

Computational Complexity Theory

Lecture 15

November 15, 2006

Let us first recall some definitions.

Definitions.

- \mathbf{AC}^0 := $\{L : L \text{ is decided by a logspace uniform family of Boolean circuits of polynomial size and constant depth}\}$
- $\mathbf{AC}^0[K]$:= $\{L : L \text{ is decided by a logspace uniform family of Boolean } K\text{-oracle circuits of polynomial size and constant depth}\}$
- \mathbf{TC}^0 := $\mathbf{AC}^0[\text{MAJORITY}]$
- \mathbf{NC}^1 := $\{L : L \text{ is decided by a logspace uniform family of Boolean circuits of polynomial size, } \mathcal{O}(\log n) \text{ depth, and fan-in } \leq 2\}$

Remark. The languages in \mathbf{AC}^0 are roughly those that can be decided in constant time by a CRCW PRAM with a polynomial number of processors.

The aim of this lecture is to establish the following.

Theorem.

$$\mathbf{AC}^0[\text{MULTIPLICATION}] = \begin{matrix} \mathbf{NC}^1 \\ \cup \\ \mathbf{TC}^0 \\ \cup \quad \cup \\ \text{ADDITION} \in \mathbf{AC}^0 \not\equiv \text{PARITY} \end{matrix}$$

Proposition. $\text{MAJORITY} \in \mathbf{NC}^1$.

Proof. We compute the sum of the input bits. A simple pairwise addition scheme will *not* work; although we get $\mathcal{O}(\log n)$ levels, we cannot add the numbers in constant depth (because of the bounded fan-in).

Instead, we repeatedly reduce the addition of three numbers to the addition of two numbers. We still get $\mathcal{O}(\log n)$ levels. To perform such a reduction in constant depth, we compute carries, but we do not propagate them, i.e., we use the identity $x + y + z = (x \oplus y \oplus z) + c$, where “ \oplus ” denotes bitwise XOR, $c_{i+1} := \lfloor \frac{1}{2}(x_i + y_i + z_i) \rfloor$, and $c_0 := 0$. To add the final two $\mathcal{O}(\log n)$ -bit numbers and compare the result to $\frac{n}{2}$, we can use a simple look-up table. \square

Corollary. $\mathbf{TC}^0 \subseteq \mathbf{NC}^1$. □

It is unknown whether $\mathbf{TC}^0 = \mathbf{NC}^1$, and even whether $\mathbf{TC}^0 = \mathbf{NP}$.

Open problem. $\mathbf{TC}^0 \stackrel{?}{=} \mathbf{NP}$.

Proposition. $\text{ADDITION} \in \mathbf{AC}^0$.

Proof. Adding two n -bit numbers x and y using the binary elementary algorithm gives an $(n + 1)$ -bit carry string c with $c_0 = 0$. Since $(x + y)_i = x_i \oplus y_i \oplus c_i$, we only have to compute c_i in constant depth.

We consider the possible *carry events*:

$$\begin{aligned} x_i \wedge y_i &\Rightarrow c_{i+1} && (\text{set}) \\ \neg x_i \wedge \neg y_i &\Rightarrow \neg c_{i+1} && (\text{clear}) \\ x_i \oplus y_i &\Rightarrow c_{i+1} = c_i && (\text{propagate}) \end{aligned}$$

and note that the carry is currently set if and only if there has been a set event with no subsequent clear events, i.e.,

$$c_i \Leftrightarrow \exists j < i : (x_j \wedge y_j \wedge \forall k, j < k < i : x_k \vee y_k).$$

The unbounded fan-in allows us to implement the \exists and \forall quantifiers using OR and AND gates, respectively. □

If we define \mathbf{AC}^0 using **DLOGTIME** uniformity, we can characterize it as the class of languages definable in first-order logic with ordering and multiplication [Lee02]. Using logarithmic space as the uniformity condition is very generous, since it gives much more power than the circuits themselves have.

Proposition. $\text{MAJORITY} \in \mathbf{AC}^0[\text{MULTIPLICATION}]$.

Proof. Given a bit string $x \in \{0, 1\}^n$, we create x' by inserting the string $0^{2 \log n}$ between each adjacent pair of bits in x . We create y from 1^n in the same fashion. When we multiply x' and y , the blocks of zeroes kill carries, prevents interference, and ensures that the bit sum of x can be read off from the middle part of the product. □

Proposition. $\text{BITSUM} \in \mathbf{TC}^0$.

Proof. The bit sum of $x \in \{0, 1\}^n$ is $\geq s$ if and only if

$$\begin{aligned} 2s \leq n \text{ and } x1^{n-2s} \in \text{MAJORITY, or} \\ 2s > n \text{ and } x0^{2s-n} \in \text{MAJORITY.} \end{aligned}$$

The bit sum of $x \in \{0, 1\}^n$ is $\leq t$ if and only if the bit sum of $1^n \oplus x$ is $\geq n - t$. □

Corollary. $\text{PARITY} \in \mathbf{TC}^0$. □

Proposition. $\text{MULTIPLICATION} \in \mathbf{TC}^0$.

Proof. Multiplying two n -bit numbers x and y using the binary elementary algorithm reduces the problem to summing an array of height and width $\mathcal{O}(n)$.

We compute the bit sum of each column, and get bit sums $b_0, b_1, \dots, b_{\mathcal{O}(n)}$. Then, we arrange the bit sums in an array, with each b_i shifted i positions. Summing this array yields the result we seek, but, unfortunately, the array is even bigger than the first one. However, it is much more sparse; each column has all its non-zero entries contained in an interval of size $\mathcal{O}(\log n)$. Thus, we can compress each column and obtain an equivalent array of height $\mathcal{O}(\log n)$.

Performing this reduction once more, we obtain an array of width $\mathcal{O}(n)$ and height $\mathcal{O}(\log \log n)$. We split it up into squares, and compute their sums $s_0, s_1, \dots, s_{\mathcal{O}(n/\log \log n)}$, using a look-up table. When we arrange the s_i 's in an array (with appropriate shifts for preserving the sum) each column will have all its non-zero entries contained in an interval of size $\mathcal{O}(1)$. Thus, we then only have to perform a constant number of additions of $\mathcal{O}(n)$ -bit numbers. \square

Theorem. $\text{PARITY} \notin \mathbf{AC}^0$.

This was first proved by Furst, Saxe and Sipser [FSS84], but the proof we outline below is due to Beame [Bea94]. We use Beame's version of Håstad's *switching lemma*.

Lemma. *If $f : \{0, 1\}^n \rightarrow \{0, 1\}$ has a DNF of width $\leq r$ and ρ is a random restriction leaving l variables free, then*

$$\Pr[f_\rho \text{ does not have a decision tree of depth } d] \leq \left(\frac{16lr}{n}\right)^d.$$

We use the switching lemma to show that functions computed by \mathbf{AC}^0 -circuits have a special property that PARITY does not have.

Theorem. *Any $\{f_i\} \in \mathbf{AC}^0$ has the property that, for sufficiently large n , it is possible to make f_n constant by fixing all but $\lfloor \sqrt{n} \rfloor$ of its inputs.*

References

- [FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial hierarchy, *Mathematical Systems Theory* 17:13-27, 1984.
- [Lee02] T. Lee. Arithmetical definability over finite structures, University of Amsterdam Technical Report PP-2002-04, 2002.
- [Bea94] Paul Beame. A switching lemma primer. University of Washington Technical Report UW-CSE-95-07-01, 1994.