COMPUTATIONAL LIMITATIONS FOR SMALL DEPTH CIRCUITS

by

Johan Torkel Håstad

B. S., Stockholm University
(1981)
S. M., Uppsala University
(1984)

Submitted to the Department of
Mathematics in Partial Fulfillment
of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1986

© Massachusetts Institute of Technology 1986

Signature of Author: Signature redacted 5/2/86
Department of Mathematics
May 1, 1986

Certified by: Signature redacted 5/2/86
Prof. Shafi Goldwasser
Thesis Supervisor

Certified by: Signature redacted
Prof. Willem Malkus
Chairman, Applied Mathematics Committee

Accepted by: Signature redacted
Prof. Nesmith Ankeny
Chairman, Departmental Graduate Committee

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

AUG 04 1986

LIBRARIES

ARCHIVES

COMPUTATIONAL LIMITATIONS FOR SMALL DEPTH CIRCUITS

by

Johan Torkel Håstad

Submitted to the Department of Mathematics on
May 2, 1986 in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy in Mathematics

ABSTRACT

We study the properties of small depth circuits. The main results are establishing lower bounds on the size of small depth circuits computing functions such as parity and majority. The main tool used in the proof of the lower bounds is a lemma, stating that any AND of small fanout OR gates can be converted into an OR of small fanout AND gates with high probaility when random values are substituted for a randomly chosen subset of the variables

We also prove that there are functions which have linear size circuits of depth k but which require exponentially large circuits for depth $k-1$. In addition we prove that if majority is computable by polynomial size constant depth gates containing AND, OR and parity gates then at least $\Omega((\log n)^{\frac{3}{2}})$ parity gates are needed.

We give some connections to relativized complexity by completing previous proofs that the above lower bound results imply the existence of an oracle A such that $PSPACE^A \neq PH^A$ and $\Sigma_i^{p,A} \neq \Sigma_{i-1}^{p,A}$ for all i.

Finally we exhibit permutations which are computable in NC^0, but are P-complete to invert.

Thesis Supervisor: Shafi Goldwasser

Title: Assistant professor

ACKNOWLEDGMENTS

First of all I would like to thank my adviser Shafi Goldwasser. She has been very supportive and has always discussed any type of problems with interest and a positive attitude. Silvio Micali's everlasting optimism and sometimes justified criticism has been very helpful. I thank Lennart Carleson for getting me interested in complexity theory and his constant support. I have had several fruitful discussions with Mike Sipser who together with Tom Leighton I thank for serving on my thesis committee. Tom also functioned well as my course adviser. Ravi Boppana has always answered or discussed any questions and I am grateful for his cooperation and permission to include Lemma 4.7. Oded Goldreich is of course very special and has the good property of commenting on early drafts.

MIT has been a stimulating environment especially because of the people mentioned above and among others Bill Aiello, Val Brezu-Tannen, Benny Chor, Charles Leiserson, Gary Miller and Ron Rivest.

Outside MIT I would like to mentioned Jeff Lagarias who has inspired a lot of work not included in this thesis.

My stay at MIT would of course have been impossible without the financial support I have received from MIT grants in particular NSF grant 8509905DCR and the IBM graduate student fellowship I have enjoyed the last year.

Finally my deepest thanks to Ingrid for being there.

CONTENTS

1. Introduction

Proving lower bounds on the amount of resources needed to compute specific functions is one of the most active branches of theoretical computer science. One of the ultimate goals is of course to resolve the $NP \neq P$ question. It seems, however, that achieving this goal is still quite far off, and that new techniques must be developed before significant progress is made in resolving this question. Several restricted models of computation have been studied in order to develop lower bound proof techniques and gain a better understanding of the problem.

In this thesis we will study *Boolean circuits.* A Boolean circuit contains AND gates, OR gates and negations configured to compute a Boolean function. It is important to remember that a single function is computed by a family of circuits, one for each length of the input. Different assumptions can be made about how hard it is to construct a member of the family for a given length of the input. We will adopt the model of nonuniform circuits which means that we will make no such assumptions.

The size of a circuit is defined to be the number of gates it contains, while the depth is defined to be the longest path from input to output. It is well known that if a problem can be solved in time $T(n)$ on a Turing Machine, then it can be solved by a family of Boolean circuits of size $O(T(n) \log T(n))$ [PF]. Thus problems in P have polynomial size circuits. Proving lower bounds on the size of general circuits computing a function yields corresponding lower bounds for the time to compute it on a Turing Machine.

Circuits can also be used to model parallel computation. In this case we interpret the size of the circuit as the number of processors computing in parallel, and the depth as maximum computation time of any processor.

Even though circuits are more powerful than Turing Machines they do not contain hard to control features like moving heads and changing states, and hence they seem better suited to lower bound proofs. Significant progress has recently been made for proving lower bounds in two restricted models of Boolean circuits. The first model is the model of *small*

depth circuits. These circuits have the complete instruction set of negations, AND and OR gates which have arbitrarily many inputs. However the depth is restricted to be small.

The subject of this thesis is the study of small depth circuits. The main result is the development of techniques for proving exponential lower bounds on the size of small depth circuits. The techniques are quite powerful and can be used to show almost optimal lower bounds on the size of small depth circuits computing several different functions such as parity and majority. Our results can be viewed as a step forward in understanding what may cause functions to be difficult to compute by small depth circuits.

Another example of a restricted model of computation is the model of monotone circuits. Monotone circuits contain just AND and OR gates, but no negations. This restricts the functions which can be computed to be monotone, i.e. changing an input from 0 to 1 cannot change the value of the function from 1 to 0. For example, monotone circuits can compute the majority function but not the parity function. Recently Razborov [R] proved superpolynomial lower bounds on the size of monotone circuits computing the NP-complete clique function. The results were improved to give exponential lower bounds by Alon and Boppana [AlBo]. Andreev [An] independently obtained exponential lower bounds for other NP-functions.

Interestingly, Razborov [R] proved the same superpolynomial lower bounds for detecting whether a graph has a perfect matching. Since this second problem is well known to be in P this result shows that even when a function can be computed by monotone circuits, these circuits may have to be significantly larger than general circuits computing the same function.

The recent results for small depth circuits and monotone circuits show that it is possible to prove exponential lower bounds in nontrivial cases. Although, it is doubtful that will apply to the $P \neq NP$ question, the results can be taken as an optimistic sign for the quest of lower bound proofs in the general circuit model.

1.1 Lower bounds for small depth circuits.

The problem of proving lower bounds for small depth circuits has attracted the attention of several researchers in the field. Functions considered have been simple functions like parity and majority. Parity is defined to be the sum modulo two of the inputs and majority is defined to be 1 when at least half the inputs are 1.

Furst, Saxe and Sipser [FSS] were the first to give superpolynomial lower bounds. They proved that circuits of depth k computing parity was of size $\Omega(n^{\log^{(3(k-2))} n})$, where $\log^{(i)} n$ denotes the logarithm function iterated i times. One of the important contributions of their paper was to introduce the concept of random restrictions. These have played a major role in subsequent papers. Ajtai [Aj] independently proved the slightly stronger bounds $\Omega(n^{c_k \log n})$.

To illustrate the difference in the power of depth k circuits versus depth $k-1$ circuits Sipser [Si] introduced a sequence of functions f_k^n which were computable by linear size circuits of depth k but seemed to require large circuits of depth $k-1$. Sipser quantified this by showing superpolynomial lower bounds for the size of depth $k-1$ circuits computing f_k^n.

Next, in the attempt to prove exponential lower bounds, the simpler case of monotone small depth circuits was studied. Valiant [V] proved that the clique problem needed exponential size circuits when the circuit depth was restricted to 3. Boppana [Bo1] proved that depth k monotone circuits computing majority had to be of size $2^{\Omega(n^{\frac{1}{k-1}})}$. Klawe, Paul, Pippenger and Yannakakis [KPPY] proved similar bounds for f_{k+1}^n.

The first breakthrough in proving exponential lower bounds without restricting the circuits to be monotone was obtained by Yao [Y2] who proved that depth k circuits computing parity had to be of size $\Omega(2^{n^{\frac{1}{4k}}})$. He also stated exponential lower bounds for the functions f_k^n. These results also have interesting consequences for relativized complexity which will be described in section 1.3.

1.2 Our results in lower bounds.

In this thesis we prove that depth k circuits computing parity have to be of size $2^{c_k n^{\frac{1}{k-1}}}$. The c_k's tend toward $\frac{1}{10}$. This is close to optimal, as there are depth k circuits of size $n2^{\frac{1}{k-1}}$ which compute parity. The key to the lower bound proof is the following technical lemma. Let a small gate denote a gate with few inputs.

Lemma: Given a depth two circuit which is an AND of small OR gates, then if random values are substituted for a randomly selected subset of the variables, it is possible to write the resulting induced function as an OR of small AND gates with very high probability.

Using this lemma lower bounds for circuits computing parity are proved by induction over the depth k.

The idea of giving random values to some of the variables was first introduced in [FSS] and weaker versions of our main lemma were proved in [FSS] and [Y2]. In [FSS], the probability of the size not increasing too much was not proved to be exponentially small and Yao only proved that the resulting OR of small ANDs was in a technical sense a good approximation of the original function. This fact significantly complicated the rest of the proof. Also, Yao did not obtain the sharp estimates for the probability of failure. Since we get almost optimal lower bounds for the size of parity circuits, our estimates are tight up to a constant.

Our nearly optimal results for the size of parity circuits imply that a polynomial size circuit computing parity has to have depth $\frac{\log n}{\log \log n} - O(\frac{\log n}{(\log \log n)^2})$. The best previous lower bounds for the depth of polynomial size parity circuits were $\Omega(\sqrt{\log n})$ by Ajtai [Aj].

We also prove that the functions f_k^n require circuits of size $2^{c_k n^{\frac{1}{k^2+k-1}}}$ when the depth is restricted to $k-1$. The proof is quite similar to the proof for the parity function. The main difference is that we are forced to choose another probability distribution for distributing values to some of the variables.

1.3 Small depth circuits and Relativized Complexity.

Because of our inability to prove lower bounds for functions and hence to separate complexity classes, a significant amount of attention has been spent on relativized complexity. This is a concept which originated in recursion theory. As a computational aid the Turing Machine is given the possibility to ask questions of the type: Is $x \in A$? Here x is an arbitrary string and A is the *oracle* set which remains fixed during the computation. It is possible to define the usual complexity classes relative to this oracle. Two important questions concerning these complexity classes can be resolved using lower bounds for small depth circuits.

Furst, Saxe and Sipser [FSS] proved that subexponential lower bounds (more precisely $\Omega(2^{(\log n)^i})$ for all i) for any constant depth circuit computing the parity function would imply the existence of an oracle separating PSPACE from the polynomial time hierarchy (PH=$\bigcup_{i=1}^{\infty} \Sigma_i$). Yao [Y2] was the first to prove sufficiently good lower bounds to obtain $PSPACE^A \neq PH^A$ for a certain oracle A. In section 7 we give a full proof of the correspondence between the two problems.

Cai [Ca] extended Yao's methods to prove that a *random oracle* separated the two complexity classes with probability 1. Namely, the set $\{A \mid PSPACE^A \neq PH^A\}$ has measure 1 in the natural measure.

The heart of the proof of Cai's result is to prove that any small circuit cannot compute parity correctly for significantly more than half of the inputs. We will address the question on what fraction of the inputs a small circuit can compute parity in section 8.

In [Si], Sipser presented the theorem that lower bounds of $\Omega(2^{(\log n)^i})$ for all i for the size of depth $k-1$ circuits computing the previously mentioned functions f_k^n would imply the existence of an oracle separating the different levels within the polynomial time hierarchy. His proof was very sketchy. We give a complete proof in section 7. The lower bounds claimed by Yao give the first oracle achieving this separation. Our bounds are also

sufficient. An interesting open question is to decide whether a random oracle achieves this separation. We conjecture that this is the case.

1.4 Is Majority harder than Parity?

The lower bounds proved for parity are also valid for majority. An interesting question is which of the two is harder. It is not hard to see that given a gate computing majority, one can construct a constant depth circuit computing parity. However the converse is not known to hold. It has been conjectured that parity gates do not help. We prove that constant depth polynomial size circuits computing majority that contain parity gates need at least $\Omega((\log n)^{\frac{3}{2}})$ parity gates. This can be taken as very weak evidence in favor of the conjecture.

1.5 Related Results

As mentioned earlier the small depth circuit model can be viewed of as a model of parallel computation. Our results then imply lower bounds on the time to compute a function (e.g parity) given a certain number of processors. In the parallel model corresponding to small depth circuits the processors can read arbitrarily many memory locations but they are only able to compute either the AND or OR of what they have read. Another model where the processors can do arbitrary computations but are restricted to read only one memory location has been studied by Beame [Be]. Surprisingly our main lemma plays a crucial role in some of his arguments.

1.6 Outline of thesis.

In section 2 we give basic definitions and some properties of small depth circuits. In section 3 we give some background and intuition for the main lemma which is proved in section 4. Section 5 applies the main lemma to get lower bounds for circuits computing parity. In section 6 we introduce the functions f_k^n and prove lower bounds on depth $k-1$ circuits computing them. This amounts among other things to reprove the main lemma in

a different setting. Section 7 contains the applications to relativized complexity and section 8 investigates on what fraction of the inputs small constant depth circuits can compute parity correctly. Finally in section 9 we discuss the problem of computing majority when parity gates of arbitrary fanin are allowed.

2. Small Depth Circuits

2.1 Computational Model

We will be working with unbounded fanin circuits of small depth. A typical example looks like this.

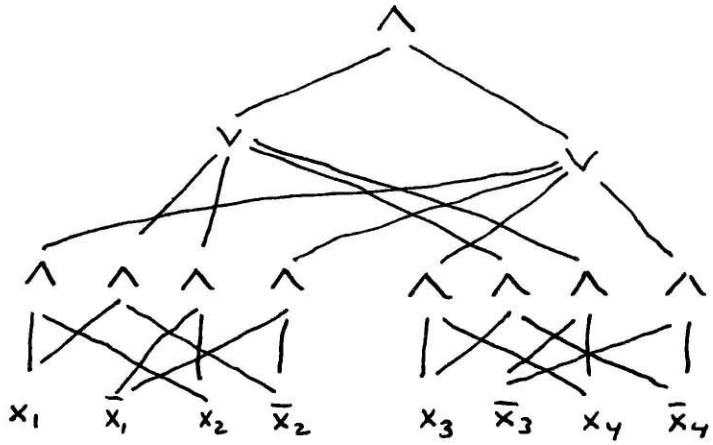

Figure 1

Without loss of generality we can assume that negations occur only as negated input variables. If there are negations higher up in the circuit we can move them down to the inputs using DeMorgan's laws. This procedure at most doubles the size of the circuit. Observe that we have alternating levels of AND and OR gates, since two adjacent gates of the same type can be collapsed into one gate.

The crucial parameters for a circuit is the depth and the size. *Depth* is defined as the length of the longest path from an input to the output and can also be thought of as

the number of levels of gates. For instance the depth of the circuit in figure 1 is 3. *Size* is defined to be the total number of AND/OR gates and the circuit in figure 1 is of size 11. The *fanin* of a gate is defined as the number of inputs to it. We put no restriction on the fanin of the gates in our circuits. We will however be interested in the *bottom fanin* of a circuit which is defined as the maximum fanin for any gate on the lowest level of the circuit and hence has variables as inputs.

2.2 General notation

The following notation will be used through out the thesis. The letter n will always denote the number of inputs to a function. The set of all finite strings from the alphabet $\{0,1\}$ is denoted by Σ^*. Strings are denoted by lower case letters. The i'th bit of a string x is denoted by x_i. All logarithms used in the paper are of the base 2.

2.3 Smallest size circuits for parity

Eventually in this thesis we will prove lower bounds on the size of small depth circuits computing parity. In this section, we present constructions of circuits whose size achieves the best upper bounds known. This construction seems to belong to the folklore of the subject.

We start by a simple observation.

Lemma 2.1: *Depth 2 circuits computing parity are of size* 2^{n-1}. *The circuit can be written either as an AND of ORs or an OR of ANDs.*

Proof: In the OR of AND case take one AND corresponding to each way of making parity 1. There are precisely 2^{n-1} such ANDs. To make it into an AND of ORs is equally easy. A few minutes reasoning will convince the reader that the presented construction is the best possible. The optimality is due to Lupanov [Lu]. ∎

In general we will have.

