Regular Resolution Lower bounds for the Weak Pigeonhole Principle

Ran Raz*
Department of Computer Science
Weizmann Institute
ranraz@wisdom.weizmann.ac.il

Toniann Pitassi†
Department of Computer Science
University of Toronto
toni@cs.toronto.edu

Abstract

In this paper, we prove exponential lower bounds for regular resolution proofs of the weak pigeonhole principle. As a corollary, we show that a particular version of the propositional statement that NP does not have polynomial-sized circuits requires exponential-size regular resolution proofs.

1 Introduction

In the last three decades, a tremendous amount of research has been directed towards understanding the mathematical structure of the satisfiability problem, and to developing algorithms for satisfiability testing and propositional theorem proving. Much of this research has centered around the method of *resolution*. The *resolution principle* says that if C and D are clauses and x is a variable, then any assignment that satisfies both of the clauses $C \vee x$ and $D \vee \neg x$ also satisfies $C \vee D$. The clause $C \vee D$ is said to be a *resolvent* of the clauses $C \vee x$ and $D \vee \neg x$ *on the variable* x. A *resolution refutation* for a CNF formula F consists of a sequence of clauses $C_1, C_2, \ldots, C_s$ where (i) each clause C_i is either a clause of F, or is a resolvent of two previous clauses and (ii) C_s is the empty clause, denoted Λ. We can represent the proof as an acyclic directed graph on vertices $C_1, \ldots, C_s$ where each clause of F has out-degree 0,

*Research supported by US-Israel BSF Grant 95-00238.
†Research supported by NSF grant CCR 9820831, US-Israel BSF Grant 95-00238, and an NSERC grant.

and any other clause has two arcs pointing to the two clauses that produced it. It is well known that resolution is a *sound* and *complete* propositional proof system, i.e., a formula F is unsatisfiable if and only if there is a resolution refutation for F. Resolution is the most widely studied approach to propositional theorem proving, and there is a large body of research exploring *resolution algorithms*, i.e., algorithms that on input an unsatisfiable formula F, output a resolution refutation of F.

A well-known restricted version of resolution that is still complete is called regular resolution. In a regular resolution proof, along any path in the directed acyclic graph, each variable can be resolved upon at most once.

There are now several very strong lower bounds for resolution, showing that various formulas require exponential-size resolution proofs, beginning with the seminal paper by Haken [Hak85]. A series of results culminating with that of Ben-Sasson and Wigderson [BT88, BKPS98, Urq87, BSW] greatly simplify Haken's method, clarifying that to prove lower bounds using this method, it is necessary and sufficient to prove that any resolution proof of the given formula must have a wide clause.

Despite this progress, there are several formulas (which seemingly have no short resolution proofs) for which the above proof method breaks down, because the wide-clause lemma for these formulas is simply false. The most ubiquitous and important example where the known proof methods break down is the weak pigeonhole principle, $WPHP_n^{2n}$, stating that there is no one-to-one mapping from $2n$ pigeons to n holes. (Here, one can replace $2n$ by any cn.) The weak pigeonhole principle is a general and basic combinatorial principle that underlies the induction principle, is used in most probabilistic counting arguments, and underlies many lower bounds in circuit complexity. Additionally, much of elementary number theory (including the existence of infinitely many primes) can be formalized in weak systems with the weak pigeonhole principle. Finally, Razborov has shown that even in weak propositional proof systems, efficient proofs asserting that P does not have polynomial-sized circuits, imply efficient proofs of the weak pigeonhole principle. Thus, lower bounds for WPHP usually imply lower bounds for a propositional formulation of $P/poly \neq NP$.

What is known about the resolution complexity of the weak pigeonhole principle, $WPHP_n^m$, as a function of m? When m is large enough (approximately $2^{\sqrt{n}\log n}$), Buss and Pitassi have shown that there are short resolution proofs of $WPHP_n^m$. When $m \leq n^2/\log n$, superpolynomial lower bounds were proven by Buss and Turan ([BT88]), using the wide-clause method mentioned above. However, when m is in the middle range, such as when $m = n^2$, no nontrivial resolution lower bounds are known. Partial progress was made by Razborov, Wigderson and Yao [RWY97] who give exponential lower bounds in m, but where the resolution proof is of a restricted form.

In this paper, we show that any regular resolution proof of the weak pigeonhole principle requires exponential size in m. As a corollary, we show that any regular resolution proof that SAT has polynomial-size circuits also requires exponential size.

2 Definitions and preliminaries

2.1 Resolution and Branching Programs

A *literal* is either an atom or the negation of an atom. A *clause* is the disjunction of a set of literals. As mentioned in the introduction, a *resolution refutation* of a CNF formula f is a sequence of clauses, such that each clause is either from f or is derived from two previous clauses by the resolution rule, and such that the final clause is the empty clause. The *size* of a resolution proof is the number of clauses in it. A resolution proof is *regular* if along every path from an initial clause to the final clause, every atom is resolved at most once.

We will exploit the equivalence [Kra96] between regular resolution proofs and read-once branching programs to prove our lower bound. For any unsatisfiable formula

$$F = C_1(p_1, \ldots, p_n) \wedge \ldots \wedge C_k(p_1, \ldots, p_n)$$

let us consider the following search problem S_F: Given a truth assignment $a \in \{0,1\}^n$, find some v such that $C_v(a) = 0$.

A (2-way or boolean) branching program in n variables is a directed acyclic graph with one source node s (sometimes called the root), with every non-sink node labelled with one of the variables $x_1, \ldots, x_n$ and with exactly two outgoing edges, labelled 0 and 1 respectively. Every input string $a \in \{0,1\}$ determines a computational path $comp(a)$ from s to a sink node. A boolean branching program solves a search problem with inputs from $\{0,1\}^n$ if its sinks can be labelled by possible solutions to the search problem in such a way that for every $a \in \{0,1\}^n$, $comp(a)$ leads to a sink labelled by a solution admissible for the input a. The *size* of a branching program is the total number of nodes. A branching program is *read-once* if along every path p, every variable is tested (appears as a node label) at most once. For a path p on a read-once branching program, $Vars(p)$ denotes the set of variables appearing as node labels on p.

