Circuit Complexity, Session 2

Lecturer: Sam Buss
Primary scribe: Marco Carmosino
Second scribe: Radheshyam Balasundaram

October 17, 2013

Today, we proved a lower bound on the size for formulas for most Boolean functions, and an upper bound on the size of circuits for Boolean functions.

Theorem 1 (Riordan and Shannon [1942])**.** *Let $\delta < 1$. Most (more than half) n-ary Boolean functions f require formula $\star$-size:*

$$L_{\{\mathcal{B}_2\}} \geq \delta \frac{2^n}{\log(n)}$$

Proof. We will count the number of Boolean formulas of size s over the basis $\mathcal{B}_2$ by encoding these formulas in reverse Polish notation (RPN), and comparing this count to the number of n-ary Boolean functions. Reverse Polish notation is a postfix representation of formulas; with fixed arities (two, in our case) it eliminates ambiguous parses of arithmetic expressions. For example, $(x \vee y) \wedge z$ would be written $xy \vee z\wedge$.

Consider a formula over $\mathcal{B}_2$ with s gates. How many formulas can be formed using inputs $x_1, \ldots, x_n$ and the 16 binary connectives in $\mathcal{B}_2$? First, assume that each gate takes two inputs: then we'll need $2s+1$ symbols total to express this formula, because every gate must have two inputs that are never re-used and may not use itself as input. So there are $s-1$ symbols to denote gates, and s "open" places are left, to be filled by input variables.

$$\binom{2s+1}{s} 16^s n^{s+1} \tag{1}$$

The binomial coefficient counts the number of ways we could assign gates to locations in the $2s+1$ length string of symbols in the RPN representation of the formula. The other two terms count the number of strings over 16 (gates) and n (variables) sized alphabets.

We are of course over-counting. Some of the strings included in our count are not valid Boolean formulas. Asymptotically, this will not matter. Now, we'll bound equation (1).

By Stirling's approximation, $\binom{2s+1}{s} \leq 4^s$. Thus, there are less than $c^s n^{s+1}$ many formulas with s gates for c a constant. Let $s = \delta \frac{2^n}{\log(n)}$ and write $\mathcal{NF}$ for the number of Boolean formulas of size s.

$$\begin{aligned}\mathcal{NF} &\leq c^s n^{s+1} \\ &\leq c^{\delta(2^n/\log(n))} n^{\delta(2^n/\log(n))+1} \\ &\leq 2^{\delta 2^n(\log(c)/\log(n))} 2^{\delta 2^n} n \\ &\leq 2^{\delta 2^n(1+\log(c)/\log(n))} n\end{aligned}$$

For sufficiently large n, we can choose $\epsilon := \delta(1 + \log(c)/\log(n))$ where $\epsilon = (1 + \delta)/2$. Note that $\epsilon < 1$ With this, we get:

$$\mathcal{NF} < 2^{\epsilon 2^n} n << 2^{2^n} \tag{2}$$

Proving that the number of n-ary formulas over $\mathcal{B}_2$ of size s is far fewer than the number of Boolean functions of n arguments (2^{2^n}). Therefore, for most Boolean functions of n arguments, we must exceed size s to compute them with a formula.

□

Our next theorem is an upper bound on the circuit $*$-size for computing any Boolean function of n arguments.

Theorem 2 (Lupanov [1958])**.** *Let $\epsilon > 0$. For all n sufficiently large, n-ary Boolean functions have circuit $\star$-size:*

$$C^*_{\{\wedge,\vee,\neg\}} \leq \frac{2^n}{n}(1 + \epsilon) \tag{3}$$

Proof. We give the (k, s)-Lupanov representation of $f : \{0, 1\}^n \to \{0, 1\}$. This representation works by cleverly de-duplicating the truth table of f and storing it efficiently in look-up circuits. Let k, s be fixed values to be specified later. The first step is to break up the arguments of f into two groups, and use them to index a table of the values of f:

$$f(\bar{x}) = f(x_1, \ldots, x_k, x_{k+1}, \ldots, x_n)$$

The first k bits of x index the rows of the table, and the rest of x indexes the columns of the table. Each entry of the table is the value $f(\bar{x})$, where $\bar{x} = x_1, \ldots, x_k, x_{k+1}, \ldots, x_n$, the concatenated values of the indices of the table.