Theorem 2.2: *There are depth* k *circuits computing parity of size* $O(n^{\frac{k-2}{k-1}}2^{n^{\frac{1}{k-1}}})$. *The*

output gate can be either an AND or an OR gate.

Proof: A natural idea is to divide the inputs in $n^{\frac{1}{2}}$ groups of $n^{\frac{1}{2}}$ variables each. First compute the parity of each group in depth 2 and size $2^{n^{\frac{1}{2}}}$ and then compute the parity of the $n^{\frac{1}{2}}$ outputs using another depth 2 and size $2^{n^{\frac{1}{2}}}$. This would lead to a circuit of depth 4 and size $O(n^{\frac{1}{2}}2^{n^{\frac{1}{2}}})$.

We have not used the full force of Lemma 2.1. Assume that the top circuit is an AND or ORs. If we let the lower circuits to be OR of ANDs it seems like we get two adjacent levels of OR gates which we then could collapse and decrease the depth to 3. This is not quite true since the top circuit also needs the negation of the outputs of the lower circuits as inputs. The way around this is to have two copies of the lower circuits, one which is an OR of ANDs and one which is an AND of ORs. The latter is used when the input to the top circuit is negated and otherwise we use the first. In both cases we get adjacent levels of OR gates and we can decrease the depth to 3.

To get the general case make a parity tree of depth $k-1$ and fanout $n^{\frac{1}{k-1}}$ and for each node in the parity tree we make on copy of it as an AND of ORs and one copy as an OR of ANDs. By every time choosing the suitable copy it is possible to make the resulting circuit have depth k.

In section 8 we will investigate how well smaller circuits can compute parity. We know that there are no small circuits which always compute parity. On the positive side we have.

Theorem 2.3: *There are circuits of depth d and size $n2^s$ which compute parity correctly for a fraction $\frac{1}{2}+2^{-\frac{n}{s^{d-1}}}$ of the inputs.*

Proof: Divide the inputs into $\frac{n}{s^{d-1}}$ sets of each s^{d-1} variables. Compute the parity of each such set by a circuit C_i of size $s^{d-1}2^s$ and depth d. Let each C_i have an OR gate as output gate. The final circuit C will be the OR of the circuits C_i. The depth of this circuit is still d and it has the prescribed size.

If C output 0 it has computed parity correctly since the parity of all the subsets are 0.

This happens for a $2^{-\frac{n}{d-1}}$ fraction of the inputs. This is the only case where the circuits is correct when the parity of the input is 0. On the other hand the circuit is always correct when the parity of the input is 1. This is because one of the subsets must have parity 1. We conclude that C has the desired advantage on parity. ∎

2.4 Some related Complexity Classes.

Let us relate to some standard definitions of Circuit Complexity. In particular let us mention let us mention the complexity class NC and AC defined in [Co]. We have $NC = \bigcup_{i=0}^{\infty} NC^i$ where NC^i is defined to be languages which are recognized by uniform constant fanin circuits of polynomial size and depth $O((\log n)^i)$. In the same way $AC = \bigcup_{i=0}^{\infty} AC^i$ where AC^i is defined to be languages which are recognized by uniform unbounded fanin circuits of polynomial size and depth $O((\log n)^i)$. Clearly $NC^i \subset AC^i \subset NC^{i+1}$. The only strict inclusions that are known are $NC^0 \neq AC^0 \neq NC^1$. The first strict inclusion is trivial since AC^0 functions might depend on all inputs while this is impossible for NC^0 functions. The second result was first proved by [FSS] and follows from the lower bounds for parity circuits since parity is in NC^1. We will see in the next section that it is possible to do nontrivial things even in NC^0.

2.5 How hard is it to invert NC^0 permutations?

The class NC^0 is the class of circuits of constant fanin and constant depth. This means in particular that the output can only depend on a constant number of inputs. It seems that nothing really interesting could be done in this model. However consider the following question. Given a permutation which can be computed by an NC^0 circuit how much effort must be spent on inverting this function? Boppana and Lagarias [BL] proved that there are permutations which were computable in NC^0 but whose inverses were as hard to compute as parity. These permutations can be said to be oneway since their inverses are much harder to compute than the permutation itself. Barrington [Bar]

gave another example of a oneway function which was computable in AC^0, but computing its inverse was LOGSPACE-complete. We prove the following stronger result.

Theorem 2.4: *There is a uniform family of NC^0 permutations which are P-complete to invert.*

Proof: We will reduce the problem of evaluating a straight line program to the problem of inverting an NC^0 permutation. Since the former problem is well known to be P-complete [La] the latter will be P-hard. We will use the term P-complete to mean P-complete under LOGSPACE-reductions. Thus if a P-complete problem is in LOGSPACE every problem in P is in LOGSPACE. In a similar manner a problem is defined to be P-hard if it has the above property but is not known to be in P.

Let us set up some notation for the straight line program. The program contains a sequence of variables which are either defined as a part of the input or in terms of two previously defined variables. To make this formal let i_k and j_k be two indices which are less than k and let f_k be arbitrary functions of two Boolean inputs to one Boolean output.

Using this notation we define an instance of straight line program.

INPUT: Boolean variables $x_1, x_2, \ldots x_n$

PROGRAM: $x_k = f_k(x_{i_k}, x_{j_k}), \quad k = n+1, n+2, \ldots m$

OUTPUT: Value of x_m.

We will reduce this problem to the question of inverting an NC^0 permutation. Let us denote the permutation by g. It will be defined from $\{0,1\}^m$ to $\{0,1\}^m$ where m is the number of variables occurring in the straight line program. Let $z_1, z_2, \ldots z_m$ denote the input bits and $g_1, g_2 \ldots g_m$ the output bits. Let $\oplus$ be exclusive or.

Then,

$$g_k(z) = z_k \quad k = 1, 2, \ldots, n$$

$$g_k(z) = z_k \oplus f_k(z_{i_k}, z_{j_k}) \quad k = n+1, n+2, \ldots m$$

where i_k, j_k and the functions f_k are the same as in the straight line program.

Let us establish that the reduction is correct.

Fact 1 g is a permutation.

We need only show that g is onto. Given $y \in \{0,1\}^m$ find $z \in \{0,1\}^m$ such that $g(z) = y$ by solving for $z_1, z_2, \ldots z_m$ in increasing order. This can be done since the equations can be written $z_k = y_k \oplus f(z_{i_k}, z_{j_k})$.

Fact 2 The m'th bit of $g^{-1}(x_1, x_2 \ldots x_n, 0, \ldots, 0)$ is the output of the straight line program.

Solving for this input as described above performs the computation of the straight line program.

Fact 3 The reduction from straight line programs to permutations is effective and g is computable by NC^0 circuits.

The reduction is trivial computationally since it just replaces equality signs by $\oplus$. The second part of Fact 3 follows from the fact that any function that only depends on a constant number of inputs can be computed in NC^0.

To get the theorem we need that there is a uniform family of instances of the straight line program problem which are hard to solve. One way to see this is as follows.

Take a Turing machine which solves the problem of evaluating straight line programs. Now take the computation tableau of this machine and convert it into a straightline program. Evaluating this straightline program for different inputs is P-complete and thus inverting the corresponding NC^0 permutation is also P-complete. ∎

We know ([BM], [GM], [Y1], [Le]) that in the sequential setting the existence of oneway functions implies the existence of good cryptosystems.

There are two obvious obstacles to using the present results to construct parallel cryptosystems.

The first problem is that the function needs to be hard to invert on a random input.

This is not quite achieved since even if we start with a straight line program which is hard to compute for a random input it is not necessarily true that the corresponding NC^0 permutation is hard to invert for random outputs. This is so as our reduction only maps to values of the permutation whose last $m - n$ bits are 0.

The second problem is that the reductions from oneway functions to cryptographic generators is sequential i.e. even if the oneway function is easy to compute in parallel the resulting cryptographic generator will require large parallel time. For a discussion of "parallel cryptography" we refer to Reif and Tygar [RT].

We have proved that there is a sequence of uniform NC^0 circuits which are P-complete to invert. An interesting open question is whether inverting every NC^0 permutation is in P.

3. Outline of Lower Bound Proof

Having established that even very limited circuits may compute reasonably complicated functions let us see how to prove lower bounds for small depth arbitrary fanin circuits. Many of lower bound proofs cited in the introduction ([FSS],[Y2] and the present paper) have the same outline. The proofs are by induction and proceed as follows.

(1) Prove that parity circuits of depth 2 are large

(2) Prove that small depth k parity circuits can be converted to small depth $k-1$ parity circuits.

Of these two steps the first step is easy and tailored for the parity function. It is the comment about optimality in the construction of Lemma 2.1. The second step is much more difficult and here lies the difference between the papers. The basic idea for doing this lies in the fact that every function can be written either as an AND of ORs or as an OR and ANDs. To give an idea of (2) assume that $k = 3$ and we have the following depth

3 circuit.

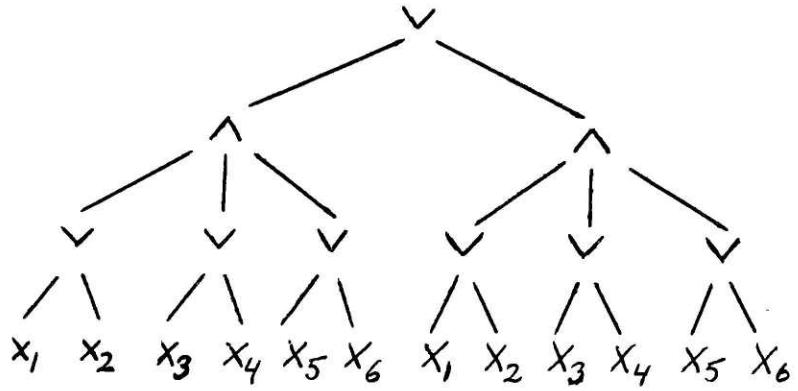

Figure 2

Take any gate at distance two from the inputs. It represents a subcircuit of depth 2. In this case this circuit will be an AND of ORs. Now observe that any function can be written either as an AND or ORs or as and OR of ANDs. Thus we can change this depth 2 circuit to and OR of ANDs which computes the same function. Thus we have the following circuit computing the same function.

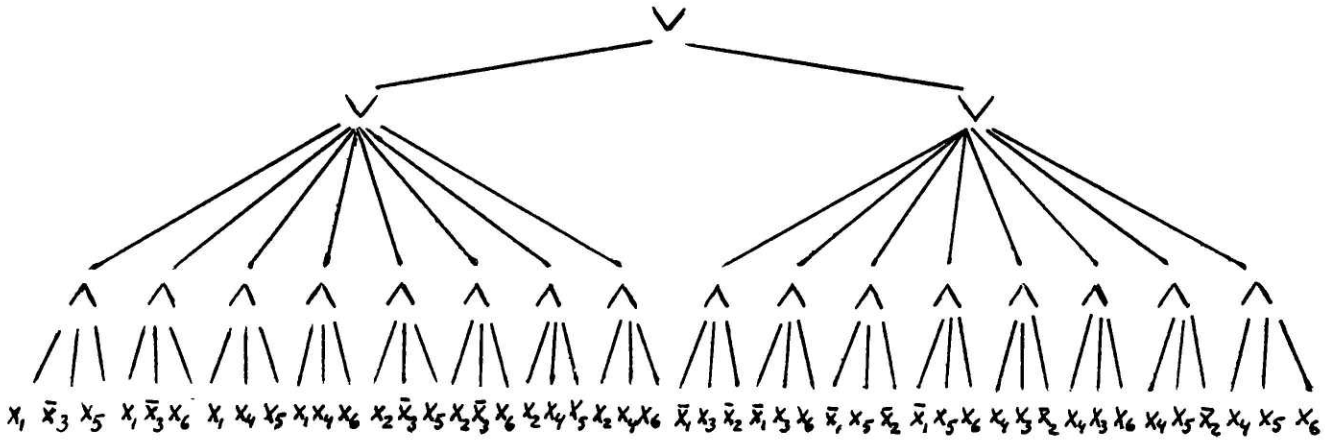

Figure 3

Observe that we have two adjacent levels consisting of OR gates.

These two levels can be merged to one level and we get the following circuit of depth 2.

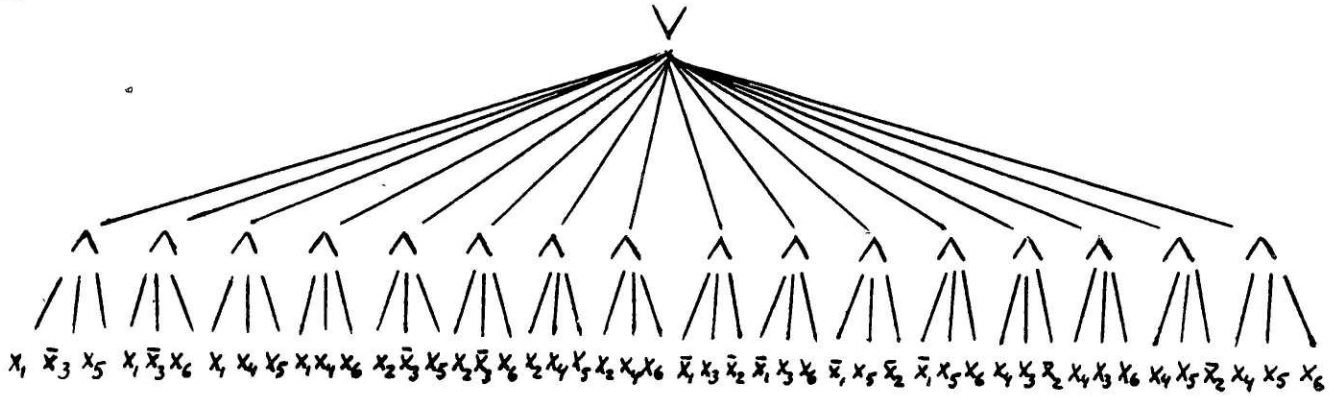

Figure 4

However doing this we run into one problem. When we convert an AND of ORs to an OR of ANDs the size of the circuit will in general increase considerably. Thus we have converted a *small* depth k circuit to a *large* depth $k-1$ circuit and hence we fail to achieve (2).

3.1 Restrictions

The way around this problem was introduced in [FSS] and works as follows. If we assign values to some of the variables we can simplify the circuit. In particular if we assign the value 1 to one of the inputs of an OR gate, the output of the OR gate will be 1 no matter what the other inputs are. In the same way we need only know that one of the inputs to an AND gate is 0 to decide that it outputs 0. This means that the value of any specific gate on the bottom level can be forced by assigning a suitable value to one of its inputs. However there are many more gates than inputs and we have to do something more sophisticated. Let us first make formal what we mean by fixing some variables.

Definition: A restriction ρ is a mapping of the variables to the set $\{0, 1, *\}$.

$\rho(x_i) = 0$ means that we substitute the value 0 for x_i

$\rho(x_i) = 1$ means that we substitute 1

$\rho(x_i) = *$ means that x_i remains a variable.

Given a function F we will denote by $F\lceil_\rho$ the function we get by doing the substitutions prescribed by ρ. $F\lceil_\rho$ will be a function of the variables which were given the value $*$ by ρ.

Example: Let $F(x_1, x_2, x_3, x_4, x_5) =$ majority of the variables and let $\rho(x_1) = 1, \rho(x_2) = *, \rho(x_3) = *, \rho(x_4) = 1$ and $\rho(x_5) = *$. Then $F\lceil_\rho(x_2, x_3, x_5) =$ at least one of x_2, x_3 and x_5 is 1.

The following simple observation will be important when we consider circuits computing parity.

Observation: Parity$\lceil_\rho$= Parity or the negation of Parity.

As pointed out above we could get rid of one gate by giving the value 0 or 1 to one of the variables. This is clearly not efficient and we have to make more clever assignments serving many purposes simultaneously. To do this explicitly seems hard and our way of avoiding this is to rely on luck. We will pick a random restriction and it will do the job for us with high probability.

We will be working with random restrictions with distributions parameterized by a real number p which usually will be small.

Definition: A random restriction $\rho \in R_p$ satisfies

$\rho(x_i) = 0$ with probability $\frac{1}{2} - \frac{p}{2}$

$\rho(x_i) = 1$ with probability $\frac{1}{2} - \frac{p}{2}$

$\rho(x_i) = *$ with probability p.

independently for different x_i.

Observe that we have probability p of keeping a variable. Thus the expected number of variables remaining is pn. Obviously, the smaller p, the more we can simplify our circuits, but, on the other hand, fewer variables remain. We must choose p as to optimize this trade off.

