**Circuits vs. space complexity (lecture notes)**

Course: Circuit Complexity, Autumn 2024, University of Chicago
Instructor: William Hoza (williamhoza@uchicago.edu)

---

## 1 Uniform space complexity

**Theorem 1.** *Uniform* $\mathsf{NC}^1$ *is contained in* $\mathsf{L}$ *(deterministic log space).*

*Proof sketch.* We are given $x \in \{0,1\}^n$ and we wish to compute $C(x)$, where $C$ is a uniform log-depth circuit with bounded fan-in. We evaluate it recursively:

1. If $C$ is a constant or a literal, consult $x$ to compute $C(x)$.
2. Otherwise, let $C_L$ and $C_R$ be the two subcircuits feeding into the output gate.
3. Recursively compute $C_L(x)$. If $C_L(x) = 0$ and the output gate is $\wedge$, or if $C_L(x) = 1$ and the output gate is $\vee$, then halt and output $C_L(x)$.
4. Otherwise, recursively compute $C_R(x)$. Halt and output $C_R(x)$.

The recursion depth is $O(\log n)$. Crucially, after computing $C_L(x)$, we can re-use the same work space for computing $C_R(x)$. Therefore, we only store a single bit of information at each "inactive" level of the recursion ($L$ or $R$). At the "active" level of recursion, we use $O(\log n)$ bits of space to compute the structure of the relevant part of the circuit; this is possible by the uniformity assumption. □

**Theorem 2.** $\mathsf{NL}$ *(nondeterministic log space) is contained in uniform* $\mathsf{AC}^1$*.*

*Proof sketch.* For every $N, k \in \mathbb{N}$ and every $s, t \in [N]$, let us construct a circuit $C_{N,s,t,k} \colon \{0,1\}^{\binom{N}{2}} \to \{0,1\}$ such that if $E$ is the adjacency matrix of a directed graph on $N$ vertices, then $C_{N,s,t,k}(E)$ indicates whether there is a directed path from vertex $s$ to vertex $t$ of length at most $2^k$. The construction is recursive. If $k = 0$, then we set $C_{N,s,t,0}(E) = M_{s,t}$. If $k > 0$, then we set

$$C_{N,s,t,k}(E) = \bigvee_{u \in [N]} (C_{N,s,u,k-1}(E) \wedge C_{N,u,t,k-1}(E)).$$

Let $C_N = C_{N,1,N,\lceil \log N \rceil}$, and observe that $C_N$ determines whether there is a directed path from vertex 1 to vertex $N$ with no length restrictions. This problem ("$s$-$t$ connectivity") is complete for $\mathsf{NL}$. The circuit $C_N$ is an $\mathsf{AC}$ circuit of depth $2\lceil \log N \rceil$ and size $O(N^3 \cdot \log N)$, because each $C_{N,s,t,k}$ contributes $O(N)$ gates, and there are $N$ possible settings of $s$, $N$ possible settings of $t$, and $\log N$ possible settings of $k$. □

The proof above actually shows that $\mathsf{NL} \subseteq \mathsf{SAC}^1$, where $\mathsf{SAC}^1$ is the subclass of $\mathsf{AC}^1$ where the $\wedge$ gates all have bounded fan-in.

## 2 Nonuniform space complexity: Barrington's theorem

A *branching program* is a generalization of a decision tree in which vertices can have an unbounded number of incoming edges. The *size* of a branching program is the number of vertices. One can show that a function $f \colon \{0,1\}^* \to \{0,1\}$ can be computed by polynomial-size branching programs if and only if it can be computed by a log-space Turing machine with polynomially many bits of advice.

A more refined way to measure space complexity is to require a *layered* branching program, and to distinguish between the program's length and its width.

**Definition 1** (Bounded-width branching programs)**.** A *width-$w$ length-$T$ branching program $B$* is a directed acyclic graph on the vertex set $[w] \times \{0, 1, \ldots, T\}$. For every $i \in [T]$, each vertex $v \in [w] \times [i-1]$ is labeled with a variable from among $x_1, \ldots, x_n$, and it has two outgoing edges labeled 0 and 1 leading to $[w] \times [i]$.

We define $\mathsf{FinalState}_B \colon [w] \times \{0,1\}^n \to [w]$ as follows. Given $u \in [w]$ and $x \in \{0,1\}^n$, we start at the vertex $(u, 0)$. After reaching a vertex $(v, i)$, labeled with a variable $x_j$, we traverse the appropriate outgoing edge based on the value of $x_j$. In this manner, we walk through the graph, arriving at a vertex $(v, T)$. We define $\mathsf{FinalState}_B(u, x) = v$.

Finally, there is a designated "start state" $u_{\text{start}} \in [w]$ and a designated "accept state" $u_{\text{accept}} \in [w]$. The program $B$ computes a Boolean function $B \colon \{0,1\}^n \to \{0,1\}$ given by

$$B(x) = \begin{cases} 1 & \text{if } \mathsf{FinalState}_B(u_{\text{start}}, x) = u_{\text{accept}} \\ 0 & \text{otherwise.} \end{cases}$$

Width (or rather $\log w$) is should be interpreted as space complexity, while length should be interpreted as time complexity. The transitions can vary from one layer to the next, which can be interpreted to mean that the program has access to a "clock."

