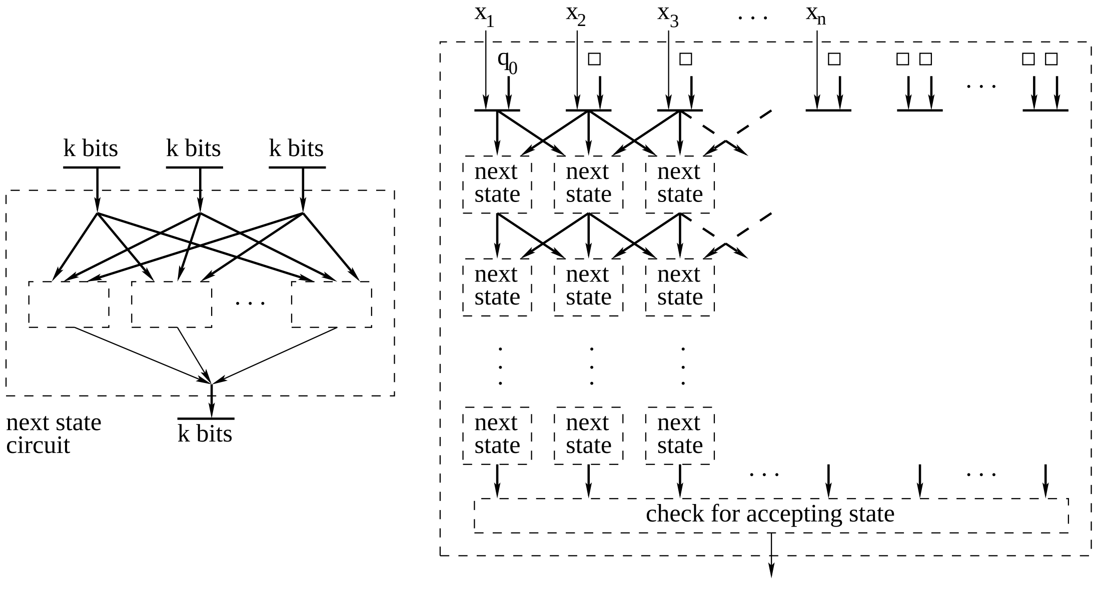

Figure 5: Circuit to simulate a Turing machine computation by constructing the tableau.

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 $\mathbf{SIZE}(2^{o(n)})$.