The Fusion Method for Lower Bounds in Circuit Complexity

Avi Wigderson

May 27, 1993

Abstract

This paper coins the term "The Fusion Method" to a recent approach for proving circuit lower bounds. It describes the method, and surveys its achievements, potential and challenges.

1 Introduction

In a recent paper, Karchmer [6] suggested an elegant way in which one can view at the same time both the "approximation method" of Razborov [13] and the "topological approach" of Sipser [15] for proving circuit lower bounds. In Karchmer's setting the lower bound prover shows that a given circuit C is too small for computing a given function f by contradiction, in the following way. She tries to combine (or 'fuse', as we propose calling it) *correct* accepting computations of inputs in $f^{-1}(1)$ by C into an *incorrect* accepting computation of an input in $f^{-1}(0)$.

It turns out that this "Fusion Method" reduces the dynamic computation of f by C into a static combinatorial cover problem, which provides the lower bound. Moreover, different restrictions on how we can fuse computations result in different combinatorial problems, which turn out to capture an amazing variety of complexity classes such as P, NP, NL, $\oplus L$ and mP.

I believe that the Fusion Method is the most viable approach we have today for proving lower bounds in a variety of computational models, most notably non-monotone circuits. My main aim in this paper is to induce people to understand this method (this is easy), and develop techniques for solving the combinatorial problems it raises (but this is hard!). For this purpose I summarize essentially all my knowledge and intuition about it, in the following order.

In Section 2 I give a semi-intuitive, semi-formal account of the Fusion Method. Section 3 is a kind of annotated bibliography, describing the evolution of ideas that led to (my view of) the Fusion Method. In Section 4 I formally define the notions from Section 2. This is done in a very general way, that will fit all existing results that use this framework. I then list all the complexity classes that can be characterized by this method, and the best lower bounds obtained by it. Section 5 is devoted to a discussion of the interesting features of this method, and Section 6 contains a list of natural directions for further research. In Section 7 I list three concrete combinatorial/algebraic problems that arise from the Fusion Method. It is completely self contained, and the impatient reader can skip everything and go there directly.

2 Intuition

2.1 Computations

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function on variables $X = \{x_1, x_2, \ldots x_n\}$, and P a straight line program that allegedly computes f. Thus $P = (g_1, g_2, \ldots g_t)$ with $g_j \in X$ for $1 \le j \le n$, and for every $j > n$, $g_j = g_{j_1} \circ_j g_{j_2}$ for some $j_1, j_2 < j$ and $\circ_j$ is a binary operation in OP, the set of operations used by P.

For every $z \in \{0,1\}^n$, let $P(z) = (g_1(z), g_2(z), \ldots, g_t(z)) \in \{0,1\}^t$ denote the *computation* of P on z. (Clearly, not every vector in $\{0,1\}^t$ is a computation of P on some input.) Let $U = f^{-1}(1) \subseteq \{0,1\}^n$ denote the set of "ones" of f. Construct a $|U| \times t$ Boolean matrix P_U which for every $u \in U$ has a row $P(u)$. The columns of P_U are Boolean functions on U. In fact, the j^{th} column is the restrictions of g_j to the domain U. Note that the first n columns are the variables x_i. Furthermore, as we assumed P computes f, P must accept all members of U and thus the last (t^{th}) column of P_U is $\overline{1}$, the all 1's vector in $\{0,1\}^U$.

2.2 Combinations

Let $\Omega \subseteq \{G : \{0,1\}^U \to \{0,1\}\}$ be some set of functionals on $\{0,1\}^U$. We will combine (fuse) the computations of P using some functional $F \in \Omega$. Applying F to a column g_j of P_U gives a Boolean value, which can be thought of as a truth value to this node in the program. Applying F to each of the t columns of P_U results in a vector $F(P_U) \in \{0,1\}^t$, which, if we are lucky, is a real computation of the program. If we are really lucky, $F(P_U)$ is in fact an accepting computation $P(z)$, of some "zero" z of f (i.e $f(z) = 0$), which will give a contradiction. Let us enumerate the requirements on F that will ensure a contradiction:

1. F **defines a "zero" of** f**:** For some $z \notin U$ and for all $1 \le i \le n$, $F(x_i) = z_i$.

2. F **is accepting:** $F(\overline{1}) = 1$.

3. F **is consistent:** For every $n < j \le t$, $F(g_{j_1}) \circ_j F(g_{j_2}) = F(g_{j_1} \circ_j g_{j_2})$.

How can we arrange a "lucky" combination F? The first two conditions are easy to ensure, simply by restricting ourselves to F satisfying them. Namely, we define $\Omega_f = \{F \in \Omega \,|\, F \text{ satisfies } (1), (2)\}$ and take $F \in \Omega_f$. However, condition (3) depends on P, which is not really given when we are trying to prove a lower bound — in fact only t is given. But the point is the following: if some P of length t computes f, all these attempts F must fail to satisfy (3).

2.3 Lower Bounds and Characterizations

The discussion above motivates the following definition.
Definition (Covering): Say that a triple $< g, h, o >$ with $g, h \in \{0,1\}^n \rightarrow \{0,1\}$, $o \in OP$ *covers* F if
$F(g) \circ F(h) \neq F(g \circ h)$. Let $\rho(f)$ denote the smallest number of such triples that cover all of Ω_f (this number depends on the choice of OP and Ω).

Then, we have just proved:

Theorem 1 (Meta-Theorem) *$\rho(f)$ is a lower bound on the length of the shortest program (over OP) for f.*

Why should we bother to restrict our fusing combinations to Ω, rather than take all possible functionals on $\{0,1\}^U$? After all, the larger Ω is, the larger the lower bound ρ is. One answer is that this set is huge — triply exponential in n for typical f, and smaller or simpler universes may be easier to handle.

Another answer has to do with the quality of the lower bound. The meta-theorem above would not be too interesting if $\rho(f)$ was small for every f. As we shall see later, for different types of models, it is possible to choose Ω in such a way that the lower bound provided by ρ is tight up to a polynomial, and sometimes even up to a constant factor. In *some* cases we will obtain a converse to the meta-theorem.

Theorem 2 (Meta-Converse) *There is a program (over OP) computing f, which is not much larger than $\rho(f)$.*

The meta-theorem together with its converse yield a characterization of many complexity classes (NL, $\oplus L$, P, mP, NP), in terms of the cover number ρ, that depends on the choice of Ω. We conclude with the ideas behind a typical proof of this converse.

Let a cover $\{< g_1, h_1, \circ_1 >, \ldots, < g_t, h_t, \circ_t >\}$ for Ω_f be given. Unlike a program, these triplets may be an arbitrary collection of unrelated "gates", and our task is to "organize" them into a program.

Now we are given an input $z \in \{0,1\}^n$, and we should compute $f(z)$. Our strategy will be to check if there is a functional F defining z which is not covered by the given cover. Let us consider the two cases.

If $f(z) = 0$, then by the definition of a cover, for every $F \in \Omega_f$, one of the "gates" in the cover will be inconsistent (will violate (3)). On the other hand, if $f(z) = 1$ (i.e. $z \in U$), then consider the "singleton" F_z defined by $F_z(g) = g(z)$. Ω will always contain these "singleton" F_z, and it is easy to see that they are consistent with every possible gate (simply since $F_z(g \circ h) = (g \circ h)(z) = g(z) \circ h(z) = F_z(g) \circ F_z(h)$). Thus we have:

Lemma 1 (Main Lemma) *$f(z) = 1$ iff $\exists\, F \in \Omega$, which defines z and is consistent (i.e. not covered) with the given cover.*

The rest is a program (over OP), that attempts to construct such an F for the given input z. A key observation is that it is sufficient to specify only the value F takes on only few "points" in 2^U, namely the inputs x_i, and the functions $g_j, h_j, g_j \circ_j h_j$ from the cover. The details depend heavily on the model, of course.