**Theorem 3** (Barrington's theorem)**.** *Let $f \colon \{0,1\}^* \to \{0,1\}$. Then $f \in \mathsf{NC}^1$ if and only if $f$ can be computed by constant-width, polynomial-length branching programs.*

*Proof of the easy direction.* Let $B$ be a width-$w$ length-$T$ branching program computing $f$ on inputs of length $n$, where $w = O(1)$ and $T = \text{poly}(n)$. Assume without loss of generality that $T$ is a power of two. We simulate $B$ by a circuit using the same idea we used to prove $\mathsf{NL} \subseteq \mathsf{AC}^1$. For every $0 \leq i \leq j \leq T$, let $B_{i \ldots j}$ be the branching program consisting of only layers $i, i+1, \ldots, j$ of $B$. Then we can recursively compute $\mathsf{FinalState}_B(u, x)$ using the formula

$$\mathsf{FinalState}_B(u, x) = v \iff \bigvee_{m \in [w]} (\mathsf{FinalState}_{B_{0 \ldots T/2}}(u, x) = m) \wedge (\mathsf{FinalState}_{B_{T/2 \ldots T}}(m, x) = v).$$

This leads to a circuit of depth $O(\log T)$ and size $\text{poly}(T, w)$, similar to the proof that $\mathsf{NL} \subseteq \mathsf{AC}^1$. However, the maximum fan-out is $w = O(1)$ this time, hence we get $B \in \mathsf{NC}^1$ rather than $\mathsf{AC}^1$. □

The other direction of Barrington's theorem is quite surprising. For example, it implies that the majority function can be computed in "constant space" and polynomial time, which is highly counterintuitive. We will present a proof due to Ben-Or and Cleve [BC92]. The crux of the matter is the following.

**Lemma 1.** *Let $C$ be a depth-$d$ circuit in which each gate is either an "AND" gate or an "XOR" gate, each with fan-in two, allowing negations at the inputs. There exists a branching program $B$ of width 8 and length $4^d$ such that for every $(r, s, t) \in \mathbb{F}_2^3 \cong [8]$ and every $x \in \{0,1\}^n$, we have $\mathsf{FinalState}_B((r, s, t), x) = (r', s, t)$, where*

$$r' = r + s \cdot C(x).$$

*Proof.* We prove it by induction on $d$. In the base case, if $d = 0$, then the lemma is trivial. Now assume $d \geq 1$. It's helpful to think of three "registers" $R, S, T$, which initially contain the values $r, s, t$. We will perform a sequence of "operations" on these registers so that the final values are $r', s, t$.

For the first case, suppose $C(x) = C_L(x) \oplus C_R(x)$ for some circuits $C_L, C_R$ of depth at most $d - 1$. In this case, we perform the following operations.

$$R \leftarrow R + S \cdot C_L(x)$$
$$R \leftarrow R + S \cdot C_R(x).$$

By induction, each of these operations can be implemented as a branching program of width 8 and length $4^{d-1}$.

Now, for the harder case, suppose $C(x) = C_L(x) \wedge C_R(x)$. In this case, we perform the following sequence of operations.

| Operation | New $R$ Value | New $S$ Value | New $T$ Value |
|---|---|---|---|
| $R \leftarrow R + T \cdot C_L(x)$ | $r + t \cdot C_L(x)$ | $s$ | $t$ |
| $T \leftarrow T + S \cdot C_R(x)$ | $r + t \cdot C_L(x)$ | $s$ | $t + s \cdot C_R(x)$ |
| $R \leftarrow R + T \cdot C_L(x)$ | $r + s \cdot C_L(x) \cdot C_R(x)$ | $s$ | $t + s \cdot C_R(x)$ |
| $T \leftarrow T + S \cdot C_R(x)$ | $r + s \cdot C_L(x) \cdot C_R(x)$ | $s$ | $t$ |

By induction, each of those operations can be performed by a branching program of width 8 and length $4^{d-1}$ (permuting the roles of $R, S, T$ if necessary). We concatenate those programs to get our program of length $4^d$. $\square$

*Proof of the hard direction of Barrington's theorem.* Suppose $f \in \mathsf{NC}^1$. Then $f$ can be computed by a log-depth circuit $C$ that only uses "AND" gates and "XOR" gates of fan-in two (and we allow negations at the inputs). This is because $a \vee b \equiv a \oplus b \oplus (a \wedge b)$. By Lemma 1, it follows that there is a branching program $B$ of width 8 and length $4^d$ such that for every $x \in \{0,1\}^n$, we have

$$\mathsf{FinalState}_B((0,1,0), x) = (C(x), 1, 0).$$

Let $u_{\text{start}} = (0,1,0)$ and $u_{\text{accept}} = (1,1,0)$. $\square$

## References

[BC92] Michael Ben-Or and Richard Cleve. "Computing algebraic formulas using a constant number of registers". In: *SIAM J. Comput.* 21.1 (1992), pp. 54–58. ISSN: 0097-5397. DOI: 10.1137/0221006.