Circuit Complexity

Circuit model aims to offer unconditional lower bound results.

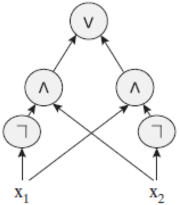
Synopsis

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

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 $S : \mathbf{N} \rightarrow \mathbf{N}$ 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 is such that $L \subseteq \{1^n \mid n \in \mathbf{N}\}$. It is accepted by a linear size boolean circuit.

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, $\mathbf{SIZE}(cS(n)) \neq \mathbf{SIZE}(S(n))$ generally. 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 an 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, we need $(7e^2S)^S \geq \epsilon 2^{2^n}$. 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.

Lower Bound

Frandsen and Miltersen translate a circuit into a stack machine, and bound the length of the machine by a combinatorial argument.

- Inputs are numbered $1, \ldots, n$.
- Boolean operations are numbered $n+1, \ldots, n+s$.
- A program is a finite sequence of push and Boolean operation.
  - *Push* i, where i is a number for an input or an early Boolean operation.
  - A Boolean operation pops two elements and pushes one element.
- After the execution of a program, there is exactly one element left in the stack.

continued on the next two slides.

Input: A circuit $C = \{1, \ldots, n\} \cup \bigcup_{i=n+1}^{n+s} \{g_i = g_{i_1} \; op \; g_{i_2}\}$, where $g_1, \ldots, g_n$ are inputs.

Output: A program P that computes the same boolean function as C.

Procedure: **begin** P is initially empty; $SM(n+s)$ **end**

$SM(i)$.

1. If g_i is an input, add "*Push i*" to P.
2. If g_i has been computed by the j-th Boolean operation in P, add "*Push* $n+j$" to P.
3. If $g_i = g_{i_1} \; op \; g_{i_2}$ has not been computed yet, then
   3.1 $SM(i_1)$.
   3.2 $SM(i_2)$.
   3.3 Add "*op*" to P.

Each push increases stack size by one, and each operation decreases stack size by at most one. Since the machine stops with one value in the stack, P has precisely $s+1$ push operations.

$|P| < (s+1)(c+\log(n+s))$ for some c since the size of the argument of *Push* is $\log(n+s)$.

There are 2^{2^n} Boolean functions. So for some function the optimal circuit has size s such that

$$(s+1)(c+\log(n+s)) \geq 2^n. \tag{2}$$

We show that (2) implies the lower bound inequality $s > \frac{2^n}{n}(1+\frac{log(n)}{n}-\frac{c}{n})$ for large n.

If $s \leq \frac{2^n}{n}(1+\frac{log(n)}{n}-\frac{c}{n})$, then $n+s \leq \frac{2^n}{n}(1+\frac{log(n)}{n}-\frac{c}{n}+\frac{n^2}{2^n})$. By $\log(1+x) < x\log(e)$,

$$\begin{aligned}
2^n &\leq (s+1)(c+\log(n+s)) \\
&\leq \frac{2^n}{n}(1+\frac{\log(n)}{n}-\frac{c}{n}+\frac{n}{2^n})(n-\log n+c+(\frac{\log n}{n}-\frac{c}{n}+\frac{n^2}{2^n})\log e) \\
&\leq \frac{2^n}{n^2}(n+\log(n)-c+\frac{n^2}{2^n})(n-\log n+c+(\frac{\log n}{n}-\frac{c}{n}+\frac{n^2}{2^n})\log e) \\
&\leq (2^n/n^2)(n^2-\log^2(n)+O(\log n)) \ < \ 2^n.
\end{aligned}$$

Upper Bound

(k, s)-Lupanov representation of n-ary boolean function f.

$$f(\mathbf{x}) = \bigvee_{i\in[p]} \bigvee_{v\in\{0,1\}^s} f^r_{i,v}(\mathbf{a}) \wedge f^c_{i,v}(\mathbf{b}).$$

- $\mathbf{x} = \mathbf{ab}$ where $\mathbf{a} \in \{0,1\}^k$ and $\mathbf{b} \in \{0,1\}^{n-k}$; and $p = 2^k/s$.
- $A_1, A_2, \ldots, A_p$ are a partition of $\{0,1\}^k$, $s = |A_1| = \ldots = |A_{p-1}| \geq |A_p| > 0$.

