Steven Rudich, Complexity Theory: From Gödel to Feynman, Lecture 8. "Natural Proof of Lower Bounds, in Steven Rudich, Avi Wigderson (eds), Computational Complexity Theory, IAS/Park City mathematics series, Vol. 10, AMS (2004), 75 - 87.

Razborov has proved an $n^{\Omega(\log n)}$ lower bound for the monotone complexity of the perfect matching function. Since a maximum matching of a graph can be computed in polynomial time, an $\Omega_0$-network of polynomial size for the perfect matching function exists. Hence, the gap between monotone and non-monotone network complexity is at least $n^{\Omega(\log n)}$. In 1986, Éva Tardos has shown that this gap is indeed exponential using another monotone Boolean function. Hence, it is not always possible to obtain large lower bounds for the $\Omega_0$-complexity from a large lower bound for the $\Omega_m$-complexity of a monotone Boolean function. But this does not exclude the possibility that techniques developed for the proof of lower bounds for the monotone complexity can also be useful for the proof of lower bounds for the $\Omega_0$-complexity of Boolean functions.

The first step will be the simplification of an arbitrary $\Omega_0$-network without increasing its size by more than a small constant factor.

Given any $\Omega_0$-network $\beta$, we can convert $\beta$ to an equivalent $\Omega_0$-network $\beta'$ where all negations occur only at the input nodes. Moreover, $|\beta'| \leq 2|\beta|$. For doing this, we start at the output nodes and apply deMorgan rules for bringing the negations to the input nodes. Since gates can be simultaneously negated and not negated, some gates have to be doubled. The resulting network is a so-called <u>standard network</u> where only input variables are negated.

**<u>Exercise:</u>**

**Work out the transformation of an $\Omega_0$-network $\beta$ into a standard network $\beta'$. Prove formally $|\beta'| \leq 2 \cdot |\beta|$.**

The <u>standard complexity</u> $C_{st}(f)$ of a function $f \in B_{n,m}$ is the size of a smallest standard network which computes $f$. Note that the standard and the $\Omega_0$-complexity of a function $f$ differs at most by the factor two. Hence, for proving a non-linear lower bound for the $\Omega_0$-complexity of a Boolean function, we can restrict us to the consideration of standard networks.

1994 at the 26th STOC, Alexander Razborov and Steven Rudich have introduced the notion of "natural proof". They argue that the known proofs of lower bounds on the complexity of explicit

Boolean functions in non-monotone models fall within their definition of natural. They show that natural proofs cannot prove $P \neq NP$ unless good pseudorandom generators do not exist. Since the existence of such generators is widely believed, natural proofs are widely accepted to be a barrier for proving $P \neq NP$ using Boolean complexity. We shall discuss this in the subsequence.

First we shall define "natural proofs". These are proofs which use a natural combinatorial property.

A <u>combinatorial property</u> is a subset $\{C_n \subset B_n \mid n \in \mathbb{N}\}$ of Boolean functions. $C_n$ is called <u>natural</u> if there is $C_n^* \subseteq C_n$ which satisfies:

1. $\forall f \in B_n$ it can be decided in $2^{O(n)}$ time if $f \in C_n^*$. (<u>constructiveness</u>)
2. $|C_n^*| \geq 2^{-O(n)} |B_n|$. (<u>largeness</u>)

The first property means that the characteristic function of $C_n^*$ can be computed in polynomial time in the size of the truth table of the input function $f \in B_n$. The second property says that a function randomly chosen from $B_n$ is contained in $C_n^*$ with non-negligible probability.

<u>$P/poly$</u> is the set of languages which are recog-

nizable by a family of Boolean networks of polynomial size. Note that $P \subseteq P/poly$. A combinatorial property is <u>useful against P/poly</u> if the network complexity of any sequence $f_1, f_2, \ldots, f_n, \ldots$ where $f_n \in C_n$ is super-polynomial; i.e., for all $k \in \mathbb{N}$ there is $n_k \in \mathbb{N}$ such that the network complexity of $f_n$ is larger than $n^k$ for all $n > n_k$. A proof that a Boolean function does not have polynomial network complexity is <u>natural against P/poly</u> if the proof uses a natural combinatorial property $C_n$ which is useful against $P/poly$.

Razborov and Rudich mention that "from experience it is plausible to say that we do not yet understand the mathematics of $C_n$ outside exponential time (as a function of $n$) well enough to use them effectively in a combinatorial style proof." This means that combinatorial properties used in a lower bound proof are constructive!

With respect to the largeness property, they write: "In Section 5 we give some solid theoretical evidence for largeness, by showing that any $C_n$ based on a "formal complexity measure" must be large." We shall discuss the "solid theoretical evidence for largeness" given by Razborov and Rudich.

