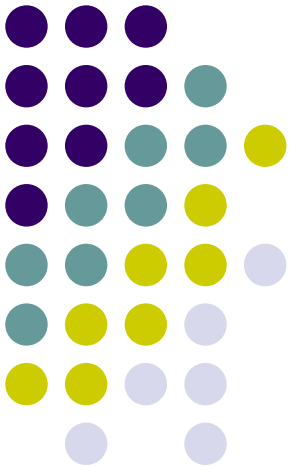
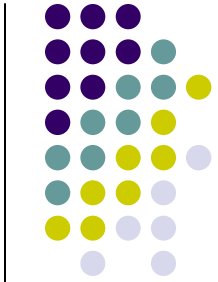
CS256
Applied Theory of Computation

Circuit Complexity III

John E Savage

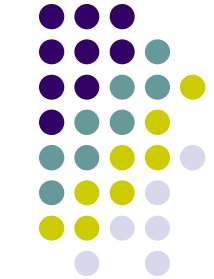


Overview

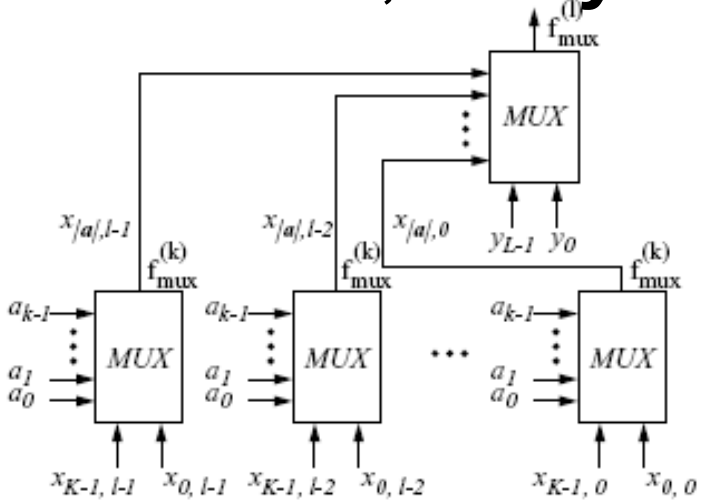


- Indirect storage access function
- Neciporuk's formula size lower bound
- Krapchenko's formula size lower bound
- Monotone function
- The path elimination method

Indirect Storage Access Function



- Let K = 2^k and L = 2^l. Consider the *indirect storage access function* $f_{ISA}^{(k,l)}(\boldsymbol{a},\mathbf{x}_{K-1}, \ldots, \mathbf{x}_0,\mathbf{y}) = y_b$, b =$|\mathbf{x}_{|\boldsymbol{a}|}|$. $\boldsymbol{a}$ is a k-bit address vector, $\mathbf{x}_j$ is an l-bit address vector, and $\mathbf{y}$ is an L-bit data vector.



Indirect Storage Access Function

- Problem 9.24 gives formula for $f_{\text{mux}}^{(n)}$ of size $3\times 2^n-2$ that uses $2(2^n-1)$ instances of address variables. Applying this to the above formula for $f_{ISA}^{(k,l)}$ gives a formula that has size $O(n^2 / \log n)$.

Neciporuk's Formula Size Lower Bound

- Neciporuk's lower bound method provides a lower bound to formula size of the same order as the upper bound for this function.
- Given $f : B^n \rightarrow B$, partition its n variables $\mathbf{X}$ into p disjoint sets $\mathbf{X}_1$, $\mathbf{X}_2$, ..., $\mathbf{X}_p$. Let $r_j(f)$ be the no. of different subfunctions of f over $\mathbf{X}_j$ when fixing the values of variables in $\mathbf{X}$-$\mathbf{X}_j$.
- We derive a lower bound on the formula size of f in terms of $r_i(f)$ for $1 \leq i \leq p$. ($r_i(f)$'s depend on the partition used. Choose the partition wisely!)

