Circuits

Note Title 4/19/2013

We're going to change our model of computation to the circuit model from Turing machines. There are several advantages of the circuit model:

- Enables discussion of parallelism.
- More manageable for lower bounds
- Introduces new ideas about uniformity & advice.

Circuit C

- directed acyclic graph
- nodes labeled with $\underbrace{\wedge \quad \vee}_{\text{in-degree } 2}$ $\neg$ (in-degree 1) $\underbrace{x \text{ (variable)} \quad 0/1}_{\text{in-degree } 0}$

(we don't actually need the 0/1 nodes since these can be removed from the ckt w/o changing the outcome)

n = # of input variables

there is one sink (node w/ out-degree 0)

A circuit computes a function $f: \{0,1\}^n \longrightarrow \{0,1\}$.

$C \Leftrightarrow f$.

Complexity of a circuit is measured in terms of the # gates.

We will also be interested in the depth of a circuit which is the longest path from an input to the output.

A formula is a ckt in which the graph is a tree (fan-out of all gates = 1 - except the output).

Every function $f: \{0,1\}^n \rightarrow \{0,1\}$ is computable by a circuit of size $O(n\,2^n)$

Take the AND of n literals that encode each x s.t. $f(x)=1$
then OR the $\leq 2^n$ such terms.

Note: circuits only work for a specific size input.
We're used to TMs which compute $f: \Sigma^* \to \{0,1\}$

Circuit Families: A circuit for each input length:
$C_1 \; C_2 \; C_3 \; \ldots$
$\{C_n\}$ computes $f: \Sigma^* \to \{0,1\}$ iff $\forall x$
$$C_{|x|}(x) = f(x)$$

We say that $\{C_n\}$ decides L where L is the language associated with f.

How do the circuit & TM models compare?

If there is a TM that decides L in time $t(n)$ then there is a circuit family that decides L where the size of C_n is $O(t(n)^2)$.
The proof of this is basically the same CVAL construction
There is a tableau corresponding to the TM computation on variable x input which can then be turned into a ckt.
Tableau (& ckt) has size $O(t(n)^2)$.

What about the other direction: if there is a circuit family that computes L, can it be computed by a TM?

Consider the following example: $C_n = (X_1 \vee \neg X_1)$ if M_n halts
$C_n = (X_1 \wedge \neg X_1)$ if M_n loops

This circuit family decides a unary version of the halting problem. We can potentially encode uncomputable information into the specs of the ckt family.

Solution: uniformity.

We require that the specification of a circuit is easy to compute.

Dfn A circuit family $\{C_n\}$ is logspace uniform iff $\exists$ TM which outputs C_n on input 1^n and runs in logspace.

Theorem: P = languages decidable by a logspace uniform poly-sized circuit families $\{C_n\}$.

P $\Rightarrow$ ckt (we did that above)

Ckt $\Rightarrow$ poly-time TM : M generates $C_{|x|}$
then evaluates $C_{|x|}(x)$
and accepts iff outcome = 1.

Turing Machines with Advice:

A circuit family without the uniformity constraint is called "non-uniform". We can regard non-uniformity as another limited resource like space or time.

→ add read-only advice tape to TM M.

→ advice only depends on n, the size of the input. ($A(n)$ = advice)

→ M decides L w/ advice $A(n)$ iff
$M(x, A(|x|))$ accepts $\Leftrightarrow x \in L$.

Complexity class: $DTIME(t(n))/f(n)$ = the set of languages for which:

$\exists A(n) \quad A: \mathbb{N} \to \Sigma^* \quad |A(n)| \leq f(n)$

there is a TM M which decides L in time $t(n)$ with advice A.

The most important of such classes is $P/poly = \bigcup_k TIME(n^k)/n^k$

<u>Theorem</u>: $L \in P/poly$ iff L decided by a family of (non-uniform) poly-sized ckts.