A <u>formal complexity measure</u> is a function $\mu : B_n \to \mathbb{R}^+$ such that

i) $\mu(f) \leq 1$ for $f \in \{x_1, x_2, \ldots, x_n, \neg x_1, \neg x_2, \ldots, \neg x_n\}$

ii) $\left.\begin{array}{l} \mu(f) \wedge \mu(g) \leq \mu(f) + \mu(g) \\ \mu(f) \vee \mu(g) \leq \mu(f) + \mu(g) \end{array}\right\} \forall f, g \in B_n.$

A <u>formula</u> is a Boolean network where the underlying graph is a tree. The <u>size</u> of a formula $\beta$ is the number of leaves in $\beta$. The <u>formula size</u> $L_{\Omega_0}(f)$ of $f \in B_n$ is the size of a smallest $\Omega_0$-formula which computes $f$.

<u>Exercise</u>

Let $\mu$ be a formal complexity measure. Prove that $\mu(f) \leq L_{\Omega_0}(f)$ $\forall f \in B_n$.

Note that $L_{\Omega_0}$ itself is a formal complexity measure.

Razborov and Rudich mention: "For any approximation model $\mathcal{M}$, we have

$$\rho(f * g, \mathcal{M}) \leq \rho(f, \mathcal{M}) + \rho(g, \mathcal{M}) + 1;$$

hence, $\rho(f, \mathcal{M}) + 1$ is a formal complexity measure", where $* \in \{\wedge, \vee\}$.

Razborov and Rudich show that "any formal complexity measure $\mu$ which takes a large value at a single function, must take large values almost everywhere". This is formalized by the

following theorem.

**Theorem 7.1**

Let $\mu$ be a formal complexity measure on $B_n$, and let $\mu(f) \geq t$ for some $f \in B_n$. Then for at least $1/4$ of all functions $g \in B_n$, $\mu(g) \geq t/4$.

**Proof:**

Let $g$ be any function in $B_n$. Define

$$h := f \oplus g.$$

Then

$$f = (f \oplus g) \oplus g = h \oplus g$$
$$= (h \wedge g) \vee (\neg h \wedge \neg g).$$

Hence,

$$(*) \qquad \mu(f) \leq \mu(g) + \mu(\neg g) + \mu(h) + \mu(\neg h).$$

Let

$$S := \{ g \in B_n \mid \mu(g) < t/4 \}.$$

Assume that

$$|S| > 3/4 \, |B_n|.$$

If we pick the above function $g$ randomly in $B_n$ with probability $|B_n|^{-1}$ then $\neg g$, $h$ and $\neg h$ are also random elements of $B_n$ (though not independent) each with the same probability.

Union bound $\Rightarrow$

Prob ( at least one of $g, \neg g, h, \neg h$ not in $S$ )

$$< 4 \cdot \frac{1}{4} = 1.$$

$\Rightarrow$

There exists at least one choice of $g$ such that $g, \neg g, h, \neg h \in S$.

Since each of these four functions has measure $< t/4$, we obtain from (*)

$$\mu(f) < t, \quad \text{a contradiction.}$$

□

Razborov and Rudich conclude that "every combinatorial property based on such a measure automatically satisfies the largeness condition in the definition of natural property". But they do not formalize what they mean that a property is based on a formal complexity measure.

We shall discuss two possible interpretations and their consequences.

1) The combinatorial property of $f \in B_n$ is

"$\mu(f) \geq t$"

for a certain bound $t$.

Then it is not clear how to prove the constructiveness of the property. Note that the formula size itself is a formal complexity measure.

Theorem 1.1 implies also for the formula size that most functions in $B_n$ need formula size $\frac{2^n}{n}$. No algorithm of time complexity $2^{O(n)}$ is known which decides for any given function $f \in B_n$ if its formula size is $> \frac{1}{4}\frac{2^n}{n}$.

2) For a Boolean function $f \in B_n$, a combinatorial property based on the structure of $f$ is used to prove $\mu(f) \geq t$.

By Theorem 7.1, most functions $g \in B_n$ satisfy

$$\mu(g) \geq t/4.$$

But this does not imply that such a function $g$ has the same combinatorial property as $f$.

$\Rightarrow$

Theorem 7.1 does not imply the largeness property for the combinatorial property used to prove $\mu(f) \geq t$.

