Math 262A Lecture Notes - Nechiporuk's Theorem

Lecturer: Sam Buss
Scribe: Stefan Schneider

October 22, 2013

Nechiporuk [1] gives a method to derive lower bounds on formula size over the full binary basis B_2. The lower bounds we have seen so far were for the parity function over the $\{\wedge, \vee, \neg\}$ basis. The parity function has a trivial linear sized formula over B_2, hence it is inevitable to study new functions.

One of the key ingredients in Nechiporuk's Theorem is the idea of a *restriction*, which sets a subset of the variables to constants.

Definition 1 (Restriction)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be an n-ary Boolean function on variable set $X = \{x_1, \ldots, x_n\}$. Further let $Y = \{y_1, \ldots, y_l\} \subseteq X$ be a subset of the variable set.*

For $Z = X - Y$, a restriction *σ is a mapping $\sigma : Z \to \{0,1\}$. The* restriction of f to σ *is the function $f_{\restriction\sigma}(y_1, \ldots, y_l) = f(x_{k_1}, \ldots, x_{k_n})$ where $x_{k_i} = y_j$ if x_{k_i} is the j-th element of Y and $x_{k_i} = \sigma(x_i)$ if $x_i \notin Y$.*

A subfunction *of f on Y is any $f_{\restriction\sigma}$ with $dom(\sigma) = X - Y$, i.e. the restriction leaves the variables Y free.*

We sometimes alternatively denote a restriction σ as a function $\sigma : X \to \{0, 1, *\}$ where $\sigma^{-1}(*) = Y$.

Theorem 1 (Nechiporuk's Theorem)**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be an n-ary Boolean function and let $Y_1, \ldots, Y_m$ be disjoint subsets of the variable set $X = \{x_1, \ldots, x_n\}$. Then*

$$L_{B_2}(f) + 1 \geq \frac{1}{4} \sum_{i=1}^{m} \log(s_i)$$

where s_i is the number of distinct subfunctions of f on Y_i

Note that $L_{B_2}(f) + 1$ corresponds to the leaf size of the formula.

Application 1: Element Distinction

Before giving the proof of Nechiporuk's Theorem, we discuss an application.

Definition 2. *The* element distinction function *on n variables is a function $ED_n : \{0,1\}^n \to \{0,1\}$ where $n = 2m\lceil \log m \rceil$ for some $m \geq 1$.*

We view the input variables $x_1, \ldots, x_n$ as m blocks of $2\lceil \log m \rceil$ bits each. Then

$$ED_n(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if the } m \text{ blocks are pairwise distinct} \\ 0 & \text{otherwise} \end{cases}$$

Proposition 1.

$$L_{B_2}(ED_n) = \Omega\left(\frac{n^2}{\log n}\right)$$

Proof. To apply Nechiporuk's Theorem we define $Y_i = x_{2i\lceil \log m\rceil+1}, \ldots, x_{2(i+1)\lceil \log m\rceil}$ as the i-th block.

The number of values each block can take on is $2^{2\lceil \log m\rceil} = m^2$. Hence the number of subfunctions on Y_i is

$$s_i \geq \binom{m^2}{m-1} \geq (m^2 - m + 1)^{m-1}$$

where we use the fact the subfunction is uniquely defined by the $m-1$ values the other blocks take on and for each setting of the other blocks such that they take on $m-1$ distinct values, the resulting subfunction on Y_i is distinct.

Since $n = 2m\lceil \log m\rceil$ we have $m = \Omega\left(\frac{n}{\log n}\right)$ and therefore

$$\log(s_i) \geq (m-1)\log(m^2 - m + 1) = \Omega(m \log m) = \Omega(n)$$

and by Nechiporuk's Theorem we get

$$L_{B_2}(ED_n) + 1 \geq \frac{1}{4}\sum_{i=1}^{m} \log(s_i) = \sum_{i=1}^{m} \Omega(n) = \Omega(mn) = \Omega\left(\frac{n^2}{\log n}\right)$$

□

The lower bound above is tight, as we can construct a formula of size $O\left(\frac{n^2}{\log n}\right)$.