3 Historical Annotated Bibliography

3.1 The "Approximation Method"

In 1985, Razborov proved superpolynomial lower bounds on monotone circuit size for the clique and matching functions ([12, 11]). The technique he introduced became known as the "approximation method". Andreev [2] used a varaiant of this method to prove exponential lower bounds for the other NP functions, and stronger results followed in [1, 17].

Much effort was subsequently put in trying to apply the "approximation method" to non-monotone circuits, but all attempts failed. In 1989, Razborov [13] explained these failures. He formalized his "approximation method" and proved that it can never yield even super-linear bounds on non-monotone circuits. In the second part of the same paper, he proposed a generalization of his method (which became known as the "generalized approxiamtion method"). He proved that this method *characterizes* circuit size (up to a polynomial), and thus can potentially be used to prove non-monotone lower bounds.

All the ideas described in the previous section are already present in his paper [13]. Indeed, the "generalized approximation method" is the method of combining computations, (with Ω as the set of all filters (monotone functionals) over 2^U). However, the description of the method in [13] takes the perspective of circuit approximation, and some readers may wonder even now where is approximation taking place in Section 2. The key is to view the consistency requirement (3) on F as an approximation of a gate; we approximate the output of $g \circ h$ by $F(g \circ h)$, even though it should have been $F(g) \circ F(h)$. The errors introduced by this approximation are those F for which $F(g \circ h) \neq F(g) \circ F(h)$ (i.e. those F covered by $< g, h, \circ >$). Thus, Razborov [13] reduced the circuit lower bound problem to a covering problem, in which one looks for the least number of gates that will together eliminate all errors for every $F \in \Omega_f$.

So far so good. But while the general approximation method was potentially powerful enough to prove non-monotone circuit lower bounds, no such bounds followed. Although the task was reduced to solving static set-cover problems, the universe and covering sets in these problems were frightfully huge — triply exponential.

It was thus extremely fortunate that Razborov wrote another paper [14], showing that the potential of the general method can materialize. Another message in this paper is that the method is applicable to other models. In [14], Razborov shows that a somewhat restricted set-cover problem can be associated with non-deterministic (and non-monotone) branching programs. In fact, the associated cover number determines exactly the size of this model for every function. Finally, he considers the Majority function, and proves a super-linear lower bound for it in this model by lower bounding the related cover number.

3.2 The "Topological Approach"

Since the early 1980's, Sipser suggested that the gap between polynomial and exponential complexity in finite circuits is analogous to the gap between countable and uncountable "circuits" in topology. Thus, we should get intuition and ideas for proving finite lower bounds by first proving their infinite analogs.

This approach was very successful in studying constant-depth circuits. The proof in [5] that parity is not in AC^0 can be thought of as a finite version of Sipser's proof [15] that no "countable parity function" is Borel. Similarly, Sipser's proof [16] that the constant-depth hierarchy in AC^0 is strict, was inspired by the classical result (see e.g. [10]) that the Borel hierarchy is strict.

In his 1984 paper [15], Sipser suggested approaching the NP vs. $coNP$ question by studying its infinite analog: the classical result of Lebesque (see e.g. [10]) that the analytic sets are not closed under complement. Sipser provides a new proof of this result, which can hopefully be "finitized". He considers the set T of all countable trees with no infinite branch. T is easily seen to be co-analytic (in fact it is complete for this class). Assume we have an analytic (i.e. countable, non-deterministic) circuit P that allegedly computes T, and assume it correctly accepts all members of T. Sipser proves that the circuit must accept some member of $\overline{T}$ as well, deriving a contradiction, in the following way.

For each member of $t \in T$, fix an accepting witness w_t. This fixes the computation of P on t, namely the values are all wires of P. Now, (using diagonalization), he picks a sequence of points in T $t_1, t_2, \ldots$ which converge to a point $t_\infty \in \overline{T}$. The choice is made such that the associated sequence of accepting computations $P(t_1), P(t_2), \ldots$ converges to an accepting computation for $P(t_\infty)$, a contradicition.