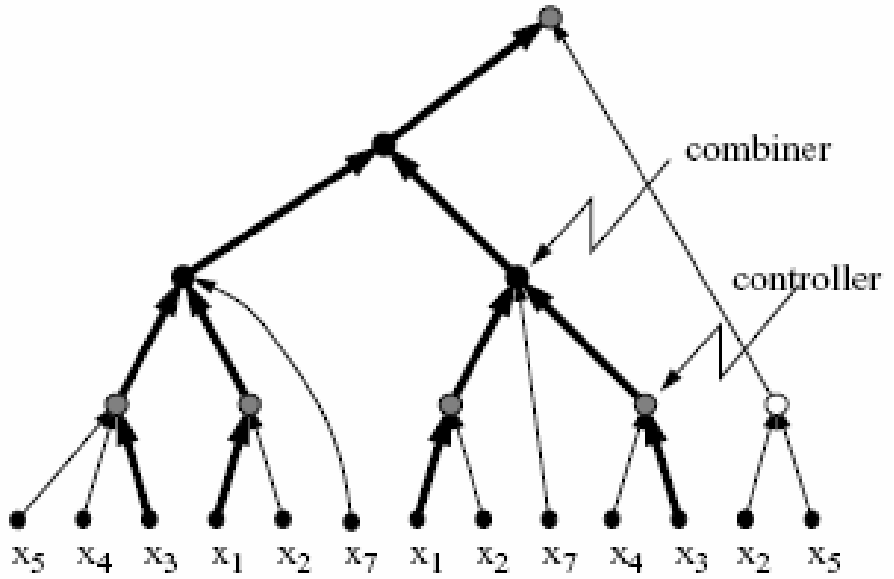
Neciporuk's Formula Size Lower Bound

Theorem For every complete basis Ω there exists constant $c_\Omega = 1/(d+2)$ such that for every $f : B^n \rightarrow B$ its formula size satisfies

$$L_\Omega(f) \geq c_\Omega \sum_{j=1}^{p} log_2\, r_i(f)$$

Neciporuk's Formula Size Lower Bound

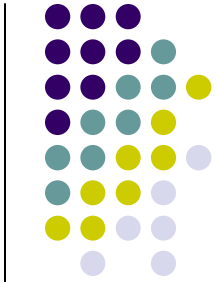
Proof Let T be a minimal fan-out 1 circuit for *f* (a tree). Let n_j be number of instances of vars in $\mathbf{X}_j$ used in T. L(*f*) $=n_1+n_2+ \ldots + n_p$.



Neciporuk's Formula Size Lower Bound

- Let T_j consist of paths from variables in $\mathbf{X}_j$ to root of T. Vertices with only one path entering them are called **controller vertices**. Others are **combiner vertices**. Assigning values to variables in $\mathbf{X}$-$\mathbf{X}_j$, each controller computes one of 4 functions on a variable x, which we represent by (a AND x) XOR b for constants a and b. (If a=0, its b, if a=1, its x XOR b.) Clearly each chain of controllers can be compressed to one controller.

Neciporuk's Formula Size Lower Bound

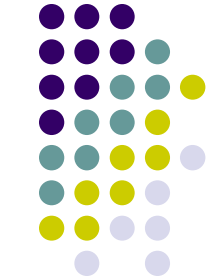


Each combiner has at least two inputs on paths from variables in $\mathbf{X}_j$. Thus, if a gate (vertex) has maximum fan-in d, at most d-2 combiner inputs are constants determined by the values of variables in $\mathbf{X}$-$\mathbf{X}_j$. By earlier lemma since T_j on $\mathbf{X}_j$ has n_j leaves, its number of vertices with fan-in ≥ 2 (combiners) is at most n_j -1. Also, T_j has ≤ 2(n_j -1) edges. Since T_j has at most one controller per edge plus one at output, it has ≤ 2n_j-1 controllers.

Neciporuk's Formula Size Lower Bound

The number of functions computed by a combiner is at most 2^{d-2} because it has at most *d-2* constant inputs. At most 4 functions are computed by a controller. Let $m = n_j$. Thus, the max number of subfunctions computed by T_j is at most $(2^{(d-2)(m-1)}4^{(2m-1)}) \leq 2^{(d+2)m}$. Thus, $(d+2)n_j \geq (\log_2 r_j(f))$ and the theorem follows. ♥

Formula Size Lower Bound



Lemma Let L = $2^l = n$ and $k = \lceil log_2 (n/l) \rceil$. Then, $L_\Omega(f_{ISA}^{(k,l)}) = \Omega(n^2 / log\ n)$.