To be more concrete, we shall give an example. Analogously to the definition of formal complexity measure, we define

a <u>formal monotone complexity measure</u> is a function $\mu' : M_n \longrightarrow \mathbb{R}^+$ such that

i) $\mu'(x_i) \leq 1$ for $1 \leq i \leq n$

ii) $\mu'(f) \cdot \mu'(g) \leq \mu'(f) + \mu'(g)$
$\mu'(f) \vee \mu'(g) \leq \mu'(f) + \mu'(g)$ $\quad \forall f, g \in M_n$

A <u>monotone formula</u> is a monotone Boolean network where the underlying graph is a tree. The size of a monotone formula $\beta$ is the number of leaves in $\beta$. The monotone formula size $L_{\Omega_m}(f)$ of $f \in M_n$ is the size of a smallest monotone formula which computes $f$.

Again it is easy to prove that for each formal monotone complexity measure $\mu'$

$$\mu'(f) \leq L_{\Omega_m}(f) \quad \forall f \in M_n.$$

$L_{\Omega_m}$ itself is a formal complexity measure.

Since the proof of Theorem 7.1 uses negations, we cannot prove in the same way an analogous result for a formal monotone complexity measure $\mu'$. But if we use the formal monotone complexity measure $L_{\Omega_m}$, the analogous result is trivially fulfilled because of the well known counting argument.

Let us consider the proofs of the exponential lower bound for the monotone complexity of the clique function. One of the combinatorial properties used in the proofs is the following:

- Only a "small" number of prime implicants contains a clique on any fixed set of $r$ nodes where $r$ is chosen appropriately.

This property implies that the complexity of any function having this property is $< \frac{2^n}{n}$.

Theorem 1.1 $\Rightarrow$

A combinatorial property which imply that each function in $B_n$ which fulfills this property has $\Omega_0$-complexity $< \frac{2^n}{n}$ cannot fulfill the largeness property!

Altogether seems that "natural proofs" built not a barrier for proving $P \neq NP$ using circuit complexity.

Therefore, it makes sence to investigate the approximation method with regard its expandability to standard networks. Before doing this, we shall investigate the computation in standard networks.

Let $\beta$ be a standard network which computes a function $f \in B_n$. Let $g$ be any node in $\beta$. As in monotone networks, the function $res_\beta(g)$ can be written as a DNF-formula: i.e.,

$$res_\beta(g) = \bigvee_{j=1}^{s_1} m_j;$$

and also as a CNF-formula; i.e.,

$$res_\beta(g) = \bigwedge_{j=1}^{s_2} d_j.$$

In contrast to monotone networks, the DNF-representation $DNF_\beta(g_0)$ of $res_\beta(g_0)$ must not contain the prime implicants of $f$ as a monomial.

To extend the proof techniques developed for monotone networks to standard networks, it is useful to recognize the prime implicants in the monomials of $DNF_\beta(g_0)$. Note that each monomial $m_j$ contains at least one prime implicant of $f$. If a monomial $m_j$ contains $\ell > 1$ prime implicants then we add $\ell - 1$ copies of $m_j$ to $DNF_\beta(g_0)$. Let

$$PIM(f) = \{p_1, p_2, \ldots, p_t\}.$$

By separating in each monomial containing a prime implicant $p_j$ the prime implicant and the other literals, we obtain

$$res_\beta(g_0) = \bigvee_{j=1}^{t} \bigvee_{i=1}^{\ell_j} p_j \wedge m'_{j,i},$$

where $DNF_\beta(g_0)$ contains $\ell_j$ monomials containing the prime implicant $p_j$, $1 \le j \le t$. The following theorem characterizes exactly $DNF_\beta(g_0)$:

Theorem 7.2

Let $\beta$ be a standard network which computes a Boolean function $f \in B_n$ at its output node $g_0$. Then the following hold:

- $DNF_\beta(g_0)$ contains only implicants of $f$. Furthermore, for each $a \in f^{-1}(1)$, $DNF_\beta(g_0)$ contains an implicant $m_a$ of $f$ such that $m_a(a) = 1$.

Proof:

Exercise

Also, the CNF-representation $CNF_\beta(g_0)$ of $res_\beta(g_0)$ must not contain the prime clauses as a clause. We can separate in each clause containing a prime clause $c_j$ the prime clause and the other literals. The following theorem characterizes exactly $CNF_\beta(g_0)$.

Theorem 7.3

Let $\beta$ be a standard network which computes a Boolean function $f \in B_n$ at its output node $g_0$. Then the following hold:

- $CNF_\beta(g_0)$ contains only $f$-clauses. Furthermore, for each $b \in f^{-1}(0)$, $CNF_\beta(g_0)$ contains an $f$-clause $d_b$ such that $d_b(b) = 0$.

Proof:

Exercise

DNF/CNF- and CNF/DNF-switches designed for formulas containing only non-negated variables can be extended to formulas which contains non-negated and negated variables.

**Exercise:**
Extend the DNF/CNF - and CNF/DNF -switches such that they can be used for formulas containing non-negated and also negated variables.