In [15], as well as in his survey [16], Sipser asks for a notion of a "finite limit" that will allow to carry out such an argument for finite computations. The reader should find it easy to match up the concepts in Sipser's proof and in Section 2. In particular, Ω can serve as a notion of a limit, and the chosen F as the converging sequence. Of course, diagonalization should be replaced by proving a lower bound on the cover problem, that will imply F exists if the given circuit is too small.

3.3 Unification

Karchmer [6] was the first to explicitely present the method of fusing computations in essentially the same way described in Section 2, (still for Ω the set of filters over 2^U). He observed that it unifies the approximation method of Razborov and the topological approach of Sipser. Moreover, he pointed out that the construction of a filter $F \in \Omega$ can be viewed as a finitary version of the ultra filter construction used in model theory (see e.g. [4]) [1]. This idea was pushed further by Ben-David, Karchmer and Kushilevitz [3], who showed how standard ultra–filter arguments in Model Theory [4] can be used to simplify Sipser's proof [15], as well as extend it to other sets.

Karchmer's paper [6] also extends the number of computational models captured by the Fusion Method. Besides restating Razborov's result, that choosing Ω as set of filters characterizes P, it contains two other results. One is an observation of Karchmer and Wigderson, that restricting Ω to the set of self-dual filters (those for which exactly one of S, $\overline{S}$ can be in F for any set $S \subseteq U$) characterizes NP.

The second, which is implicitely stated only, is that taking Ω to be all filters again, but weakening the way in which a filter $F \in \Omega$ defines an input, results in a characterization of mP, the class of functions computed by monotone, polynomial-size circuits.

Based on this last result, Karchmer presents Razborov's proof of the monotone clique lower bound [12] within the new method. While the combinatorial arguments are identical to those of [12, 1], the proof is essentailly different, as the construction of the filter F does depend on the small circuit that allegedly computes the clique function. This again, like [14], provides evidence that some cover problems arising from the new method are amenable to analysis.

3.4 Algebraic Variants

In two recent papers [7, 8], Karchmer and Wigderson suggested taking Ω to be the set of all linear functionals over 2^U. The rationale was two-fold. First, there are exponentially fewer linear functionals than monotone functionals, so perhaps the cover problems that arise would be more comprehensible. Second and more important, the algebraic structure may allow the application of techniques from linear algebra to attack these cover problems.

Luckily (I am not kidding!) this choice turned out to be meaningful. In [7] we showed that the cover problem for circuits with polynomial cover number characterize the class (can you guess?) NP. In [8], we restricted these problems as was done in [14], to discover that here polynomial cover number characterizes the class (well? no, not NL. This happens for filters in [14]) $\oplus L$, the logspace analog of $\oplus P$. These are the languages recognized by branching programs of polynomial size that count the number of accepting paths modulo 2.

In [7] we presented several concrete combinatorial algebraic problems, for any of which a non-trivial lower bound will imply a super-linear lower bound on the size of non-deterministic (as well as deterministic) circuits. We were not able to prove such bounds. However, in [8] we succeeded in applying Razborov's techniques from [14], together with linear algebra, to prove the first super-linear lower bound on the size of counting (mod 2) branching programs that compute Majority. The cover problems that arise in this paper suggest an elegant linear-algebraic model of computation we call Span Programs. This model is independently interesting, and we show some connections it has with notions in complexity and cryptogra-

[1] this was undoubtedly known to both Sipser and Razborov

phy.

4 Definitions and Results

4.1 Different straight-line Programs

We will consider a variety of circuits and branching programs, which use different gates and computation modes. It will be convenient to describe all of these in terms of straight line programs.

Four parameters define the computation of the straight line program. A particular choice for these parameters will be later denoted by Δ.