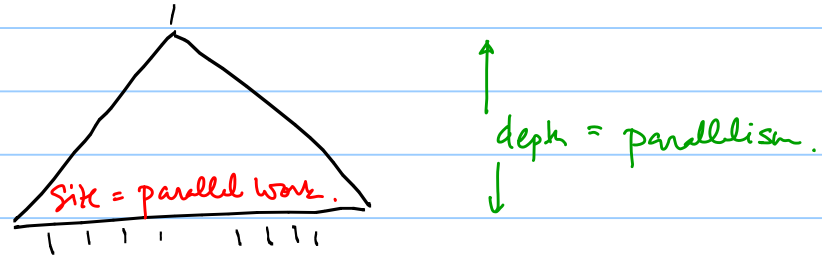
Ckts $\to P/poly$ $\quad A(n)$ is the description of C_n
on input x, TM simulates $C_{|x|}(x)$

=

We believe that $NP \not\subseteq P$.

Generally believed also that $NP \not\subseteq P/poly$
(i.e. SAT does not have poly-sized ckts, even without the uniformity constraint).

Parallelism: Uniform circuits allow for a refinement of polynomial time:

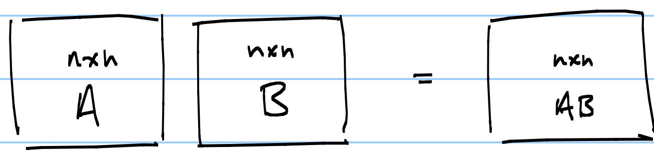


NC ("Nick's Class") Hierarchy of log-space uniform circuits.

NC_k = languages computable by families of log-space uniform circuits w/ poly # gates + depth $O(\log^k n)$.

$NC = \bigcup_{k \geq 1} NC_k$. "Efficiently parallelizable problems."

<u>Example</u>: Matrix Multiplication:



What is the parallel complexity of this problem
Work = poly(n)
parallel time = $\log^k(n)$ – for which k?

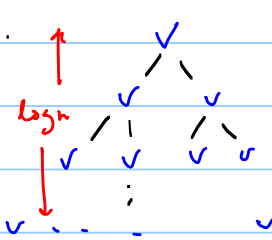
Let's look at this problem where entries are boolean:

$$(AB_{ij}) = \bigvee_k (a_{ik} \wedge b_{kj})$$

To make the output a single bit, input (A, B, i, j)
$\longrightarrow (AB)_{ij}$.

to get all of (AB) can have n^2 matrices in parallel.

Boolean Matrix Multiplication $\subseteq NC_1$
level 1: Compute $(a_{ik} \wedge b_{kj})$ $\forall k$ in parallel.
Compute n-wise OR with a binary tree of height log n.



(for single bit output, this is actually a formula).

$NL \subset NC_2 \Rightarrow$ ST-CONN $\in NC_2$

Let A be the adjacency matrix for G w/ self loops added (1's on diagonal).

$(A^2)_{ij} \longleftrightarrow \exists$ path from i to j of length ≤ 2.

We want: $(A^n)_{st}$ $\quad A \to A^2 \to A^4 \ldots A^{2^{\log n}}$

log n matrix multiplications

$\Rightarrow \log^2 n$ depth.

Now we would like to establish that the notion of P-completeness extends to the class NC as well (i.e. If a P-complete problem is in NC, then $NC = P$).

In order to do this, we need to ensure that logspace reductions can be converted to log-space uniform NC circuits. That is:

If $L \propto_\ell L'$ and $L' \in NC$ then $L \in NC$.

We need to show an NC circuit that computes the reduction. Furthermore, the circuit needs to be from a logspace uniform ckt family.

Consider a TM M that takes in a pair (x, i) s.t. $i \leq |R(x)|$. It accepts iff the i^{th} bit of $R(x)$ is one. M runs in logspace.

The idea is that the circuit will compute ST-CONN on the configuration graph for M's computation on x. Recall that we add a special node t to this graph and connect all accepting configurations to t. The output of the whole circuit will be the (c_{start}, t) entry of $(A_x)^n$ where A_x is the configuration graph for M's computation on input x.

