Math 262A: Circuit Complexity

Relations between $\mathcal{B}_2$ circuits and Constant Depth Circuits

Lecturer: Sam Buss
Scribe: Radheshyam Balasundaram
Secondary Scribe: Bob Chen

November 20, 2013

Recall that Constant Depth Circuits are made of AND ($\bigwedge$), OR ($\bigvee$) gates of unbounded fan-in and negations ($\neg$) are pushed down to literals and depth of the circuit does not depend on number of inputs. In this lecture we see some relations between $\mathcal{B}_2$ circuits and Constant Depth Circuits.

Theorem 1. *Let C be a $\mathcal{B}_2$ formula of size S and depth d computing an n-ary boolean function f. Let $k < d$ and $m = 2^k$. Then, f has a Constant Depth Circuit of depth $d' = \lceil \frac{d}{k} \rceil + 1$ and size $S' \leq S\frac{2^m}{m-1}$.*

Proof. Split C into subformula with each subtree (corresponding to the subformula) has height k and M gates. For each subtree with M gates, we would have $M+1$ inputs, with $M+1 \leq 2^k$.

Convert each such subtree into either CNF of size $\leq 2^{M+1}$ or DNF of size $\leq 2^{M+1}$ so that we alternate between CNF and DNF at each level. This way, we can collapse the AND gates of CNF with AND gates DNFs previous level and OR gates of CNF with OR gates of DNF in next level (and vice versa for DNFs).

Each replacement of subtree with corresponding DNF/CNF increases size from M to (atmost) 2^{M+1}. Hence, the size of the resulting circuit $\leq S\frac{2^{M+1}}{M}$. The depth of the circuit is changed from k to 2 and we perform the collapsing described in previous paragraph. So, depth of new ciruit $\leq \lceil \frac{d}{k} \rceil + 1$

□

We can use a similar proof for the following theorem which deals with $\mathcal{B}_2$ circuits instead of formulas. But here, we may have to modify the circuit in such a way that gates at any level i will take inputs from level $i+1$. We do this by adding extra identity gates. This increases the size by at most $d \cdot S$.

Theorem 2. *Let C be a $\mathcal{B}_2$ circuit of size S and depth d computing an n-ary boolean function f. Let $k < d$ and $m = 2^k$. Then, f has a Constant Depth Circuit of depth $d' = \lceil \frac{d}{k} \rceil + 1$ and size $S' \leq Sd2^m$.*

Corollary 3. *For any fixed $d \geq 2$, Parity function has depth d circuits of size $2^{O(n^{\frac{1}{d-1}})}$*

For example, parity function has depth 3 circuit of size $2^{O(\sqrt{n})}$.

Proof. We know that Parity has $\{\oplus\}$-formula of size $n-1$ and depth $\log n$. Use Theorem 1 with $k = \left\lceil \frac{\lceil \log n \rceil}{d-1} \right\rceil$. We have,

$$\begin{aligned} d' &= \left\lceil \frac{d}{k} \right\rceil + 1 \\ &= \frac{\log n}{\left\lceil \frac{\lceil \log n \rceil}{d-1} \right\rceil} + 1 \\ &\leq \frac{\log n}{\frac{\log n}{d-1}} + 1 = d \end{aligned}$$

And for size, we get

$$\begin{aligned} S' &= (n-1)\frac{2^{2^k}}{2^k} \\ &= (n-1)\frac{2^{2^{\left\lceil \frac{\lceil \log n \rceil}{d-1} \right\rceil}}}{2^{\left\lceil \frac{\lceil \log n \rceil}{d-1} \right\rceil}} \\ &\leq \frac{n-1}{n^{\frac{1}{d-1}}} 2^{2^{1+\frac{1}{d-1}+\frac{\log n}{d-1}}} \\ &= \frac{n-1}{n^{\frac{1}{d-1}}} 2^{O(n^{\frac{1}{d-1}})} \\ &= 2^{O(n^{\frac{1}{d-1}})} \end{aligned}$$

□

Theorem 4. *Let C be a $\mathcal{B}_2$-circuit of size s and depth d for a boolean function f. Let $l = \lceil \log d \rceil$. Let $r < l$, set $r' = l - r, k = 2^r, m = 2^k$. Then function f has a Σ_3 circuit of size $\leq 2^{\frac{r'}{l}2s} \cdot \frac{r'}{l}2s \cdot 2^m$*

For the given circuit C, look at the underlying Directed Acyclic Graph (DAG) with gates represented by nodes and output from one gate and corresponding input into another gate is represented by a directed edge between nodes corresponding to the gates.