A read-once branching program is *uniform* if for each path p beginning at the root s, $Vars(p)$ depends only on the terminal node v of p, and for every sink node t, $Vars(t)$ contains all variables.

It is not hard to see that any read-once branching program can be made uniform with little increase in the size:

Proposition 2.1 *Any branching program in n variables can be simulated by a uniform branching program whose size is larger by at most a factor of n.*

Proof:
The proof is by induction on the level of the branching program. Clearly the proposition is true for all level zero sink nodes. Now assume that the proposition is true for all nodes up to level l, and let x be a node at level $l+1$, and let $y_1, \ldots y_k$ be the nodes that are children of x. By induction, we know that for each $i \leq k$, for every path p from the root node to y_i, $Vars(p)$ is the same set, so we will call this set $Vars(y_i)$. For each $i \leq k$, let $Vars(i)$ be the set of variables queried along a path to x through y_i, and let ALL be $\cup_i Vars(i)$. The

intuition is as follows: since the branching program is read-once, once we arrive at node x, we cannot ask about any variable in ALL, so we may as well have already asked about each of these variables along any path to x. Formally, for each i, we will add a single dummy path where we will query all variables in $ALL - Vars(y_i)$, and such that the dummy path will end at node x. The resulting branching program will increase by at most a factor of n, will still be read once, and will have the property that for every node x, there exists a set $Vars(x)$ such that for each path p from the source node to x, the variables read along p are equal to $Vars(x)$. To finish off the proof, if for any sink node s, $Vars(s)$ does not equal the entire set of variables, then again we will add a dummy path of queries to the remaining variables.

The following theorem due to Krajíček gives the connection between branching programs and regular resolution.

Lemma 2.1 *Let F be an unsatisfiable CNF formula. Then the minimum size of any regular resolution proof of F is equal to the minimum size of any read-once branching program solving the search problem S_F.*

For our lower bound, we will only need the easy direction of Krajicek's lemma, namely that any regular resolution proof of F can be transformed into a read-once branching program for S_F. For this direction, the BP is obtained by retaining the structure of the resolution proof, and if we resolve on x, the BP queries x. The proof that the transformation works is by induction on the size of the Resolution proof. For the base case, the resolution proof has size one, and is therefore the empty clause; in this case the BP is trivial. Assume now that we have a regular resolution proof $S_1, \cdots S_{k+1}$, and inductively, we have a read-once BP corresponding to $S_1 \ldots S_k$. Also assume that $S_{k+1} = (C \vee D)$ was derived from $S_i = (C \vee x)$ and $S_j = (D \vee \neg x)$. Any truth assignment that falsifies $(C \vee D)$ falsifies either $(C \vee x)$ or $(D \vee \neg x)$ and which one can be decided by querying x.

For the other direction, let B be the read-once branching program for S_F. We will define a corresponding regular resolution proof by induction on the depth of B. The sink nodes of B are labelled by clauses of F, and these will be the initial clauses of the regular resolution proof. Now for the intermediate nodes, suppose that a node a of B is labelled by the atom p, with outgoing edges (labelled 0 and 1 respectively) connected to nodes b and c respectively. Suppose that $Clause(b)$ and $Clause(c)$ are the two clauses already defined inductively, corresponding to nodes b and c. Then $Clause(a)$ is obtained by applying the resolution rule to $Clause(b)$ and $Clause(c)$ on atom p.

Now we want to show that the resulting sequence of clauses is in fact a regular resolution proof of F. It is clear that the sink clauses are clauses of F, and that each intermediate clause follows from two previous clauses by the resolution rule, and that the proof is regular. To show that the final source node corresponds to the empty clause, we prove by induction (on l) that for any node x at level l of the branching program, for all truth assignments a, if the computational path $comp(a)$ passes through node x, then a falsifies $Clause(x)$.

2.2 The weak pigeonhole principle

The propositional pigeonhole principle, PHP_n^m, states that there is no one-to-one mapping from m pigeons to n holes. The underlying boolean variables, $x_{i,j}$, $i \leq m$ $j \leq n$ represent whether or not pigeon i is mapped to hole j. The negation of the pigeonhole principle, $\neg PHP_n^m$ is expressed in conjunctive normal form (CNF) as the conjunction of *pigeon* clauses and *hole* clauses. There are m pigeon clauses, $(x_{i,1} \vee \ldots \vee x_{i,n})$, for $i \leq m$, stating that pigeon i maps to some hole. There are $\binom{m}{2} \cdot n$ hole clauses, $(\neg x_{i1,j} \vee x_{i2,j})$, for $i1, i2 \leq m$, $i1 \neq i2$, $j \leq n$, stating that no two pigeons $i1$ and $i2$ both map to hole j.

For a set $S \subset [n]$, we will use the notation $x_{i,S}$ to denote the disjunction of variables $x_{i,j}$, $j \in S$.

3 The Lower Bound for Branching Programs

3.1 Overview of proof

There is a spectrum of good tactics that can be used to prove the weak pigeonhole principle, when the number of pigeons (m) is large compared to the number of holes (n). For all known upper bounds, the queries are made one hole at a time. That is, the branching program is levelled into n phases, where in phase 1, we can ask about the variables $x_{i,1}$, $i \leq m$, and in phase 2 we ask about $x_{i,2}$, $i \leq m$ and so on. Let us assume further that we mainly care about the behavior of our branching program on those inputs where only the pigeon axioms are violated. Our branching program will remember negative information, where at any point in the program, we will remember that a set of variables $x_{i,j}$ have all been set to zero. Consider the first phase of the branching program where we ask questions about hole 1. One straightforward tactic is to consider only the minimal number of pigeons, some $n+1$ of them, and to ask all queries pertaining to these pigeons and hole 1, and to remember all negative information. This will mean remembering a set of n pigeons, and the knowledge that none of these pigeons are mapped to hole 1. In the second phase, we ask about all of the remaining n pigeons and hole 2, and in this round we remember which $n-2$ pigeons are not mapped to holes 1 and 2. Finally, after n phases, we know one particular pigeon that was not mapped to any hole. Note however that the size of this branching program is exponential.