Proposition 2.

$$L_{\{\wedge,\vee,\neg\}}(ED_n) = O\left(\frac{n^2}{\log n}\right)$$

Proof. For every $0 \leq i, i' \leq m-1$ with $i \neq i'$ we construct the formula

$$\bigwedge_{j=1}^{2\lceil \log m\rceil} (x_{2i\lceil \log m\rceil+j} \equiv x_{2i'\lceil \log m\rceil+j})$$

which returns true if and only if blocks i and i' are identical. We then take the disjunction of the $\binom{m}{2}$ formulas and negate the output to get a formula for ED_n.

For two inputs a and b, the function $a \equiv b$ requires a constant number of gates. Furthermore, for every pair i, i' the formula contains $2\lceil \log m\rceil - 1$ conjunctions and the disjunction requires $\binom{m}{2} - 1$ gates. The total size of the formula is therefore

$$O\left(\binom{m}{2}\log m\right) + O\left(\binom{m}{2}\right) = O\left(m^2 \log m\right) = O\left(\frac{n^2}{\log n}\right)$$

□

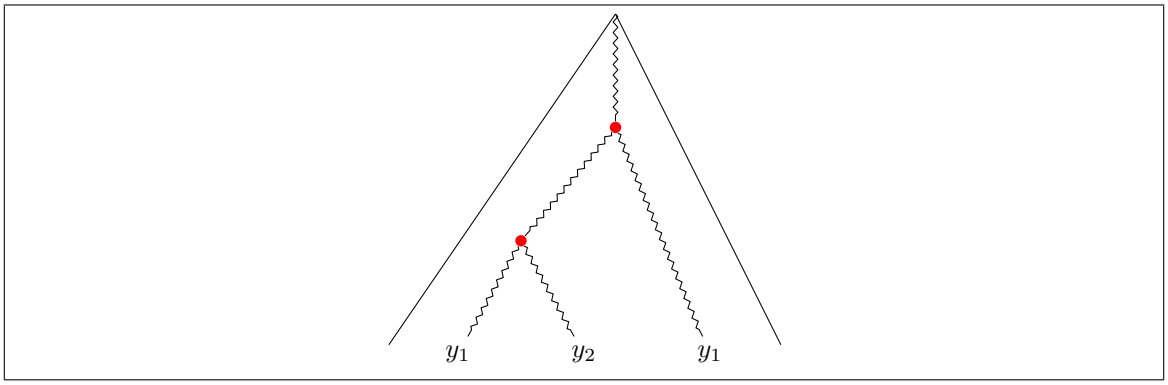

Figure 1: A function restricted to a subtree for $Y_i = \{y_1, y_2\}$. The set W_i is colored red.

Proof of Nechiporuk's Theorem

The basic idea of the proof is the look at what happens to the formula when a restriction is applied.

Proof. We prove

$$l_i \geq \frac{1}{4}\log(s_i)$$

where l_i is the number of occurrences of variables in Y_i in a formula φ on basis B_2 for f. The theorem then follows immediately.

Given φ we form a subtree with l_i leaves that contains all paths from any leaf labeled by a variable in Y_i to the output gate. Let W_i be the set of gates that have two inputs in this tree. Since a binary tree with l_i leaves has exactly $l_i - 1$ vertices, we have $|W_i| = l_i - 1$. Figure 1 sketches such a subtree. Note that an edge in this subtree might contain an arbitrarily long chain of gates that only have a single input in the subtree.

Consider a subfunction of φ on Y_i. The formula then simplifies immediately to the subtree described above. Furthermore, each path from an input gate or a gate in W_i to the next gate in W_i or the output is then a function on one variable. Note that there are only 4 Boolean functions on one input, $h(x) = x$, $h(x) = \neg x$, $h(x) = 1$ and $h(x) = 0$.