The main improvement of our result over previous work is that we make a tighter analysis of how much a restriction simplifies a circuit. We will prove a lemma which basically tells us that if we hit a depth two circuit with a random restriction then we can change an AND of ORs to an OR of ANDs without increasing the size. We prove that this fails with only exponentially small probability.

We will need some notation. A *minterm* is a minimal way to make a function 1. We will think of a minterm σ for a function F as a partial assignment with the following two properties.

(1) σ forces F to be true.

(2) No subassignment of σ forces F to be true.

Thus (2) says that σ is minimal satisfying (1).

Example Let $F(x_1, x_2, x_3)$ be the majority function. Then the minterms are σ_1, σ_2 and σ_3 where

$$\sigma_1(x_1) = 1, \sigma_1(x_2) = 1, \sigma_1(x_3) = *$$

$$\sigma_2(x_1) = 1, \sigma_2(x_2) = *, \sigma_2(x_3) = 1$$

$$\sigma_3(x_1) = *, \sigma_3(x_2) = 1, \sigma_3(x_3) = 1$$

The size of a minterm is defined as the number of variables to which it gives either the value 0 or the value 1. All three of the above minterms are of size 2. Observe that it is possible to write a function as an OR of ANDs where the ANDs precisely correspond to its minterms. The size of the ANDs will be the size of the minterms since x_i will be input precisely when $\sigma(x_i) = 1$ and $\bar{x}_i$ will be input precisely when $\sigma(x_i) = 0$.

4. Main Lemma

Our main lemma will tell us that if we apply a restriction we can with high probability convert an AND of ORs to an OR of ANDs. This will provide the tool for us to carry through the outline of the proof described in section 3.

Main Lemma 4.1: *Let G be an AND of ORs all of size $\leq t$ and ρ a random restriction from R_p. Then the probability that $G\lceil_\rho$ cannot be written as an OR of ANDs all of size $< s$ is bounded by α^s where α is the unique positive root to the equation.*

$$(1+\frac{4p}{1+p}\frac{1}{\alpha})^t = (1+\frac{2p}{1+p}\frac{1}{\alpha})^t + 1$$

Remark 1 By looking at $\neg G$ one can see that it is possible to convert an OR of ANDs to an AND or ORs with the same probability.

Remark 2 There are two versions of the proof of the main lemma which are almost identical except for notation. Our original proof was in terms of a labeling algorithm used by Yao [Y2] in his proof. The present version of the proof, avoiding the use of such an algorithm, was proposed by Ravi Boppana [Bo2].

It turns out that it is easier to prove a stronger version of the main lemma. First we will require all minterms of $G\lceil_\rho$ to be small. By the remark above this implies that $G\lceil_\rho$ can be written as an OR of small ANDs. A more significant difference is that we prove that the probability of the existence of large minterms remain small even if we condition upon an arbitrary function being forced to be 1. This facilitates induction.

For notational convenience let $min(G) \geq s$ denote the event that $G\lceil_\rho$ has a minterm of size at least s.

Stronger Main Lemma 4.2: *Let $G = \wedge_{i=1}^{w} G_i$, where G_i are OR's of fanin $\leq t$. Let F be an arbitrary function and ρ a random restriction in R_p. Then*

$$Pr[min(G) \geq s \mid F\lceil_\rho \equiv 1] \leq \alpha^s$$

Remark 3: The stronger main lemma implies the main lemma by choosing $F \equiv 1$ and the fact that a function has a circuit which is an OR of ANDs corresponding to its minterms.

Remark 4 If there is no restriction ρ satisfying the condition $F\lceil_\rho \equiv 1$ we use the convention that the conditional probability in question is 0.

Proof: We prove the stronger main lemma by induction on w the number of ORs in G. A picture of G which is good to keep in mind is the following.

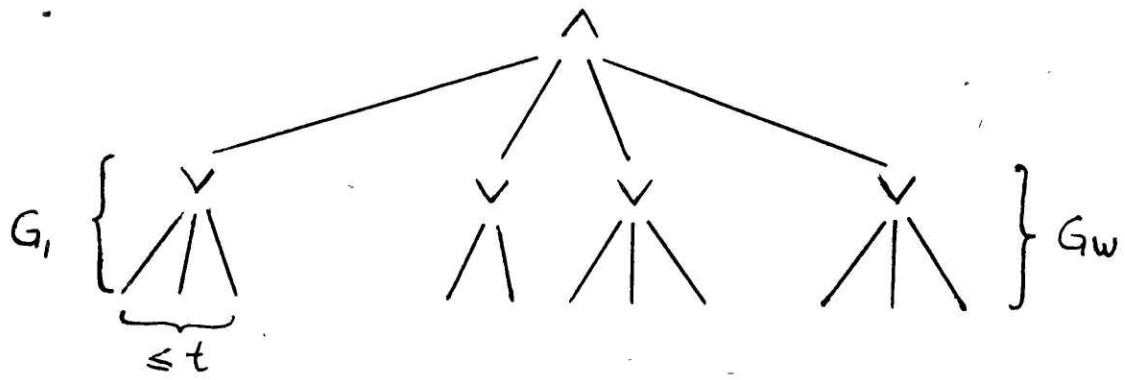

Figure 5

If $w = 0$ the lemma is obvious ($G \equiv 1$). Suppose now that the statement is true for all values less than w. We show that it is true for w. We first study what happens to G_1, the first OR in the circuit. We have two possibilities, either it is forced to be 1 or it is not. We estimate these two probabilities separately. We have

$$Pr[min(G) \geq s \mid F\lceil_\rho \equiv 1] \leq$$

$$\max(Pr[min(G) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \equiv 1], Pr[min(G) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1])$$

The first term is

$$Pr[min(G) \geq s \mid (F \wedge G_1)\lceil_\rho \equiv 1]$$

However in this case $G\lceil_\rho = \wedge_{i=1}^{w} G_i\lceil_\rho = \wedge_{i=2}^{w} G_i\lceil_\rho$ since we are only concerned about ρ's which forces G_1 to be 1. Thus $min(G) \geq s$ is equivalent to saying that $\wedge_{i=2}^{w} G_i\lceil_\rho$ has a minterm of size at least s. But this probability is $\leq \alpha^s$ by the inductive hypothesis since we are talking about a product of size $w - 1$. We are conditioning upon another function being 1 but this is OK since we are assuming that the induction hypothesis is true for an arbitrary F. It is precisely the fact that the conditioning keeps changing that "forced" us to introduce the stronger version of the main lemma.

Now consider the second term $(Pr[min(G) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1])$. For notational convenience we will assume that G_1 is an OR of only positive literals, i.e.

$$G_1 = \vee_{i \in T} x_i$$

where $|T| \leq t$. We do not lose generality by this since we can interchange x_i and $\bar{x}_i$.

Let $\rho = \rho_1\rho_2$, where ρ_1 is the restriction of the variables in T and ρ_2 is the restriction of all other variables. The condition $G_1\lceil_\rho \not\equiv 1$ is equivalent to ρ_1 never assigning the value 1. As this is only a condition on ρ_1 we rewrite this as $G_1\lceil_{\rho_1} \not\equiv 1$.

Before going on let us at this point give some intuition what is going on and why the proof works. Assume that $p < \frac{1}{10t}$.

We are now studying the case where the first OR, G_1 is not forced to be 1 by the restriction. There are two possibilities, either G_1 is forced to be 0 or it remains undetermined. In the first case all our troubles are over since G itself is forced to be 0. The second case is the bad case and we have to do further work. However by our choice of p the first case will occur with at least constant probability and that is what makes the proof work. Let us return to the formal proof.

By definition a minterm of $G\lceil_\rho$ makes $G\lceil_\rho$ true. Since we are now conditioning upon the fact that G_1 is not made true by the restriction, we know that G_1 has to be made true by every minterm σ i.e. there must be an $i \in T$ such that $\sigma(x_i) = 1$. Observe that σ might give values to some other variables in T and that these values might be both 0 and 1. Partition the minterms of $G\lceil_\rho$ according to which variables in T they give values to. Call a typical such subset Y.

The fact that the minterm gives values to the variables in Y implies that the variables in Y were given the value $*$ by ρ_1. This fact will be denoted by $\rho_1(Y) = *$. Further let $min(G)^Y \geq s$ denote the event that $G\lceil_\rho$ has a minterm of size at least s whose restriction to the variables in T assigns values to precisely those variables in Y. Using this notation we get

$$Pr[min(G) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1] \leq \sum_{Y \subset T, Y \neq \emptyset} Pr[min(G)^Y \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1] =$$

$$\sum_{Y \subset T, Y \neq \emptyset} Pr[min(G)^Y \geq s \wedge \rho_1(Y) = * \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1] =$$

$$\sum_{Y \subset T, Y \neq \emptyset} Pr[\rho_1(Y) = * \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1]$$

$$\times Pr[min(G)^Y \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1 \wedge \rho_1(Y) = *]$$

The inequality and the first equality follows by the reasoning above. The last equality follows by the definition of conditional probability. Let us estimate each of the two factors in each term of the above sum starting with the first factor (i.e. $Pr[\rho_1(Y) = * | F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1]$). To make life simpler we will first ignore the condition $F\lceil_\rho \equiv 1$.

Lemma 4.3: $Pr[\rho_1(Y) = * \mid G_1\lceil_{\rho_1} \not\equiv 1] = (\frac{2p}{1+p})^{|Y|}$.

Proof: As remarked above the condition $G_1\lceil_{\rho_1} \not\equiv 1$ is precisely equivalent to $\rho_1(x_i) \in \{0, *\}$ for $i \in T$. The induced probabilities are $Pr[\rho(x_i) = 0] = \frac{1-p}{1+p}$ and $Pr[\rho(x_i) = *] = \frac{2p}{1+p}$. The lemma follows since the probabilities are independent. ∎

Now we must take the condition $F\lceil_\rho \equiv 1$ into account. The intuition for handling this case is as follows. The fact that F is determined to be 1 cannot make stars more likely since having a lot of stars is in a vague sense equivalent to things being undetermined. This argument can be made formal in several ways. The one presented here was proposed by Mike Saks.

We first need an elementary fact from probability theory. Let A, B and C be three arbitrary events

Lemma 4.4: $Pr[A \mid B \wedge C] \leq Pr[A \mid C]$ *iff* $Pr[B \mid A \wedge C] \leq Pr[B \mid C]$.

This lemma follows from use of definition of conditional probability and trivial algebra. Our final estimate is

Lemma 4.5: $Pr[\rho_1(Y) = * \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1] \leq (\frac{2p}{1+p})^{|Y|}$.

Proof: Let $A = (\rho_1(Y) = *), B = (F\lceil_\rho \equiv 1)$ and $C = (G_1\lceil_{\rho_1} \not\equiv 1)$. By the above lemmas we only have to verify that

$$Pr[F\lceil_\rho \equiv 1 \mid \rho_1(Y) = * \wedge G_1\lceil_{\rho_1} \not\equiv 1] \leq Pr[F\lceil_\rho \equiv 1 \mid G_1\lceil_{\rho_1} \not\equiv 1]$$

This is clear from inspection since requiring that some variables are $*$ cannot increase the probability that a function is determined. ∎

Next we to estimate the other factor. Namely

$$Pr[min(G)^Y \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1 \wedge \rho_1(Y) = *]$$

To do this think of a minterm of $G\lceil_\rho$ which give values to the variables in Y and no other variable in T as consisting of two parts:

(1) Part σ_1 which assigns values to the variables of Y.

(2) Part σ_2 which assigns values to some variables in the complement $\bar{T}$ of T.

This partitioning of a minterm is possible since we are assuming that it assigns no values to variables in $T - Y$. Observe that σ_2 is a minterm of the function $(G\lceil_\rho)\lceil_{\sigma_1}$. This obviously suggests that we can use the induction hypothesis. We only have to get rid of the unpleasant condition that $G_1\lceil_{\rho_1} \not\equiv 1$. This we do by maximizing over all ρ_1 satisfying this condition. Let $min(G)^{Y,\sigma_1} \geq s$ denote the event that $G\lceil_\rho$ has a minterm of size at least s which assigns the values σ_1 to the variables in Y and does not assign values to any other variables in T. We have

$$Pr[min(G)^Y \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1 \wedge \rho_1(Y) = *] \leq$$

$$\sum_{\sigma_1 \in \{0,1\}^{|Y|}} \sum_{\sigma_1 \neq 0^{|Y|}} \left(\max_{\rho_1(Y)=*,\ \rho_1(T) \in \{0,*\}^{|T|}} Pr_{\rho_2}[min(G)^{Y,\sigma_1} \geq s \mid (F\lceil_{\rho_1\sigma_1})\lceil_{\rho_2} \equiv 1] \right)$$

The two last conditions have disappeared because they involve only ρ_1 and the probability is taken over ρ_2 only. By (2) above we know that $min(G)^{Y,\sigma_1} \geq s$ implies that $(G\lceil_{\rho_1\sigma_1})\lceil_{\rho_2}$ has a minterm of size at least $s - |Y|$ on the variables in $\bar{T}$. Thus we can estimate the probability by $\alpha^{s-|Y|}$ by the induction hypothesis. The stars of ρ_1 are substituted by taking AND of the two formulas resulting by substituting 0 and 1. This can be done since both $F\lceil_\rho \equiv 1$ and the property of having a certain minterm are properties of making a function 1. Making a function 1 even when some variable is undetermined is the same as making the function 1 in the two cases where 0 and 1 are substituted for the variable. Finally observe that $G\lceil_{x_i=0} \wedge G\lceil_{x_i=1}$ does not contain more ORs than G. The reason is that ORs that do not contain x_i are duplicated and one of the copies can be removed. ORs that contain either x_i or $\bar{x}_i$ occur only in one of $G\lceil_{x_i=0}$ and $G\lceil_{x_i=1}$.

To sum up, each term in the sum is estimated by $\alpha^{s-|Y|}$ and we have $2^{|Y|}-1$ possible σ_1. This is because σ_1 must make G_1 true and hence cannot be all 0. Thus we get the total bound $(2^{|Y|}-1)\alpha^{s-|Y|}$.

Finally we must evaluate the sum and since the term corresponding to $Y = \emptyset$ is 0 we can include it.

$$\sum_{Y \subset T} (\frac{2p}{1+p})^{|Y|}(2^{|Y|}-1)\alpha^{s-|Y|} = \alpha^s \sum_{i=0}^{|T|} \binom{|T|}{i} [(\frac{4p}{1+p}\frac{1}{\alpha})^i - (\frac{2p}{1+p}\frac{1}{\alpha})^i] =$$

$$\alpha^s((1+\frac{4p}{1+p}\frac{1}{\alpha})^{|T|} - (1+\frac{2p}{1+p}\frac{1}{\alpha})^{|T|}) \leq \alpha^s((1+\frac{4p}{1+p}\frac{1}{\alpha})^t - (1+\frac{2p}{1+p}\frac{1}{\alpha})^t = \alpha^s$$

The second equality comes from the fact that $(1+x)^t = \sum_{i=0}^{t} \binom{t}{i} x^i$ and the last equality follows from the definition of α. This finishes the induction step and the proof of the stronger main Lemma.

4.1 Improving the constant

It is possible to prove a slightly stronger version of the main lemma. This result is due

to Ravi Boppana [Bo2] and we are grateful for his permission to include it here. Shlomo Moran [M] independently produced a similar proof for the same result. The few mistakes occurring in that proof was easily fixed.

Lemma 4.7: *Let G be an AND of ORs all of size $\leq t$ and ρ a random restriction from R_p. Then the probability that $G\lceil_\rho$ cannot be written as an OR of ANDs all of size $< s$ is bounded by α^s where α is the unique positive root to the equation.*

$$(1+\frac{2p}{1+p}(\frac{2}{\alpha}-1))^t = (1+\frac{2p}{1+p}(\frac{1}{\alpha}-1))^t + 1$$

Furthermore we have the same estimate on the probability even if we condition upon an arbitrary function F being forced to 1 by ρ.

To get this result we only have to do a few slight modifications to the original proof. Observe that we are not any more claiming that all the minterms are small but only that the resulting function can be written as an OR of small ANDs. To see the difference look at $f(x_1.x_2,x_3) = (x_1 \wedge x_2) \vee (\bar{x}_1 \wedge x_2)$. $x_2 = 1$ and $x_3 = 1$ gives a minterm which does not correspond to an AND. By this relaxation we can choose the set Y to be precisely the set of variables given the value $*$ by ρ. Thus by the same reasoning as before we are led to estimating

$$\sum_{Y \subset T, Y \neq \emptyset} Pr[\rho_1(Y) = * \wedge \rho_1(T-Y) = 0 \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1]$$

$$\times Pr[AND(G)^Y \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_{\rho_1} \not\equiv 1 \wedge \rho_1(Y) = * \wedge \rho_1(T-Y) = 0]$$

The second factor will be estimated by induction as before. Denote the first term by P_Y. Previously we used the bound $P_Y \leq (\frac{2p}{1+p})^{|Y|}$ provided by Lemma 4.5. No stronger bound for P_Y can be obtained in general. However Lemma 4.5 really tells us that

$$\sum_{Y_0 \subset Y} P_Y \leq (\frac{2p}{1+p})^{|Y_0|}$$

and this is what we will use.