Proof Recall that $f_{ISA}^{(k,l)}(\boldsymbol{a},\mathbf{x}_{K-1}, ..., \mathbf{x}_0,\mathbf{y}) = y_b$, $b = |\mathbf{x}_{|\boldsymbol{a}|}|$. where $\boldsymbol{a}$ is a k-bit address vector, $\mathbf{x}_j$ is an l-bit address vector, and $\mathbf{y}$ is an L-bit data vector.

Let $p = K = 2^k$ and let $\mathbf{X}_j$ contain the variables in $\mathbf{x}_j$ and possibly other variables that are fixed (which cannot increase $r_j(f)$).

If $|\boldsymbol{a}| = j$, $r_j(f) \geq 2^L$ because each of 2^L assignments of $\mathbf{y}$ defines a different function of the variables in $\mathbf{X}_j$.

It follows that the formula size is at least $c_\Omega KL$. But $K=2^k \geq n/l$ where $l = log_2\ n$ and $L=n$, from which the result follows. ♥

Krapchenko Lower Bound

- Krapchenko's lower bound to formula size applies to the bases: a) standard, b) {AND, NOT}, and c) {OR, NOT}. It provides for slightly stronger result than Neciporuk's.

- We shall use Krapchenko's method to show that parity on n inputs, $f_{\otimes}^{(n)}$, has $L(f_{\otimes}^{(n)}) = \Omega(n^2)$.

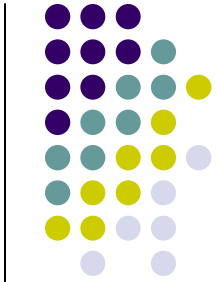
Krapchenko Lower Bound

- We now give an upper bound on formula size for it. $L(f_{\otimes}^{(2)}) \leq 4$ because the following works on 2 inputs:

  $$(x_1 \text{ AND } \text{NOT}\{x_2\}) \text{ OR } (\text{NOT}\{x_1\} \text{ AND } x_2)$$

- We compute $f_{\otimes}^{(n)}$ on n variables by separately computing it on each half of the variables and then combining the results with the above formula.
- If $n = 2^k$, let $P(k) = L(f_{\otimes}^{(n)})$. It follows from the above formula that $P(k) = 4 * P(k\text{-}1) = 4k = n^2$. Thus, the upper bound will agree with the lower bound for this function.

Krapchenko Lower Bound



- **Definition** Let A and B be disjoint sets of B^n. The neighborhood of A and B, N(A,B), is the set of pairs (**x**,**y**) of *n*-tuples, **x** in A and **y** in B, that differ in exactly one position.
- For $f : B^n \rightarrow B$ let $f^{-1}(0)$ and $f^{-1}(1)$ be the sets of *n*-tuples that cause f to assume values 0 and 1, respectively.

  Theorem For $f : B^n \rightarrow B$ and any $A \subseteq f^{-1}(0)$ and any $B \subseteq f^{-1}(1)$ the following holds over the standard basis:

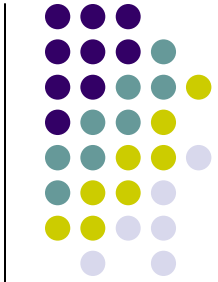
$$L(f) \geq |N(A,B)|^2 / |A||B|$$

Krapchenko Lower Bound

Proof For two-input bases the number of leaves in the optimal formula is one more than the number of two-input gates. Since we count only inputs, we can assume that gates used consist of AND and NOT.

The proof is by induction. Base case: $n = 1$. Here f is either a constant ($|N(A,B)| = 0$ and $L(f) = 0$) or it is x or $\bar{x}$. If $f(x) = x$, $f^{-1}(0) = 0$ and $f^{-1}(1) = 1$, and with $A = \{0\}$, $B = \{1\}$ we have $|N(A,B)| = 1$. A similar argument holds if f is $\bar{x}$. Since $|A| = |B| = 1$, the base case holds.

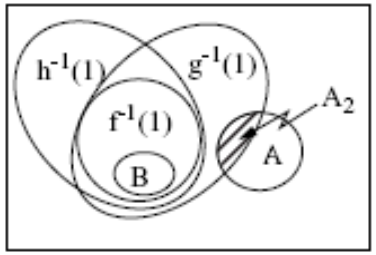
Krapchenko Lower Bound



Assume that $L(f) \geq |N(A,B)|^2/|A||B|$ holds for f such that $L(f) \leq L_0 - 1$ and show it holds for $L(f) = L_0 \geq 2$. Without loss of generality assume that the output gate is AND and computes $f = g$ AND h. Since formula for f is optimal, this is also true for g and h.

