Polylog Thresholds are computable in AC^0

D.Vamsi Krishna
CS09B006

AC^0

- Consists of all families of circuits of depth O(1) and polynomial size, with unlimited-fanin AND gates and OR gates. (We allow NOT gates only at the inputs).
- Given n bits ,

$$Th_r^n(x_1,x_2,..,x_n) = 1 \text{ if atleast r bits of n bits are 1}$$

$$= 0 \text{ otherwise.}$$

- $Th_r^n(x_1,x_2,..,x_n)$,when r = O(1) is in AC^0.

Log Threshold

- $Th_r^n(x_1,x_2,..,x_n)$, where r = O(log(n)) has no obvious AC^0 circuit.
- Can we some how reduce the problem ?

Idea

- The idea is to hash the input bits which are 1 without collisons on to a set of size t (= $2.\log^2(n)$) .
- Can we can do this (Hashing without collisons) when number of 1's in input bits are atmost log(n)? (Relaxing the above condition).
- This is sufficient in our case as we have to check only whether atleast log(n) input bits are 1.

Idea

- The task that remains is , to find a AC^0 circuit for deciding a log(n) threshold out of t(= $2.\log^2(n)$) bits.
- $Th^t_{\log(n)}(z_1, z_2, ..., z_t)$
  $= \neg$ COMP(log(n),LogltAdd($w_1, w_2, ..., w_{\log(n)}$))
- w_i = Bcount($z_{1+(i-1).2\log(n)}, z_{2+(i-1).2\log(n)}, ..., z_{i.2\log(n)}$)
- Each w_i depends on log(n) bits and hence in AC^0.
- COMP , LogltAdd are in AC^0.

LogItAdd

- Given log(n) , n bit numbers , can we get a AC^0 circuit for generating the sum ?

LogItAdd

- Given log(n) , n bit numbers , can we get a AC^0 circuit for generating the sum ?
- A truth table for log(n) bits would have $2^{\log(n)}$ (= n) rows with loglog(n) output bits.
- Each of loglog(n) bits can be realized by an AC^0 circuit of depth 2 with o(n) gates.

LogItAdd

- Given log(n) , n bit numbers , can we get a AC^0 circuit for generating the sum ?
- A truth table for log(n) bits would have $2^{\log(n)}$ (= n) rows with loglog(n) output bits.
- Each of loglog(n) bits can be realized by an AC^0 circuit of depth 2 with o(n) gates.
- How to add the obtained n loglog(n) bits ?

LogItAdd

- Take the diagonals and lay out them horizontally.