In the final summation we are estimating

$$\sum_{Y \subset T} P_Y (2^{|Y|} - 1) \alpha^{s-|Y|}$$

Using partial summation this is equal to

$$\sum_{Y \subset T} ((2^{|Y|} - 1)\alpha^{s-|Y|} - \delta_{|Y|}) \sum_{Y \subset Y_0} P_{Y_0}$$

where $\delta_i = 0$ if $i = 0$ and $(2^{i-1} - 1)\alpha^{s+1-i}$ otherwise.

Now the term outside the inner sum is positive in $\alpha < 1$ which is the only case we are interested in. Thus we can use the upper bound $(\frac{2p}{1+p})^{|Y|}$ for the inner sum. Observing that the case of equality is $P_Y = (\frac{2p}{1+p})^{|Y|}(\frac{1-p}{1+p})^{|T|-|Y|}$ we can simplify our calculations by immediately substitute this expression for P_Y in the original sum. Observe that we have proved that $F \equiv 1$ is the worst case. Now estimating the final sum is done as usual.

$$\sum_{Y \subset T} (\frac{2p}{1+p})^{|Y|}(\frac{1-p}{1+p})^{|T|-|Y|}(2^{|Y|} - 1)\alpha^{s-|Y|}$$

$$\alpha^s(\frac{1-p}{1+p})^{|T|} \sum_{i=0}^{|T|} \binom{|T|}{i} [(\frac{4p}{1-p}\frac{1}{\alpha})^i - (\frac{2p}{1-p}\frac{1}{\alpha})^i] =$$

$$\alpha^s(\frac{1-p}{1+p})^{|T|}((1 + \frac{4p}{1-p}\frac{1}{\alpha})^{|T|} - (1 + \frac{2p}{1-p}\frac{1}{\alpha})^{|T|}) \leq$$

$$\alpha^s((1 + \frac{2p}{1+p}(\frac{2}{\alpha} - 1))^t - (1 + \frac{2p}{1+p}(\frac{1}{\alpha} - 1))^t = \alpha^s$$

The last equality follows by the definition of α. This finishes the proof.■

4.2 Estimates on the size of α

To make Lemma 4.1 easier to apply we need more explicit bounds on the size of α. These are provided by the following lemma.

Lemma 4.8: *Let α solve the equation*

$$(1+\frac{4p}{1+p}\frac{1}{\alpha})^t = (1+\frac{2p}{1+p}\frac{1}{\alpha})^t + 1$$

and let $\phi = \frac{\sqrt{5}+1}{2}$. Then $\alpha \leq \frac{2pt}{\ln \phi} + O(tp^2)$. In particular $\alpha < 5pt$ for $p < p_0$ for some absolute constant $p_0 > 0$.

Proof: Neglecting terms of order tp^2 the equation can be written

$$e^{\frac{4pt}{\alpha}} = e^{\frac{2pt}{\alpha}} + 1$$

This equation is solved by $e^{\frac{2pt}{\alpha}} = \phi$ leading to the claimed expression.

5. Lower bounds for small depth circuits

Parity is the first function for which we prove lower bounds.

Theorem 5.1: *There are no depth k parity circuits of size $2^{(\frac{1}{10})^{\frac{k}{k-1}} n^{\frac{1}{k-1}}}$ for $n > n_0^k$ for some absolute constant n_0.*

Remark 5: Observe that this is quite close to optimal since by Theorem 2.2 parity can be computed by depth k circuits of size $n2^{n^{\frac{1}{k-1}}}$. The best previous lower bounds were $\Omega(2^{n^{\frac{1}{4k}}})$ by Yao [Y2].

As in the case of the main lemma we first prove a result suitable to induction, and later show that this result implies Theorem 5.1.

Theorem 5.2: *Parity cannot be computed by a depth k circuit containing $\leq 2^{\frac{1}{10}n^{\frac{1}{k-1}}}$ gates of distance at least 2 from the inputs and which has bottom fanin $\leq \frac{1}{10}n^{\frac{1}{k-1}}$ for $n > n_0^k$, where n_0 is some absolute constant.*

Proof: We prove the theorem by induction over k. The base case $k = 2$ follows from the well known fact that depth 2 parity circuits must have bottom fanin n. The induction step

is done as outlined in section 3. We proceed by contradiction. Assuming that a depth k circuit exists with the the described parameters we construct a depth $k-1$ circuit with the corresponding parameters. The key step is of course provided by the main lemma which enables us to control the size of the depth $k-1$ circuit.

Suppose, without loss of generality, that our depth k circuits are such that the gates at distance 2 from the inputs are AND gates and hence represents a depth 2 circuit with bottom fanin bounded by $\frac{1}{10}n^{\frac{1}{k-1}}$. Apply a random restriction from R_p with $p = n^{-\frac{1}{k-1}}$. Then by our lemma every individual depth two subcircuit can be written as an OR of ANDs of size bounded by s with probability $1-\alpha^s$. By the chosen parameters and Lemma 4.8 α is bounded by a constant less than $\frac{1}{2}$. If we choose $s = \frac{1}{10}n^{\frac{1}{k-1}}$ the probability that it is not possible to write all depth two circuits as OR of ANDs of size s is bounded by $2^{\frac{1}{10}n^{\frac{1}{k-1}}}\alpha^s = (2\alpha)^s$. Thus with probability at least $1-(2\alpha)^s$ we can interchange the order of AND and OR in all depth 2 subcircuits giving adjacent levels of OR gates and still maintain bottom fanin bounded by s. The adjacent levels of OR's which can be collapsed to decrease the depth of the circuit to $k-1$. Observe that gates at distance at least two from the inputs in this new circuit corresponds to gates at distance at least three from the inputs in the old circuit and are hence bounded in number by $2^{\frac{1}{10}n^{\frac{1}{k-1}}}$.

The number of remaining variables is expected to be $pn = n^{\frac{k-2}{k-1}}$ and with probability greater than $\frac{1}{3}$ we will get at least this number for $n > n_0^k$. Thus with nonzero probability the new circuit of bottom fanin $\leq \frac{1}{10}n^{\frac{1}{k-1}}$ having $\leq 2^{\frac{1}{10}n^{\frac{1}{k-1}}}$ gates of distance at least 2 from the inputs and which computes the parity of $n^{\frac{k-2}{k-1}}$ variables. In particular such a restriction exists. Applying this restriction and letting $m = n^{\frac{k-2}{k-1}}$ we have a circuit certified not to exist by the induction hypothesis. The proof of Theorem 5.2 is complete.

■

Let us now prove Theorem 5.1. Consider the circuit as a depth $k+1$ circuit with bottom fanin 1. Hit it with a restriction from R_p using $p = \frac{1}{10}$ and by using our main

lemma with $s = \frac{1}{10}(\frac{n}{10})^{\frac{1}{k-1}}$ we see that we get a circuit which does not exist by Theorem 5.2.

Since there are no constants depending on k hidden in the theorem we get the following corollary

Corollary 5.3: *Polynomial size parity circuits must have depth at least* $\frac{\log n}{c+\log\log n}$ *for some constant* c.

Observe that this is tight since for every constant c there are such polynomial size circuits.

Observe that in the above proof the properties of parity were hardly used. Only the lower bound for $k = 2$ and the fact that it behaves well with respect to restrictions. Thus our main lemma can be used to improve lower bounds for sizes of small depth circuits computing other functions as well. Let us do majority

Theorem 5.4: *Majority requires size* $2^{(\frac{1}{10})^{\frac{k}{k-1}} n^{\frac{1}{k-1}}}$ *depth* k *circuits for* $n > n_0^k$ *for some absolute constant* n_0.

Proof: The proof is almost identical. First we observe that the base case $k = 2$ holds. To do the induction step we will use a restriction from R_p. However this time we will require the following properties of the restriction

(i) All depth 2 subcircuits within in the circuit can be converted to the other type.

(ii) ρ gives out the same number of 0's and 1's.

(iii) ρ gives out at least $(1-\epsilon)np$ $*$'s.

The probability that the restriction violates either (i) or (iii) is exponentially small. The probability that it satisfies (ii) is $\approx n^{-\frac{1}{2}}$. Thus with a positive probability all three conditions are satisfied.

Now the inductionstep can be done since (i) implies that the depth of the circuit decreases and (ii) together with (iii) imply that the remaining circuit computes majority

of large number of variables.

Suppose we want to prove lower bounds for a function f. In general it is not necessary that $f\lceil_\rho$ is the same function f on fewer variables. As can be seen from the proof it is sufficient that $f\lceil_\rho$ for a severe restriction ρ has a reasonable probability of having large minterms. We leave the details to the interested reader.

5.1 Improving the constant.

Since we did not use optimal use even of Lemma 4.1 it is clear that $\frac{1}{10}$ is not the optimal constant. Let us improve the constants slightly by using Lemma 4.7 and not being so careless.

Theorem 5.5: *Parity cannot be computed by a depth k circuit containing* $\leq 2^{.1437n^{\frac{1}{k-1}}}$ *subcircuit of depth at least 2 and bottom fanin* $\leq .1136n^{\frac{1}{k-1}}$ *for* $n > n_0^k$ *for some absolute constant* n_0.

Proof: The proof is identical to that of Theorem 5.2. We use $s = t = .1136n^{\frac{1}{k-1}}$ and $p = n^{-\frac{1}{n^{\frac{1}{k-1}}}}$ and use the bounds of lemma 4.7 giving $\alpha = .415904$. ∎

The restriction of the bottom fanin can be removed as before and we can get theorem 5.1 with the c_k tending towards .1437.

6. Functions requiring depth k to have small circuits.

In [Si], Sipser defined a set of functions f_k^m which could be computed in depth k and polynomial size. He showed , however, that these functions required superpolynomial size for depth $k-1$.

The functions were defined by a depth k circuit as follows:

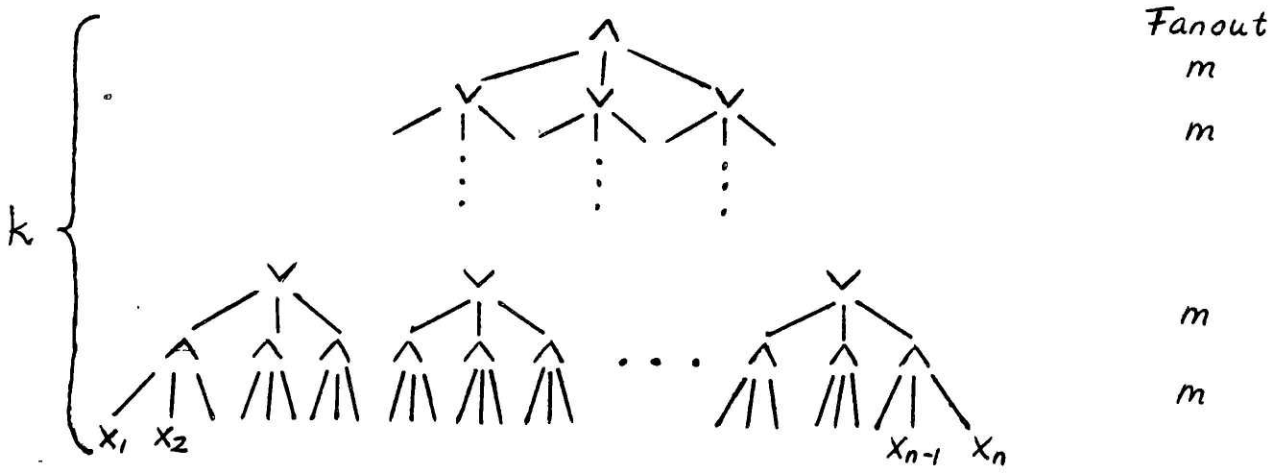

Figure 6

To avoid confusion we will refer to the above circuit as the defining circuit of f_k^m. The defining circuit is thus a tree with fanout m, depth k, in which each variable occurs only once. Yao has claimed exponential lower bounds for these functions. The proof has not yet appeared but is supposed to be as complicated as in the case of the parity function. Therefore, we include our lower bound proof, even though the bounds are not quite optimal.

First let us redefine the functions f_k^m slightly. Let g_k^m be defined by the following circuit:

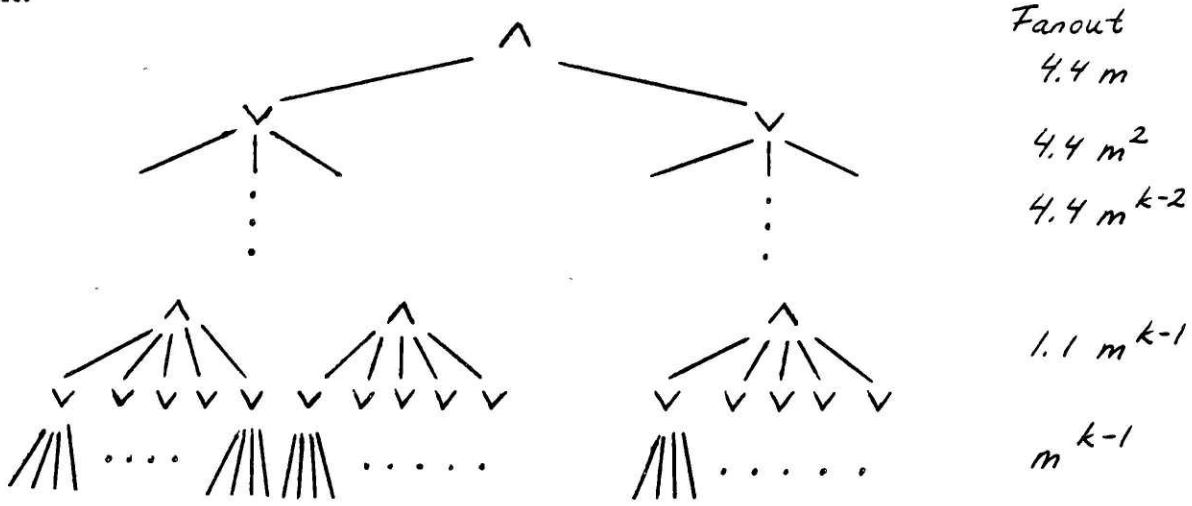

Figure 7

Note that the only difference between f_k^m and g_k^m is that the fanouts in the defining tree varies for g_k^m. Observe that g_k^m is a function of $(1.1)^{k-1}4^{k-2}m^{\frac{k^2+k-2}{2}}$ variables. The g_k^m might seem more complicated than f_k^m but they will simplify the notation in the proof. The constant 1.1 is any constant greater than 1 and 4 is any constant greater than $e^{1.1}$.

For the g_k^m we have the following theorem.

Theorem 6.1: *Depth $k-1$ circuits computing g_k^m are of size at least $2^{\frac{1}{10}m}$ for $m > m_1$ where m_1 is some absolute constant.*

Let us make explicit the relation between f_k^m and g_k^m. Define a function h_1 be a restriction of a function h_2 if fixing some of the inputs to h_2 it is possible to make the induced function on the remaining variables equal to h_1. If this is the case it is clear that any circuit for h_2 can be converted to a circuit for h_1 without increasing the size.

Using this definition we note that f_k^m is a restriction of g_k^m and g_k^m is a restriction of $f_k^{2m^{k-1}}$ and thus the sizes of depth $k-1$ circuits computing the two functions are related. This gives the following corollary to Theorem 6.1.

Corollary 6.2: *Depth $k-1$ circuits computing f_k^m has to be of size at least $2^{\frac{1}{10}(\frac{m}{2})^{\frac{1}{k-2}}}$ for $m > m_1^{k-2}$ for some absolute constant m_1.*

6.1 New random restrictions

One would like to prove Theorem 6.1 with the aid of the main lemma. However here one runs into problems not encountered in the case of the parity function. If a restriction from R_p is applied to either f_k^m or g_k^m the resulting function will be a constant function with very high probability. This happens since the gates at the bottom level are quite wide and with very high probability all gates will be forced. To get around this problem we will define another set of restrictions which will be more suitable to the present functions.