Lemma 5 (Valiant [1977])**.** *It is possible to remove $\frac{r'}{l}2n$ edges such that their removal splits the DAG into pieces in which every path has length $\leq \frac{d}{2^{r'}} \leq \frac{2^l}{\frac{l}{2^r}} = 2^r$*

Proof. We begin by assigning a *level* to each node in DAG so that for an edge going from vertex g to g', $level(g) < level(g')$. For each edge e, we define $i(e)$ as follows:

Look at the binary representation of integers of $level(g)$ and $level(g')$. $i(e) =$ index of the first bit from left to right where $level(g)$ differs from $level(g')$. Clearly, $i(e)$ can take values in $\{1, \ldots, l\}$. Among these l possible values, choose r' least occuring values. That is, let us chooose $i_1, i_2, \ldots, i_{r'}$ amongst values for i and the corresponding edges with these $i(e)$ values. Note that the number of such edges is $\leq \frac{r'}{l}2s$ because expected fraction of edges having any particular i value is $\frac{1}{l}$ and there are a total of at most $2s-1$ edges.

Now we calim that after removing of this set of edges, any path has length $< 2^{l-r'}$. This is true because along any path within any connected component of DAG, if we look at the remaining $l-r'$ bits, their values should be increasing. $\square$

Proof of Theorem 4. We use Lemma 5 and identify the $\frac{r'}{l} \cdot 2s$ edges and use them to build a Σ_3 circuit. Here's a top-down construction of the circuit:

- Outermost OR gate is taken among all possible truth assignments to the $\frac{r'}{l} \cdot 2s$ many removed edges. This OR gate will have (at most) $2^{\frac{r'}{l} \cdot 2s}$ inputs.
- Each input to the OR gate above will correspond to a particular assignment to the removed edges. So, we construct the circuit for this particular assignment with an AND over all edge assignments. This AND gate will take $\frac{r'}{l} \cdot 2s$ inputs. Here, output of some edges will depend on assignment to other edges, which we do in next step.
- Consider any edge which depends on assignment to other edges. From Valiant's lemma above, we know that maximum length of any path is 2^r. So, the circuit to evaluate this edge will have a maximum depth of $2^r = k$. This can have at most $2^k = m$ inputs and hence can be converted to a CNF of size $\leq 2^m$ in which AND gates can be collapsed with AND gates of previous step to get an overall depth 3.
- For any particular assignment for the removed edges, we need to make sure that the final output gate evaluates to TRUE. So we add an AND gate corresponding to the same.

The overall circuit will look like:

$$\bigvee_{\substack{\text{All possible truth assignments to the } \frac{r'}{l} \cdot 2s \text{ removed edges}}}^{2^{\frac{r'}{l} \cdot 2s}} \left[\bigwedge_{\substack{\frac{r'}{l} \cdot 2s \\ \text{removed edges}}} \left(\begin{array}{c}\text{This edge in terms of} \\ \text{other edges written as} \\ \text{a CNF of size} \leq 2^m\end{array} \right) \bigwedge \left(\begin{array}{c}\text{Output Gate should} \\ \text{evaluate to TRUE}\end{array} \right) \right]$$

$\square$

Definition 1. *Let $f = \{f_n\}_{n \geq 0}$ be a family of n-ary boolean functions. f is said to be NC^k if each f_n has a $\mathcal{B}_2$-circuit of size $n^{O(1)}$ and depth $(\log n)^k$*

Definition 2. *Let $f = \{f_n\}_{n \geq 0}$ be a family of n-ary boolean functions. f is said to be AC^k if each f_n has a $\{\bigwedge, \bigvee, \neg\}$-circuit of size $n^{O(1)}$ and depth $(\log n)^k$*

So, AC^0 circuits have polynomial size and constant depth. And, NC^1 circuits have $\mathcal{B}_2$-circuits of polynomial size and depth $O(\log n)$.

Definition 3. *A family f of n-ary boolean functions is said to be NC^1_{lin} if f has a $\mathcal{B}_2$-circuit of size $O(n)$ and depth $O(\log n)$.*

Theorem 6. *If $f \in NC^1_{lin}$, then f has Σ_3 circuits of size $2^{O(\frac{n}{\log \log n})}$*

Proof. Use Theorem 4 with
$l = \log d = \log(c_1 \log n)$ for some constant c_1,
$r' = \log(\frac{2c_1}{\epsilon})$, for some constant ϵ,
$r = l - r'$,
$k = 2^r = \frac{2^l}{2^{r'}} = \epsilon \log n$,
$m = n^\epsilon$,
$s = c_2 n$, for some constant c_2 as size of circuit is linear.
Notice that $\frac{r'}{l} = \theta(\frac{1}{\log \log n})$.
Using these substitutions into the theorem, we can get a Σ_3 circuit with size:

$$\begin{aligned} &\leq 2^{\frac{r'}{l}} 2s \cdot \frac{r'}{l} 2s \cdot 2^m \\ &= 2^{\frac{2c_2 n}{\log \log n}} \cdot \frac{2c_2 n}{\log \log n} \cdot 2^{n^\epsilon} \\ &= 2^{O(\frac{n}{\log \log n})} \end{aligned}$$

□

References

Leslie G Valiant. *Graph-theoretic arguments in low-level complexity.* Springer, 1977.