On the other hand, if we are willing to utilize more pigeons, then we can remember less. For example, if we ask about $m \geq 2n$ pigeons and hole 1, then we can divide the relevant pigeons into two equal sized groups, A and B, and just remember that either no pigeon in A was mapped to hole 1, or remember that no pigeon in B was mapped to hole 1. However, in order to continue this algorithm for n phases, we require that m is exponential in n initially.

An intermediate strategy (between the above two extremes) might remember that one group of pigeons $S_1, S_2, \ldots S_k$ does not map to hole 1, where each group has size at least n, and the intersection of the groups is disjoint.

These upper bounds demonstrate that a lower bound argument must simultaneously block different kinds of programs: those that learn a lot about a small number of pigeons ($n+1$ pigeons), versus those that learn more general information about larger blocks of pigeons.

Because of the different ways of making progress, it will be useful to generalize the problem by adding additional axioms of type k, for every k, $1 \leq k \leq n^{\epsilon}$. An axiom of type k is a set A of pigeons, $|A| = m_k = 2^{n^{\epsilon}-k}$, and for each pigeon a_i in A, a corresponding set of holes H_i, $|H_i| = n_k = kn^{1-\epsilon}$, and the axiom is the disjunction of the variables x_{a_i,H_i}. In other words, to falsify this axiom, we must know that none of the pigeons a_i in A map to any of their respective holes in H_i. We will intuitively think of an axiom of type k as an n_k-by-m_k block of variables. By adding these axioms, we make the branching program search problem potentially easier, since now we are done whenever we reach either an original axiom, or one of the new axioms, and therefore the lower bound for the original weak pigeonhole principle will follow from a lower bound with these additional axioms. In what follows we will assume that the axioms are this extended set.

The intuition behind the argument is as follows. Assume for sake of contradiction that there is a read-once branching program of small size solving the search problem associated with these axioms. We will define a distribution on inputs (based on this branching program) such that for each extended axiom, the probability of arriving at this axiom with respect to the distribution is exponentially small, and thus if the branching program has small size, it cannot be correct.

We will define the distribution through an adversary strategy. That is, for each node and path of the branching program, we will specify the probability of following each outgoing edge from that node. Our strategy will have the property that the probability of arriving at a hole axiom will be zero, and for each k, the probability of arriving at a pigeon axiom of type k will be exponentially small. To simultaneously insure that pigeon axioms of each type (corresponding to the different ways of making progress) are not reached very often, we will partition the *holes* into groups $S_1, \ldots S_k$. This partitions the *nodes* of the branching program into k types in the natural way; a node that queries a hole from S_j will be of type j. We will choose the partition probabilistically in such a way that for each node u and for each pigeon p and for each type k, the number of queries made about p within S_j at node u is in proportion to the number of queries asked about p within all of $[n]$. Now there will be different strategies for each of the k types; the nodes of type k will collectively insure that any type k axiom is reached with exponentially small probability.

What should the strategy be for type k axioms? To reach a type k axiom, we know that there are m_k pigeons such that at least n_k queries must be made to each of these pigeons. We will therefore be interested in a *relevant* interval of queries for a type k axiom; that is, a node u of type k is relevant for pigeon i if at node u, the number of queries asked so far about pigeon i is in the interval $I_k = 1/3n^{1-3\epsilon}(k-1)$ and $1/3n^{1-3\epsilon}k$. (Think of the relevant interval for type k as those nodes where we are close to getting to a type k axiom, but are still far enough away that we can now actively avoid getting there.) Since node u is of type k and is not a terminating node, we also know that it is not relevant for too many pigeons i since if it was relevant for more than $2m_k$ pigeons, then u would be a type $k-1$ axiom.

Therefore, for a node u of type k not in the relevant interval, we want to follow the zero path with probability 1. The other case is when u is in the relevant interval where node u queries $x_{i,j}$, and the number of queries to i made so far is in the interval I_k. In this case, we want to follow the edge setting $x_{i,j}$ to 1 with reasonable probability, to prevent reaching a type k axiom easily. If will follow the edge setting $x_{i,j}$ to 1 with probability $p = 1/2m_k$, and the edge setting $x_{i,j}$ to 0 with probability $1 - 1/2m_k$, then the overall probability of arriving at the type k axiom saying that A is not mapped to an n_k block of holes will be at most $(1 - 1/2m_k)^{m_k n_k}$, which is exponentially small as desired.

Of course the above strategy becomes complicated because we also have to make sure that we never end the game early by arriving at a hole axiom, stating that some two pigeons are mapped to the same hole. To be sure that this does not happen, we need to also remember for each node u the set of $x_{i,j}$'s that have been set to one (along every path to u). The above strategy needs to be modified so that for each node u, querying a variable $x_{i,j}$, if some variable $x_{i',j}$ has been set to 1 along every path to node u, then at node u we always follow $x_{i,j} = 0$ with probability one.

The crux of the argument is to show that when we remember the set of 1's for each node, and change the adversary strategy accordingly to avoid hole axioms, it does not destroy the probability distribution too much. This complicates the argument, since we can only show that for at least half of the m pigeons, the probability distribution is not changed too much.

