Computational Complexity Theory, Fall 2010 October 6

Lecture 13: Basic circuit constructions

Lecturer: Kristoffer Arnsfelt Hansen Scribe: Mikael Harkjær Møller

1 Problem definitions

Initially we will define some problems for later usages.

Problem 1 (ADDITION)**.** Given n-bit numbers x and y compute $z = x + y$.

Problem 2 (MULT)**.** Given n-bit numbers x and y compute $z = x \cdot y$.

Problem 3 (ITADD)**.** Given n,n-bit numbers $a^1, a^2, \ldots, a^n$, compute $z = a^1 + a^2 + \ldots + a^n$.

Problem 4 (BCOUNT)**.** Given n bits $a_1, a_2, \ldots, a_n$, compute $z = a_1 + a_2 \ldots + a_n$.

2 Basic circuit constructions

Lets recall our non-uniform hierarchy of circuit classes.

Definition 5 (Circuit classes)**.** The Non-uniform hierarchy of circuit classes we study is as follows,

$$\mathrm{AC}^0 \subseteq \mathrm{NC}^1 \subseteq \mathrm{AC}^1 \subseteq \mathrm{NC}^2 \ldots \subseteq \mathrm{NC} \subseteq \mathrm{P/poly}\,.$$

We can also define uniform log circuit classes, i.e. log-space versions of AC^i and NC^i;

$$\mathrm{U_L\text{-}AC}^i \quad \text{and} \quad \mathrm{U_L\text{-}NC}^i\,.$$

The goal of this lecture is to classify the problems defined above and to also prove the following hierarchy;

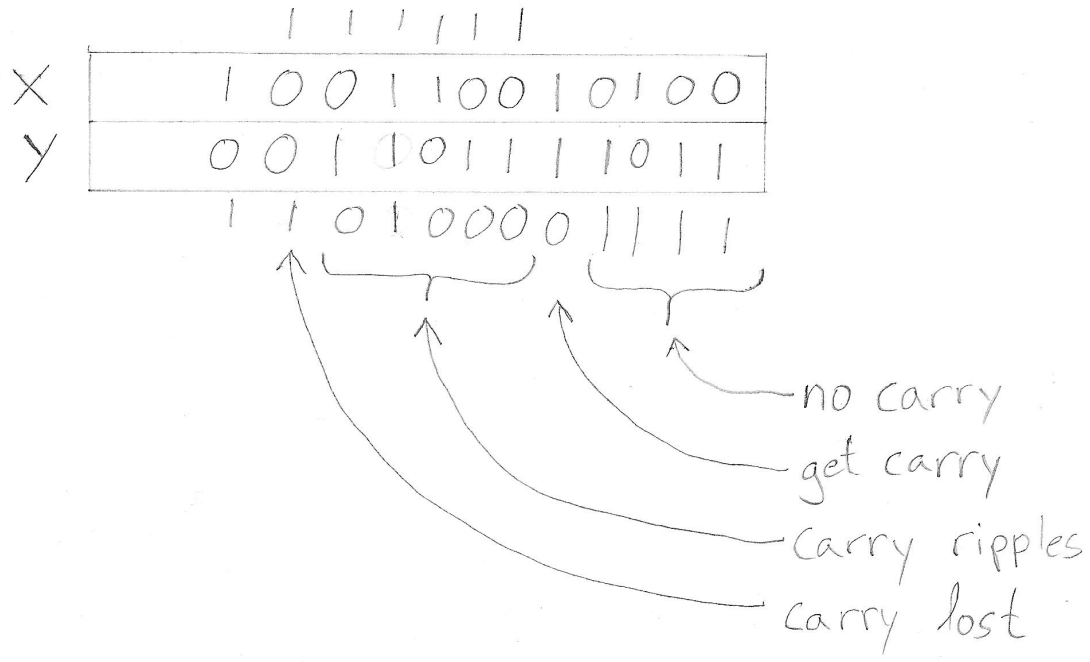
$$\mathrm{U_L\text{-}AC}^0 \subseteq \mathrm{U_L\text{-}NC}^1 \subseteq \mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{U_L\text{-}AC}^1 \subseteq \mathrm{U_L\text{-}NC}^2 \ldots \subseteq \mathrm{U_L\text{-}NC} \subseteq \mathrm{P},$$

more precisely we will prove that $\mathrm{U_L\text{-}NC}^1 \subseteq \mathrm{L}$ and that $\mathrm{NL} \subseteq \mathrm{U_L\text{-}AC}^1$.

Theorem 6. ADDITION $\in \mathrm{AC}^0$.