- **Inputs** — the set of input functions
- **Gates** — the set of allowed operations
- **Type** — Circuit/Branching program (BP)
- **Mode** — Deterministic/Non-deterministic

The set of *GATES* we will consider will be either the Boolean operations $\{\wedge, \vee\}$ or the *GF(2)* operations $\{\wedge, \oplus\}$. We use $+$ to denote either $\vee$ or $\oplus$, depending on the choice. A straight line program over a set *GATES* of operations and a set *INPUTS* of input functions is a sequence of $P = (g_1, g_2, \ldots, g_t)$ satisfying either:

1. $g_i \in \mathit{INPUTS}$
2. $g_i = g_j + g_k$ with $j, k < i$
3. $g_i = g_j \wedge g_k$ with $\begin{cases} j, k < i \text{ (circuit computation)} \\ j < i \text{ and } g_k \in \mathit{INPUTS} \text{ (Branching program computation)} \end{cases}$

Let $X = \{x_1, x_2, \ldots x_n\}$ be a set of variables. $\overline{X} = \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n\}$ their negation, and 1 the constant 1 function. Let $Y = \{y_1, y_2 \ldots\}$ be a set of independent (non-deterministic) variables. *INPUTS* will always contain X, and the program P will define a function $f : \{0,1\}^n \to \{0,1\}$ over X in the standard way. If P is deterministic, *INPUTS* will not include Y, and $f = g_t$. For non-deterministic P, *INPUTS* will contain $Y \cup \overline{Y}$, and $f(x) = 1$ iff $\exists y$ s.t. $g_t(x, y) = 1$.

For a given function $f : \{0,1\}^n \to \{0,1\}$ (and with a fixed choice Δ of *INPUTS*, *GATES*, *TYPE* and *MODE*), we let $s_\Delta(f)$ denote the smallest t such that there is a program P of lenght t computing f. As usual, we say that a language $L \subseteq \{0,1\}^*$ is computed in size $s : \mathcal{N} \to \mathcal{N}$ if for every n, $L_n = L \cap \{0,1\}^n$ satisfies $s_\Delta(L_n) \leq s(n)$. The polynomial (non-uniform) complexity $\mathcal{C}_\Delta$ associated with a fixed choice Δ is the set of all languages computed in size $n^{O(1)}$.

4.2 Cover Problems

For $f : \{0,1\}^n \to \{0,1\}$, let $U = f^{-1}(1)$, as well as the $|U| \times n$ matrix whose rows contain the vectors in U. Let 2^U denote the power set of U, endowed with the structure of either the Boolean lattice $\{0,1\}^U$ or the vector space $\mathrm{GF}(2)^U$. For a set $S \subseteq U$ we let $\overline{S} = U \setminus S$, and set $\bar{1}$ to be the all 1's vector of length $|U|$.

Three parameters define the cover problem. A particular choice for these parameters will be later denoted Γ.

- Ω — the set of fusing functionals.
- **Input Definition** — the way in which a functional defines an input.
- **Cover kind** — which pairs of functions are allowed to participate in the cover.

We will consider here three classes Ω of Boolean functionals on the set 2^U.

- Filters $= \{F : \{0,1\}^U \to \{0,1\} \mid$ for all $S \subseteq T \subseteq U, F(S) = 1 \Rightarrow F(T) = 1\}$
- Self Dual Filters (SDF) $= \{F \in \text{Filters} \mid \text{for all}\, S \subseteq U, F(S) \neq F(\overline{S})\}$[2]
- Affine $= \{F \in \mathrm{GF}(2)^U \mid F \cdot \bar{1} = 1 \,\text{and for}\, g \in \mathrm{GF}(2)^U,\ F(g) =< F \cdot U >\}$

The literals $X \cup \overline{X}$ are naturally associated with members of 2^U, namely $x_i(u) = u_i$, $\overline{x}_i(u) = \overline{u}_i$, for all $u \in U$.

We distinguish two ways in which a functional $F \in \Omega$ defines an input $z \in \{0,1\}^n$.

- F *weakly* defines z if $F(x_i) = z_i$ for all $i \in [n]$
- F *strongly* defines z if $F(x_i) = z_i$ and $F(\overline{x}_i) = \overline{z}_i$ for all $i \in [n]$.

Note that if Ω is either a Self Dual Filter of Affine, these two definitions are equvialent!