		Different values for $x_{k+1}, x_{k+2}, \ldots, x_n$
		$(000\ldots00)$ $(000\ldots01)$ $\ldots$ $(111\ldots11)$
Different values for $x_1, x_2, \ldots, x_k$	$00\ldots0$ $00\ldots1$ $\vdots$	
	$\vdots$	Block of s rows - A_i
	$\vdots$ $\vdots$	
	$\vdots$	Block of s rows - A_j
	$\vdots$ $11\ldots1$	

Now we break the table into blocks of s rows. Denote the number of such blocks by p. Though it is necessary to round up to obtain exact values for p and other quantities, we will drop rounding from this proof for clarity, and asymptotically it will be inconsequential.

$$p = \frac{\#\text{rows}}{sizeofblock} = \left\lceil \frac{2^k}{s} \right\rceil \tag{4}$$

Name each block A_i, for $i \in [p]$. We will use the structure of this table to decompose f. Then, we will implement each part of the decomposition and keep track of the number of gates used for each of these steps.

Let $f_i(\bar{x})$ be the function f restricted to block A_i. Formally:

$$f_i(\bar{x}) := \begin{cases} f(\bar{x}) & \text{if } x_1, \ldots, x_k \in A_i \\ 0 & \text{otherwise} \end{cases}$$

So we can write f as:

$$f(\bar{x}) = \bigvee_{i=1}^{p} f_i(\bar{x}) \tag{5}$$

The big OR **costs** $(p-1)$ $\vee$ gates arranged in a binary tree.

Now we fix a value $i \in [p]$ and examine a particular block A_i. Observe that A_i has at most 2^s distinct columns. By de-duplicating these columns and storing them in efficient decoder circuits, we will achieve size-savings over the naive solution. To begin, restrict f further, according to the set of unique columns in A_i. We abuse notation slightly, writing A_i to mean the set of unique columns of block A_i in our table.

$$f_{i,\bar{b}}(\bar{x}) := \begin{cases} f_i(\bar{x}) & \text{if } x_{k+1}, \ldots, x_n \text{ names a column of } A_i \text{ containing } \bar{b} \\ 0 & \text{otherwise} \end{cases}$$

Observe that we can now re-express f_i:

$$f_i(\bar{x}) = \bigvee_{\bar{b} \in A_i} f_{i,\bar{b}}(\bar{x}) \tag{6}$$

The **cost** of the big OR per f_i is $(2^s - 1)$, and so total cost for all f_i is $p2^s$.

Next we fix $\bar{b} \in A_i$ and decompose $f_{i,\bar{b}}$. We will have two functions: one to tell us if a particular input maps to one of the $\bar{b}$ columns, and one to tell us the values in $\bar{b}$.

$$f^{(\text{col})}_{i,\bar{b}}(x_{k+1}, \ldots, x_n) := \begin{cases} 1 & \text{if } x_{k+1}, \ldots, x_n \text{ names a column of } A_i \text{ equal to } \bar{b} \\ 0 & \text{otherwise} \end{cases}$$

$$f^{(\text{row})}_{i,\bar{b}}(x_1, \ldots, x_k) := \begin{cases} 1 & \text{if } \bar{b} \text{ has a 1 in row } x_{k+1}, \ldots, x_n \\ 0 & \text{otherwise} \end{cases}$$

The function $f^{(\text{col})}_{i,\bar{b}}$ is key to the efficiency of the (k, s)-Lupanov representation. Consider that the table we construct has exactly 2^{n-k} columns. As noted above, each A_i has at most 2^s unique columns $\bar{b}$. For suitable values of (k, s), 2^s is sufficiently smaller than 2^{n-k} such that we will use less hardware by storing only the *unique* $\bar{b}$ for each A_i.

Using these functions, we can rewrite f entirely:

$$f(\bar{x}) = \bigvee_{i=1}^{p} \bigvee_{\bar{b} \in A_i} f^{(\text{row})}_{i,\bar{b}}(x_1, \ldots, x_k) \wedge f^{(\text{col})}_{i,\bar{b}}(x_{k+1}, \ldots, x_n) \tag{7}$$

To compute $f^{(\text{row})}_{i,\bar{b}}$, we write it in DNF. By definition, $f^{(\text{row})}_{i,\bar{b}}$ has at most s 1-values (corresponding to the 1's in a column $\bar{b}$ of A_i), meaning that the total **cost** per $f^{(\text{row})}_{i,\bar{b}}$ is at most $(s-1)(k-1) < sk$ gates. Thus, repeating this construction for all $(i, \bar{b})$ pairs costs $p2^s sk$ gates total.