Definition: Let p_1, p_0 and p_* be real numbers satisfying $p_1 + p_0 + p_* = 1$ and $(B_i)_{i=1}^r$ a partition of the variables (The B_i are disjoint sets of variables and their union is the set

of all variables). Let $R^{+}_{p_1,p_0,p_*,B}$ be the probability space of restrictions which takes values as follows.

For $\varrho \in R^{+}_{p_1,p_0,p_*,B}$ and every B_i, $1 \leq i \leq r$ independently

With probability p_1, $\rho(x_j) = 1$ for all $x_j \in B_i$.

With probability $p_0 + p_*$ choose a random $x_k \in B_i$. Let $\rho(x_j) = 1$ for $j \neq k$ and $\rho(x_k) = 0$ or $*$ with probability $\frac{p_0}{p_0+p_*}$ and $\frac{p_*}{p_0+p_*}$ respectively.

Similarly a $R^{-}_{p_0,p_1,p_*,B}$ probability space of restrictions can be defined by interchanging the roles played by 0 and 1.

These sets of restrictions do not assign values to variables independently as our previous restrictions did but they are nice enough so that the proof of our main lemma will go through with only minor modifications. Define q to be $max(\frac{p_*}{p_0+p_*}, (\frac{p_*}{p_1|B_i|+p_*})_{i=1}^{r})$.

Lemma 6.3: *Let G be an AND of ORs all of size $\leq t$ and ρ a random restriction from $R^{+}_{p_0,p_1,p_*,B}$. Then the probability that $G\lceil_\rho$ cannot be written as an OR of ANDs all of size $< s$ is bounded by α^s, where α is the unique positive root to the equation*

$$(1 + \frac{2q}{\alpha})^t = (1 + \frac{q}{\alpha})^t + 1.$$

Remark 6 The same is true for $R^{-}_{p_0,p_1,p_*,B}$.

Remark 7 The probability of converting an OR of ANDs to an AND of ORs is the same.

As in the case of the main lemma, before proving Lemma 6.3, we prove a stronger lemma stating that the resulting function has large minterms is very unlikely even when conditioning upon an arbitrary formula being forced to 1 by the restriction.

Lemma 6.4: *Let $G = \wedge_{i=1}^{w} G_i$, where G_i are OR's of fanin $\leq t$. Let F be an arbitrary function. Let ρ be a random restriction in $R^{+}_{p_1,p_0,p_*,B}$. Then*

$$Pr[min(G\lceil_\rho) \geq s \mid F\lceil_\rho \equiv 1] \leq \alpha^s$$

where α is the unique positive root to the equation

$$(1+\frac{2q}{\alpha})^t = (1+\frac{q}{\alpha})^t + 1.$$

Remark 8: Lemma 6.4 implies Lemma 6.3 as the strong main lemma implies the main lemma.

Remark 9: Recall that, if there is no restriction ρ satisfying the condition $F\lceil_\rho \equiv 1$ then the conditional probability in question is defined to be 0.

Proof: The proof of Lemma 6.4 will be done the same way as the proof of the stronger main lemma (Lemma 4.2). We therefore only outline the proof, and give details only where the proofs differ.

As before

$$Pr[min(G\lceil_\rho) \geq s \mid F\lceil_\rho \equiv 1]$$

$$\leq max(Pr[min(G\lceil_\rho) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \equiv 1], Pr[min(G\lceil_\rho) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1])$$

The first term,

$$Pr[min(G\lceil_\rho) \geq s \mid (F \wedge G_1)\lceil_\rho \equiv 1]$$

is taken care of by the induction hypothesis, as discussed in the proof of Lemma 4.2.

The second term, $Pr[min(G\lceil_\rho) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1]$ is estimated as in Lemma 4.2 with the only difference that here we cannot assume that G_1 is an OR of only positive literals since the restrictions employed here assign 0 and 1 nonsymmetrically.

We denote the set of variables occurring in G_1 by T, and $|T| \leq t$. We still know that G_1 must be made true by every minterm of $G\lceil_\rho$, and partition the minterms of $G\lceil_\rho$ according to what set of variables $Y \subset T$ they assign values to.

We get

$$Pr[min(G\lceil_\rho) \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1] \leq$$

$$\sum_{Y \subset T, Y \neq \emptyset} Pr[\rho(Y) = * \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1] \times$$
$$Pr[min(G\lceil_\rho)^Y \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1 \wedge \rho(Y) = *]$$

The factors in the above sum can be estimated separately. The estimates are made in a slightly different way than in the proof of Lemma 4.2. This is the main difference between the present proof and that proof.

Let us start with the first factor $Pr[\rho(Y) = * \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1]$.

Lemma 6.5 investigates which restrictions satisfy the conditions $F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1$ and how this might effect the probability of a set of variables taking the value $*$ under ρ.

Lemma 6.5: *Let $x_i \in Y$. Then if an assignment ρ satisfies the condition $F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1$ and has $\rho(x_i) = *$ and x_i ($\bar{x}_i$) occurs in G_1. Then the corresponding assignment $\tilde{\rho}$ which is the same as ρ with the exception of $\tilde{\rho}(x_i) = 0$ (1) instead of $*$ also satisfies the condition.*

Proof: Clearly, the condition $G_1\lceil_\rho \not\equiv 1$ presents no problem. The other condition $F\lceil_\rho \equiv 1$ is easily verified since the fact that $F\lceil_\rho \equiv 1$ and $\rho(x_i) = *$ implies that a change in the value of x_i cannot change the value of $F\lceil_\rho$.

Next we prove

Lemma 6.6: $Pr[\rho(Y) = * \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1] \leq q^{|Y|}$.

Proof: The proof of Lemma 6.6 will use Lemma 6.5. By the definition of conditional probability we want to prove

$$\frac{\sum'_{\rho(Y)=*} Pr(\rho)}{\sum' Pr(\rho)} \leq q^{|Y|}$$

Here the $'$ indicates that we are only summing over ρ's satisfying the condition $F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1$. If this quotient is $\frac{0}{0}$ we use the convention that it takes on the value 0. Now observe that if ρ gives a nonzero contribution to the numerator, then by Lemma 6.5, all possible ρ obtained by changing arbitrary stars of Y to nonzero values required by $G_1\lceil_\rho \not\equiv 1$ will give contributions to the denominator. Calculation shows that this contribution is at

least a factor $q^{-|Y|}$ larger. This clearly proves the lemma. We will carry out the detailed calculation for $|Y| = 2$, the general case being similar.

Suppose Y contains x_i and x_j and that x_i and $\bar{x}_j$ occurs in G_1. Assume for notational convenience that $x_i \in B_i$ and $x_j \in B_j$. We can assume that $B_i \neq B_j$ since ρ never gives out two $*$'s to the same block and thus in the case $B_i = B_j$ the above probability is 0. Take a restriction ρ which satisfies $\rho(Y) = * \wedge F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1$. Define $\rho_{a,b}$ by $\rho_{a,b}(x_k) = \rho(x_k)$ for $k \neq i, j$, $\rho(x_i) = a$ and $\rho(x_j) = b$. By Lemma 6.5 $\rho_{a,b}$ satisfies the condition $F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1$ for $(a, b) \in \{(*, *), (*, 1), (0, *), (0, 1)\}$. Thus corresponding to the contribution of $Pr(\rho)$ in the numerator we get the the following contribution in the denominator:

$$Pr(\rho) + Pr(\rho_{*,1}) + Pr(\rho_{0,*}) + Pr(\rho_{0,1}) =$$

$$= Pr(\rho) + Pr(\rho)\frac{p_1}{\frac{p_*}{|B_j|}} + Pr(\rho)\frac{p_0}{p_*} + Pr(\rho)\frac{p_0}{p_*}\frac{p_1}{\frac{p_*}{|B_j|}}$$

By the definition of q, we have $\frac{1}{q} - 1 \leq \frac{p_0}{p_*}$ and $\frac{1}{q} - 1 \leq \frac{p_1}{\frac{p_*}{|B_j|}}$. Using this the above expression is bounded from below by

$$Pr(\rho)(1 + (q^{-1} - 1) + (q^{-1} - 1) + (q^{-1} - 1)^2) = Pr(\rho)q^{-2}$$

Next we try to estimate the factor

$$Pr[min(G\lceil_\rho)^Y \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1 \wedge \rho(Y) = *]$$

We think of every minterm as consisting of two parts:

(1) Part σ_1 which assigns values to the variables of Y.

(2) Part σ_2 which assigns values to some variables in the complement $\bar{T}$ of T.

To use the induction hypothesis we have to get rid of the condition that $G_1\lceil_\rho \not\equiv 1$. We will maximize over the behavior of ρ on T. Let $\rho\lceil_T$ denote the behavior of ρ on T and $\rho\lceil_{\bar{T}}$ the behavior on $\bar{T}$.

$$Pr[min(G)^Y \geq s \mid F\lceil_\rho \equiv 1 \wedge G_1\lceil_\rho \not\equiv 1 \wedge \rho(Y) = *] \leq$$

$$\sum_{\sigma_1 \in \{0,1\}^{|Y|}\ G_1\lceil_{\sigma_1} \equiv 1} (\max_{\rho\lceil_{\bar{T}}} Pr_{\rho\lceil_{\bar{T}}}[min(G)^{Y,\sigma_1} \geq s \mid F\lceil_\rho \equiv 1])$$

The two last conditions have disappeared since the probability is taken over $\rho\lceil_{\bar{T}}$. By (2) above we know that $min(G)^{Y,\sigma_1} \geq s$ implies that $(G\lceil_{\rho\sigma_1})$ has a minterm of size at least $s - |Y|$ on the variables in $\bar{T}$.

We have to be slightly careful since the values of ρ on T and $\bar{T}$ are not independent. We get around this as follows. If $\rho(x_i) = 0$ or 1 for $x_i \in T$ then we incorporate that in the condition $F\lceil_\rho \equiv 1$. If $\rho(x_i) = *$ substitute all the other values in the the same block B_i (they are now known to be all 1) and in the future we take the probability over a restriction with one block less. In F and G we substitute as in the case of the stronger main lemma. Thus we can use the induction hypothesis and we get the estimate $\alpha^{s-|Y|}$ for each individual σ_1. Since σ_1 has to make G_1 true there are $2^{|Y|} - 1$ possible σ_1. Thus we get the total bound

$$(2^{|Y|} - 1)\alpha^{s-|Y|}$$

Finally we evaluate the sum to get

$$\sum_{Y \subset T} q^{|Y|}(2^{|Y|} - 1)\alpha^{s-|Y|} \leq \alpha^s$$

This finishes the induction step and the proof of the lemma 6.4. ∎

An interesting question is for what probability distributions on the space of restrictions is it possible to prove the equivalent lemma of Lemma 6.3 and Lemma 4.1. The general proof technique uses two crucial properties of the distribution.

(1) The condition $F\lceil_\rho \equiv 1$ for an arbitrary F does not bias the value of any variable too much towards $*$. This should also remain true even if we know that a specific variable is not 1 or 0.

(2) It is possible to eliminate the variables of G_1 and use induction on a similar restriction over the remaining variables.

Condition (1) was taken care of by Lemmas 6.6 and 4.5. Condition (2) seems easier to satisfy and was so obviously satisfied that no formal lemma was needed. The verification was basically done where we claimed that induction could be used after getting rid of G_1.

6.2 Back to proof of Theorem 6.1

Let us continue with the present restriction space $R^+_{p_*,p_1,p_0,B}$ and prove Theorem 6.1. We first prove a slightly stronger technical theorem.

Theorem 6.7: *There is no depth k circuit computing g^m_k with bottom fanin $\leq \frac{1}{10}m$ and size $\leq 2^{\frac{1}{10}m}$ for $m > m_0$ some absolute constant m_0.*

Note that Theorem 6.7 implies Theorem 6.1 since a depth $k-1$ circuit can be considered as a depth k circuit with bottom fanin 1.

Theorem 6.7 is proved by induction over k. The base case for $k = 2$ is quite easy and is left to the reader.

For the induction step we use one of the restrictions defined above. Assume for definiteness that k is odd, so that the gates on the bottom level are AND gates. Define the sets B_i in the partition to be the set of variables leading into an AND gate. Recall that since the defining circuit of g^m_k is a tree the blocks are disjoint. Set $p_1 = m^{1-k}, p_0 = 1 - m^{-1} - m^{1-k}$ and $p_* = m^{-1}$ and apply a random restriction from $R_{p_1,p_0,p_*,B}$. Since the size of every block B_i is m^{k-1}, we get $q = max(\frac{p_*}{p_0+p_*}, (\frac{p_*}{p_1|B_i|+p_*})^r_{i=1}) = \frac{m^{-1}}{1-m^{1-k}}$.

In the case of the parity function even after applying a restriction, the remaining circuit still computed parity or the negation of parity. In the case of g^m_k, we will prove that the new restrictions used transform g^m_k into something that is very close to g^k_{k-1}. It was precisely the fact that the R_p restrictions simplified g^m_k too much that forced us to define the new probability space of restrictions.

Lemma 6.8: *If k is odd then the circuit that defines $g_k^m\lceil_\rho$ for a random $\rho \in R^+_{p_1,p_0,p_*,B}$ will contain the circuit that defines g_{k-1}^m with probability at least $\frac{2}{3}$, for all $m > m_1$, where m_1 is some absolute constant.*

Remark 10: Lemma 6.8 holds for even k when R^+ is replaced by R^-.

Proof: The fact that k is odd implies that the three lower levels look like:

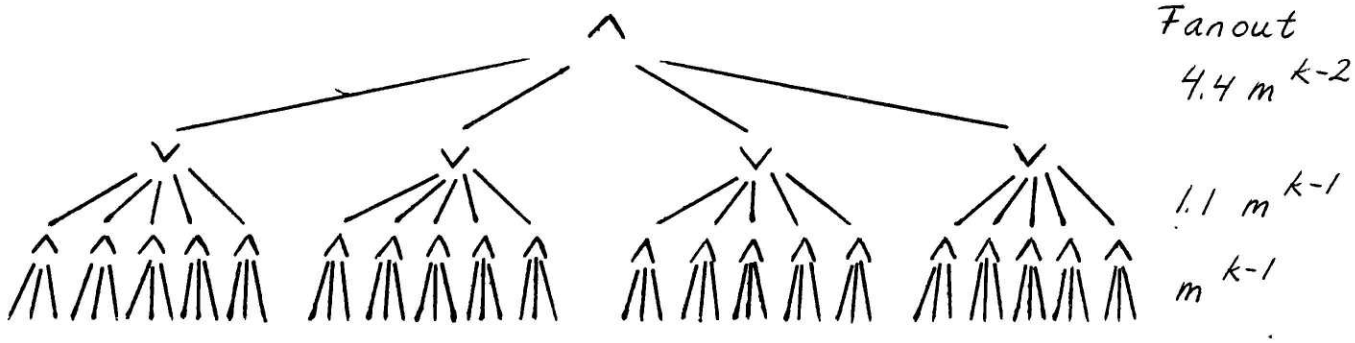

Figure 8

The restriction can be viewed as giving values to the AND gates. The value 1 is given with probability p_1, the value 0 with probability p_0 and $*$ with probability p_*.

An OR gate is determined to 1 as soon as it has one input which is 1. This means that the probability that an individual OR gate on level $k-1$ will not be determined to 1 is $(1-m^{1-k})^{1.1m^{k-1}}$. For large m this is approximately $e^{-1.1}$. Look at an AND gate on level $k-2$. The expected number of nondetermined OR gates leading into this gate is $e^{-1.1}4.4m^{k-2}$. The probability that the number of surviving OR gates is at least $1.1m^{k-2}$ is $1-2^{-cm^{k-2}}$ for some constant c for $m > m_0$ some absolute constant m_0. Thus the probability that this will be true for all AND gates is $\geq \frac{5}{6}$ if $m > m_1$ for some absolute constant m_1.

If an OR gate is not determined to 1 then the expected number of $*$'s in it is $1.1m^{k-2}$. The probability that an individual OR gate contains at least m^{k-2} $*$'s is $1-2^{-cm^{k-2}}$. Hence the probability that all OR gates will have at least this many $*$'s as inputs is $\geq \frac{5}{6}$ for $m > m_1$.