For a fixed Ω and notion of input definition, we set $\Omega_f = \{F \mid F \,\text{defines}\, z \,\text{with}\, f(z) = 0\}$. For $g, h \in 2^U$, we say that the pair (g,h) *covers* F if $F(g) \wedge F(h) \neq F(g \wedge h)$. The set $\{(g_1,h_1),(g_2,h_2),\ldots,(g_t,h_t)\}$ is a cover of f if every $F \in \Omega_f$ is covered by some pair (g_i,h_i). The cover satisfies that for every $i \in [t]$, $g_i \in 2^U$, and we distinguish three kinds depending on whether the h_i, $i \in [t]$ are restricted to 2^U, $X \cup \overline{X}$, and X respectively.

For a fixed choice Γ of Ω, input definitions and cover kind, we set $\rho_\Gamma(f)$ to be the smallest t for which such a cover of size t exists for f.

[2]for our purposes it is equivalent to demand only that $F(S) \vee F(\overline{S}) = 1$

4.3 Theorems

In Table 1 we give the known pairs of choices (Δ, Γ) of computational model Δ and cover problem Γ such that for every function f, $\rho_\Gamma(f) \le s_\Delta(f)$. For all these choices, the reader should be able to prove the inequality using the intuition in Section 2.

In all cases, we also have an upper bound $s_\Delta(f) \le (\rho_\Gamma(f))^C$, and in some of them, a stronger bound $s_\Delta(f) \le C \cdot \rho_\Gamma(f)$. The constant C is independent of n, and in fact the choice $C = 4$ is satisfied in all theorems.

The proofs of the upper bounds also follow the intuition given in Section 2, but are typically somewhat harder as they are more "model dependent".

	Reference	[13]	[14]	[6]	[6]	[7]	[8]
PROGRAMS (Δ)	**Inputs**	$X \cup \overline{X}$	$X \cup \overline{X}$	$X \cup \overline{X} \cup Y \cup \overline{Y}$	X	$X \cup \{1\} \cup Y$	$X \cup \{1\}$
	Gates	$\{\wedge, \vee\}$	$\{\wedge, \vee\}$	$\{\wedge, \vee\}$	$\{\wedge, \vee\}$	$\{\wedge, \oplus\}$	$\{\wedge, \oplus\}$
	Type	Circuit	BP	Circuit	Circuit	Circuit	BP
	Mode	Det	Det	Nondet	Det	Nondet	Det
COVERS (Γ)	Ω	Filters	Filters	SDF	Filters	Affine	Affine
	Input def	strong	strong	weak	weak	weak	weak
	Cover kind	2^U	$X \cup \overline{X}$	2^U	2^U	2^U	X
RESULTS	$\mathcal{C}_\Delta$	P	NL	NP	mP	NP	$\oplus L$
	$\rho_\Gamma(f) \le s_\Delta(f) \le$	$(\rho_\Gamma(f))^C$	$C \cdot \rho_\Gamma(f)$	$C \cdot \rho_\Gamma(f)$	$(\rho_\Gamma(f))^C$	$C \cdot \rho_\Gamma(f)$	$(\rho_\Gamma(f))^C$
	Some Lower Bounds on ρ_Γ		$\rho_\Gamma(\text{MAJORITY}) = \Omega(\log\log\log^* n)$		$\rho_\Gamma(\text{CLIQUE}) = \exp(\Omega(n^{1/8}))$		$\rho_\Gamma(\text{MAJORITY}) = \Omega(\log\log\log^* n)$

Table 1: Summary of Results

Remarks:

1. Similar results can be obtained for arithmetic computation over arbitrary rings R, considering appropriate functionals over R^U (as is done in [8]).

2. Some of the results in the table appear only implicitly (or very implicitly) in the given references.

3. Note that to define the non-deterministic and counting classes NL and $\oplus L$ of branching programs one uses the deterministic mode.

4. The classes $coNP$ and $coNL$ can easily be defined by taking $U = f^{-1}(0)$.

5 Discussion

The Fusion Method differs from other lower bound methods in circuit complexity in several aspects, that we enumerate below.

