Circuit Complexity

Circuit model aims to offer unconditional lower bound results.

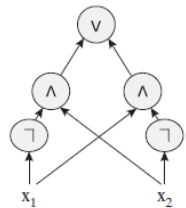
Synopsis

1. Boolean Circuit Model
2. Uniform Circuit
3. $\mathbf{P}_{/\mathbf{poly}}$
4. Karp-Lipton Theorem
5. $\mathbf{NC}$ and $\mathbf{AC}$
6. $\mathbf{P}$-Completeness
7. Håstad Switching Lemma

Boolean Circuit Model

Boolean circuit is a nonuniform computation model that allows a different algorithm to be used for each input size.

Boolean circuit model appears mathematically simpler, with which one can talk about combinatorial structure of computation.



Boolean Circuit

An n-input, single-output boolean circuit is a dag with

- n sources (vertex with fan-in 0), and
- one sink (vertex with fan-out 0).

All non-source vertices are called gates and are labeled with

- $\vee, \wedge$ (vertex with fan-in 2), and
- $\neg$ (vertex with fan-in 1).

A Boolean circuit is monotone if it contains no $\neg$-gate.

If C is a boolean circuit and $x \in \{0,1\}^n$ is some input, then the output of C on x, denoted by $C(x)$, is defined in the natural way.

Example

1. $\{1^n \mid n \in \mathbf{N}\}$.
2. $\{\langle m, n, m+n\rangle \mid m, n \in \mathbf{N}\}$.

Boolean Circuit Family

The size of a circuit C, notation $|C|$, is the number of gates in it.

Let $S : \mathbf{N} \to \mathbf{N}$ be a function.

An $S(n)$-size boolean circuit family is a sequence $\{C_n\}_{n \in \mathbf{N}}$ of boolean circuits, where C_n has n inputs and a single output, and $|C_n| \leq S(n)$ for every n.

Circuit Model Accepts Undecidable Language

A unary language $L \subseteq \{1^n \mid n \in \mathbf{N}\}$ is accepted by a linear size circuit family.

Fact. Not every unary language is decidable.

Nonuniform Complexity Class

A problem L is in $\mathbf{SIZE}(S(n))$ if there exists an $S(n)$-size circuit family $\{C_n\}_{n\in\mathbf{N}}$ such that for each $x \in \{0,1\}^n$, $x \in L$ iff $C_n(x) = 1$.

Unlike in a uniform model, $\mathbf{SIZE}(cS(n)) \neq \mathbf{SIZE}(S(n))$ for $c > 2$. This follows from Circuit Hierarchy Theorem.

Boolean Functions are Hard

Shannon. Most n-ary boolean functions have circuit complexity $> \frac{2^n}{n} - o(\frac{2^n}{n})$.

Lupanov. The size complexity of the hardest n-ary boolean function is $< \frac{2^n}{n} + o(\frac{2^n}{n})$.

Lutz. The size complexity of the hardest n-ary boolean function is $> \frac{2^n}{n}(1 + c\frac{\log n}{n})$ for some $c < 1$ and all large n.

---

1. C. Shannon. The Synthesis of Two-Terminal Switching Circuits. Bell System Technical Journal. 28:59-98, 1949.
2. O. Lupanov. The Synthesis of Contact Circuits. Dokl. Akad. Nauk SSSR (N.S.) 119:23-26, 1958.
3. J. Lutz. Almost Everywhere High Nonuniform Complexity. JCSS, 1992.

Shannon's Counting Argument

Fixing output gate, the number of functions defined by S-size circuits is bounded by

$$\begin{aligned}
\frac{(S+n+2)^{2S}3^S}{(S-1)!} &< \frac{(S+n+2)^{2S}(3e)^S}{S^S}S \\
&= (1+\frac{n+2}{S})^S(3e(S+n+2))^S S \\
&< (e^{\frac{n+2}{S}}3e(S+n+2))^S S \\
&< (3e^2(S+n+2))^S S \\
&< (6e^2 S)^S S.
\end{aligned}$$

To define an ϵ-fraction of the functions, $(7e^2S)^S \geq \epsilon 2^{2^n}$ must be valid. It follows that

$$S(\log(7e^2) + \log S) \geq 2^n - \log \epsilon^{-1}. \tag{1}$$

It is easy to see that $S \leq \frac{2^n}{n} - \log(\frac{1}{\epsilon})$ would contradict (1).