Let $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$. That is, $f(x) = 0$ for x in A and $f(x) = 1$ for x in B. It follows that $g(x) = h(x) = 1$ when x in B. I.e., $f^{-1}(1) \subseteq g^{-1}(1)$ and $f^{-1}(1) \subseteq h^{-1}(1)$. Thus, $B \subseteq g^{-1}(1)$ and $B \subseteq h^{-1}(1)$ Let $B_1 = B_2 = B$. Let $A_1 = A \cap g^{-1}(0)$ and $A_2 = A - A_1$.

Krapchenko Lower Bound



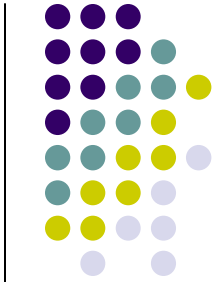
If x in A, $f(x) = 0$. If x also in A_2, $g(x) = 1$. This $\Rightarrow h(x) = 0$ or $A_2 \subseteq h^{-1}(0)$ which $\Rightarrow N(A_1,B_1)$ and $N(A_2,B_2)$ are disjoint (recall that $B_1 = B_2 = B$) and $N(A,B) = N(A_1,B_1) + N(A_2,B_2)$.

Because g and h are independent, by induction

$$L(g) \geq |N(A_1,B_1)|^2 / |A_1||B_1|,$$

$$L(h) \geq |N(A^2,B_2)|^2 / |A_2||B_2|$$

Krapchenko Lower Bound



Also, L(f) = L(g) + L(h). In the above we have |B| as a common denominator. The remaining expression is of the form

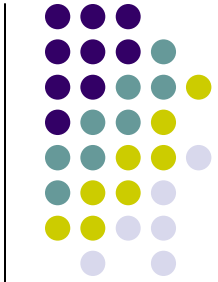
$$S = (n_1^{\,2} / a_1) + (n^2 / a_2)$$

where $n_1 + n_2 = N(A,B)$. It is straightforward to show that

$$S \geq (n_1 + n_2)^2 /(a_1 + a_2)$$

from which the result follows. ♥

Normal Forms



- Shannon expansion of a monotone function

  $f(x_1, x_2, ..., x_n) = f(0, x_2, ..., x_n) \vee (x_1 \wedge f(1, x_2, ..., x_n))$

  - Does this show that every monotone function can be realized by AND and OR gates (the monotone basis)?
- Later we show some monotone functions require exponential size circuits when realized over monotone basis.
- The disjunctive and conjunctive normal forms can be realized by bounded-depth circuits in which AND and OR have potentially unbounded fan-in.

Monotone Disjunctive Normal Form

- **Definition** Let $f : B^n \rightarrow B$ and $g : B^n \rightarrow B$ be monotone increasing. g satisfies $g \Rightarrow f$ if for all $(x_1, x_2, \ldots, x_n)$ in B^n such that $g(x_1, x_2, \ldots, x_n) = 1$, then $f(x_1, x2, \ldots, x_n) = 1$. p, the AND of uncomplemented variables of a function, is a **monotone implicant** of f if $p \Rightarrow f$.
- A **monotone prime implicant** of f is a monotone implicant p of f such that for no other monotone implicant q is $p \Rightarrow q$. (p is the AND of the fewest variables that implies f.)

Monotone Disjunctive Normal Form

Theorem Every monotone Boolean function has a unique expansion as the OR of its monotone prime implicants.

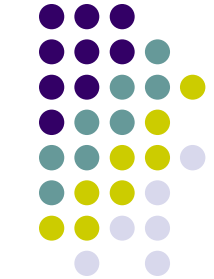
- **Definition** The expansion of a Boolean function as the OR of its monotone prime implicants is called its **monotone disjunctive normal form**.

Path Elimination Method

- The **path elimination** method to derive lower bounds on the monotone circuit size of simple monotone functions.

- Much stronger lower bounds are possible using more sophisticated techniques when applied to complex functions, such as the **clique function** $f_{\text{clique},k}^{(n)}$ defined below.