- $f^c_{i,v}(\mathbf{b}) = 1$ if v is the function table of $f(\_, \mathbf{b})$ on A_i.
- $f^r_{i,v}(\mathbf{a}) = 1$ if $\mathbf{a} = A_i(j)$ for $j \in [s]$, necessarily unique, and $v(j) = 1$.

Upper Bound

A Circuit for f can be constructed in three steps.

1. Use $2(2^k + 2^{n-k})$ gates to compute all minterms over **a** and **b**.
2. Use $p2^{n-k}$, respectively $p2^s$, or-gates to compute $f_{i,\mathbf{v}}^{(c)}$, respectively $f_{i,\mathbf{v}}^{(r)}$, for all $i, \mathbf{v}$.
3. Use $2p2^{n-k}$ gates to compute f.

So we may use at most $2(2^k + 2^{n-k}) + p2^{n-k} + 3p2^{n-k}$ gates to compute f.

- Using $p \leq 1 + 2^k/s$, we get the upper bound $2^n/s + 2 \cdot 2^k + 3(2^{n-k} + 2^s + 2^{k+s}/s)$.
- Set $k = 2\log n$ and $s = n - 3\log n$. The upper bound is dominated by $2^n/s$. Now

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

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

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

It follows from the assumption that

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

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

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

Uniform Circuit

A boolean 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 **P**

Theorem. A language is accepted by a uniform circuit family if and only if it is in **P**.

'$\Rightarrow$': Trivial.

'$\Leftarrow$': A detailed proof is given next.

From **P** to Uniform Circuit Family

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

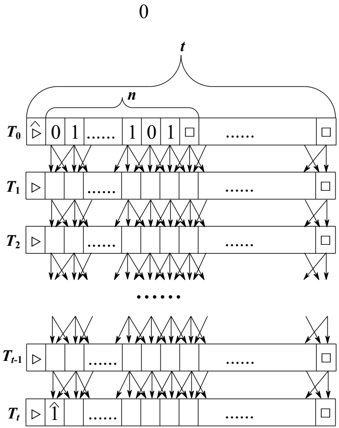
- Given input 1^n one writes down $t = cn^c$ in logspace.
- Let $T_0, \ldots, T_t$ be the configurations.
- Let T_{ij} be the j-th symbol of T_i, where $0 \leq j \leq t+1$.
  - T_{i0} is $\triangleright$, and $T_{i(t+1)}$ is $\square$.
  - Use $\widehat{s}$ to indicate that s is in the cell the reader is pointing to.

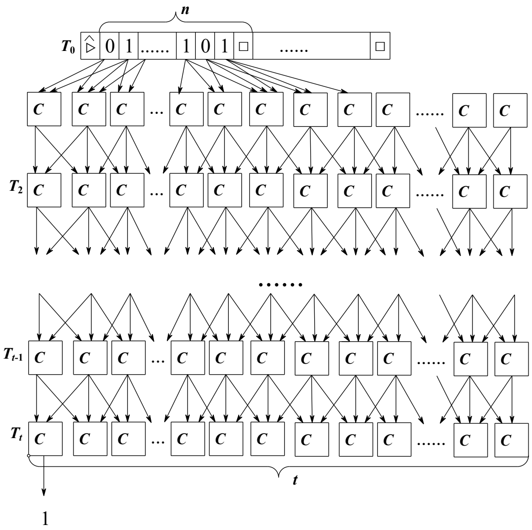
Let C_{ij} for $i, j \in [t]$ be the circuit that computes T_{ij}.