We would also like to write $f^{(\text{col})}_{i,\bar{b}}$ in DNF. Fixing i and $\bar{b}$, the 1 values of $f^{(\text{col})}_{i,\bar{b}}$ occur only when $x_{k+1}, \ldots, x_n$ names a column containing $\bar{b}$ in block A_i. Observe that by definition these values are disjoint for different $\bar{b}$. Therefore, with 2^{n-k} $\vee$ gates, we can compute each $f_{i,\bar{b}}$ from the full minterms over $x_{k+1}, \ldots, x_n$; we split them into separate binary trees of $\vee$s for each $\bar{b} \in A_i$. Since we are building a circuit we can re-use these full minterms, so we must add the **cost** of computing all full minterms exactly once. At this point, we require a lemma (to be proved later):

Lemma 3 (Efficient Computation of Full Minterms)**.** *There is a circuit computing all the full minterms on $x_1, \ldots, x_m$ of size $2^m(1 + o(1))$*

Gathering together the costs of each individual part of the circuit we have constructed, we obtain the following expression for the size S of the (k, s)-Lupanov representation of f:

$$
\begin{aligned}
S &\leq (p+1) && \text{disjunct over } i && (8)\\
&+ 2p2^s && \text{disjunct over } \bar{b} && (9)\\
&+ p2^s sk && \text{construct each } f^{(\mathrm{row})}_{i,\bar{b}} && (10)\\
&+ p2^{n-k} && \text{disjunct over each } f^{(\mathrm{col})}_{i,\bar{b}} && (11)\\
&+ 2^{n-k}(1+o(1)) && \text{all minterms over } x_{k+1},\ldots,x_n && (12)
\end{aligned}
$$

Setting $s = \lceil n - 5\log(n) \rceil$ and $k = 3\log(n)$, we obtain the desired upper bound on S:

$$
\begin{aligned}
S &\leq (p+1) + 2p2^s + p2^s sk + p2^{n-k} + 2^{n-k}(1+o(1))\\
&\leq \left(\frac{2^k}{s}+1\right) + 2\frac{2^k}{s}2^s + \frac{2^k}{s}2^s sk + \frac{2^k}{s}2^{n-k} + 2^{n-k}(1+o(1))\\
&\leq 2^{k+s}k(1+o(1)) + \frac{2^n}{s}(1+o(1))\\
&\leq 2^{3\log(n)+(n-5\log(n))}3\log(n)(1+o(1)) + \frac{2^n}{n-5\log(n)}(1+o(1))\\
&\leq \frac{2^n}{n^2}3\log(n)(1+o(1)) + \frac{2^n}{n-5\log(n)}(1+o(1))\\
&\leq \frac{2^n}{n}(1+o(1))
\end{aligned}
$$

□

What goes wrong if we attempt a (k,s)-Lupanov representation of formulas? Only two things: first, Lemma 3 gives a *circuit* for computing all minterms, and in the definition of $f^{(\mathrm{col})}_{i,\bar{b}}$ we re-use our minterm circuits – the rest of the Lupanov representation is actually a formula. Patching these issues, we obtain:

Theorem 4 (Lupanov [1965])**.** *Let $\epsilon > 0$. For all n sufficiently large, n-ary Boolean functions have formula $\star$-size:*

$$L^*_{\{\wedge,\vee,\neg\}} \leq \frac{2^n}{\log(n)}(1+\epsilon) \tag{13}$$

Proof. We shall prove only that

$$L^*_{\{\wedge,\vee,\neg\}} \leq \frac{2^{n+1}}{\log(n)}(1+\epsilon)$$

See the citations below for proof of the theorem as stated.

As above, fix $i \in [p]$. We begin by implementing $f^{(\mathrm{col})}_{i,\bar{b}}$ with a formula. Consider how many times $f^{(\mathrm{col})}_{i,\bar{b}}$ evaluates to 1. Specifically, let $q_i(t)$ be the number of $\bar{b}$ such that $f^{(\mathrm{col})}_{i,\bar{b}}$ has t-many 1 outputs. Observe:

$$\sum_t t q_i(t) = 2^{n-k} = \text{total \# of columns} \tag{14}$$

and

$$\sum_t q_i(t) \leq 2^s \tag{15}$$

because (recall) there are at most 2^s unique columns per block A_i.