To sum up, with probability at least $\frac{2}{3}$ all AND gates at level $k-2$ will remain undetermined. Furthermore all such AND gates will have at least $1.1m^{k-2}$ OR gates leading into it each having at least m^{k-2} inputs. This constitutes the defining circuit for g^m_{k-1}. The lemma is proved. ∎

Let us now finish the proof Theorem 6.7. We need to perform the induction step. This is done using the same argument as in the proof of Theorem 5.2. Apply a restriction from $R^+_{p_1,p_0,p_*,B}$ to the circuit. By Lemma 6.8 the defining circuit still computes a function as difficult as g^m_{k-1} and setting some of the remaining variables the circuit can be made into the defining circuit of g^m_{k-1}.

In the circuit of smaller depth we can by Lemma 6.3 with high probability change the order of $ANDs$ and ORs in the last two levels and still maintain a small bottom fanin. Thus we get a circuit which does not to exist by the induction hypothesis. ∎

7. Application to relativized complexity

In this chapter we will outline the connection between relativized complexity and lower bounds for small depth circuits. The proofs and theorems of this section are not due to the author of the thesis. The proofs that sufficiently strong lower bounds for small depth circuits computing certain functions would imply the existence of certain oracles, were obtained by Furst, Saxe and Sipser [FSS] and Sipser [Si]. The first sufficiently strong lower bounds to imply the existence of such oracles were obtained by Yao [Y2]. As some of the available proofs are somewhat sketchy, in this section we give the complete proofs.

Let us introduce the necessary notation.

Definition: An *oracle* A is a subset of Σ^*. A *Turing machine with oracle* A is a Turing machine with the extra feature that it can ask questions of the form: Is the string x in the set A. These questions are answered correctly in one timestep.

Let us introduce the model of an Alternating Turing Machine (ATM) [CKS], in order to define the polynomial time hierarchy (abbreviated as PH) [St].

An ATM is a nondeterministic Turing machine, whose states have one of four labels. The labels "accept" and "reject" occur on halting states. The non halting states are marked either by $\wedge$ or $\vee$. These states may have several possible next configurations. We will assume for notational convenience that the number of possible next configurations is at most 2. For each input x the computation can be viewed as a directed tree with the interior nodes marked with either $\wedge$ or $\vee$ and the leaves marked either "accept" or "reject". The machine *accepts* the input iff the root is evaluated to 1 under the natural evaluation of the tree. (Replace "accept" by 1, "reject" by 0 and evaluate the $\wedge$ and $\vee$ gates as logical functions). We make the following definition:

Definition: A Σ_i^p-*machine* is an ATM for which each path from the root to a leaf is of polynomial length and such that the number of consecutive alternating runs of $\wedge$ and $\vee$ along any path is at most i. The machine starts in a state labeled $\vee$.

We use the almost the same notation for languages.

Definition: A language L is in Σ_i^p iff if it is accepted by some Σ_i^p machine.

It is clear that $NP = \Sigma_1^p$. The polynomial time hierarchy is defined as

$$PH = \bigcup_{i=0}^{\infty} \Sigma_i^p$$

By allowing the alternating machine to have access to an oracle A we define the complexity classes $\Sigma_i^{p,A}$ of languages accepted by machines having at most i alternations and which run in polynomial time. PH^A is defined as $\bigcup_{i=0}^{\infty} \Sigma_i^{p,A}$. In a similar way $PSPACE^A$ is defined to be the set of languages which are accepted in polynomial space by a Turing machine with oracle A. One of the original motivations for [FSS] was to try to construct an oracle A such that $PSPACE^A \neq PH^A$. Let us first do some preliminaries.

Definition: A *weak* oracle machine is one which asks at most one oracle question on each

computation branch. Furthermore this question is asked at the end of the computation branch.

This might seem like a very weak machine but alternation is powerful.

Lemma 7.1: *For every oracle A, every $\Sigma_i^{p,A}$ language L is accepted by some $\Sigma_{i+2}^{p,A}$ weak oracle machine.*

Proof: Take a $\Sigma_i^{p,A}$ machine M^A which accepts L. We convert this into a weak Σ_{i+2}^{p} machine M_1^A accepting the same language. M_1^A will guess the answers to the questions that M^A asks along a computation branch and then verify the answers in the end of the branch using some extra alternations. Let us make this formal.

To simulate the answer to a query $x \in A$? asked by machine M^A, M_1^A proceeds as follows.

If M^A is asking the question while in an $\wedge$ state, M_1^A enters an $\wedge$ branch and assumes along one branch that the answer was 0 and along the other that it was 1. It remembers the question and its guess of the answer for the rest of the computation.

If M^A is asking the question while in an $\vee$ state, M_1^A enters an $\vee$ branch and assumes along one branch that the answer was 0 and along the other that it was 1. It again remembers the question and its guess of the answer for the rest of the computation.

When we arrive at a leaf of M^A we have to determine whether M_1^A accepts or rejects. Let Q_i denote the $i'th$ guess made by M_1^A for an oracle answer along the present branch. M_1^A accepts iff

$$([M^A \text{ accepts on this branch}] \wedge [\forall i\ Q_i \text{ is a correct guess}])$$
$$\vee\ (\ \exists i[Q_i \text{ is an incorrect and-guess}] \wedge [\forall j < i\ Q_j \text{ is a correct guess}])$$

The second line reflects the fact that we did not really want to do an $\wedge$ guess but rather an $\vee$ guess.

Claim: M_1^A accepts an input x iff M^A accepts x.

Look at the tree corresponding to the computation of M_1^A. It contains all the branches of M^A. These correspond to branches of M_1^A where only correct guesses were made. However we also have branches sticking out from this tree corresponding to incorrect guesses.

Observe now that as soon as we leave the tree of correct guesses the value of the branch is determined. If the first incorrect guess was made at an $\wedge$ branch the corresponding branch will accept by the second criteria. If the first incorrect guess was made at an $\vee$ branch the corresponding branch will reject. In neither case does the incorrect guess effect the value of the tree and hence the claim is established.

To finish the proof of the lemma we need only observe that M_1^A is a $\Sigma_{i+2}^{p,A}$ weak oracle machine. No new alternations are introduced by the internal guesses since we always use the same connective as the previous state. To realize the acceptance criteria we need only add two more alternations at the end. ∎

In the future we will identify an oracle A with a set of Boolean variables y_z^A defined for every $z \in \Sigma^*$ by $y_z^A = 1$ iff $z \in A$. Using this notation we can establish the relationship between circuits and weak oracle machines.

Lemma 7.2: *Let M^A be a $\Sigma_i^{p,A}$ weak oracle machine which runs in time t on input x. Then there is a depth i circuit C of size 2^t with has a subset of the y_z^A as inputs such that for every oracle A, M^A accepts x precisely when C outputs 1 on inputs y_z^A.*

Observe that the structure of the circuit depends on M^A and the input x.

Proof: Write down the computation tree of M^A on input x. At each leaf one of four things happens. M^A accepts without looking at the oracle, M^A rejects without looking at the oracle, M^A asks $z \in A$? and accepts iff $y_z^A = 1$ or M^A asks $z \in A$? accepts iff $y_z^A = 0$. In the corresponding circuit we will write the constant 1, the constant 0 , the variable y_z^A and the variable $\bar{y}_z^A$ respectively. By the definition of acceptance by an ATM the circuit

will output 1 precisely when the machine accepts x. By the definition of being in $\Sigma_i^{p,A}$ the computation tree does not have more than i alternations along any branch. This implies that the corresponding circuit can be collapsed to be of depth i. Finally since the depth of the original tree was $\leq t$ the size of the resulting circuit is bounded by 2^t. ∎

7.1 An oracle such that $PSPACE^A \neq PH^A$.

Having taken care of the preliminaries we go on to the essentials of this section.

Theorem 7.3 ([FSS], [Y2]): *There is an oracle A such that $PSPACE^A \neq PH^A$.*

Proof: Since clearly $PH^A \subseteq PSPACE^A$ for any A we want to display an oracle such that the inclusion is proper. The language which is in $PSPACE^A$ will not be in PH^A for a later determined choice of A is.

$$L(A) = \{1^n \mid \text{the number of strings of length } n \text{ in } A \text{ is odd } \}$$

The connection to parity is clear since to determine whether 1^n is in $L(A)$ is precisely to compute the sum mod 2 of the y_z^A, where $|z| = n$. This can clearly be done in polynomial space since we can ask the oracle about all strings of length n. Thus $L(A) \in PSPACE^A$ for all A and hence it remains to construct A to make $A \notin PH^A$. We will do this by diagonalization.

Let $M_i^A, i = 1, 2 \ldots$ be an enumeration of all weak oracle machines that belong to $\Sigma_j^{p,A}$ for some constant j. We construct A by determining the variables y_z^A in rounds. In round i we will make sure that M_i^A does not accept $L(A)$.

The idea of the construction is as follows. To recognize $L(A)$ M_i^A wants to compute the parity of $y_z^A, |z| = n$. However we know by Lemma 7.2 that the output of M_i^A corresponds to the output of a circuit with inputs y_z^A of constant depth and relatively small size. We know by Theorem 5.1 that such a circuit cannot compute parity.

Initialize $n_0 = 1$.

Round i. Suppose M_i^A runs in time cn^c and has j alternations. Find an m_i such

that $100cm_i^c < 2^{\frac{m_i}{j-1}}$ and $m_i > n_{i-1}$. Look at the computation tree of M_i^A on the input 1^{m_i}. By Lemma 7.2 the value of this computation is determined by the value of a circuit with inputs y_z^A for some z's. We know that the depth of this circuit will be j and the size will be at most $2^{cm_i^c}$. First substitute the values of all y_z^A which has been previously determined. Then substitute some arbitrary values (e.g. 0) for all other y_z^A with $|z| \neq m_i$. Now the remaining circuit is a circuit of depth j in the 2^{m_i} variables y_z^A with $|z| = m_i$. We know by Theorem 5.1 that a depth j circuit computing parity of 2^{m_i} variables have to be of size $2^{\frac{1}{10}\frac{j}{j-1}2^{\frac{m_i}{j-1}}}$. Thus by the choice of m_i the present circuit does not compute parity correctly. Give values to the $y_z^A, |z| = n$ such that the circuit does not compute parity on this input. Finally let n_i be the largest integer such that y_z^A has been determined for some z with $|z| = n_i$. Set $y_z^A = 0$ for all the so far undetermined variables with $|z| \leq n_i$.

Fact 1: A is well defined.

This is clear since A is uniquely determined by the variables y_z^A and each of these variables is given a value precisely once.

Fact 2: M_i^A does not decide $L(A)$ correctly on 1^{m_i}.

This is by construction. $1^n \in L(A)$ precisely when parity of the variables y_z^A, $|z| = n$ and A was chosen such that M_i^A did not compute parity of y_z^A, $|z| = m_i$.

Using these two facts we see that $L(A)$ is not accepted by any weak oracle machine which has a bounded number of alternations. Finally using Lemma 7.1 we have proved the theorem. ∎

7.2 An oracle such that $\Sigma_i^{p,A} \neq \Sigma_{i-1}^{p,A}$ for all i.

Having established the firm connection between oracle machines and circuits by Lemmas 7.1 and 7.2 it is natural to proceed and try to use Theorem 6.1 to get a similar result.

Theorem 7.4 ([Si], [Y2]): *There is an oracle A such that $\Sigma_i^{p,A} \neq \Sigma_{i-1}^{p,A}$ for all i.*

Remark 11: Yao claimed sufficient lower bounds to give Theorem 7.4. Our Theorem 6.1 gives the first proof of such lower bounds.

To prove Theorem 7.4 we first need a preliminary remark.

Lemma 7.5: *If $\Sigma_i^{p,A} \neq \Sigma_{i-3}^{p,A}$ for $i > 3$ then $\Sigma_i^{p,A} \neq \Sigma_{i-1}^{p,A}$ for all i.*

Proof: In fact $\Sigma_i^{p,A} = \Sigma_{i-1}^{p,A}$ implies that $PH^A = \Sigma_{i-1}^{p,A}$ and in particular $\Sigma_{i+2}^{p,A} = \Sigma_{i-1}^{p,A}$. ∎

Thus it is sufficient to construct an oracle satisfying the hypothesis of Lemma 7.5. The construction is almost identical to the previous one. Start by defining a suitable language.

$$L_i(A) = \{1^n | \exists x_1, x_2, \ldots x_{\frac{n}{i}} \forall x_{\frac{n}{i}+1} \ldots x_{\frac{2n}{i}} \exists \ldots \ x \in A\}$$

Thus $1^n \in L_i(A)$ iff the function $f_i^{2^{\frac{n}{i}}}$ of section 6 evaluates to 1 on the inputs y_z^A where $|z| = n$.

Since the i alternations are built into the definition of $L_i(A)$ it is clear that $L_i(A) \in \Sigma_i^{p,A}$. We will construct A such that $L_i(A) \notin \Sigma_{i-3}^{p,A}$. The proof is very similar to the proof of Theorem 7.3.

As before let $M_i^A, i = 1, 2 \ldots$ be an enumeration of $\Sigma_j^{p,A}$ weak oracle machines for all constants j.

The idea of the proof is that a weak oracle machine with $i-1$ alternations corresponds to a relatively small circuit of depth $i-1$ with inputs y_z^A. We know by Corollary 6.2 that a small circuit of depth i cannot compute $f_i^{2^{\frac{n}{i}}}$.

Initialize $n_0 = 1$.

Round i. Suppose M_i^A runs in time cn^c and has j alternations. Thus we want M_i^A not to accept $L_{j+1}(A)$. Find an m_i such that $10cm_i^c < 2^{\frac{m_i}{200j^2}}$ and $m_i > n_{i-1}$. Look at the computation tree of M_i^A on the input 1^{m_i}. This is a circuit of depth j and size at most $2^{cm_i^c}$ with inputs y_z^A for some z's. First substitute the values of all y_z^A which have

been previously determined. Then substitute some arbitrary values (e.g. 0) for all other y_z^A with $|z| \neq m_i$. Now the remaining circuit is a circuit in the 2^{m_i} variables y_z^A with $|z| = m_i$. We know by Corollary 6.2 that it is too small to compute $f_{j+1}^{2^{\frac{m_i}{j+1}}}$ correctly. Give a set of values to the y_z^A such that the circuit makes an error for this input. Finally let n_i be the largest integer such that y_z^A has been determined for some z with $|z| = n_i$. Set $y_z^A = 0$ for all the so far undetermined variables with $|z| \leq n_i$.

Fact 1: A is well determined.

This is clear since A is uniquely determined by the variables y_z^A and each of these variables is given a value precisely once.

Fact 2: No weak oracle machine with j alternations computes L_{j+1} correctly. More precisely, if the index of the machine is i it makes an error on 1^{m_i}.

By Lemma 7.1 no $\Sigma_{j-3}^{p,A}$ machine accepts $L_j(A)$ and hence we have proved Lemma 7.5 and the theorem follows. ∎

Of course it is possible to interleave the two constructions and get an oracle which achieves the two separations simultaneously. This gives us the following theorem.

Theorem 7.6 ([FSS], [Si], [Y2]): *There is an oracle B such that $PSPACE^B \neq PH^B$ and $\Sigma_i^{p,B} \neq \Sigma_{i-1}^{p,B}$ for all i.*

7.3 Separation for random oracles.

The above constructions give very little information what the oracles look like. One nice extra piece of evidence was provided by by Cai [Ca].

Theorem 7.7 ([Ca]): *With probability 1, $PSPACE^A \neq PH^A$ for a random oracle A.*

Let us first make clear what the theorem means. It is possible to look at the set $\mathcal{B}$ of oracles A satisfying $PSPACE^A \neq PH^A$. There is also a natural measure on the set

of subset of Σ^*, this measure corresponding to Lebesgue measure on $[0,1]$. The theorem states that $\mathcal{B}$ has measure 1. For related results see [BG].

Proof: (Due to Babai [Bab]) In view of the previous theorems and proofs it is not surprising that the theorem will rely on the following question. How well can small depth circuits compute parity? By this we mean for what fraction of the inputs can a small circuit be correct. The lemma that will be sufficient for our present purpose is the following.

Lemma 7.8: *For any constant k there is a constant c_k such that for $n > n_0^k$ a depth k circuit which computes parity correctly for 60% of the inputs is of size at least $2^{n^{c_k}}$. Here n_0 is an absolute constant.*

We will prove much stronger forms of Lemma 7.8 in the next section. However we will here establish this weaker result by a simpler proof. Lemma 7.8 follows from Theorem 5.1 and the following result of Ajtai and Ben-Or [AjBe].