The intuition behind this is as follows. First, a node cannot remember too many 1's since otherwise very few truth assignments make it to that node in the first place. Therefore, we can assume now that a node remembers very few 1's. Further, the hard case of interest is when some variables $x_{i',j}$ are set to 1 and then very shortly thereafter (more precisely within the relevant interval for k and i), we ask about conflicting variables $x_{i,j}$. Consider a particular path p. Suppose that along p a query $x_{i',j}$ is made where j has type k, and on our path, $x_{i',j}$ is set to 1. Since it has been set to one, it was a good variable in that we were not forced to set it to zero. Now when $x_{i',j}$ is set to 1, this has the potential of forcing at most $2m_k$ future $x_{i,j}$'s to 0 during this stage. But the probability of setting the variable to 1 in the first place is $1/2m_k$. Therefore, in the expectation, we create at most one bad (forced) pigeon for every good one. Thus in expectation, the number of bad pigeons is at most half the total number.

In order to be sure that *all* of the m pigeons do not fill up too quickly (that is, for each k and for each pigeon p, we have very small probability of learning that pigeon p is not mapped to a large group of holes within S_k, where large is relative to k), the nodes working against type k axioms will be further divided into phases. In the first phase, we will make sure that at least half of the m pigeons do not fill up too quickly. In the second phase, we will insure that half of the remaining $m/2$ pigeons ($m/4$ in total) do not fill up too quickly, and so on. To accomplish this, we further partition the holes into groups $S_{k,l}$, where $k \leq n^\epsilon$ and $l \leq 3n^\epsilon$. As before, this partition of the holes gives rise to a natural partition of the nodes of the branching program. The groups $S_{k,*}$ will act against type k axioms. Again, the partition will be probabilistically chosen so that for every node u, pigeon p and k, l, the number of variables queried at node u about pigeon p in $S_{k,l}$ is in the right proportion to the number of variables queried at node u about pigeon p in all of $[n]$. As before, we define a relevant interval $I_{k,l}$ of queries for each type (k, l), with the property that the interval of queries for (k, l) are after that for (k', l') whenever $k > k$, or $k' = k$ and $l' > l$.

Let a node u be of type (k, l) if it queries a hole in $S_{k,l}$. Intuitively a pigeon is *stubborn* with respect to a node u of type (k, l) and a path p to u if: u is in the relevant interval with for pigeon i and type (k, l) (that is, we know that a substantial number of queries to pigeon i have already been made), and during all of the first $l - 1$ stages for i, k, l, for most of the queries to this pigeon the adversary strategy had to follow the 0-path in order to avoid satisfying a hole axiom. We will show inductively on l, that at each phase the number of stubborn pigeons drops by half, using the reasoning discussed above. That is, the particular group of nodes of type (k, l) will be the l^{th} round of nodes acting against type k axioms. Any node of type (k, l) will be working against a group of stubborn pigeons of size at most $m/2^{l-1}$, and after round l, the number of pigeons left that will still be stubborn will be reduced by half, to roughly $m/2^l$.

3.2 Basic notations

In this section we prove our lower bound on the size of branching programs for the weak pigeon hole principle. We denote by n the number of holes, and by m the number of pigeons. We denote by $Holes$ the set of holes, and by $Pigeons$ the set of pigeons. That is,

$$\mathbf{Holes} = \{1, ..., n\}.$$

$$\mathbf{Pigeons} = \{1, ..., m\}.$$

We will usually denote a hole by j, and a pigeon by i. By $x_{i,j}$ we denote the variable corresponding to pigeon i and hole j. By $Hole_j$ we denote the set of variables corresponding to the j^{th} hole, and by $Pigeon_i$ the set of variables corresponding to the i^{th} pigeon. That is,

$$\mathbf{Hole_j} = \{x_{i,j} | i \in Pigeons\}.$$

$$\mathbf{Pigeon_i} = \{x_{i,j} | j \in Holes\}.$$

We will consider read once branching programs that query the variables $x_{i,j}$. By u we will usually denote a node in the branching program (BP). We say that $u' < u$ if there is a path in the BP from u' to u. By p we will usually denote a path in the BP. We denote by ϵ a small fixed constant (say $\epsilon = 1/20$). For simplicity, we assume that n is large enough (say $n \geq 10^{100}$). We will assume that our BP is of size $< 2^{n^\epsilon/10}$, and we will show that such a BP cannot solve the weak pigeon hole principle (we do not attempt here to optimize the value of ϵ). Note that since the size of the BP is lower than $2^{n^\epsilon/10}$, we can assume w.l.o.g. that $m < 2^{n^\epsilon/10}$ as well. For simplicity, we assume that expressions like $n^\epsilon, n^{1-\epsilon}/2$, etc., are all integers.

By Proposition 2.1, we can assume that the set of variables queried along each path to a node u is the same (i.e., the set of queries is independent of the path and depends only on the final node u). For a non-leaf node u define:

Label(u) = the variable $x_{i,j}$ queried at u.

For any node u, define:

Queries(u) = the set of variables $x_{i,j}$ queried along paths to u (not including the variable $Label(u)$). As mentioned above, by proposition 2.1, this set is independent of the path taken to u.

Queries$_i$(u) = $Queries(u) \cap Pigeon_i$.

For any node u, define $Ones(u)$ to be the set of variables that the node u "remembers" to be 1, and $Zeros(u)$ to be the set of variables that the node u "remembers" to be 0, that is:

Ones(u) = the set of variables $x_{i,j}$ in $Queries(u)$ that get the value 1 along every path to u.

Zeros(u) = the set of variables $x_{i,j}$ in $Queries(u)$ that get the value 0 along every path to u.

We can now define for any node u,

OpenPigeons(u) = the set of pigeons i with $Queries_i(u) \subset Zeros(u)$.