Proof. Let

$$x = x_{n-1}x_{n-2}\ldots x_0 \quad \text{and} \quad y = y_{n-1}y_{n-2}\ldots y_0.$$



Define the following
$g_i = x_i \wedge y_i$ (generate carry),
$p_i = x_i \vee y_i$ (propagate carry),
$c_i = \bigvee_{j=0}^{i-1} \left(g_j \bigwedge_{k=j+1}^{i-1} (p_k) \right)$ and $c_0 = 0$.
Thus it follows that

$$z_i = x_i \oplus y_i \oplus c_i \qquad \text{for } 0 \leq i \leq n,$$

and $z_n = c_n$. Note that the XOR $\oplus$ can easily be replaced by 2 fan in AND, OR and NOT gates. □

Theorem 7. $\mathrm{U_L\text{-}NC^1} \subseteq L$.

Proof. Let M be a machine that on given input x, evaluates the circuit for the input of length $n = |x|$ on x, generating the circuit on the fly, recursively.

We need to store $O(1)$ bits per level. Since depth of circuit is $O(\log(n)$, the total space is $O(\log(n)$. □

Before we go on to prove the next theorem, lest recall multiplication of Boolean matrices. Let B, C be two boolean $n \times n$, then the product BC is defined by

$$[BC]_{i,j} = \bigvee_{k=1}^{n} (B_{i,k} \wedge C_{k,j}) .$$

We will also need to define the transitive closure of a boolean matrix B. Denoted B^* and defined as

$$B^* = \bigvee_{j \geq 0} B^j.$$

Observe that if B is a $n \times n$ matrix then it holds that

$$B^* = (A \vee I)^{n-1}, \quad \text{,where } I \text{ is the identity matrix.}$$

Theorem 8. $\mathrm{NL} \subseteq \mathrm{U_L\text{-}AC^1}$.

The proof of this theorem is similar to the one for Savitch theorem, but we present a bottom up algorithm.

Proof. We construct circuits for STCON. Given a graph G, states s, t is there a path from s to t. Let D be the $n \times n$ boolean adjacency matrix of G. Note that D^* is then the adjacency if paths in G, i.e. STCON is true if $[D^*]_{s,t} = 1$.

We will compute $(D \vee I)^{n-1}$ by iterated squaring. We can compute $(D \vee I)^{n-1}$ by $O(\log n)$ matrix multiplications. Each matrix multiplication is done in depth 2 (with unbounded fan in). Then STCON must be in $\mathrm{U_L\text{-}AC^1}$. □

Now lets define some more Boolean functions;

Definition 9 (Majority)**.**

$$\mathrm{MAJ}(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} x_i \geq \frac{n}{2}, \\ 0 & \text{oterwise.} \end{cases}$$

Definition 10 (Threshold)**.**

$$\mathrm{T}_k(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} x_i \geq k, \\ 0 & \text{oterwise.} \end{cases}$$

We can now define a now circuit class

Definition 11 ("TC-zero")**.** $\mathrm{TC^0}$ is the class of languages computed by $O(1)$ depth $n^{O(1)}$ size circuits using only MAJ function as gates.

NOTE: $\mathrm{TC^0}$ is a possible theoretical model of Neural Networks.

Lemma 12.

$$T_k(x_1, \ldots, x_n) = \mathrm{MAJ}(x_1, \ldots, x_n, \overbrace{1, 1, \ldots, 1}^{n-2k}) \quad \textit{for } k \leq \frac{n}{2}.$$

Lemma 13.

$$T_k(x_1, \ldots, x_n) = \mathrm{MAJ}(x_1, \ldots, x_n, \overbrace{0, 0, \ldots, 0}^{2k-n}) \quad \textit{for } k > \frac{n}{2}.$$