Lemma 7.9: *Suppose there is a depth k circuit of size s which computes parity correctly for 60% of the inputs. Then for some absolute constant c there is a circuit of depth $k+4$ and size cns which computes parity exactly.*

For the sake of completeness we will give an outline of this result. Define a (p,q) family of circuits to be a set of circuits such that for any fixed input x a random member of this family outputs 1 on input x with probability $\geq p$ if $parity(x) = 1$ and outputs 1 with probability $\leq q$ if $parity(x) = 0$. Using the circuit C which is correct for 60% we will construct a $(.6,.4)$ family C_y. For $y \in \{0,1\}^n$ define C_y starting from C by

Interchange x_i and $\bar{x}_i$ iff $y_i = 1$.

Negate the resulting circuit if $\sum_{i=1}^{n} y_i = 1 \bmod 2$.

This will give a $(.6,.4)$ family since for every fixed input the probability that C_y is correct for a random y is precisely the probability that C is correct for a random input.

Construct a new family of circuits by taking the AND of $c \log n$ randomly chosen C_y.

For a suitably chosen c this gives a $(n^{-\frac{1}{2}+\epsilon}, n^{-\frac{1}{2}-\epsilon})$ family. Now take the OR of $n^{\frac{1}{2}}$ random members of this new family. This gives a $(1-e^{-n^{\epsilon}}, n^{-\epsilon})$ family. By choosing the correct parameters and continuing the construction for two more rounds it is possible to obtain a $(1-2^{-n-1}, 2^{-n-1})$ family.

Consider a random element from this $(1-2^{-n-1}, 2^{-n-1})$ family. The probability that it is incorrect for any particular input is 2^{-n-1}. Thus the probability that it is incorrect for any input is bounded by $\frac{1}{2}$. Thus in particular there is a member of this family which is correct for all inputs. This concludes the outline of the proof of Lemma 7.9.

Let us see how Theorem 7.7 follows from Lemma 7.9. We will again use the language

$$L(A) = \{1^n \mid \text{the number of strings of length } n \text{ in } A \text{ is odd } \}$$

To prove that $Pr[L(A) \in PH^A] = 0$ we only need to prove that $Pr[M^A \textit{ accepts } L(A)] = 0$ for any $\Sigma_i^{p,A}$ machine M^A. This is sufficient since there is only a countable number of machines.

Fix a machine M^A with running time cm^c and j alternations. Define m_1 such that $cm_1^c < 2^{m_1 c_j}$ with the constant c_j from Lemma 7.8. Then recursively define $m_i = cm_{i-1}^c + 1$. Define "M^A agrees with $L(A)$ on input x" to mean that M^A outputs 1 on input x and $x \in L(A)$ or M^A outputs 0 on input x and $x \notin L(A)$. Then we have

Lemma 7.10: *$Pr[M$ agrees with $L(A)$ on $1^{m_i} \mid M$ agrees with $L(A)$ on $1^{m_j}, j < i] \leq .6$.*

Proof: By the bound for the running time M^A cannot write down any string of length m_i during its computations on $1^{m_j}, j < i$. In particular it cannot ask the oracle about any such string. Thus the condition that M^A behaves correctly on these inputs shed no information on which strings of length m_i the oracle A contains. Thus look at the circuit corresponding to the computation of M^A on input 1^{m_i}. We know by Lemma 7.8 that for any fixed setting of the values of y_z for $|z_i| \neq m_i$ the probability that M^A gives the right answer is $\leq .6$. The lemma follows. ∎

Lemma 7.10 clearly implies that the probability that M^A agrees with $L(A)$ on

1^{m_i}, $i = 1, \ldots k$ is $\leq (.6)^k$. Thus the probability that M^A accepts $L(A)$ is 0. By the previous discussion Theorem 7.7 follows. ∎

To prove that a random oracle separates the different levels within the polynomial time hierarchy one would have to strengthen Corollary 6.2 to say that no depth $k-1$ circuit computes a function which agrees with f_k^m for most inputs. This is not true since if k is even the constant function 1 agrees with f_k^m for most inputs. However perhaps it is possible to get around this by defining other functions more suited to this application.

8. How well can we compute parity in small depth?

The information on the size of a circuit which computes parity exactly is now quite complete. As we saw in the end of the last section the question of on what fraction of the inputs a small constant depth circuit could compute parity correctly had some interesting consequences. We believe that the question is interesting in its own right and we will try to answer it as well as we can. Let us fix the following notation.

Definition: *Let $h(s,k,n)$ be the function such that no depth k circuit of size 2^s computes parity correctly for more than a $\frac{1}{2} + h(s,k,n)$ fraction of the inputs.*

Ajtai [Aj] was the first researcher to consider this question. He derived the best known results when the size is restricted to be polynomial. Using our notation he proved $h(c \log n, k, n) < 2^{-n^{1-\epsilon}}$ for all constants $c, k, \epsilon > 0$ and sufficiently large n. Cai [Ca] when proving the separation result proved a considerably stronger statement than we used in section 7, namely that $h(n^{\frac{1}{4k}}, k, n) = o(1)$. Together with Ravi Boppana we have obtained the following results.

Theorem 8.1: *The following bounds hold for* $h(s,k,n)$:

(i) $h(s,2,n) < 2^{-\Omega(\frac{n}{s})}$ for all s.

(ii) $h(s,k,n) < 2^{-\Omega(\frac{n}{s^{k-1}})}$ for $k > 2$ and $s > n^{\frac{1}{k}}$.

(iii) $h(s,k,n) < 2^{-\Omega((\frac{n}{s})^{\frac{1}{k-1}})}$ for $k > 2$ and $s < n^{\frac{1}{k}}$.

Remark: The first two bounds are optimal except for a constant.

Proof: The basic idea is to apply a random restriction from R_p and use our main lemma of section 4. Let us set up some further notation. Suppose that a function f agrees with parity for a fraction α of the inputs. Define the *advantage*, $\Delta(f)$ of f by $\Delta(f) = \alpha - \frac{1}{2}$. Observe that this number can be negative. By definition $h(s,k,n)$ is the maximal advantage of any function on n variables which can be computed by a circuit of size 2^s and depth k. Observe that we get the same bounds for $|\Delta(f)|$ as for $\Delta(f)$. This follows since we can negate a circuit without increasing its size.

We will need a basic property of how the advantage of a function behaves with respect to random restrictions. Let E denote expected value.

Lemma 8.2: $\Delta(f) \leq E(|\Delta(f\lceil_\rho)|)$ **for** $\rho \in R_p$.

Proof: Suppose that ρ give non $*$ values to the variables in the set S. Let $w(\sigma)$ denote the weight of σ i.e. the number of σ_i which are 1. Then

$$\Delta(f) = 2^{-|S|} \sum_{\sigma \in \{0,1\}^S} (-1)^{w(\sigma)} \Delta(f\lceil_\sigma)$$

This expression can be interpreted as an expected value over all possible assignments to S. Thus for every fixed S the inequality of Lemma 8.2 holds since introducing the absolute values only increases the right hand side. The fact that S is chosen randomly does not matter since the inequality is true for any fixed S. ∎

The way to use the above lemma to prove Theorem 8.1 is as follows. Suppose f is computed by a small circuit of depth k. Then for suitably chosen parameter p, $f\lceil_\rho$ will, with very high probability, be computed by a circuit of depth $k-1$. In this case we can

use induction. In the very few case where $f\lceil_\rho$ is not computed by a depth $k-1$ circuit we estimate the advantage by $\frac{1}{2}$. Thus the crucial parameter will be the probability of this failure. Even though it will be exponentially small it will prevent us from getting the optimal results for some ranges of s. Let us start by the case $k=2$ and small bottom fanin. For this case we get optimal results.

Lemma 8.3: *Any depth 2 circuit with bottom fanin bounded by t can not agree with parity for more than $\frac{1}{2}+2^{-\Omega(\frac{n}{t})}$ of the inputs.*

Proof: Assume that the circuit is an AND of ORs of size $\leq t$. We will use a slight modification of the proof of Lemma 4.7. Lemma 4.7 tells us that if we apply a restriction from R_p with $p=\frac{1}{10t}$ we can write the resulting function as an OR of ANDs of size $\leq r$ with probability $1-2^{-r}$. Looking more closely at the proof we see that the inputs accepted by the different ANDs form disjoint sets.

Now suppose that the number of remaining variables is m. Any AND of size $< m$ is satisfied by an equal number of odd and even strings. Thus for the circuit to accept a different number of odd and even strings it is necessary that there are some ANDs of full size.

The probability that less than $\frac{n}{20t}$ variables remain is $\leq 2^{-\Omega(\frac{n}{t})}$. The probability of getting any AND of size $\geq \frac{n}{20t}$ is $\leq 2^{-\Omega(\frac{n}{t})}$. Thus with probability $1-2^{-\Omega(\frac{n}{t})}$ the resulting circuit has no correlation with parity. Even if we assume that in the remaining case we get perfect agreement with parity we have established Lemma 8.3 since the probability of this case is $2^{-\Omega(\frac{n}{t})}$. ∎

However not all small circuit have only gates with small fanin. The next lemma takes care of large bottom fanin.

Lemma 8.4: $h(s,2,n)\leq 2^{-\Omega(\frac{n}{s})}$.

Ideally one would like to apply a random restriction to the circuit to get rid of the large bottom fanin. Unfortunately the probability of failure ($\approx 2^{-s}$) and this is way too

much if $s < n^{\frac{1}{2}}$. However in these simple cases when we only want to decrease the bottom fanin we can proceed by the greedy method. For notational convenience let us denote the implicit constant in Lemma 8.3 by c. We prove

Lemma 8.5: *A depth 2 circuit of size 2^s which has at most k gates of size $\geq 20s$ does not agree with parity for more than $\frac{1}{2} + (1+k)^{\frac{cn}{40s^2}} 2^{-\frac{cn}{20s}}$ of the inputs for $s > s_0$ for some absolute constant s_0.*

Lemma 8.5 clearly implies Lemma 8.4 since $k \leq 2^s$. Assume that the circuit is an AND of ORs, the other case can be handled similarly. Furthermore denote the function computed by the circuit by f. We will prove the lemma by induction over k and n. The base case $k = 0$ is already taken care of by Lemma 8.3 and $n = 0$ is trivial. Find the variable which appear in the most large clauses (A clause is large if it is of size at least $20s$). Clearly there is one which occurs in at least $\frac{20ks}{n}$ clauses. Without loss of generality let this variable be x_1. We will substitute values for x_1. It is true that

$$\Delta(f) = \frac{1}{2}(\Delta(f\lceil_{x_1=0}) - \Delta(f\lceil_{x_1=1}))$$

We estimate this number by induction. Let k_i denote the number of large clauses you get by substituting $i for x_1$, $i = 0, 1$. By induction the above expression is bounded by

$$\frac{1}{2}((1+k_0)^{\frac{c(n-1)}{40s^2}} 2^{-\frac{c(n-1)}{20s}} + (1+k_1)^{\frac{c(n-1)}{40s^2}} 2^{-\frac{c(n-1)}{20s}})$$
$$\leq (1+k)^{\frac{cn}{40s^2}} 2^{-\frac{cn}{20s}} (2^{\frac{c}{20s}-1}((\frac{1+k_0}{1+k})^{\frac{cn}{40s^2}} + (\frac{1+k_1}{1+k})^{\frac{cn}{40s^2}}))$$

Now since the variable x_1 occurs in at least $\frac{20sk}{n}$ large clauses either the literal x_1 or the literal $\bar{x}_1$ occurs in at least $\frac{10sk}{n}$ clauses. Assume without loss of generality that this is true for x_1. This implies that $k_1 < k(1 - \frac{10s}{n})$. Clearly $k_0 \leq k$. Thus

$$(\frac{1+k_0}{1+k})^{\frac{cn}{40s^2}} + (\frac{1+k_1}{1+k})^{\frac{cn}{40s^2}} \leq 1 + (1 - \frac{5s}{n})^{\frac{cn}{40s^2}} \leq 1 + e^{-\frac{c}{8s}}$$

Finally

$$2^{\frac{c}{20s}-1}(1 + e^{-\frac{c}{8s}}) < 1$$

for sufficiently large s and the lemma follows.■

For general k we will proceed in the same matter, first establishing the theorem for the case when the bottom fanin is bounded.

Lemma 8.6: *A circuit of depth k, size 2^s and bottom fanin s can not compute parity correctly for a fraction of the inputs which is greater than*

(i) $\frac{1}{2} + 2^{-\Omega(\frac{n}{s^{k-1}})}$ for $s > n^{\frac{1}{k}}$.

(ii) $\frac{1}{2} + 2^{-\Omega((\frac{n}{s})^{\frac{1}{k-1}})}$ for $s < n^{\frac{1}{k}}$.

Proof: Suppose that the circuit computes the function f. We prove the lemma by induction over k. We have already established the base case $k = 2$. To do the induction step apply a random restriction from R_p with $p = \frac{1}{10s}$ to the circuit and use Lemma 8.2. Let $r = \max(s, (\frac{n}{s})^{\frac{1}{k-1}})$. Observe that $r = s$ precisely when $s > n^{\frac{1}{k}}$. By our main lemma with probability $1 - 2^{-\Omega(r)}$, $f\lceil_\rho$ can be computed by a circuit of depth $k-1$, size 2^s and bottom fanin bounded by r.

With probability $1 - 2^{-\Omega(\frac{n}{s})}$ the number of remaining variables is at least $\frac{n}{10s}(1-\epsilon)$. If these two conditions are true then by induction with $s = r$ the advantage is bounded by $2^{-\Omega(\frac{\frac{n}{s}}{r^{k-2}})}$. Observe that now we are always in case (i) by out choice of r.

In the case where there either remain too few variables or we cannot decrease the depth of the circuit we estimate the advantage by $\frac{1}{2}$. Since the probability of this happening is $2^{-\Omega(r)}$ we have the the following estimate for $\Delta(f)$:

$$2^{-\Omega(\frac{\frac{n}{s}}{r^{k-2}})} + 2^{-\Omega(r)}$$

Substituting the two possibilities of r we get Lemma 8.6. ■

Finally we need a last lemma.

Lemma 8.7: *We have the following bounds for $h(s,k,n)$:*

(i) $h(s,k,n) \leq 2^{-\Omega(\frac{n}{s^{k-1}})}$ for $k > 2$ and $s > n^{\frac{1}{k}}$.

(ii) $h(s,k,n) \leq 2^{-\Omega((\frac{n}{s})^{\frac{1}{k-1}})}$ for $k > 2$ and $s < n^{\frac{1}{k}}$.

Proof: The proof is very similar to the proof of Lemma 8.5. One uses the power $\frac{n}{s^d}$ for $s > n^{\frac{1}{k}}$ and $\frac{n^{\frac{1}{k-1}}}{s^{\frac{k}{k-1}}}$ in the other cases. ∎

Finally observe that we have proved Theorem 8.1 since part (i) is Lemma 8.4 and parts (ii) and (iii) is Lemma 8.7. ∎

9. Is majority harder than parity?.

When proving lower bounds for parity circuits it was crucial that the gates in the circuit were AND and OR gates. A natural extension would be to allow parity gates of arbitrary fanin. In this case parity trivially has small circuits. It is far from clear whether there are more efficient circuits computing majority. It is not too difficult to see that computing parity is easy given gates computing majority.

If we can compute majority, we can compute the function "at least k" for any k and by negating this function one gets the function "at most k" for any k. Taking AND of these two functions we get the function "exactly k" and finally the OR of circuits computing "exactly k" for odd $k \leq n$ is parity of n variables. In fact majority is as hard as any symmetric function. We will now provide a weak piece of evidence that majority might be harder to compute than parity.

Theorem 9.1: *A circuit containing AND, OR and parity gates of constant depth d, size $2^{c(\log n)^{\frac{3}{2}}}$ which computes majority, contains at least $\Omega((\log n)^{\frac{3}{2}})$ parity gates. Here $c < c_d$, where c_d is a constant depending on d.*

The reason that makes this theorem harder to prove and somewhat weak in conclusion is that parity gates are quite different from AND and OR gates. An AND gate which has a 0 as input will output 0 no matter what the other inputs are. To know the output of a parity gate we must always know all the inputs. In some sense this makes the parity gates very powerful. Thus we need different techniques.

Proof: Let us first give an outline of the proof. We will get rid of the parity gates by "guessing" the outputs of them. To be more precise we will look at all possible outcomes of the parity gates. Next we will hit all resulting circuits with one restriction from $R_{\sqrt{n}^{-1}}$. We will think of this restriction as d restrictions each being picked randomly from $R_{n^{-\frac{1}{2d}}}$. Using the main lemma of section 4 we get that all these circuits are transformed into circuits which are very simple and cannot compute anything even remotely resembling majority. Finally we establish that at least one of the settings of the parity gates did not introduce too much error. Let us make this formal.