Since there is one path originating at each leaf in the subtree and each vertex in W_i, the number of paths is $l_i + l_i - 1 = 2l_i - 1$. Also, since the functions described by each of the paths fully specify the restricted subfunction, the number of subfunctions of f on Y_i is bounded by

$$s_i \leq 4^{2l_i - 1} < 4^{2l_i} = 2^{4l_i}$$

and therefore

$$l_i > \frac{1}{4}\log(s_i)$$

□

Application 2: Storage Access Functions

In this section we discuss the *storage access function* or *universal function*. The function was first defined in the same paper as Nechiporuk's Theorem.

Definition 3. *The* storage access function *is a Boolean function* $SA_n : \{0,1\}^n \to \{0,1\}$ *with* $SA_n(x_1, \ldots, x_n) = SA_n(y_0, \ldots, y_{m-1}, z_1, \ldots, z_l)$ *where* $l = \lceil \log m \rceil$ *and* $n = m+l$. *Let* $0 \leq i \leq m-1$ *be the integer encoded by the bits* $\vec{z}$. *Then*

$$SA_n(\vec{y}, \vec{z}) = y_i$$

The function is universal in the sense that by restricting the variables $y_0, \ldots, y_{m-1}$ we can get an arbitrary function on $\lceil \log m \rceil$ inputs.

Definition 4. *The* indirect storage access function *is a Boolean function* $ISA_n : \{0,1\}^n \to \{0,1\}$ *with* $ISA_n(x_1, \ldots, x_n) = ISA_n(y_0, \ldots, y_{m-1}, z_1, \ldots, z_m, u_1, \ldots, u_k)$ *where* $k = \log\left(\frac{m}{\lceil \log m \rceil}\right)$. *We view* $\vec{z}$ *as* $\frac{m}{\lceil \log m \rceil}$ *blocks of* $\lceil \log m \rceil$ *bits each. Let* $0 \leq i \leq \frac{m}{\lceil \log m \rceil} - 1$ *be the integer encoded by the bits* $\vec{u}$ *and let* j_i *be the integer encoded by the* i*-th block of* $\log m$ *bits in* $\vec{z}$. *Then*

$$ISA_n(\vec{y}, \vec{z}, \vec{u}) = y_{j_i}$$

Proposition 3.

$$L_{B_2}(ISA_n) = \Omega\left(\frac{n^2}{\log n}\right)$$

Proof. Let Y_i be the i-th block of $\vec{z}$. For a fixed $0 \leq i \leq \frac{m}{\lceil \log m \rceil} - 1$ consider a restriction such that $\vec{u}$ encodes i and all but the i-th block of z is restricted to some arbitrary values. Then the vector $\vec{y}$ is the function table of the subfunction on Y_i. Hence the number of subfunctions is given by $s_i = 2^m$. Therefore

$$L_{B_2}(ISA_n) + 1 \geq \frac{1}{4} \sum_{i=1}^{m/\lceil \log m \rceil} \log(s_i) = \frac{m^2}{\lceil \log m \rceil} = \Omega\left(\frac{n^2}{\log n}\right)$$

where we use $n = 2m + \log\left(\frac{m}{\lceil \log m \rceil}\right)$ for the last step. □

For upper bounds, we consider SA_n first. A possible naïve approach is to construct a DNF as follows.

$$\bigvee_{i=0}^{m} (\text{“}\vec{z} \text{ encodes integer i”} \wedge y_i)$$

The check if $\vec{z} = z_1, \ldots, z_{\lceil \log m \rceil}$ encodes an integer i is a conjunction of $\lceil \log m \rceil$ variables or negated variables. Hence the size of the DNF is $O(m \log m) = O(n \log n)$.

To get an upper bound of $O(n)$ we construct a formula based on a decision tree. Note that a matching lower bound of $\Omega(n)$ is trivial.