By Shannon Theorem and Lupanov Theorem, the circuit C_f for the hardest n-ary boolean function f has the following bounds:

$$|C_f| = \frac{2^n}{n} \pm o\left(\frac{2^n}{n}\right).$$

Frandsen and Miltersen have provided a proof of the following.

$$\frac{2^n}{n}\left(1 + \frac{\log n}{n} - O\left(\frac{1}{n}\right)\right) \leq |C_f| \leq \frac{2^n}{n}\left(1 + 3\frac{\log n}{n} + O\left(\frac{1}{n}\right)\right).$$

---

1. G. Frandsen and P. Miltersen. Reviewing Bounds on the Circuit Size of the Hardest Functions. Information Processing Letters, 2005.

Circuit Hierarchy Theorem

Theorem. If $n < (2+\epsilon)S(n) < S'(n) \ll 2^n/n$ for $\epsilon > 0$, then $\mathbf{SIZE}(S(n)) \subsetneq \mathbf{SIZE}(S'(n))$.

Let

$$m = \max m. \left(S'(n) \geq \left(1 + \frac{\epsilon}{4}\right) \frac{2^m}{m} \right).$$

The condition guarantees that $0 < m < n$ for large n. It follows from the assumption that

$$S(n) < \left(1 - \frac{\epsilon}{4+2\epsilon}\right) \frac{2^m}{m}.$$

Consider the set $\mathcal{B}_{m,n}$ of all n-ary Boolean functions that depend only on the first m inputs.

- By Lupanov Theorem, $\mathcal{B}_{m,n} \subseteq \mathbf{SIZE}(S'(n))$ for large n.
- By Shannon Theorem, $\mathcal{B}_{m,n} \not\subseteq \mathbf{SIZE}(S(n))$ for large n.

Uniform Circuit

A circuit family $\{C_n\}_{n\in\mathbf{N}}$ is uniform if there is an implicitly logspace computable function mapping 1^n to C_n.

Uniform Circuit Family and $\mathbf{P}$

Theorem. A language is accepted by a uniform circuit family if and only if it is in $\mathbf{P}$.

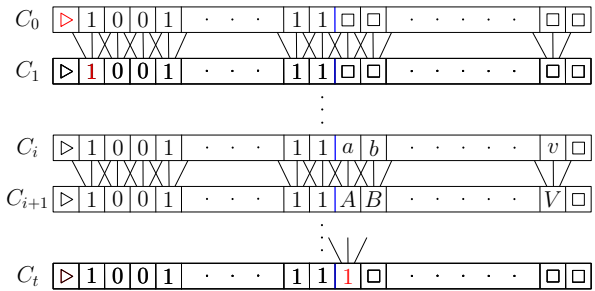
'$\Rightarrow$': Trivial.

'$\Leftarrow$': A proof is given on the next two slides.

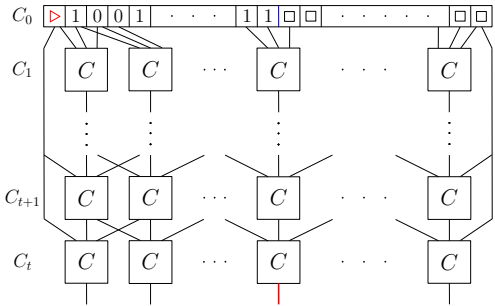
From **P** to Uniform Circuit Family

Suppose $\mathbb{M}$ is a one tape TM bounded in time by $T(n) = cn^c$.

- Given input 1^n one writes down $t = cn^c$ in logspace.
- Let $C_0, \ldots, C_t$ be the configurations. Let C_{ij} be the j-th symbol of C_i.



Boolean Formula for Turing Computation



Input to C_{ij} = output of $C_{(i-1)(j-1)}$, $C_{(i-1)j}$, $C_{(i-1)(j+1)}$.

All C_{ij}'s can be computed by a fixed circuit C whose size depends only on $\mathbb{M}$.

Circuit Satisfiability

A binary string is in CKT-SAT if it represents an n-input boolean circuit C such that $\exists u \in \{0,1\}^n . C(u) = 1$.

From **NP** to Circuit Satisfiability

Lemma. $L \leq_L$ `CKT-SAT` for every $L \in \mathbf{NP}$.

Let $L \in \mathbf{NP}$, p be a polynomial, and $\mathbb{M}$ be a P-time TM such that

$$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)}.\mathbb{M}(x,u) = 1.$$

Now apply to $\mathbb{M}(x,u)$ the logspace reduction defined on page 18.

From CKT-SAT to SAT

Lemma. CKT-SAT $\leq_L$ SAT.

Introduce a boolean variable for every gate input and every gate output.
The formula is a big conjunction of the clauses relating input and output variables.

Cook-Levin reduction is computable in logspace.