Number the occurring parity gates by $G_1, G_2, \ldots G_t$, $t \leq c(\log n)^{\frac{3}{2}}$. Let $\alpha \in \{0,1\}^t$ and let C_α be the circuit obtained by substituting α_i for G_i for all i. This is a usual depth d circuit of size $2^{c_k(\log n)^{\frac{3}{2}}}$ containing only AND and OR gates. These circuits do not compute majority any more but we will prove that at least one of them computes something that is reasonably close. Let F_α be the function computed by C_α

Let the i'th parity gate G_i have d_i inputs. These inputs are in general subcircuits. In each of these subcircuits substitute values for its own parity gates as prescribed by α. Call the resulting subcircuit $C_{\alpha,i,j}$ for $1 \leq i \leq t, 1 \leq j \leq d_i$. Denote the function computed by $C_{\alpha,i,j}$ by $F_{\alpha,i,j}$. In section 3 we introduced the concept of a minterm. Here we also need the dual concept of a maxterm.

Definition: A *maxterm* of a function f is a minimal assignment that forces f to be 0.

We have the following lemma

Lemma 9.2: *Let ρ be a random restriction from $R_{n^{-\frac{1}{2}}}$. Then for $n > n_0$ for some constant n_0 with a positive probability all the following are true.*

(i) $F_\alpha\lceil_\rho$ *has minterms and maxterms of size* $\leq \frac{1}{2}\sqrt{\log n}$ *for all* α.

(ii) $F_{\alpha,i,j}\lceil_\rho$ *has minterms and maxterms of size* $\leq \frac{1}{2}\sqrt{\log n}$ *for all* α, i, j.

(iii) ρ *gives out the same number of* 0*'s and* 1*'s.*

(iv) ρ *gives out at least* $\frac{\sqrt{n}}{2}$ $*$*'s.*

Proof: First use the following observation.

Lemma 9.3: *Let ρ_1 be a random restriction from R_{p_1} and ρ_2 be a random restriction from R_{p_2} on the variables given the value $*$ by ρ_1. Then applying ρ_1 and then ρ_2 gives the same distribution as using a random restriction from $R_{p_1 p_2}$.*

Proof: Under the composition of ρ_1 and ρ_2 a variable gets the value $*$ precisely when it receives the value $*$ under both restrictions. The probability of this event is hence $p_1 p_2$. To see that we get the same distribution as $R_{p_1 p_2}$ we just observe that the variables are treated independently and the probability of 0 is equal to the probability of 1. ∎

Let us return to the proof of Lemma 9.2. By repeatedly applying Lemma 9.3 we can interpret the restriction ρ as d consecutive restrictions from R_p with $p = n^{-\frac{1}{2d}}$. Let us start by analyzing the probability of failing (i). Fix first an individual F_α. We reason as we did in proof that parity had large circuits. Using our main lemma with $s = t = \frac{1}{2}\sqrt{\log n}$ and $p = n^{-\frac{1}{2d}}$ $d-1$ times we see that we can reduce C_α to be of depth two with bottom fanin $\frac{1}{2}\sqrt{\log n}$ with probability $1-(d-1)(5pt)^s = 1-2^{c_d(\log n)^{\frac{3}{2}}}$. We even have a stronger statement that all minterms are small. If we apply a final restriction from R_p we get with the same high probability that also all maxterms are bounded in size by $\frac{1}{2}\sqrt{\log n}$. Thus the probability of failure for any individual F_α is $2^{-\Omega((\log n)^{\frac{3}{2}})}$ and we have $2^{c(\log n)^{\frac{3}{2}}}$ different α. If we have chosen c small enough the probability of failing (i) is $2^{-\Omega((\log n)^{\frac{3}{2}})}$. The probability of failing (ii) can be estimated by the same bound in the same way.

Continuing, the probability of failing the condition (iv) is bounded by $2^{-\Omega(\sqrt{n})}$ by standard argument. Finally the probability of success of (iii) is $\approx \frac{1}{\sqrt{n}}$. But since the probability of failing any of the other conditions is exponentially small we have proved Lemma 9.2. ∎

We will need an important consequence of having both small minterms and maxterms.

Lemma 9.4: *If a function has minterms only of size $\leq s$ and maxterms only of size $\leq t$ then it depends on at most $2^{st}-1$ variables.*

Proof: We prove the lemma by induction over s and t. The lemma is clearly true for $s = t = 1$. Now suppose that $t > 1$. Observe now that any minterm must at least one variable in common with every maxterm (Otherwise we could simultaneously force the function to be 0 and 1.). Take any minterm σ which hence is of size at most s. Now try all the 2^s possible settings of the variables in σ. Call the resulting functions f_β for $\beta \in \{0,1\}^s$. We will bound the sizes of the minterms and maxterms of the f_β.

Let δ be a minterm of f_β. By definition δ together with the assignment prescribed by β forces the f to 1. Thus f has a minterm σ_1 which is a subassignment of δ plus β. By the fact that δ is a minterm σ_1 must contain all of δ. Thus the size of δ is bounded by the size of σ_1 which is bounded by assumption by s.

Now the same reasoning can be applied to the maxterms. However in this case we know that the corresponding maxterm of f has to have some variable in common with the substituted minterm σ. Using this fact we can bound the size of the maxterms of f_β by $t - 1$.

Thus by induction each of f_β only depends on at most $2^{s(t-1)} - 1$ variables. Thus f only depends on at most $s + 2^s(2^{s(t-1)} - 1) \leq 2^{st} - 1$ variables. ∎

Now let us recall a well known lemma which we prove for the sake of completeness.

Lemma 9.5: *Each boolean function f of m variables can be written as a $GF(2)$ polynomial of degree at most m.*

Proof: Let $\beta \in \{0,1\}^m$. Define

$$g_\beta(x) = \prod_{i=1}^{m}(x_i + \beta_i + 1)$$

Then $g_\beta(\beta) = 1$ while $g_\beta(x) = 0$ for $x \neq \beta$. Note that g_β is a polynomial of degree m. To prove the lemma observe that

$$\sum_{\beta, f(\beta)=1} g_\beta(x)$$

is a polynomial of degree at most m which takes the same values as f. ∎

In what follows we always think of a polynomial over $GF(2)$ as a sum of monomials with all variables occurring to at most their first power. This is possible since $x^i = x$ for all $i \geq 1$ for $x \in GF(2)$. Let us combine Lemma 9.4 and Lemma 9.5 and state conditions (i) and (ii) in Lemma 9.2 in a different manner.

Lemma 9.6: *The conditions (i) and (ii) of Lemma 9.2 imply that $F_\alpha\lceil_\rho$ and $F_{\alpha,i,j}\lceil_\rho$ can be written as polynomials of degree at most $n^{\frac{1}{4}}$ for all α, i and j.*

Proof: By Lemma 9.4 the functions depend on at most $n^{\frac{1}{4}}$ variables and the degree bound follows from Lemma 9.5. ∎

The true value of G_i is $\sum_{j=1}^{d_i} F_{\alpha,i,j}$. Since the degree does not increase when we add two polynomials this means that the value of G_i is a polynomial $P_{\alpha,i}(x)$ of degree at most $n^{\frac{1}{4}}$. Now we can make precise in what way the circuits C_α almost computes majority.

Lemma 9.7: *The circuit C_α computes majority correctly except for those inputs x where $\prod_{i=1}^{s}(P_{\alpha,i}(x) + \alpha_i + 1) = 0$.*

Proof: If the product is nonzero then each term is nonzero. Thus if the product is nonzero then $P_{\alpha,i}(x) + \alpha_i + 1 = 1$ for all i and replacing the parity gates by the α_i did not change the circuit. ∎

Define $P_\alpha = \prod_{i=1}^{s}(P_{\alpha,i}(x) + \alpha_i + 1) = 0$. Note that the degree of P_α is bounded by $n^{\frac{1}{4}} c(\log n)^{\frac{s}{2}}$. We will sometimes call P_α the error polynomials.

The reason that the degree bound will give us information is the following lemma.

Lemma 9.8: *Let $P(x)$ be a polynomial over $GF(2)$. If majority of m variables agrees with the constant function 1(0) except when $P(x) = 0$ then either $P(x) \equiv 0$ or the degree on P is at least $\frac{m}{2}$.*

Proof: We give the proof in the case of 1, the other case being similar. The condition implies that $P(x) = 0$ for all x of weight at most $\frac{n}{2}$. Assume that P is nonzero and let $\prod_{i \in S} x_i$ be a monomial of minimal degree of P. S is here a subset of $\{1, 2, \ldots n\}$. Look at

P evaluated at $x_i = 1, i \in S, x_j = 0,\ j \notin S$. This value must clearly be 1. Thus $|S| > \frac{n}{2}$. ∎

There are a two problems in using Lemma 9.8 to prove Theorem 9.1. Firstly, the remaining circuit does not compute the constant function and secondly we do not know that the polynomial in question is nonzero.

The first problem can be taken care of as follows. If we replace the final circuit by a constant b we only make an error where $F_\alpha + b + 1 = 0$. Since this is a polynomial of degree $\leq n^{\frac{1}{4}}$ we only add this polynomial the product defining the error polynomial obtaining a new error polynomial $P_{\alpha,b}$.

The second problem comes from the fact that polynomials over $GF(2)$ is not an integral domain i.e. a product can be zero even if none of the factors is 0. (e.g. $x(1+x) = 0$). Even if the proof is not difficult let us state it as a lemma.

Lemma 9.9: *There is an α and a b such that $P_{\alpha,b}(x) \not\equiv 0$.*

Proof: Take an arbitrary input x to the original circuit $C\lceil\rho$. For this input the parity gates evaluate to some values α_i and the circuit evaluates to b. The polynomial $P_{\alpha,b}$ is nonzero, in particular it takes the value 1 at x. ∎

Let us see how the lemmas fit together to finish the proof of Theorem 9.1. We take the C_α which corresponds to the α of Lemma 9.9. Replace the circuit by the constant b. Now supposedly majority of the remaining variables is b except when $P_{\alpha,b}(x) = 0$. This is a contradiction by Lemma 9.8. ∎

CONCLUSIONS

We have established exponential lower bounds for the size of small depth circuits. However there are still many open questions concerning small depth circuits. It seems that the lower bounds proved for the size of circuits computing parity, are essentially the best bounds one could hope to obtain for any function if the tool used is our main lemma.

It would be very interesting to establish lower bounds which are better than our bounds for parity on the size of constant depth circuits computing some NP-complete function like clique. Hopefully, a proof establishing such lower bounds would not only rely on the restrictive nature of the model but also on the difficulty of computing the clique function. Such a proof would shed light on what makes a function difficult to compute.

Another interesting direction for future research is to extend the rather weak results of section 9. Adding parity gates to the circuit seems to complicate the situation considerably. Our main lemma seems to be an insufficient tool and to get substantially better results we believe that new insight is required.

One of the few open problems of this thesis which potentially might not need new ideas but only a bit of cleverness is to prove that $\Sigma_i^{p,A} \neq \Sigma_{i-1}^{p,A}$ for a random oracle A. One way to prove this is to prove that there are functions that are computable by small depth k circuits which cannot be approximated well by small circuits of smaller depth. Here smaller depth can be interpreted as depth $d(k)$ for any function $d(k)$ which goes to infinity with k.

Finally a question of another type is to determine the complexity of inverting a NC^0 permutation. By the results in section 2 this is at least as hard as P, and clearly the problem is in $NP \bigcap coNP$. We conjecture that this problem is not in P.

References

[Aj] Ajtai M. "$\sum_1^1$-Formulae on Finite Structures", *Annals of Pure and Applied Logic* 24(1983) 1-48

[AjBe] Ajtai M. and Ben-Or M. "A theorem on probabilistic constant depth computations" *Proceedings of 16th Annual ACM Symposium on the Theory of Computation*, Washington D.C. 1984, pp.471-477.

[AlBo] Alon N. and Boppana R. "The Monotone Circuit Complexity of Boolean Functions", Submitted to *Combinatorica.*

[An] Andreev A.E. "On one method of obtaining lower bounds of individual monotone function complexity" *Dokl. Ak. Nauk.* 282 (1985), pp 1033-1037.

[Bab] Babai L. "Random oracles separate PSPACE from the polynomial time hierarchy" Technical Report 86-001, University of Chicago, January 1986.

[Bar] Barrington D. personal communication 1985

[Be] Beame P. "Limits on the Power on Concurrent-Write Parallel Machines", to appear in *Proceedings of the 18th Annual ACM Symposium on Theory of Computation* 1986.

[BG] Bennet C. and Gill J. "Relative to a Random Oracle A, $P^A \neq NP^A \neq co-NP^A$ with Probability 1", *SIAM Journal on Computing 10* (1981), 96-113.

[BM] Blum M. and Micali S. "How to generate Cryptographically Strong Sequences of Pseudo Random Bits", *Proceedings of 23'rd IEEE symposium on Foundations of Computer Science* 1982 pp 112-117. Also *SIAM J. on Computing* 13 (1984) pp 850-864.

[Bo1] Boppana R. "Threshold Functions and Bounded Depth Monotone Circuits" *Proceedings of 16th Annual ACM Symposium on Theory of Computing*, 1984, 475-479. To appear in *Journal of Computer and System Sciences*, 1986.

[Bo2] Boppana R. Personal communication 1986.

[BL] Boppana R. and Lagarias J. "One way functions and circuit complexity" to be presented a Structure in Complexity conference.

[Ca] Cai J. "With Probability One, a Random Oracle Seperates PSPACE from the Polynomial Time Hierarchy", to appear in *Proceedings of the 18th Annual ACM Symposium on Theory of Computation* 1986.

[CKS] Chandra A.K, Kozen D.C. and Stockmeyer L.J. "Alternation", *Journal of the ACM 28*, 1 (January 1981), pp 114-133.

[Co] Cook S. "A taxonomy of problems with fast parallel algorithms", , preprint , 1985.

[FSS] Furst M., Saxe J. and Sipser M., "Parity, Circuits, and the Polynomial Time Hierarchy", *Proceedings of 22nd Annual IEEE Symposium on Foundations of Computer Science*, 1981, 260-270.

[GM] Goldwasser S. and Micali S., "Probabilistic Encryption" *Journal of Computer and System Sciences* Vol. 28, No.2 pp 270-299.

[KPPY] Klawe M.,Paul W, Pippenger N. and Yannakakis M. "On Monotone Formulae with Restricted Depth" *Proceedings of 16th Annual ACM Symposium on Theory of Computing*, 1984, 480-487.

[La] Ladner, R.E., "The Circuit Value Problem is Log Space Complete for P", SIGACT News, vol. 7, No 1, Jan 1975 (18-20).

[Le] Levin, L.A., "One-Way Functions and Pseudorandom Generators", *Proceedings of the 17'th Annual ACM Symposium on Theory of Computing* 1985 pp 363-365.

[Lu] Lupanov O. "Implementing the algebra of lgoic functions in terms of constant depth formulas in the basis $\&, \wedge, \neg$", *Dokl. Ak. Nauk. SSSR* 136 (1961), 1041-1042 (in Russian). English translation in *Sov. Phys. Dokl* 6 (1961), 107-108.

[M] Moran S., "An Improvement of the Hastad-Yao Lower Bound on Parity Circuits",

preprints.

[PF] Pippenger N. and Fischer M.J., "Relationships Among Complexity Measures", IBM Research Report RC-6569 (1977), Yorktown Heights.

[R] Razborov A.A. "Lower Bounds for the Monotone Complexity of some Boolean Functions" *Dokl. Ak. Nauk.* 281 (1985), pp 798-801.

[RT] Reif J.H. and Tygar J.D., "Efficient Parallel Pseudo-Random Number Generation", presented at Crypto 85.

[Si] Sipser M. "Borel Sets and Circuit Complexity", *Proceedings of 15th Annual ACM Symposium on Theory of Computing*, 1983, 61-69.

[St] Stockmeyer L.J. "The Polynomial-Time Hierarchy", *Theoretical Computer Science 3* 1977, 1-22.

[V] Valiant L. "Exponential Lower Bounds for Restricted Monotone Circuits" *Proceedings 15th Annual ACM Symposium on Theory of Computing*, 1983, 110-117.

[Y1] Yao A. "Theory and Applications of Trapdoor Functions", *Proceedings of 23'rd IEEE Symposium on Foundations of Computer Science* (1982) pp 80-91.

[Y2] Yao A. "Separating the Polynomial-Time Hierarchy by Oracles" *Proceedings 26th Annual IEEE Symposium on Foundations of Computer Science*, 1985, 1-10.