Path Elimination Method



- Graph G = (V,E), |V| = n specified by $n(n-1)/2$ Boolean variables $\{x_{i,j}\}$. Edge (v_i,v_j) is in E if and only if $x_{i,j}$ =1. G has a k-clique U $\subseteq$ V, if |U| = k, and $x_{i,j}$ =1 for each i, j in U, $i \neq j$.

  Definition $f_{\text{clique},k}^{(n)} : B^{n(n-1)/2} \rightarrow B$ has value 1 if its variables define a graph containing a clique on k vertices. It is monotone.

- It can be shown that $f_{\text{clique},k}^{(n)}$ has monotone circuit size exponential in n.

The Path Elimination Method

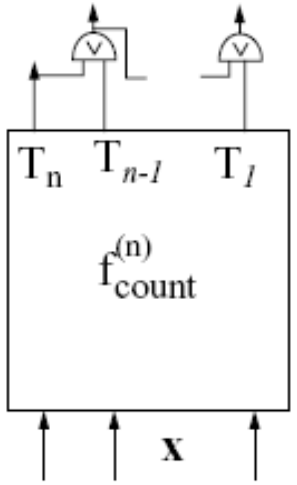
- The binary sorting function $f_{sort}^{(n)} : B^n \rightarrow B$ sorts its n binary inputs into an ascending sequence. $T_t : B^n \rightarrow B$ is 1 if t or more of its inputs are 1 for $t \geq 1$.

$$f_{sort}^{(n)}(x_1, x_2, \ldots, x_n) = (T_1, T_2, \ldots, T_n)$$

Lemma $C_\Omega(f_{sort}^{(n)}) = O(n)$, Ω the standard basis.

The Path Elimination Method

- a) Construct a counting circuit giving number of 1's among the inputs as a binary number. (Sect. 2.11)
- b) Apply decoder to result. $T_j = 0$ unless count = j.
- c) Attach a cascade of OR gates t propagate a 1 in the output to all lower numbered positions.



The Path Elimination Method

- If one input variable is fixed, we show that in a monotone circuit for $f_{sort}^{(n)}$ all gate outputs on a path from an input to some output have constant values and can be eliminated.
- **Theorem** $C_{mon}(f_{sort}^{(n)})$, monotone circuit size for $f_{sort}^{(n)}$, is at least $(nk - 2^k)$ for $k = \lceil \log_2 n \rceil$ and at most $O(n \log n)$.

The Path Elimination Method

Proof The upper bound follows from efficient sorting network due to Ajtai, Komlos, and Szemeredi. A compare/exchange operation produces (*max, min*) of two inputs, which are (OR, AND).

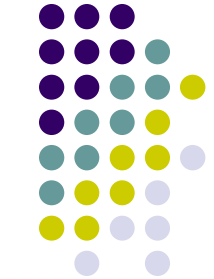
To derive the lower bound consider an input x_i to a monotone circuit for $f_{sort}^{(n)}$ whose path to T_1 is longest. This path has length at least $\lceil log_2\, n \rceil$ because the length of the longest path is shortest when the paths from T_1 to inputs form a balanced binary tree.

The Path Elimination Method

Now set all inputs except x_i to 0. It follows that T_1 has value x_i. There must be a path from x_i to T_1 on which each gate computes x_i. Fix $x_i = 1$. Gate outputs remain fixed for all values of the remaining inputs due to the fact that each gate computes a monotone function. Remove the gates on the path to produce the same function on one fewer inputs. It follows that

$$C_{mon}(f_{sort}^{(n)}) \geq C_{mon}(f_{sort}^{(n-1)}) + \lceil \log_2 n \rceil$$

The Path Elimination Method



- Also, $C_{mon}(f_{sort}^{(2)}) = 2$ because $T_1 = x_1 \wedge x_2$ and $T_2 = x_1 \vee x_2$. For $n = 2^k$ let $y_k = C_{mon}(f_{sort}^{(n)})$. Then,

$$y_k \geq y_{k-1} + 2^{k-1}k$$

$$y_1 = 2 + 2^0 1$$

$$y_k \geq S_k(r) = \Sigma_{j=0}^{k-1} r^j(j+1) \text{ for } r = 2.$$

- Not hard to show that

$$S_k(r) = (k+1)2^k - 2^{k+1} + 1$$