Claim 3.1 *If $u' \leq u$ then*

$$OpenPigeons(u) \subseteq OpenPigeons(u').$$

Proof:
Let $i \in OpenPigeons(u)$. For any $x_{i,j} \in Queries_i(u')$,

$$x_{i,j} \in Queries_i(u') \subset Queries_i(u) \subset Zeros(u).$$

Hence, $x_{i,j}$ gets the value 0 along every path to u. Since $u' < u$ and $x_{i,j} \in Queries_i(u')$, $x_{i,j}$ must get the value 0 along every path to u'. Hence, $x_{i,j} \in Zeros(u')$. Since this is true for any $x_{i,j} \in Queries_i(u')$, we conclude that $Queries_i(u') \subset Zeros(u')$, and hence, $i \in OpenPigeons(u')$. □

3.3 Types of axioms and types of holes

For integer $1 \leq k \leq n^\epsilon$, define

$$n_k = k \cdot n^{1-\epsilon},$$

and

$$m_k = 2^{n^\epsilon - k}$$

(recall that we assume that $n^\epsilon, n^{1-\epsilon}$ are integers). We will say that a node u is a pigeon-axiom of order k if there is a set $A \subset OpenPigeons(u)$, such that $|A| = m_k$ and for every $i \in A$, $|Queries_i(u)| \geq n_k$. Note that a pigeon-axiom of order $k = n^\epsilon$ is just a standard pigeon-axiom of the pigeon hole principle. We say that a node u is a hole-axiom if there exists a hole j and two different pigeons i_1, i_2, such that $x_{i_1,j}, x_{i_2,j} \in Ones(u)$. Note that this is just a standard hole-axiom of the pigeon hole principle. In our proof, we allow our BP to find a pigeon-axiom of any order k (and not only pigeon-axioms of order $k = n^\epsilon$ as in the usual weak pigeon hole principle). That is, we say that the BP solves the weak pigeon hole principle if it always stops at an axiom (i.e., all leaves of the BP are either hole-axioms or pigeon-axioms of some order). We assume w.l.o.g. that in our BP no non-leaf node is an axiom (otherwise, the BP can just stop at that node). In particular, no non-leaf node is a pigeon-axiom of order k, for any k.

Let $S_1 \cup \cdots \cup S_{n^\epsilon}$ be a random partition of $Holes$ into n^ϵ sets of size $n^{1-\epsilon}$ each. The intuition is that the set of holes S_k will be used against pigeon-axioms of order k. For each $1 \leq k \leq n^\epsilon$, let $S_{k,1} \cup \cdots \cup S_{k,n^\epsilon}$ be a random partition of S_k into n^ϵ sets of size $n^{1-2\epsilon}$ each.

Let I be the segment of integers $[1, ..., n^{1-2\epsilon}]$, We think of I as a set of indices. Let $I_1, ..., I_{3n^\epsilon}$ be the partition of I into $3n^\epsilon$ segments of length $n^{1-2\epsilon}/3n^\epsilon = (1/3) \cdot n^{1-3\epsilon}$ each. That is,

$$I_r = \left[(1/3) \cdot n^{1-3\epsilon} \cdot (r-1) + 1, ..., (1/3) \cdot n^{1-3\epsilon} \cdot r\right].$$

For each $1 \leq r \leq 3n^\epsilon$, let $I_{r,1}, ..., I_{r,3n^\epsilon}$ be the partition of I_r into $3n^\epsilon$ segments of length $(1/3) \cdot n^{1-3\epsilon}/3n^\epsilon = (1/9) \cdot n^{1-4\epsilon}$ each. That is,

$$I_{r,s} = [b_{r,s}, ..., e_{r,s}],$$

where

$$b_{r,s} = (1/3) \cdot n^{1-3\epsilon} \cdot (r-1) + (1/9) \cdot n^{1-4\epsilon} \cdot (s-1) + 1,$$

and

$$e_{r,s} = (1/3) \cdot n^{1-3\epsilon} \cdot (r-1) + (1/9) \cdot n^{1-4\epsilon} \cdot s.$$

Denote

$$\hat{n} = |I_{r,s}| = (1/9) \cdot n^{1-4\epsilon}.$$

The relevant set of indices for S_k will be I_{3k-1}. The relevant set of indices for $S_{k,l}$ will be $I_{3k-1,3l-1}$. Define:

$\mathbf{Queries_i^{k,l}(u)}$ = the set of variables $x_{i,j} \in Queries_i(u)$ such that $j \in S_{k,l}$.

$\mathbf{Stage_i^{k,l}}$ = the set of all nodes u such that $|Queries_i^{k,l}(u)| + 1 \in I_{3k-1,3l-1}$ (that is, $b_{3k-1,3l-1} \leq |Queries_i^{k,l}(u)| + 1 \leq e_{3k-1,3l-1}$).

$\mathbf{Over_i^{k,l}}$ = the set of all nodes u such that $|Queries_i^{k,l}(u)| + 1 > e_{3k-1,3l-1}$.

$\mathbf{Before_i^{k,l}}$ = the set of all nodes u such that $|Queries_i^{k,l}(u)| + 1 < b_{3k-1,3l-1}$.

The following claim shows that it is possible to choose a partition $\{S_{k,l}\}$ such that for every node u, for every pigeon i and for every k, l, the number of queries asked at u about pigeon i within $S_{k,l}$ is proportional to the number of queries asked at u about pigeon i within all of $[n]$.

Claim 3.2 *With exponentially high probability (over the choice of the partition $\{S_{k,l}\}$), the following is satisfied for every k, l, i, u:*

1.
$$n^{-2\epsilon} \cdot |Queries_i(u)| < |Queries_i^{k,l}(u)| + \hat{n}/2.$$

2.
$$n^{-2\epsilon} \cdot |Queries_i(u)| > |Queries_i^{k,l}(u)| - \hat{n}/2.$$

Proof:
Let a partition S be *bad* if there exists a set $S_{k,l}$ in the partition, and a set $Queries_i(u)$, such that the size of the intersection of $S_{k,l}$ and $Queries_i(u)$ is not within a reasonable range of the expected value. That is, if $|Queries_i^{k,l}(u)| \geq n^{-2\epsilon} \cdot |Queries_i(u)| + \hat{n}/2$ or $|Queries_i^{k,l}(u)| \leq n^{-2\epsilon} \cdot |Queries_i(u)| - \hat{n}/2$.