$\mathbf{P}_{/\mathbf{poly}}$

Turing Machines that Take Advice

Let $T, a : \mathbf{N} \to \mathbf{N}$ be functions. The class of languages decidable by $T(n)$-time TM's with $a(n)$ bits of advice, denoted

$$\mathbf{DTIME}(T(n))/a(n),$$

contains every L such that there exists a countable sequence $\{\alpha_n\}_{n\in\mathbf{N}}$ of strings with $\alpha_n \in \{0,1\}^{a(n)}$ and a TM $\mathbb{M}$ satisfying

$$x \in L \text{ iff } \mathbb{M}(x, \alpha_n) = 1$$

for all $x \in \{0,1\}^n$, where on input x, α_n the machine $\mathbb{M}$ stops in $O(T(n))$ steps.

Complexity Class $(\_)_{/\mathbf{poly}}$

Complexity class defined by TM using advice.

- $\mathbf{P}_{/\mathbf{poly}}$ is the class of languages decidable by P-time TM using P-size advice.
- $\mathbf{NP}_{/\mathbf{poly}}$ is the class of languages decidable by P-time NDTM using P-size advice.
- $\mathbf{L}_{/\mathbf{poly}}$ ...

---

1. R. Karp and R. Lipton. Turing Machines that Take Advice. STOC, 1980.

$\mathbf{P}_{/\mathbf{poly}}$

Theorem. $\mathbf{P}_{/\mathbf{poly}} = \bigcup_c \mathbf{SIZE}(cn^c)$.

If L is computable by a P-size $\{C_n\}_{n\in\mathbf{N}}$, we use the description of C_n as advice.

Conversely we apply the reduction defined on page 18 to the TM's that take advice, and hard-wire the advice to the circuit.

Karp-Lipton Theorem

It follows from $\mathbf{P} \subseteq \mathbf{P}_{/\mathbf{poly}}$ that $\mathbf{NP} \not\subseteq \mathbf{P}_{/\mathbf{poly}}$ would imply $\mathbf{NP} \neq \mathbf{P}$.

Karp-Lipton Theorem. If $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ then $\mathbf{PH} = \sum_2^p$.

- $\prod_2 \mathtt{SAT}$ is the set of the true QBFs of the form $\forall\_\exists\_.\_$.
- $\sum_2 \mathtt{SAT}$ is the set of the true QBFs of the form $\exists\_\forall\_.\_$.

1. R. Karp and R. Lipton. Turing Machines that Take Advice. STOC, 1980.

The basic idea:

1. Construct some P-time $\mathbb{M}$ such that, for some polynomial q, $\psi \in \prod_2 \mathtt{SAT}$ if and only if $\exists w{\in}\{0,1\}^{q(|\psi|)}.\forall u{\in}\{0,1\}^{q(|\psi|)}.\mathbb{M}(\psi) = 1$.
2. Non-uniformity of circuits is dealt with by $\exists$ quantifier.

If $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, then SAT would be solved by a P-size circuit family $\{C_n\}_{n\in\mathbf{N}}$.

1. Given $\psi = \forall u{\in}\{0,1\}^m.\exists v{\in}\{0,1\}^m.\varphi(u,v)$, there is a P-time machine that upon input u outputs the formula $\varphi(u,v)$.
2. By assumption $\varphi(u,v) \in \mathtt{SAT}$ is decided by a P-size circuit C'.
3. By self reducibility there exists a circuit C of polynomial q size such that $C(u) = v$ whenever $\varphi(u,v)$ is satisfiable.

Conclude that $\forall u{\in}\{0,1\}^m.\exists v{\in}\{0,1\}^m.\varphi(u,v)$ if and only if

$$\exists C{\in}\{0,1\}^{q(|\psi|)}.\forall u{\in}\{0,1\}^m.(C \text{ is a boolean circuit}) \wedge \varphi(u, C(u)).$$

Meyer Theorem. If $\mathbf{EXP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ then $\sum_2 \mathtt{SAT}$ is $\mathbf{EXP}$-hard.

Let $L \in \mathbf{EXP}$ be decided by a $2^{p(n)}$-time one tape TM $\mathbb{M}$. Given input $x \in \{0,1\}^n$, there is an exponential time TM computing the i-th configuration C_i and the head position h_i. There is another exponential time TM computing the i-th configuration C_{i_p} and the head position h_{i_p}, where $i_p = \max j.j < i \wedge h_j = h_i$.

If $\mathbf{EXP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, there are P-size circuit D and D_p, say of size $q(n)$ such that $x \in L$ if and only if

$$\exists D, D_p \in \{0,1\}^{q(n)} \forall i \in \{0,1\}^{p(n)}.\mathbb{T}(x, D(i), D(i-1), D_p(i)) = 1,$$

where the P-time TM $\mathbb{T}$ is defined by the transition function.