Now, we employ a lemma that gives us a cheap way to compute functions with a bounded number of 1 values using formulas:

Lemma 5 (Finkov 57)**.** *If g is m-ary and has t-many 1 values for $t \geq (\log(m))^2)$ and m sufficiently large, then:*

$$L^*_{\{\wedge,\vee,\neg\}}(g) \leq \frac{2tm}{\log(m)}(1 + o(1)) \tag{16}$$

Homework: Prove the above lemma.

***Homework:** Improve the above lemma to $\frac{mt}{\log(m)}(1 + o(1))$ (unknown if this is possible).

Define $L^*(t, m)$ to be the maximum size required to compute an m-ary function with t 1 values. We want to bound the summation of this quantity over all relevant values t:

$$\sum_t L^*(t,m)q_i(t) \leq \sum_{t<(\log(m))^2} L^*(t,m)q_i(t) + \sum_{t>(\log(m))^2} L^*(t,m)q_i(t) \tag{17}$$

For the first term on the left, because of the small number of 1-values t, we can simply write the formula in DNF. For the second term on the left, we apply Lemma 5. Substituting in these bounds and $m = n - k$ (the arity of $f^{(\text{col})}_{i,\bar{b}}$) and bounding $\sum q_i(t)$ above by 2^s, we obtain:

$$\begin{aligned}\sum_t L^*(t,m)q_i(t) &\leq (\log(m)^2)m2^s + \frac{m2}{\log(m)}2^{n-k}(1 + o(1)) \\ &\leq (\log(n-k))^2(n-k)2^s + \frac{2(n-k)}{\log(n-k)}2^{n-k}(1 + o(1))\end{aligned}$$

Now, we substitute in the bound above for 8 in the summation over costs for constructing f, which reflects our reimplementation of $f^{(\text{col})}_{i,\bar{b}}$ using a formula. After substituting in $k = 2\log(n)$ and $s = n - 3\log(n)$ and simplifying, we obtain:

$$\begin{aligned}L^*_{\{\wedge,\vee,\neg\}}(f) &\leq o\left(\frac{2^n}{\log(n)}\right) + \frac{2^{2\log(n)}}{s}\frac{2(n - 2\log(n))}{\log(n)}2^{n-2\log(n)} \\ &\leq \frac{2^{n+1}}{\log(n)}(1 + o(1)) + 1\end{aligned}$$

□

Finally, we will prove Lemma 3, which was necessary for the (k, s)-Lupanov representation of f with a circuit.

Proof. We operate recursively, forming all full minterms over $x_1, \ldots, x_{\lceil m/2 \rceil}$ on the left and $x_{\lfloor m/2 \rfloor}, \ldots, x_m$ on the right. Denote by S_m the size of the circuit computing all full minterms on m inputs. Using the strategy above, we have:

$$\begin{aligned}
S_m &= 2^m + S_{\lceil \frac{m}{2} \rceil} + S_{\lfloor \frac{m}{2} \rfloor} \\
&\leq 2^m 2 S_{\lceil \frac{m}{2} \rceil} \\
&\leq 2^m 2 (2^{\lceil \frac{m}{2} \rceil} + 2 S_{\lceil \frac{m}{4} \rceil}) \\
&\leq 2^m 2 (2^{\lceil \frac{m}{2} \rceil} + 2 S_{\lceil \frac{m}{4} \rceil}) \\
&\leq 2^m + 2^{\lceil \frac{m}{2} \rceil + 2} \text{ (sum descending powers of 2)} \\
&\leq 2^m
\end{aligned}$$

□

References

Lupanov. On a method of circuit synthesis. *Izvestia VUZ Radiofizika*, 1(1):120–140, 1958. Upper bounds on sizes of circuits for all functions (in Russian).

Lupanov. On the realization of functions of logical algebra by formulae of finite classes (formulae of limited depth). *Problems of Cybernetics*, 6(6):1–14, 1965. Upper bounds on sizes of formulas for all functions (English translation of Problemy Kibernetiki 6 (1961) 5-14.).

Riordan and Shannon. The number of two-terminal series-parallel networks. *J. of Mathematics and Physics*, 21(21):83–93, 1942. Lower bounds on size of formulas for most functions.

Shannon. The synthesis of two-terminal switching circuits. *Bell System Technical Journal*, 28(28):59–98, 1949. Lower bounds on size of circuits for most functions.