It suffices to show that $Pr(\text{S is bad})$ is exponentially small over all partitions. This probability is at most

$$\sum_{k,l,i,u} Pr[(S_{k,l}, Queries_i(u)) \text{ is bad}],$$

where the probability is over all random partitions. But this probability is equal to the same equation, but where the probability is over all subsets S of size exactly $n^{1-2\epsilon}$.

Fix a particular k, l, i, u. We will first upper bound the probability that $(S_{k,l}, Queries_i(u))$ is bad. The expected size of the intersection of $S_{k,l}$ and $Queries_i(u)$ is $|Queries_i(u)|n^{-2\epsilon}$. Let S be a random variable representing the domain of $S_{k.k}$. By Chernoff-Hoeffding bounds on the tail of the hypergeometric distribution, we have

$$Pr[|Queries_i(u) \cap S| \geq \alpha n] \leq e^{-2(\alpha-p)^2 n},$$

where pn is the expectation. Thus,

$$Pr[|Queries_i(u) \cap S| \geq |Queries_i(u)|n^{-2\epsilon} + \hat{n}/2] \leq e^{-2n^{1-8\epsilon}/18^2}.$$

Using similar bounds on the tail of the hypergeometric distribution from the other end, we also acheive exponentially small probability that $|Queries_i(u) \cap S| \leq |Queries_i(u)|n^{-2\epsilon} - \hat{n}/2]$.

The overall probability is therefore the above probability times the number of choices of k, l, i, u. The number of k and l is $3n^{\epsilon}$; the number of i is at most $2^{n^{\epsilon}/10}$, and the number of u is at most $2^{n^{\epsilon}/10}$. Thus, the total number of choices for k, l, i, u is at most $92^{n^{\epsilon}/10} \cdot 2^{n^{\epsilon}/10} \cdot n^{\epsilon} \cdot n^{\epsilon}$, which is much less than $e^{2n^{1-8\epsilon}/18^2}$.

□

Claim 3.3 *With exponentially high probability (over the choice of the partition $\{S_{k,l}\}$), the following is satisfied for every k, l, k', l', i, u:*

1. $$|Queries_i^{k,l}(u)| < |Queries_i^{k',l'}(u)| + \hat{n}.$$

2. $$|Queries_i^{k,l}(u)| > |Queries_i^{k',l'}(u)| - \hat{n}.$$

Proof:
Immediate from Claim 3.2. □

We say that $(k', l') < (k, l)$ if either $(k' < k)$ or $(k' = k$ and $l' < l)$.

Claim 3.4 *With exponentially high probability (over the choice of the partition $\{S_{k,l}\}$), the following is satisfied for every i, u and every $(k', l') < (k, l)$:*

1. $$u \in Stage_i^{k,l} \Longrightarrow u \in Over_i^{k',l'}.$$

2. $$u \in Stage_i^{k',l'} \Longrightarrow u \in Before_i^{k,l}.$$

Proof:
Immediate from Claim 3.3 and the definitions of $Stage_i^{k,l}$, $Before_i^{k,l}$, $Over_i^{k,l}$. □

Assume from now on that the partition $\{S_{k,l}\}$ is some fixed partition satisfying Claim 3.2, Claim 3.3 and Claim 3.4.

> We say that a variable $x_{i,j}$ is of **type** $(\mathbf{k}, \mathbf{l})$ if $j \in S_{k,l}$. We say that a non-leaf node u is of **type** $(\mathbf{k}, \mathbf{l})$ if $Label(u)$ is of type (k, l).

The reason for the intervals is as follows. If u is a type l axiom, there must exist at least m_l pigeons, such that for each such pigeon i, $|Queries_i(u)|$ is at least n_l. And thus by Claim 3.2, for each pigeon i, and type k, l, $|Queries_i^{k,l}(u)|$ is roughly $n^{-2\epsilon} n_l$. The relevant set of indices, $I_{3k-1,3l-1}$ for $S_{k.l}$, intuitively, will be a relatively large interval $(I_{k,l})$ of the roughly $n^{-2\epsilon} n_l$ queries that nodes of type (k, l) will act upon in order to insure that we do not arrive at a type l axiom very often. The definition $Stage_i^{k,l}$ identifies the set of nodes u of type k, l in the BP such that the number of queries to i within $S_{k,l}$ $(|Queries_i^{k,l}(u)|)$ is in this relevant interval, $I_{k,l}$. Note that the interval of relevant indices for (k, l) are always before the interval for $(k + 1, l)$ since the strategy proceeds in stages. Also, the intervals are separated by gaps to be sure that since our measurements are being taken with respect to $S_{k,l}$ rather than $[n]$, the actual segments are nonoverlapping.

3.4 The adversary strategy

We will define a probabilistic adversary strategy to answer queries of the BP. The answer given by the strategy depends not only on the node u but also on the path p that was taken to the node u (this is done for simplicity). In all the following, p is a path from the root to the node u (in some cases p will be a path starting from the root and u any node on it, and then we just think of p as a path to u and ignore the rest of it). We will define the strategy by defining it separately for each set $S_{k,l}$. The strategy for $S_{k,l}$ will answer queries $x_{i,j}$ of type (k,l). The answer $Strategy(p,u)$ for a node u of type (k,l) will be defined by induction on l, together with a set of pigeons $Stubborn^{k,l}(p,u)$. Thus, the definition of the strategy is actually done in $n^{\epsilon} \cdot n^{\epsilon}$ steps. Note that in step (k,l) of the definition, $Strategy(p,u)$ will be defined only for nodes u with $Label(u) = x_{i,j}$ such that $j \in S_{k,l}$. In contrast, the set $Stubborn^{k,l}(p,u)$ will be defined for any node u. First define for all k, p, u, the set

$$Stubborn^{k,0}(p,u) = Pigeons.$$

We will now assume that $Stubborn^{k,l-1}(p,u)$ is already defined and we will define the answer $Strategy(p,u)$ for nodes u of type (k,l). Let u be a node in the BP, with $Label(u) = x_{i,j}$ of type (k,l). Let p be a path from the root to u.