---

1. R. Karp and R. Lipton. Turing Machines that Take Advice. STOC, 1980.

$\mathbf{NC}$ and $\mathbf{AC}$

Massively Parallel Computer

Off-the-shelf micro processers linked via interconnection network.

The processors compute in lock-step; and the communication overhead is $O(\log(n))$, where n is the number of processor.

Efficient Parallel Algorithm

A problem has an efficient parallel algorithm if, for each n, it can be solved for inputs of size n using a parallel computer of $n^{O(1)}$ processors in $\log^{O(1)}(n)$ time.

Matrix Multiplication

There is an efficient parallel algorithm for the multiplication of two $(n \times n)$-matrices of numbers using n^3 processors and $\log(n)$ time.

Massively Parallel Computing in Circuit Model

Computations in a circuit can be largely carried out in parallel.

The flatter a circuit is, the more parallel its computation can be.

NC

A language L is in $\mathbf{NC}^d$ if L can be decided by a uniform circuit family $\{C_n\}_{n\in\mathbf{N}}$ of $\mathrm{poly}(n)$ size and of $O(\log^d(n))$ depth.

$$\mathbf{NC} = \bigcup_{d\in\mathbf{N}} \mathbf{NC}^d.$$

Nick's Class was introduced by Nick Pippenger.

AC

$\mathbf{AC}^d$ extends $\mathbf{NC}^d$ by admitting unbounded fan-in's.

$$\mathbf{AC} = \bigcup_{d \in \mathbf{N}} \mathbf{AC}^d.$$

- A P-size fan-in can be simulated by a tree of bounded fan-in's of depth $O(\log(n))$.
- Consequently $\mathbf{NC}^i \subseteq \mathbf{AC}^i \subseteq \mathbf{NC}^{i+1}$.
- Hence

$$\mathbf{AC} = \mathbf{NC}.$$

Gates in a circuit of unbounded fan-in can be arranged in layers in alternating fashion. This is convenient when reasoning about circuits.

Boolean Matrix Multiplication

Suppose A is a boolean $(n \times n)$-matrix. Now

$$(A^2)_{ij} = \bigvee_{k=1}^{n} A_{ik} \wedge A_{kj}.$$

If there are n^3 processors, then the calculation of all $A_{ik} \wedge A_{kj}$'s requires one parallel step, and the calculation of all $(A^2)_{ij}$'s needs $\log(n)$ parallel steps.

- A^2 is in $\mathbf{NC}^1$.
- A^n is in $\mathbf{NC}^2$.

Reachability is in $\mathbf{NC}^2$

Using matrix representation we see immediately that graph reachability is just the boolean matrix multiplication problem.

Parallel Prefix Algorithm

Given $x_1, \ldots, x_n$, we compute x_1, $x_1 + x_2$, $x_1 + x_2 + x_3$, ..., $x_1 + x_2 + \ldots + x_n$.

- In one parallel step we get the following

$$x_1 + x_2,\ x_3 + x_4,\ \ldots,\ x_{n-1} + x_n.$$

- In $2(\log(n) - 1)$ parallel steps we get inductively the sequence

$$x_1 + x_2,\ x_1 + x_2 + x_3 + x_4,\ x_1 + x_2 + x_3 + x_4 + x_5 + x_6,\ \ldots.$$

- We get all the sums in one more parallel step.

The time complexity is $2\log(n)$, discounting the network cost.

Carry Lookahead Addition

Problem: Calculate $\sum_{i=0}^{n} a_i 2^i + \sum_{i=0}^{n} b_i 2^i$ with $a_n = b_n = 0$.

Let x_i be the carry at the i-th position, where $0 \le i \le n-1$. Define

$$\begin{aligned} g_i &= a_i \wedge b_i, \text{ the carry generate bit;} \\ p_i &= a_i \vee b_i, \text{ the carry propagate bit.} \end{aligned}$$

Now $x_i = g_i \vee (p_i \wedge x_{i-1}) = g_i \vee (p_i \wedge g_{i-1}) \vee (p_i \wedge p_{i-1} \wedge x_{i-2})$. Introduce the operation

$$(g', p') \odot (g, p) = (g \vee (p \wedge g'), p \wedge p').$$

Let $(g_0, p_0) = (a_0 \wedge b_0, 0)$. The carries $x_0, \ldots, x_{n-1}$ can be calculated in $2 \log n$ parallel steps as

$$(g_0, p_0),\ (g_0, p_0) \odot (g_1, p_1),\ (g_0, p_0) \odot (g_1, p_1) \odot (g_2, p_2),\ \ldots.$$

Finally $a_1 \oplus b_1 \oplus x_0, \ldots, a_n \oplus b_n \oplus x_{n-1}$ can be calculated in parallel, where $\oplus$ is the exclusive or.