Now, we shall investigate the approximation method with regard its expendability to standard networks. We shall do this for DNF/CNF - approximators first. Then we shall consider Razborov - approximators.

## 8. DNF/CNF - approximators for non-monotone computations.

Our intention is to give some evidence that DNF/CNF - approximators cannot be extended to prove a lower bound for the standard complexity of any Boolean function $f \in B_n$. With respect to the extended approximators, we distinguish two cases:

1) Only the non-negated or only the negated variables are approximated.
2) Both kind of literals are approximated.

Case 1: Only one kind of literals is approximated

W.l.o.g., we consider only the subcase that the non-negated variables are approximated. The other subcase can be discussed in the same way

Let $\beta$ be a standard network which computes a Boolean function $f \in B_n$ at its output node $g_0$. Analogously to monotone networks, we approximate for each node $g$ in $\beta$ the function $res_\beta(g)$ by two formulas

$$D_g^r = \bigvee_{i=1}^{t_1} m_i \quad \text{and} \quad C_g^k = \bigwedge_{j=1}^{t_2} d_j$$

where each $m_i$ is a monomial of size $< r$ and each $d_j$ is a clause of size $< k$.

The sizes of a monomial $m_i$ or of a clause $d_j$ depend only on the non-negated variables in it. This implies that any number of negated variables can be in $m_i$ or in $d_j$. We consider the case that the size is the number of distinct variables in $m_i$ and $d_j$, respectively.

As observed above, $DNF_\beta(g_0)$ contains for each $a \in f^{-1}(1)$ a monomial $m_a$ such that $m_a(a) = 1$. Only for $a = (1,1,\ldots,1)$, we can exclude that $m_a$ contains any negated variable. Each clause $d$ in $CNF_\beta(g_0)$ contains a literal in $m_a$. For $a \in f^{-1}(1) \setminus \{(1,1,\ldots,1)\}$, this literal can be a negated variable.

If $C_{g_0}^k$ is not the constant function 1 then $C_{g_0}^k$ must contain at least one clause $d$. For each $a \in f^{-1}(1)$ which is computed correctly by $C_{g_0}^k$, the clause $d$ must contain a literal

of a monomial $m_a$ with $m_a(a) = 1$. For $a \neq (1,1,\ldots,1)$ this literal could be a negated variable. Since $d$ can contain any number of negated variables, for each input $a \in f^{-1}(1) \setminus \{(1,1,\ldots,1)\}$, the clause $d$ can contain such a literal. Hence, no input in $f^{-1}(1) \setminus \{(1,1,\ldots,1)\}$ can be excluded to be computed correctly by the approximator $C^k_{g_0}$. With respect to the approximator $D^r_{g_0}$, a similar consideration can be performed.

$\Rightarrow$

A DNF/CNF-approximator which proves a lower bound for the standard complexity of a Boolean function has to approximate both kinds of literals.

Case 2: Both kinds of literals are approximated.

Now, the size of a monomial $m_i$ or of a clause $d_j$ is the maximum of the number of non-negated variables and of the number of negated variables. This implies that the length of each monomial in $D^r_g$ is less than $2r$ and that the length of each clause in $C^k_g$ is less than $2k$.

It is no problem to switch from a CNF-approximator of size $k$ to a DNF-approximator of size $r$ by performing a CNF/DNF-switch where at most

$(2k-1)^{2r}$ monomials of length $2r$ are replaced by zero. But in contrast to the monotone case, the number $(2k-1)^{2r}$ is too large. To see this fix any $2r$ variables $x_{i_1}, x_{i_2}, \dots, x_{i_{2r}}$. Consider any $a \in f^{-1}(1)$. Let

$$m'_a := y_{i_1} y_{i_2} \cdots y_{i_{2r}}$$

where for $1 \leq j \leq 2r$

$$y_{ij} := \begin{cases} x_{ij} & \text{if } a_{ij} = 1 \\ \neg x_{ij} & \text{if } a_{ij} = 0. \end{cases}$$

Note that $m'_a(a) = 1$ and $|m'_a| = 2r$.

$\Rightarrow$

$\leq 2^{2r}$ monomials of length $2r$ could suffice such that their replacement by zero could destroy the correct computation of $f(a)$ for all $a \in f^{-1}(1)$.

Since $(2k-1)^{2r} > 2^{2r}$, the construction of one approximator $D_g^r$ could destroy the correct computation of $f(a)$ for all $a \in f^{-1}(1)$.

An analogous argumentation holds with respect to switching from a DNF-approximator of size $r$ to a CNF-approximator of size $k$.

Altogether, we have given some evidence that extended DNF/CNF-approximators cannot be used to prove a lower bound for the Standard complexity of any function $f \in B_n$.