Definition 5. *A* decision tree *is a binary tree with internal vertices labeled by input variables and the leaves labeled by either* 0 *or* 1. *The* branching program *of a decision tree starts at the root node. If the current vertex is labeled* x_i, *then the variable* x_i *is queried. If the result is* 0, *the computation continues at the left child, otherwise it continues at the right child. The value of the leaf reached is the output of the computation.*

Proposition 4.

$$L_{\{\wedge,\vee,\neg\}}(SA_n) = O(n)$$

Proof. Consider a decision tree T for SA_n, where T is a complete binary tree of depth $\lceil \log m \rceil$ with every node on level i being labeled by the variable z_i. Here, the root is level 1. The vertices on the last level are labeled by y_i, where i is the integer encoded by the queried values of $\vec{z}$. The decision tree is a direct implementation of the intuition of first reading $\vec{z}$ and then looking up of the appropriate value y_i. Figure 2 shows the decision tree described above.

Converting this decision tree into a formula we get

$$T = (\neg z_1 \wedge T_0) \vee (z_1 \wedge T_1)$$

where

$$T_0 = (\neg z_2 \wedge T_{00}) \vee (z_2 \wedge T_{01})$$

and

$$T_1 = (\neg z_2 \wedge T_{10}) \vee (z_2 \wedge T_{11})$$

Continuing this construction for $\lceil \log m \rceil$ levels we eventually set $T_{\vec{z}} = y_i$, where $\vec{z}$ encodes the integer i. Figure 3 shows the constructed formula.

The formula contains 2^j leaves labeled z_j for every j and one leaf labeled y_i for every i. Hence the size of this formula is given by

$$L_{\{\wedge,\vee,\neg\}}(SA_n) \leq m + \sum_{j=1}^{\lceil \log m \rceil} 2^j = O(m) = O(n)$$

□

Note that more generally, any decision tree of depth d can be converted into a formula of leaf size $O(2^d)$.

Using a similar technique, we also get a tight upper bound for the indirect storage access function.

Proposition 5.

$$L_{\{\wedge,\vee,\neg\}}(ISA_n) = O\left(\frac{n^2}{\log n}\right)$$

Proof. We construct a decision tree of depth $\log\left(\frac{m}{\lceil \log m \rceil}\right) + \lceil \log m \rceil$. We first query the variables in $\vec{u}$ and then use the fact that $ISA_n(\vec{y}, \vec{z}, \vec{u}) = SA_n(\vec{y}, \vec{z_j})$ where $\vec{z_j}$ is the j-th block of $\vec{z}$ and j is the integer encoded by $\vec{u}$.

The decision tree consists of $\log\left(\frac{m}{\lceil \log m \rceil}\right)$ levels that query the variables in $\vec{u}$ and at every root we attach a decision tree with $\lceil \log m \rceil$ levels equivalent to the decision tree of $SA_n(\vec{y}, \vec{z_j})$. Figure 4 shows the decision tree.

The depth of the decision tree is $\log\left(\frac{m}{\lceil \log m \rceil}\right) + \lceil \log m \rceil$ and can therefore be converted into a formula of size

$$L_{\{\wedge,\vee,\neg\}}(ISA_n) = O\left(2^{\log\left(\frac{m}{\log m}\right)+\log m}\right) = O\left(\frac{m^2}{\log m}\right) = O\left(\frac{n^2}{\log n}\right)$$