$\mathbf{NC}$ = Problems with Efficient Parallel Algorithm

Theorem. L has efficient parallel algorithms if and only if $L \in \mathbf{NC}$.

Let $L \in \mathbf{NC}$ be decided by a circuit family of $O(n^c)$-size and $O(\log^d(n))$-depth. Assign a processor to each node. The running time of the computer is $O(\log^{d+1}(n))$.

Conversely a processor is replaced by a small circuit, and the interconnection network is replaced by circuit wires.

- The running time of a processor is in polylog.

$\mathbf{NC}$ is the class of problems that have efficient parallel algorithms.

P-Completeness

Does every problem in $\mathbf{P}$ have an efficient parallel algorithm?

We intend to characterize a class of P-time solvable problems that are most unlikely to have any parallel algorithms.

What is the right notion of reduction?

Lemma. An implicitly logspace computable function is efficiently parallel.

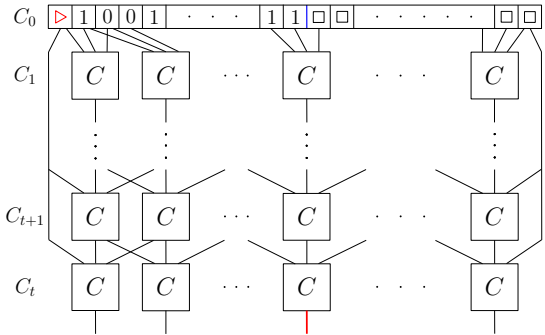
Here is the argument:

1. All output bits can be calculated in parallel.
2. The adjacency matrix of the configuration graph of an implicitly logspace computable function can be constructed by an efficient parallel algorithm.
3. Reachability is in $\mathbf{NC}^2$.

$\mathbf{P}$-Completeness

A language is $\mathbf{P}$-complete if it is in $\mathbf{P}$ and every problem in $\mathbf{P}$ is logspace reducible to it.

Circuit Evaluation is **P**-Complete



`Circuit-Eval` is the language consisting of all pairs $\langle C, x \rangle$ where C is an n-input circuit and $x \in \{0,1\}^n$ is such that $C(x) = 1$.

Monotone Circuit Evaluation is **P**-Complete

We can recycle the reduction defined on the previous page.

- We turn C into C' by pushing negation operations downwards and remove them. Similarly we construct $\overline{C'}$ from $\overline{C}$.
- The monotone circuit is defined in terms of C' and $\overline{C'}$.

Notice that input may be doubled in length.

`Monotone-CKT-SAT` however is a very different story.

The Most "Difficult" Problems in $\mathbf{P}$

Theorem. Suppose L is $\mathbf{P}$-complete. Then $L \in \mathbf{NC}$ iff $\mathbf{P} = \mathbf{NC}$.

---

1. An implicitly logspace computable function is efficiently parallel.
2. We are done by composing two efficient parallel algorithms.

Inside P

Theorem. $\mathbf{NC}^1 \subseteq \mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{NC}^2 \subseteq \ldots \subseteq \mathbf{NC}^i \subseteq \ldots \mathbf{P}$.

---

1. $\mathbf{NL} \subseteq \mathbf{NC}^2$. This is because PATH is in $\mathbf{NC}^2$.
2. $\mathbf{NC}^1 \subseteq \mathbf{L}$. Let $\{C_n\}_{n\in\mathbf{N}}$ accepts $L \in \mathbf{NC}^1$ and let $x \in \{0,1\}^n$.
   - A string of length no more than $\log(n)$ is used to indicate the position of the current gate of C.
   - The initial value of this string is $0^{O(\log n)}$.

Using the depth first strategy, we only have to record the value of the current gate.

All we know is that $\mathbf{AC}^0$ and $\mathbf{NC}^1$ are separated by parity function.

This is Håstad Switching Lemma.

Håstad Switching Lemma

The following strict inclusions have been proved.

$$\mathbf{NC}^0 \subsetneq \mathbf{AC}^0 \subsetneq \mathbf{NC}^1.$$

The parity function $\oplus_n(x_1, \ldots, x_n)$ is in $\mathbf{NC}^1$ but not in $\mathbf{AC}^0$.

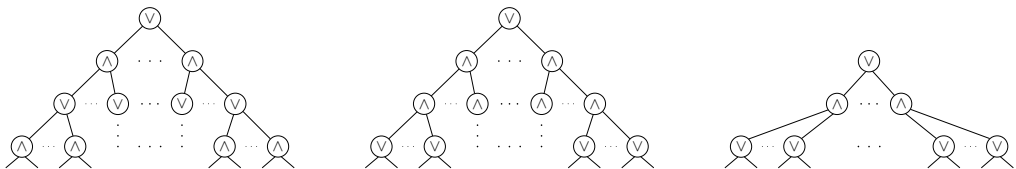
One powerful tool for proving lower bounds for constant height circuits is Håstad's Switching Lemma, which allows one to switch two layers of an alternating circuit, leading to the removal of one layer.