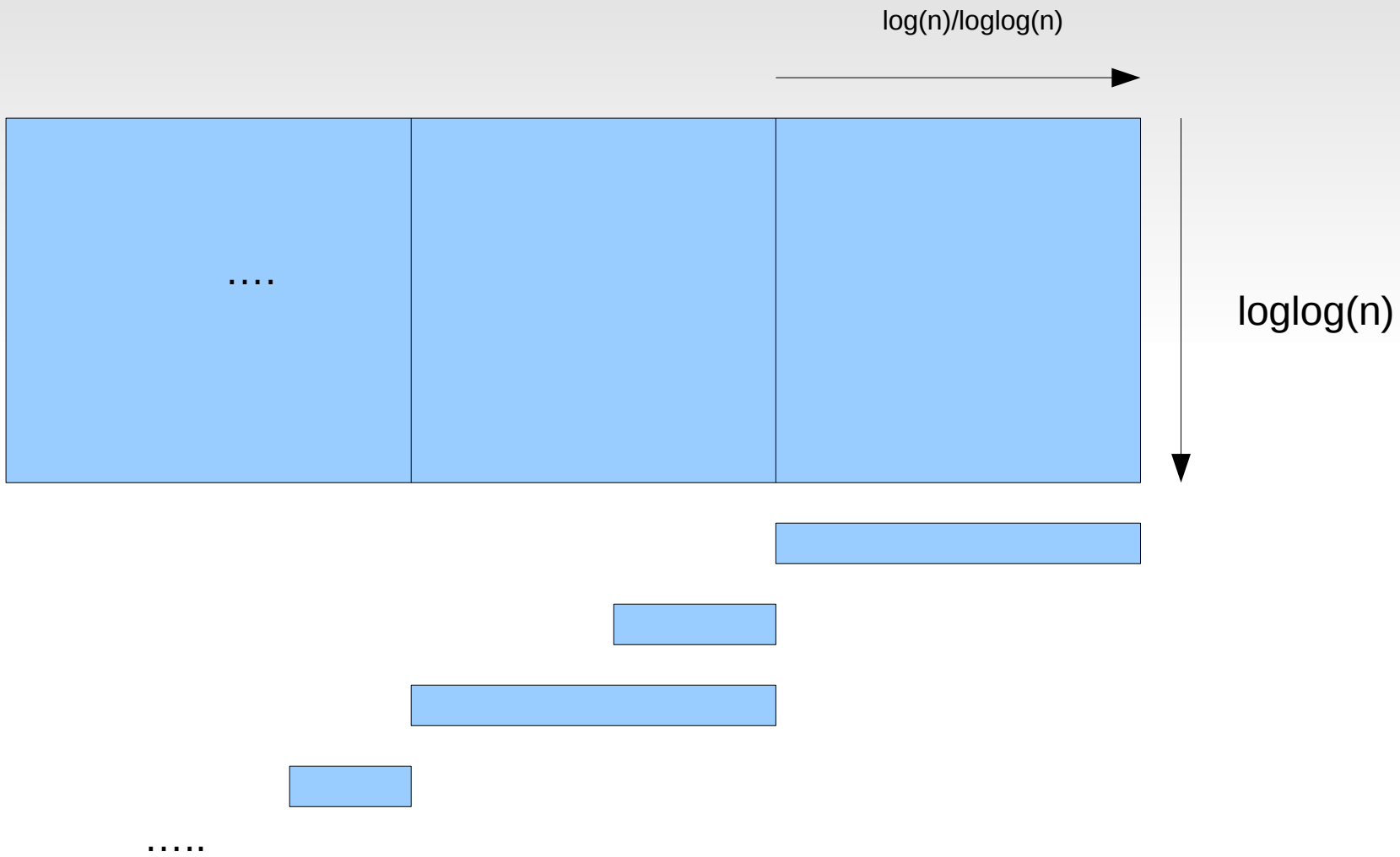
LogItAdd

- Take the diagonals and lay out them horizontally.
- Our problem now reduces to adding loglog(n) , n-bit numbers .

LogItAdd

- Take the diagonals and lay out them horizontally.
- Our problem now reduces to adding loglog(n) , n-bit numbers .
- Recursion !
- Close recursion by brute force.
- Take log(n)/loglog(n) columns .

LogItAdd



LogItAdd

- The positioning is such that the carry for a block fully overlaps with the sum of the next block.
- Now we have to add these two set of numbers which can be done in AC^0.

Log Threshold

- We can get a AC^0 circuit using a hashing family H , which is as follows :
  - Pick any prime number 'p' in the range n,....,2n (such a prime must exist ?).

Log Threshold

- We can get a AC^0 circuit using a hashing family H , which is as follows :
  - Pick any prime number 'p' in the range n,....,2n (such a prime must exist - Bertrand Postulate).

Log Threshold

- We can get a AC^0 circuit using a hashing family H , which is as follows :
  - Pick any prime number 'p' in the range n,....,2n (such a prime must exist - Bertrand Postulate).
  - Then the functions h_α for each $\alpha \in [p-1]$, where $h_\alpha(u)=(\alpha \cdot u \bmod p) \bmod t$, $t \in N$.
  - For Log Threshold $t = 2\log^2 n$ gives us a AC^0 circuit.

Hash Family

- The hash family H ensures us for some input x with atmost log(n) ones , there exists a h_α which doesn't witnesses a collison of 1s.
- Proof:
  - Assume the contrary that for some input x with atmost log(n) ones , every h_α witnesses a collison of 1s. Without the loss of generality , assume that x has exactly log(n) ones.

Proof Continued..

- Let the input bits are indexed by the set [n] $=\{1,2,....,n\}$.
  $W=\{(\alpha,u,v) \mid \alpha \in [p-1], u,v \in S, h_\alpha(u)=h_\alpha(v)\}$.
  $S \subseteq [n]$ be the set of positions where the input bits are 1.
- Clearly W has atleast one triple for each α ,
  $|W| \geq p-1$.
- Consider any pair of distinct elements $u,v \in S$.

Proof Continued..

- For a collision to occur , we should have

  $(\alpha .u \bmod p) = (\alpha . v \bmod p) \pmod t$.

  $\Rightarrow (\alpha .u \bmod p) - (\alpha . v \bmod p) = q.t$ for some $q \in \{- \text{Floor}((p-1)/t),....,0,...,\text{Floor}((p-1)/t)\}$.

- As p is prime there are atmost 2.Floor((p-1)/t) -1 bad α 's for a fixed pair.

- $|W| \leq$ (# of pairs $u,v \in S$) . (# of bad α 's for u,v)

  $\leq {}^{\log(n)}C_2$. 2.Floor((p-1)/t) -1

Proof Continued..

- Using $t = 2 \log^2(n)$ we get

  $p-1 \leq |W| \leq (p-1)/2$

  that is $(p-1) \leq (p-1)/2$.

- A contradiction !
- So our assumption is false proving our claim.

Some Definitions

- For $\alpha \in [p-1]$, $j \in T$, $i \in [n]$

  $B_{\alpha,j,i} = 1$ if $i \in [n]$ is mapped by h_α to j.

  0 otherwise.

- Clearly each of $B_{\alpha,j,i}$ is independent of x and depends only on α,j,i and hence can be hardwired.

- For any input x ,

  $D_{\alpha,j}(x) = 1$ if there is a collison of 1's form input x into position j.

  = 0 other wise .

Some Definitions

$C_\alpha(x) = 1$ if h_α perfectly hashes S

$= 0$ otherwise .

One can see that

- $C_\alpha(x) = \bigwedge_{j=1}^{t} \neg D_{\alpha,j}(x)$.
- $D_{\alpha,j}(x) = Th_2^n(x_1 \wedge B_{\alpha,j,1}, x_2 \wedge B_{\alpha,j,2}, \ldots, x_n \wedge B_{\alpha,j,n})$

- Clearly , these all are in AC^0.

Final Circuit

- $$\left[\bigwedge_{\alpha \in [p-1]} \neg C_\alpha \right] \vee \left[\bigvee_{\alpha \in [p-1]} \left(C_\alpha \wedge \mathrm{Th}^t_{\log(n)}(z_{1,\alpha}, z_{2,\alpha}, \ldots, z_{t,\alpha}) \right) \right].$$
- $$z_{j,\alpha} = \bigvee_{i \in [n]: B_{\alpha,j,i} = 1} x_i = \bigvee_{i \in [n]} x_i \wedge B_{\alpha,j,i}$$

Conclusion

- The idea used here can be extended to prove that polylog thresholds are in AC^0.

Thank You