□

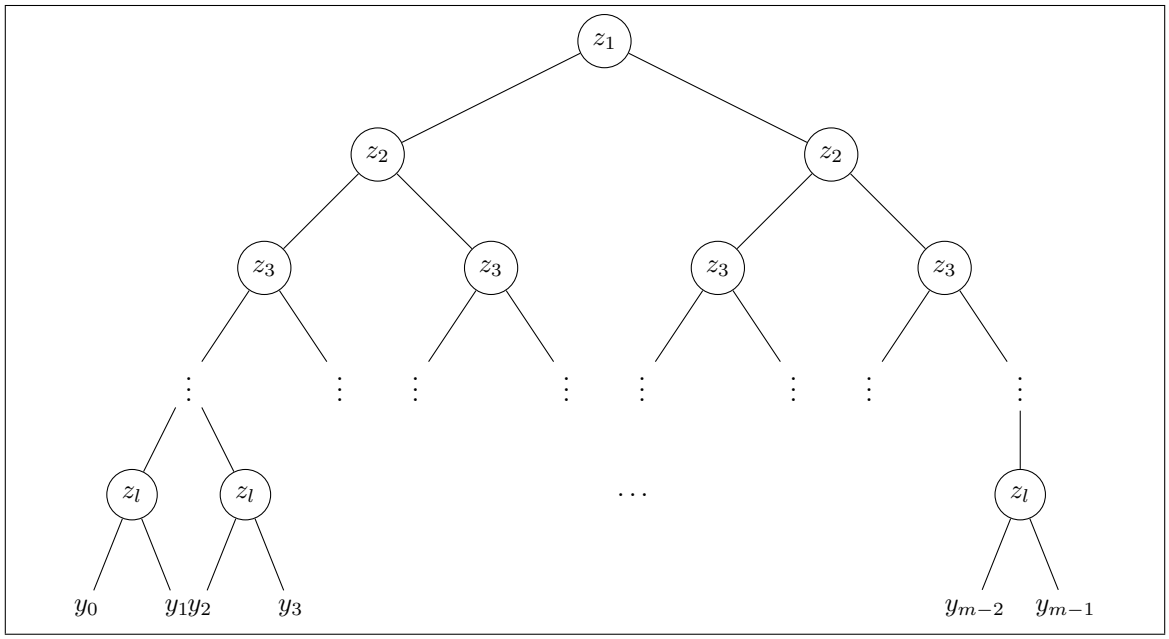

Figure 2: A decision tree for SA_n

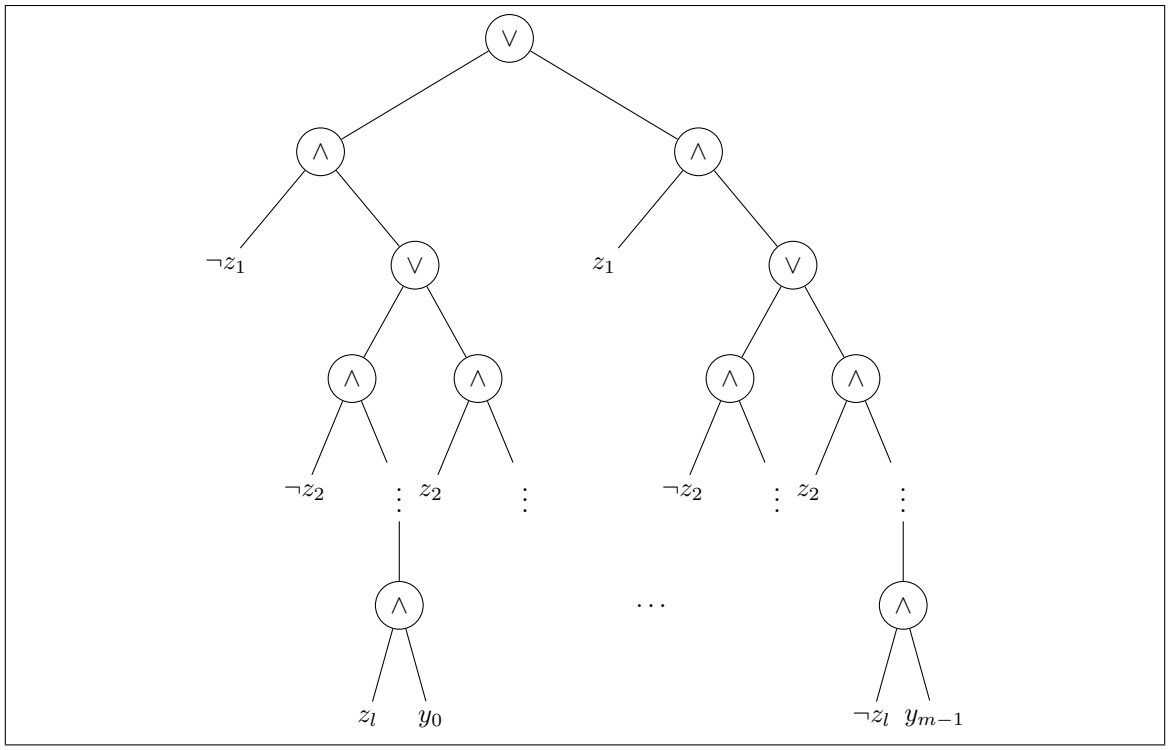

Figure 3: A $\{\wedge, \vee, \neg\}$-formula equivalent to the decision tree in Figure 2