By repeating the switching operation, one ends up with a circuit for computing a constant function.

---

1. M. Furst, J. Saxe and M. Sipser. Parity, circuits, andthe polynomial time hierarchy. Mathematical Systems Theory, 17:13–27, 1984. Prelim version FOCS'81.
2. A. Yao. Separating the polynomial-time hierarchy by oracles. FOCS'85.
3. J. Håstad. Almost optimal lower bounds for small depth circuits. STOC'86.

The key is to transform a t-CNF to an s-DNF.



A d-alternating circuit is a generalization of CNF/DNF.

1. A d-alternating circuit consists of d layers of $\wedge$-gates/$\vee$-gates in an alternating fashion.
2. In the bottom layer an input of a gate is either an input variable or the output of a $\neg$-gate connected to an input variable.

Minterm

A minterm of $x_1, \ldots, x_n$ is a conjunction of n literals such that every variable occurs precisely once.

Suppose α is a binary string of length n. Let x_α be the minterm that is true by the assignment α.

Every n-ary Boolean function f is equivalent to a principal DNF:

$$f(x) = \bigvee_{f(\alpha)=1} x_\alpha.$$

Suppose $X = \{x_1, \ldots, x_n\}$.

We say that $\rho : X \to X \cup \{0, 1\}$ is a restriction of X if $\rho(x) \in \{x, 0, 1\}$ for all $x \in X$.

- x is unrestricted if $\rho(x) = x$,
- $sup(\rho) \stackrel{\text{def}}{=} \{x \mid \rho(x) \neq x\}$ is the support of ρ.

We say that ρ' is a sub-restriction of ρ if $sup(\rho') \subseteq sup(\rho)$ and ρ', ρ coincide on $sup(\rho')$.

If f is defined on X, then f_ρ is the Boolean function $f(\rho(x_1), \ldots, \rho(x_n))$ whose set of input/free variables is $\{x \in X \mid \rho(x) = x\}$.

Minterm of Boolean Function

Suppose $f(x_1, \ldots, x_n) : \{0,1\}^n \to \{0,1\}$, and ρ is a restriction of $\{x_1, \ldots, x_n\}$.

The restirction ρ defines a minterm of f if

1. the value of f_ρ is 1 no matter what the remaining variables are assigned and
2. ρ has no proper sub-restriction that satisfies 1.

Let $\min(f)$ be the maximal size (number of literals/variables) of the minterms of f.

Lemma. If $\min(f) \leq s$, then $f\colon \{0,1\}^n \to \{0,1\}$ can be represented by an s-DNF.

We know that $f(x)$ can be represented by the principal DNF $\bigvee_{f(\alpha)=1} x_\alpha$.

Let x_β be a minterm of f. There must be some α such that $\beta \subseteq \alpha$.

If x_β is false, x_α must be false, and consequently $\bigvee_{f(\alpha')=1}^{\alpha' \neq \alpha} x_{\alpha'}$ and $f(x)$ are equivalent.

If x_β is true, $f(x)$ must be true.

Therefore

$$f(x) \Leftrightarrow x_\beta \vee \bigvee_{f(\alpha')=1}^{\alpha' \neq \alpha} x_{\alpha'}.$$

We are done by induction.

In fact $f(x)$ is equivalent to the $\bigvee$ of all the minterms.

Suppose $0 < n - u \leq u < n$.

A u size random restriction ρ to $X = \{x_1, \ldots, x_n\}$ is defined as follows: Choose a u size random subset of X, and to each chosen variable x assign a value by tossing a coin, that is

$$\rho(x) \quad = \quad \begin{cases} 1, & \text{with probability } 1/2, \\ 0, & \text{with probability } 1/2. \end{cases}$$

Let R^u be the set of all restrictions to u variables. Obviously

$$|R^u| \quad = \quad \binom{n}{u} 2^u. \tag{2}$$

Let $\mathrm{Bad}_f(u, s) = \{\rho \in R^u \mid \min(f_\rho) > s\}$. An element ρ of $\mathrm{Bad}_f(u, s)$ is bad because f_ρ is not equivalent to any s-DNF.

Razborov Lemma. If f is a t-CNF, then $|\mathrm{Bad}_f(u,s)| \leq |R^{u+s}| \cdot (4t)^s$.