> We say that u (with $Label(u) = x_{i,j}$ of type (k,l)) is **Relevant** for the path p (from the root to u) if all the following are satisfied:
>
> 1. $i \in OpenPigeons(u)$.
> 2. $i \in Stubborn^{k,l-1}(p,u)$.
> 3. $u \in Stage_i^{k,l}$.
>
> If u is *Relevant* for p and there exists i' such that $x_{i',j} \in Ones(u)$ we say that for the path p, $x_{i,j}$ is **Forced** to 0 at u by $x_{i',j}$.

We now define $Strategy(p,u)$ in the following way:

> **Strategy(p, u)**, (for u with $Label(u) = x_{i,j}$ of type (k,l)):
>
> - If u is not *Relevant* for p answer 0.
> - If for the path p, $x_{i,j}$ is *Forced* to 0 at u answer 0.
> - Otherwise, answer 1 with probability $1/2m_k$ and 0 with probability $1 - (1/2m_k)$.

The set of pigeons $Stubborn^{k,l}(p,u)$ is now defined (for any node u) in the following way:

> $\mathbf{Stubborn^{k,l}(p,u)}$ = the set of pigeons i, such that all the following are satisfied:
>
> 1. $i \in Stubborn^{k,l-1}(p,u)$.

2. $u \in Over_i^{k,l}$.

3. At least $0.9 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were $Forced$ to 0 at some node of the path p (up to the node u).

Claim 3.5 *Let p be a path from the root to u and let $u' < u$ be another node on the path p. Then for every k, l,*

$$Stubborn^{k,l}(p,u') \subseteq Stubborn^{k,l}(p,u).$$

Proof:
The proof is by induction on l. The base case, $l = 0$, follows by the definition of $Stubborn^{k,0}(p,u)$. For $l > 0$, let $i \in Stubborn^{k,l}(p,u')$. We will show that $i \in Stubborn^{k,l}(p,u)$.

1. By the inductive assumption, $i \in Stubborn^{k,l-1}(p,u')$ implies $i \in Stubborn^{k,l-1}(p,u)$.

2. If $u' \in Over_i^{k,l}$ and $u' < u$ then obviously $u \in Over_i^{k,l}$ (by the definition of $Over_i^{k,l}$).

3. Since $u' < u$, if at least $0.9 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were $Forced$ to 0 at some node of the path p up to u' then all these variables were $Forced$ to 0 at some node of the path p up to u.

Hence, $i \in Stubborn^{k,l}(p,u)$. □

Claim 3.6 *Let p be a path from the root to u and let $u' < u$ be another node on the path p. Let k, l, i be such that $u, u' \in Over_i^{k,l}$ and $i \in Stubborn^{k,l}(p,u)$. Then*

$$i \in Stubborn^{k,l}(p,u').$$

Proof:
The proof is by induction on l. The base case, $l = 0$, follows by the definition of $Stubborn^{k,0}(p,u')$. For $l > 0$, let us show that $i \in Stubborn^{k,l}(p,u)$ implies $i \in Stubborn^{k,l}(p,u')$.

1. By Claim 3.4, $u, u' \in Over_i^{k,l-1}$. Hence, by the inductive assumption, $i \in Stubborn^{k,l-1}(p,u)$ implies $i \in Stubborn^{k,l-1}(p,u')$.

2. $u' \in Over_i^{k,l}$ by the assumption.

3. Since $u' \in Over_i^{k,l}$, every node in the path p after the node u' is also in $Over_i^{k,l}$ and hence is not $Relevant$ for the path p. Therefore, if at least $0.9 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were $Forced$ to 0 at some node of the path p up to u then all these variables were $Forced$ to 0 at some node of the path p up to u'.

Hence, $i \in Stubborn^{k,l}(p,u')$. □

Let **Path** be the random path defined by applying the adversary strategy for answering questions of the BP and let **Leaf** be the random leaf of the BP reached by $Path$.

3.5 The main claim

The following is the main claim needed for our lower bound. The rest of the subsection is dedicated for the proof of the claim.

Claim 3.7 *With probability exponentially close to 1, for every k, l,*

$$|Stubborn^{k,l}(Path, Leaf)| \leq (0.9)^l \cdot m.$$

Proof:
The proof is by induction on l. The base case, $l = 0$, is true (with probability 1) by the definition of $Stubborn^{k,0}(p, u)$.

We say that a path p is **Cheating** if for some $u \in p$, $|Ones(u)| \geq 2n^\epsilon$.

Claim 3.8 *The probability that $Path$ is $Cheating$ is at most 2^{-n^ϵ}.*

Proof:
First note that by the definition of $Strategy(p, u)$, the adversary strategy never answers a query by 1 with probability more than 1/2 (i.e., the probability for 0 is always at least 1/2). We say that a node u is $Cheating$ if $|Ones(u)| \geq 2n^\epsilon$. Let u be a node that is $Cheating$. Then there exist at least $2n^\epsilon$ variables $x_{i,j}$ in $Ones(u)$. Each one of these variables is in $Queries(u)$ and it gets the value 1 along every path to u. In order to have $u \in Path$, each one of these variables must be queried along $Path$ (and must get the value 1 along $Path$). Since the adversary strategy answers by 1 with probability of at most 1/2, the probability that all these variables get the value 1 along $Path$ is at most 2^{-2n^ϵ}. Hence, the probability that one particular $Cheating$ node u satisfies $u \in Path$ is at most 2^{-2n^ϵ}. Since the total number of nodes in the BP is (by our assumption) at most $2^{n^\epsilon/10}$, the probability that there exists a $Cheating$ node $u \in Path$ is at most $2^{n^\epsilon/10} \cdot 2^{-2n^\epsilon} < 2^{-n^\epsilon}$. □

Let p be a path starting at the root. Define:

$\mathbf{Forced^{k,l}(p)}$ = the set of variables $x_{i,j}$ of type (k, l) such that $x_{i,j}$ was $Forced$ to 0 at some node of the path p.