- **Static**: Almost every existing lower bound is obtained by introducing some progress measure, and showing that the model cannot make too much progress too fast in computing the given function. This is present in all top-down and bottom-up proofs. It is also consistent with our view of computation as a dynamic process. In contrast, the Fusion Method converts the computation into a static cover problem, considering the program as a completely unstructured and unordered collection of "abstract gates".

- **Versatile**: The Fusion Method can in principle be used to prove lower bounds on a variety of computational models; Boolean and Arithmetic, monotone and non-monotone, deterministic and non-deterministic.

- **Tight**: As the cover number is a lower bound on size, the method is sensitive enough to provide super-linear lower bounds.

- **Complete**: On the other hand, since (in the examples in the table) the cover number is polynomially related to size, every super-polynomial lower bound on size can be proven via this method. This allows for the characterization of complexity classes.

- **Assymetric**: A unique feature of the Fusion Method is that the 'ones' and 'zeros' of the function play entirely different roles; the 'zeros' are defined by subsets of 'ones'.

6 Further Research

- **Lower Bounds**: Clearly the most important task is to prove lower bounds on the various cover problems, mainly those that apply to non-monotone circuit size. These often can be presented as concrete and elegant combinatorial/algebraic problems (e.g. see [7, 8], and the next section). Still, the choice of function f is problematic, and we may have to consider functions different than our favorite combinatorial and graph theoretic ones.

As we have little experience and techniques to deal with the covering problems suggested by the Fusion Method, it will be of interest to prove new and reprove old lower bounds even in the weaker (monotone, branching program) models in this framework, as was done in [6].

- **Find New Characterizations**: It will be interesting to find cover problems that characterize other classes, such as L. It is also interesting to characterize P using other choices of Ω (as is possible for NP).

- **Explain Existing Characterizations**: We have no complete understanding of why particular choices of Ω characterize particular classes. Why does the choice of Ω as Filters give the deterministic class P for circuits and the non-deterministic class NL for branching programs? Why choosing Ω=Affine gives the non-deterministic class NP for circuits and the counting class $\oplus L$ for branching programs? Why do both Affine and Self Dual Filters define NP? Can we characterize all choices of Ω that will define a particular class (say NP)? In fact the existing characterizations suggest comparing complexity classes by comparing the sets Ω defining them, which may be "function independent". The NP vs. $coNP$ vs. P question can already be studied in this light.

7 Concrete Open Problems

In this section I present three cover problems that arise from the Fusion Method. The selection of functions and covers types is somewhat arbitrary, and was intended to provide some variety.

We need some notation. For a field K and vector space K^t over it, denote by $\bar{1}$ the all 1's vector in the space (after a fixed choice of basis). For $g, h \in K^t$ let $g \cdot h \, (\in K^t)$ denote their component–wise product, and $< g, h >$ $(\in K)$ their inner product. Finally let $[m] = \{1, 2, \cdots, m\}$.

1. Here $K = \Re$, the reals. Let $A_1, A_2, \cdots, A_t$ be an enumeration of the $n \times n$ adjacency matrices of maximal bipartite graphs having no perfect matching. For $\alpha \in \Re^t$ let $A_\alpha = \sum_{i=1}^{t} \alpha_i A_i$, and set $PER = \{\alpha \mid \, < \alpha, \bar{1} >= 1 \, , \, permanent(A_\alpha) \neq 0\}$.

   Say that a pair $g, h \in \Re^t$ *covers* $\alpha \in \Re^t$ if $< g, \alpha >< h, \alpha > \neq < (g \cdot h), \alpha >$.

   Problem: Bound the smallest number of pairs covering PER.

   Remark: This number is a lower bound on the arithmetic complexity of the permanent function.

2. Here $K = GF(2)$. Let $A_1, A_2, \cdots, A_t$ be an enumeration of all nonsingular $n \times n$ matrices over $GF(2)$. For $\alpha \in GF(2)^t$ set $DET = \{\alpha \mid \, < \alpha, \bar{1} >= 1 \, , \, det(A_\alpha) = 0\}$.

   A 3-partition T_1, T_2, T_3 of $[t]$ *covers* $\alpha \in GF(2)^t$ if for every $j \in [3]$ the characteristic vector τ_j of the part T_j satisfies $< \tau_j, \alpha >= 1$.