Theorem 14. $\mathrm{ITADD} \in \mathrm{NC^1}$

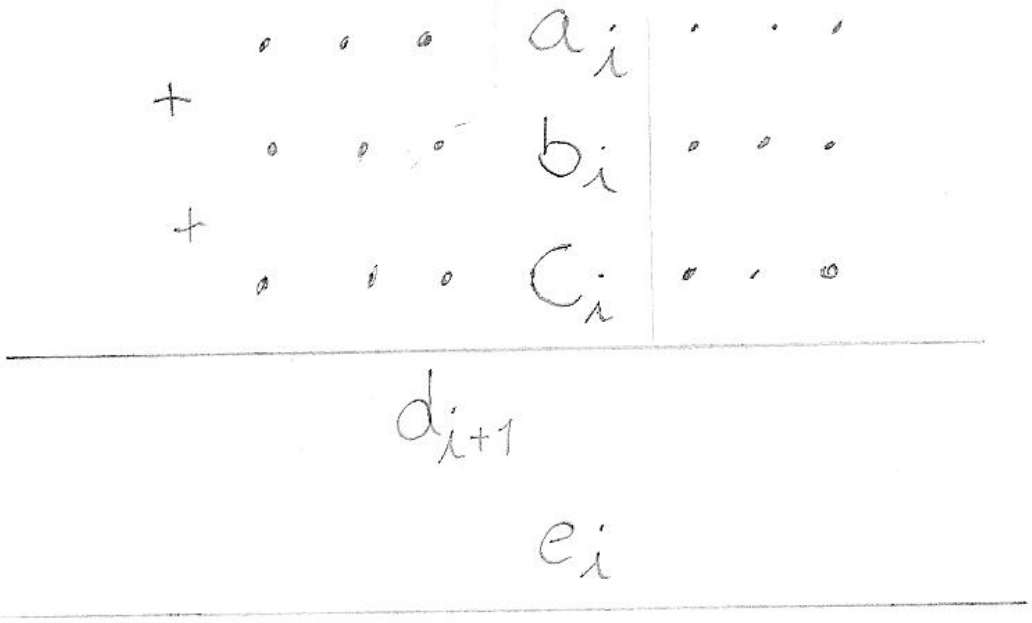
Proof. We will proof this using the "$3 \to 2$" addition trick. Let a, b, c be n-bit numbers

$$\begin{aligned} a = &\quad a_{n_1} \cdots a_1\ a_0 \\ b = &\quad b_{n_1} \cdots b_1\ b_0 \\ a = &\quad c_{n_1} \cdots c_1\ c_0. \end{aligned}$$

Now lets define

$$\begin{aligned} d &= d_n\ d_{n_1} \cdots d_1\ 0 \\ e &= 0 \quad e_{n_1} \cdots e_1\ e_0, \end{aligned}$$

where $d_i = \mathrm{T}_2(a_{i-1}, b_{i-1}, c_{i-1})$ and $e_i = a_i \oplus b_i \oplus c_i$. We can compute these using fanin 2 and constant depth.



By this, using $O(1)$ levels of fan in 2 gates we can reduce adding n - n-bits numbers to adding $\frac{2}{3} \cdot n$ - $(n+1)$ bit numbers. We can do this $O(\log n)$ times to get 2 numbers with $n + O(\log n)$ bits. All this we can do in depth $O(\log n)$.

We can now add these two numbers, this is also possible to do in depth $O(\log n)$ (Here we use the AC^0 circuit we constructed and convert it to a $O(\log n)$ fanin 2 circuit). Thus it follows that ITADD $\in \mathrm{NC}^1$. □

From this theorem we also immediately get the following corollary.

Corollary 15. $\mathrm{TC}^0 \subseteq \mathrm{NC}^1$

Now lets define a new kind of reduction.

Definition 16 (Constant depth reduction)**.** We say that constant depth reducible denoted $A \leq_{\text{cd}} B$ if A can be computed in constant depth and $n^{O(1)}$ size using unbounded fanin AND, OR and oracle gates for B.

Theorem 17. MULT $\leq_{\text{cd}}$ ITADD.