- All C_{ij}'s are copy of a fixed C whose size depends only on $\mathbb{M}$.
- Input to C_{ij} = output of $C_{(i-1)(j-1)}$, $C_{(i-1)j}$, $C_{(i-1)(j+1)}$
  $+$ input $x + \triangleright + \square$.

Using adjacency matrix it is easy to see that the reduction is computable in logspace.

Configuration Circuit





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

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 22.

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

From $\texttt{CKT-SAT}$ to $\texttt{SAT}$

Lemma. $\texttt{CKT-SAT} \leq_L \texttt{SAT}$.

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

Advice to Turing Machine

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.
- $\mathbf{L}_{/\mathbf{poly}}$ $\ldots$

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 $\{C_n\}_{n\in\mathbf{N}}$, we can just use the description of C_n as advice.

Conversely, we can apply the reduction defined on page 22 to the TM's that take advice, and hard-wire the advice to the circuit.

Karp-Lipton Theorem

If $\mathbf{NP} = \mathbf{P}$ then $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$.

Intuitively an NP-complete problem cannot be solved by a P-size circuit family because the number of logical operations required to solve the problem grows faster than any polynomial.

Karp-Lipton Theorem. If $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ then $\prod_2 \mathtt{SAT} \leq_K \sum_2 \mathtt{SAT}$.

- $\prod_2 \mathtt{SAT}$ is the set of the true formulae of the form $\forall\_\exists\_.\_$.
- $\sum_2 \mathtt{SAT}$ is the set of the true formulae 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, $\psi \in \prod_2$ SAT iff $\exists w \in \{0,1\}^{q(|\psi|)}.\forall u \in \{0,1\}^{q(|\psi|)}.\mathbb{M}(\psi) = 1$.
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 $\{C_n\}_{n\in\mathbf{N}}$.

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 $\forall u \in \{0,1\}^m.\exists v \in \{0,1\}^m.\varphi(u,v) = 1$ iff

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

In terms of polynomial hierarchy $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ implies $\mathbf{PH} = \sum_2^p$.

The next theorem says that $\mathbf{EXP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ implies $\mathbf{EXP} = \sum_2^p$.

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

Let $L \in \mathbf{EXP}$ be decided by a $2^{p(n)}$-time oblivious k-tape TM $\mathbb{M}$.

If $\mathbf{EXP} \subseteq \mathbf{P}_{/\mathbf{poly}}$, a P-size circuit C, say of size $q(n)$, computes the i-th snapshot z_i.

It follows that $x \in L$ if and only if

$$\exists C \in \{0,1\}^{q(n)} \forall i, i_1, .., i_k \in \{0,1\}^{p(n)}.\mathbb{T}(x, C(i), C(i_1), .., C(i_k)) = 1,$$

where the P-time TM $\mathbb{T}$ checks the local properties of the snapshot sequence of $\mathbb{M}(x)$.

---

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

NC and **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 logspace 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 $\mathfrak{A}$ is a boolean $(n \times n)$-matrix. Now

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

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

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

Prefix Sums Problem

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

- 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 $1 \leq i \leq n-1$. 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 $x_0, \ldots, x_{n-1}$ are

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

which can be calculated in $O(\log(n))$ parallel steps.

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 $\mathrm{polylog}$.

NC is the class of problems that have efficient parallel algorithms.

P-Completeness

Does every problem in **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 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 $\mathbf{NC}^2$.

P-Completeness

A language is **P**-complete if it is in **P** and every problem in **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 $C(x) = 1$.

The reduction is defined on page 22.

Monotone Circuit Evaluation is **P**-Complete

We can recycle the reduction defined on page 22.

- 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 **P**

Theorem. Suppose L is **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 $\mathbf{P}$

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.

Heribert Vollmer.

- Introduction to Circuit Complexity. Springer, 1999.