Problem: Bound the smallest number of 3-partitions covering DET.

Remark: This number is a lower bound on the Boolean complexity of the determinant function (and thus also of matrix multiplication).

3. Let $\chi_i : \{0,1\}^n \to [k]$, for $i \in [n]$, be a collection of n $k-$colorings of the n-dimensional cube.

   For a triple of distinct vectors $X, Y, Z \in \{0,1\}^n$, say that coordinate $i \in [n]$ is *interesting* if not all three vectors agree in this coordinate. Say that χ_i is *proper* on this triple if the three colors $\chi_i(X), \chi_i(Y), \chi_i(Z)$ are distinct. We now define the collection of colorings *good* if every for every triple of vectors there is an interesting coordinate i for which χ_i is proper on this triple.

   Problem: Bound the smallest number k for which such a good collection exists.

   Remarks: This is (indirectly) related to the size of branching programs of the $\oplus L$ type computing certain linear codes. This problem was suggested by Karchmer and Wigderson [9]. They noted that Proposition 1 in [8] implies that the number of colors k has to increase with n. They also proved the tight lower bound $k = \Omega(n)$ in the special case that the colorings χ_i are monotone (i.e. $X < Y$ imply $\chi_i(X) \leq \chi_i(Y)$). This lower bound holds even if need to properly color only ordered triples ($X < Y < Z$). On the other hand, we don't even know that more than constant number of colors is needed for ordered triples if we remove the monotonicity condition.

Acknowledgements

I am greatful to M. Karchmer, N. Nisan, A. Razborov and M. Sipser for their comments on an earlier version of this paper.

References

[1] A. Alon and R. Boppana. The monotone circuit complexity of boolean circuits. *Combinatorica*, 7(1):1–22, 1987.

[2] A.E. Andreev. On a method for obtaining lower bounds for the complexity if individual monotone functions. *Dokl. Akad. Nauk USSR*, 282(5):1033–1037, 1985. English translation in Soviet Math. Dokl. 31(3) (1985) 530-534.

[3] S. Ben-David, M. Karchmer, and E. Kushilevitz, 1992. private communication.

[4] C. C. Chung and H. J. Keisler. *Model Theory* North–Holland, 1990.

[5] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. *Math. Syst. Theory*, 17:13–27, 1984.

[6] M. Karchmer. On proving lower bounds for circuit size. In *Proceedings of the 8th Annual Symposium on Structure in Complexity Theory*, 1993.

[7] M. Karchmer and A. Wigderson. Characterizing non-deterministic circuit size. In *Proceedings of the 25th STOC*, 1993.

[8] M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the 8th Annual Symposium on Structure in Complexity Theory*, 1993.

[9] M. Karchmer and A. Wigderson. Private communication.

[10] Y. Moschovakis. *Descriptive Set Theory*. North–Holland, 1980.

[11] A. Razborov. A lower bound on the monotone network complexity of the logical permanent. *Mat. Zametki*, 37(6):887–900, 1985. English translation in Math. Notes 37(6) (1985) 485-493.

[12] A. Razborov. Lower bounds on the monotone complexity of some boolean functions. *Dokl. Akad. Nauk USSR*, 281(4):798–801, 1985. English translation in Soviet Math. Dokl. 31 (1985) 354-357.

[13] A. Razborov. On the method of approximation. In *Proceedings of the 21st ACM Symposium on Theory of Computing*, pages 167–176, 1989.

[14] A. Razborov. Lower bounds on the size of switching-and-rectifier networks for symmetric Boolean functions (in Russian). *Mathematical Notes of the Academy of Sciences of the USSR*, 48(6):79–91, 1990.

[15] M. Sipser. A topological view of some problems in complexity theory. *Colloquia Mathematica Societatis János Bolyai*, 44:387–391, 1984.

[16] M. Sipser. The history and status of the P versus NP question. In *Proceedings of* 24^{th} *STOC*, pages 603–618, 1992.

[17] É. Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8:141–142, 1988.