Razborov's idea is surprisingly simple:
Construct an encoding function $\mathfrak{e} : \mathrm{Bad}_f(u,s) \to R^{u+s} \times S$ and a decoding function, where $|S| \leq (4t)^s$.

Suppose f is a t-CNF. Fix an order of the clauses of f and an order of the literals.

Suppose $\rho \in \mathrm{Bad}_f(u, s)$, meaning that f_ρ has a minterm τ' of size $s' > s$.

Obtain τ by removing $s' - s$ literals from τ'. These $s' - s$ literals are made unrestricted.

Here are some observations about f_ρ:

1. Some clauses of f disappear from f_ρ.
2. Some literals in some clauses of f disappear from the clauses in f_ρ.
3. No clause of f_ρ is equivalent to 0 because f_ρ has a minterm.
4. $f_{\rho\tau}$ cannot be the constant 1 function because τ is not a minterm for f_ρ; it cannot be the constant 0 function according to 3.

Let C_1 be the first clause of f that is undetermined by ρ but is 1 by $\rho\tau$. [why does it exist?]

---

1. Let τ_1 be τ confined to the t variables in C_1.
   Let $\alpha_1 \in \{0,1\}^t$ be the characteristic function of the support of τ_1, meaning that

$$\alpha_1(x) \quad = \quad \begin{cases} 1, & x \in \mathit{sup}(\tau_1), \\ 0, & x \notin \mathit{sup}(\tau_1). \end{cases}$$

   It follows from the definition of C_1 that some x in C_1 exists such that $\alpha_1(x) = 1$.
2. Let $\overline{\tau_1}$ be the unique restriction defined as follows: the characteristic function of its support is precisely α_1, and the value of C_1 by $\overline{\tau_1}$ is undetermined.

It should be clear that with the help of C_1 and α_1, we can compute $\overline{\tau_1}$.

Replace ρ by $\rho\tau_1$, and τ by $\tau \setminus \tau_1$. Repeat the above construction inductively, we finally get

$$\tau_1, \tau_2, \ldots, \tau_m;\ \overline{\tau_1}, \overline{\tau_2}, \ldots, \overline{\tau_m};\ \alpha_1, \alpha_2, \ldots, \alpha_m,$$

where $m \leq s$ and $\tau = \tau_1\tau_2 \ldots \tau_m$.

We need some extra information to recover $\tau_1, \tau_2, \ldots, \tau_m$ from $\overline{\tau} \stackrel{\text{def}}{=} \overline{\tau_1\tau_2} \ldots \overline{\tau_m}$.

- Let $\beta \in \{0,1\}^s$ be defined by $\beta(x) = 1$ if $\tau(x) = \overline{\tau}(x)$ and by $\beta(x) = 0$ otherwise.

Now the encoding $\mathfrak{e}$ is defined as follows:

$$\mathfrak{e}(\rho) \quad \stackrel{\text{def}}{=} \quad \langle \rho\overline{\tau_1\tau_2} \ldots \overline{\tau_m}, \alpha_1, \alpha_2, \ldots, \alpha_m, \beta \rangle.$$

We can recover ρ from $\mathfrak{e}(\rho)$. Identify the first clause of f that is not set to 1 by $\rho\overline{\tau_1\tau_2} \ldots \overline{\tau_m}$.
Since no $\overline{\tau_i}$ sets C_i to 1, the identified clause must be C_1.
Recover $\overline{\tau_1}$ from C_1 and α_1. Recover τ_1 from $\overline{\tau_1}$ and β. We then get $\rho\tau_1\overline{\tau_2} \ldots \overline{\tau_m}$.
By induction we eventually get $\rho\tau_1\tau_2 \ldots \tau_m$, and ρ as well.

Let's estimate the size of $\mathfrak{e}(\rho)$.

---

1. Since $\rho\overline{\tau_1\tau_2}\ldots\overline{\tau_m} \in R^{u+s}$, the number of such restrictions is bounded by $|R^{u+s}|$.
2. Suppose α_i contains k_i number of 1's. Then $k_i \geq 1$, and $k_1 + \ldots + k_m = s$. The number of strings $(\alpha_1, \alpha_2, \ldots, \alpha_m) \in \{0,1\}^{mt}$ is bounded by

$$\prod_{i\in[m]} \binom{t}{k_i} \leq \prod_{i\in[m]} t^{k_i} = t^s.$$

   The number of $(k_1, \ldots, k_m)$ satisfying $k_1 + \ldots + k_m = s$ is $\binom{s-1}{m-1} \leq 2^s$.
3. The number of $\beta \in \{0,1\}^s$ is bounded by 2^s.

Conclude that $|\mathfrak{e}(\rho)| \leq |R^{u+s}| \cdot (4t)^s$.