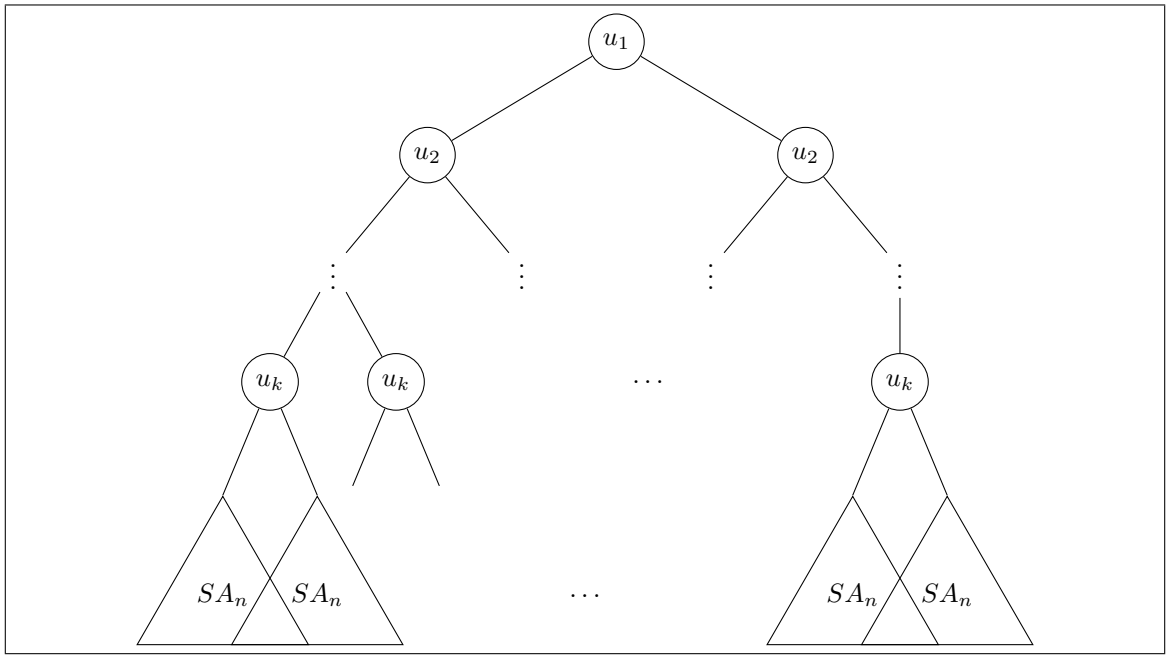

Figure 4: A decision tree for ISA_n

References

[1] E. I. Nechiporuk, *On a Boolean function*. Soviet Math. Dokl. 7(4), 999-1000, 1966.