Proof. We need to compute the following

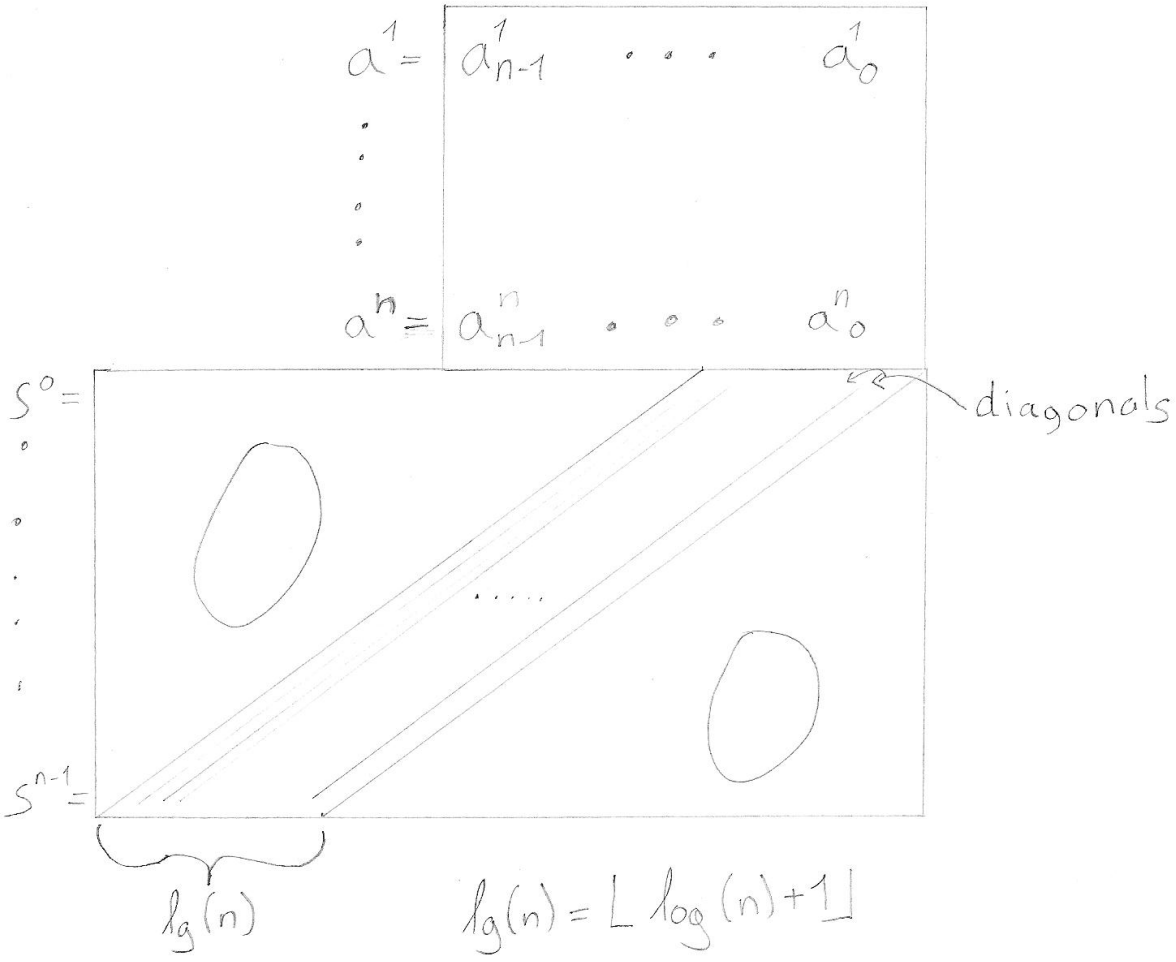
$$(x_{n-1}\ x_{n-2}\ \cdots\ x_0) \cdot (y_{n-1}\ y_{n-2}\ \cdots\ y_0).$$

We do this by adding the numbers $y \cdot 2^i$ for all i where $x_i = 1$.

Thus we get less than n numbers with less than $2n$ bits we need to add. □

Theorem 18. ITADD $\leq_{\text{cd}}$ BCOUNT

Proof. We want to add $a^1, \ldots, a^n$ which are all n-bit numbers.



Compute the column sums $s^i = \sum_{j=1}^{n} a^j_i$. Now consider the $\lg(n)$ diagonals as $n + \lg(n)$ bit numbers. (column sum i is shifted by i positions). Now it holds that $\sum_{i=0}^{n-1} a^i$ =sum of the diagonals.

Thus we have that adding n - n-bit numbers is reducible to adding $\lg(n)$ - $n+\lg(n)$-bit numbers. Repeating this we can again reduce this to adding $\lg\lg((n))$ - $n + \lg(n) + \lg(\lg(n))$-bit numbers. These reductions can be done in constant depth using BCOUNT.

If we continue by repeating this *mentally* until $\lg^k(n) \leq 2$, then we have reduced the problem to adding ≤ 2 numbers each of $n + \lg(n) + \cdots + \lg^{(n)}(n)$ bits, and each of the bits in these last ≤ 2 numbers depends on at most $\lg^{(2)}(n) \cdot \lg^{(3)}(n) \cdots \lg^{(n)}(n)$ bits of the previous $\lg(\lg(n))$-bit numbers. We claim that all these reductions can be done all at once in constant depth and polynomial size.

Notice that

$$\lg^{(2)}(n) \cdot \lg^{(3)}(n) \cdots \lg^{(n)}(n) = O(\log(\log(n))^{\lg^*(n)}) = O(\log(n)).$$

That is *each* output bit depends on only $O(\log n)$ previously computed bits. Thus we can do this computation by $2^{O(\log(n)} = n^{O(1)}$ size circuits. That is, exponential size circuits on $O(\log n)$ inputs are of polynomial size in n. □

Theorem 19. BCOUNT $\in$ TC^0

Proof. Given n bits $a_1, a_2, \ldots, a_n$. Define

$$R_i = \{j \| i\text{'th bit of } j\text{in binary is1}\}$$

Then

$$s_i = \bigvee_{j \in R_i} (\mathrm{T}_j(a_1, \ldots, a_n) \wedge \not\mathrm{T}_j + 1(a_1, \ldots, a_n))$$

□