Håstad Switching Lemma. Suppose f is a t-CNF of n variables and $p < \frac{1}{9t}$. Let ρ be a random restriction of size $u = (1-p)n$. Then

$$\Pr_{\rho \in_{\mathrm{R}} R^u}[\min(f_\rho) > s] < (9pt)^s.$$

It follows from Razborov Lemma, the equality (2) and the inequality $p < \frac{1}{5t}$ that

$$\frac{\mathrm{Bad}_f(u,s)}{|R^u|} \leq \frac{\binom{n}{u+s}2^{u+s}(4t)^s}{\binom{n}{u}2^u} \leq \left(\frac{n-u}{u}\right)^s (8t)^s = \left(\frac{p}{1-p}\right)^s (8t)^s < \left(\frac{9}{8}\right)^s (8pt)^s = (9pt)^s.$$

The justification of the second inequality is as follows:

$$\binom{n}{u+s} \Big/ \binom{n}{u} = \frac{u!}{(u+s)!} \cdot \frac{(n-u)!}{(n-u-s)!} < \frac{1}{u^s} \cdot (n-u)^s.$$

Remark. The upper bound is independent of n.

Theorem. A $(d+1)$-alternating circuit computing $\oplus_n$ requires $2^{\Omega(n^{1/d})}$ gates.

Corollary. $\oplus_n \notin \mathbf{AC}^0$.

---

1. M. Furst, J. Saxe and M. Sipser. Parity, circuits, andthe polynomial time hierarchy. Mathematical Systems Theory, 17:13–27, 1984. Prelim version FOCS'81.
2. A. Yao. Separating the polynomial-time hierarchy by oracles. FOCS'85.
3. J. Håstad. Almost optimal lower bounds for small depth circuits. STOC'86.

Given a $(d+1)$-height S-size alternating circuit for $\oplus_n$, we start by fixing the number of fan-ins of the gates at the bottom layer. Wolg, assume that the bottom gates are $\vee$-gates.

Let $p = 1/18$, $t = 1$ and $s = 2\log S$.

1. Every bottom $\vee$-gate is a 1-DNF.
2. By Håstad Switching Lemma, a $\vee$-gate is equivalent to an s-CNF with probability at least $1 - (9pt)^s = 1 - \frac{1}{S^2}$.

Conclude that there is a u size restriction that allows one to switch the two bottom layers.

- By merging the two consecutive layers of $\wedge$-gates, we obtain a $(d+1)$-height alternating circuit whose bottom layer $\vee$-gates have fan-in $s = 2\log S$.
- After the switching, the new circuit has at most $pn = \frac{n}{18}$ input variables.

Set $k = t = s = 2\log S$ and $p = \frac{1}{18k}$.

---

1. The two bottom layers contain less than S number of s-CNFs.
2. By the same argument, all s-CNFs can be turned into s-DNFs with probability > 0.
3. The new circuit has $\frac{1}{18k} \cdot \frac{n}{18}$ input variables.
4. By merging the two consecutive layers of $\vee$-gates, we obtain a d-height alternating circuit whose bottom layer $\wedge$-gates have fan-in $s = 2\log S$.

Finally we get a two layer alternating circuit with at most $\frac{1}{(18k)^{d-1}} \cdot \frac{n}{18} = \frac{n}{O((\log S)^{d-1})}$ input variables and every gate in the bottom layer has fan-in $2\log S$.

The two layer alternating circuit is either a $(2\log S)$-CNF or a $(2\log S)$-DNF.

The two layer circuit has constant output by fixing an assignment to $2\log S$ variables.

It follows that by fixing an assignment to

$$n - \frac{n}{O((\log S)^{d-1})} + 2\log S$$

input variables of the $(d+1)$-alternating circuit, it becomes a constant output circuit.

For $\oplus_n$ to be a constant function every input variable must be assigned a value. Thus

$$n \leq n - \frac{n}{O((\log S)^{d-1})} + 2\log S.$$

Consequently $S = 2^{\Omega(n^{1/d})}$.

The output of the n-ary threshold function Th_k^n is 1 if there are at least k inputs are 1.

$Th_k^n \notin \mathbf{AC}^0$. This is because the parity function is the same as

$$\bigvee_{k \in [n]}^{k \text{ is odd}} \left(Th_k^n \wedge \neg (Th_{k+1}^n) \right).$$

$Maj_n \notin \mathbf{AC}^0$.

Books.

1. Heribert Vollmer. Introduction to Circuit Complexity. Springer, 1999.
2. Stasys Jukna. Boolean Function Complexity: Advances and Frontiers. Springer, 2011.
3. Ryan O'Donnell. Analysis of Boolean Functions. CUP, 2014.