We already saw how to compute A^n for an arbitrary matrix A. Now we need to discuss how to determine A_x.

Consider two nodes in A_x corresponding to two configurations:

$$C = (\underbrace{i}_{\text{location of head on input tape}}, \underbrace{j}_{\text{location of head on work tape.}}, \sigma_1 \cdots \sigma_s, q) \qquad (i', j', \sigma_1' \cdots \sigma_s', q') = C'$$

The entry in A_x corresponding to these two nodes is either $0, 1, x_i$ or $\neg x_i$.

→ The only input bit on which this question (is (c,c') an edge?) can depend is x_i. The rest of this decision is determined by the configurations themselves.

The log-space TM that computes the circuit can determine which of these four possibilities should be the bit $(A_x)_{c,c'}$.

Circuit Lower Bounds:

Major effort in complexity theory: prove lower-bounds for size of circuits that compute problems in NP.

(a super-poly l.b. would actually show $NP \not\subseteq P/poly$).

The best known lower bound is $4.5n$ (w/o uniformity constraint)

(don't even have super-poly l.b.'s for problems in NEXP).

There are, however, l.b.'s for restricted classes of circuits.

Frustrating Fact: almost all functions require huge circuits, just by a counting argument.

Theorem (Shannon) w.p. $\geq 1 - o(1)$ a random function $f: \{0,1\}^n \to \{0,1\}$ requires a circuit of size $\Omega(2^n/n)$.

Proof: $B(n) = 2^{2^n}$ is the # of boolean functions with n inputs.

$C(n,s)$ = the # of circuits w/ n inputs and size s:

$$C(n,s) \leq \Big(\underbrace{(n+3)}_{\text{gate types}}\underbrace{s^2}_{\text{inputs}}\Big)^{s \,\to\, \text{\# gates.}}$$

$$C(n, c\,2^n/n) < \left(2n\, c^2 \cdot 2^{2n}/n^2\right)^{c\,2^n/n}$$

$$< o(1)\, 2^{2c\,2^n} < o(1)\, 2^{2^n} \qquad (\text{if } c < 1/2)$$

The probability that a randomly selected function has a ckt of size $< s = (1/2\; 2^n/n)$ is

$$\leq \frac{C(n,s)}{B(n)} = o(1). \quad //$$

Naturally the best candidates for circuit lower bounds are NP complete problems.

Recent work has focused on special classes of circuits.

Monotone languages: $L \subset \{0,1\}^n$

$x \in L \Rightarrow x' \in L \quad \forall x' \quad \underbrace{x < x'}_{\text{switching 0's to 1's.}}$

Flipping an input bit from a 0 to 1 either changes the output from 0 to 1 or not at all.

Some NP-Complete problems are monotone:
(Hamiltonian cycle, Clique, Set cover)
but not others: SAT, graph coloring...

A Monotone circuit has $\wedge$ + $\vee$ gates, but no $\neg$ gates.
Monotone circuits can only compute monotone functions.

*⇒ Do all poly-time computable monotone functions have poly-size monotone circuits?
(this is true in the non-monotone case).

A monotone ckt. for CLIQUE (n,k)
For each subset $S \subseteq V$ $|S| = k$ $\quad \bigwedge_{(i,j) \in S} x_{ij}$
take the OR of all the S's.
Size = $\binom{n}{k}\binom{k}{2}$
For $k = n^{1/4}$ size $\sim n^{n^{1/4}}$

Theorem (Rasborov '85)
Any Monotone ckt for $\text{Clique}_{n,k}$ w/ $k = n^{1/4}$ has size $\geq 2^{\Omega(n^{1/8})}$

Note that a "yes" answer to (*) would imply $P \neq NP$.

Unfortunately, Razborov also later proved:
Any monotone circuit for MATCHING also requires exponentially large circuits.