$\mathbf{Tough_i^{k,l}(p)}$ = the set of variables $x_{i,j}$ of type (k, l) such that all the following are satisfied:

1. For some $u \in p$ and some pigeon i', $x_{i,j}$ was $Forced$ to 0 at u by $x_{i',j}$.
2. $x_{i',j}$ was queried at $u' \in p$ such that $u' \in Before_i^{k,l}$.

As mentioned in the overview, a path is cheating if it remembers a lot of one's. Any cheating path has very low probability of being followed since it must be consistent with a large setting of one's, and therefore, we can focus our attention on the non-cheating paths.

Now consider a path that is not cheating, and consider nodes that are working against axioms of type l. Initially (for type $(1,l)$ nodes) the set of stubborn pigeons is all m of them. We want to show that for each k, the set of stubborn pigeons for the nodes of type (k,l) is at most half of the set of stubborn pigeons for nodes of type $(k,l-1)$. In order to do this, we need to understand how a stubborn pigeon i stays stubborn at the next stage. This happens if for many of the queries $x_{i,j}$ to a stubborn pigeon i during stage (k,l) $x_{i,j}$ had to be forced to 0 because previously some $x_{i',j}$ was set to 1 and remembered. Now there are two ways in which a variable $x_{i,j}$ gets forced to 0: either (i) a variable $x_{i',j}$ was set to one during this same stage and later in the same stage the BP asks about $x_{i,j}$. (That is, $x_{i',j}$ was queried by a node $u \in Stage_i^{k,l}$); or (ii) a variable $x_{i',j}$ was set to 1 during a previous stage and remembered for a long time. When case (ii) happens, variable $x_{i,j}$ is called tough. The following claim shows that as long as a path is not cheating, there cannot be too many tough variables.

Claim 3.9 *If a path p is not Cheating then for every k,l,i,*

$$|Tough_i^{k,l}(p)| < 2n^{\epsilon}.$$

Proof:
Denote (in the proof of this claim) $n' = 2n^{\epsilon}$. Let v be the first node on p such that $v \in Stage_i^{k,l}$. Assume for a contradiction that $|Tough_i^{k,l}(p)| \geq n'$. Then there exist n' different nodes $u_1, ..., u_{n'} \in p$, and n' different nodes $u'_1, ..., u'_{n'} \in p$, and n' different variables $x_{i,j_1}, ..., x_{i,j_{n'}}$ of type (k,l), and n' different variables $x_{i'_1,j_1}, ..., x_{i'_{n'},j_{n'}}$ (of type (k,l)), such that for every $1 \leq r \leq n'$:

1. x_{i,j_r} was *Forced* to 0 at u_r by $x_{i'_r,j_r}$.
2. $x_{i'_r,j_r}$ was queried at u'_r and $u'_r \in Before_i^{k,l}$.

By (1) we know that $u_1, ..., u_{n'} \in Stage_i^{k,l}$. Hence,

$$v \leq u_1, ..., u_{n'}.$$

By (2) we know that $u'_1, ..., u'_{n'} \in Before_i^{k,l}$. Hence,

$$u'_1, ..., u'_{n'} < v.$$

By (2), each variable $x_{i'_r,j_r}$ was queried at u'_r and hence

$$x_{i'_r,j_r} \in Queries(v).$$

By (1), each variable $x_{i'_r,j_r}$ satisfies $x_{i'_r,j_r} \in Ones(u_r)$ and hence

$$x_{i'_r,j_r} \in Ones(v).$$

Therefore,

$$|Ones(v)| \geq 2n^{\epsilon},$$

that is, the path p is *Cheating*. □

The other case, case (i) is when a path p is not cheating, and pigeon i is stubborn for (k,l), and variables $x_{i,j}$ are forced to 0 during this stage because some variable $x_{i',j}$ was set to one and remembered during this same stage. (That is, both $x_{i,j}$ and $x_{i',j}$ are queried by nodes in $Stage_i^{k,l}$.) The following definition, $Shadow(p,u')$, identifies those variables $x_{i,j}$ that are potentially affected by setting variable $x_{i',j}$ to 1 at u'. Then Claim 3.10 shows that the number of such $x_{i,j}$'s that are forced at the same stage by setting some $x_{i',j}$ to 1, cannot be too large.

Let u' be a non-leaf node of type (k,l) such that $Label(u') = x_{i',j}$, and let p be a path from the root to u'. Define:

> $\mathbf{Shadow(p,u')}$ = (for u' with $Label(u') = x_{i',j}$ of type (k,l)) the set of variables $x_{i,j}$ such that all the following are satisfied:
>
> 1. $i \in OpenPigeons(u')$.
> 2. $i \in Stubborn^{k,l-1}(p,u')$.
> 3. $u' \in Stage_i^{k,l}$.

Claim 3.10 *For any non-leaf u' of type (k,l) and any path p from the root to u':*

1.

$$|Shadow(p,u')| < 2m_k.$$

2.

$$|Shadow(p,u')| \leq |Stubborn^{k,l-1}(p,u')|.$$

Proof:
Part (2) is obvious by the definition of $|Shadow(p,u')|$. We will prove part (1).

If $k = 1$ then $2m_k = 2^{n^\epsilon}$ and the claim follows by our assumption that $m < 2^{n^\epsilon/10}$. For $k > 1$, assume for a contradiction that $|Shadow(p,u')| \geq 2m_k$. Then there exist $2m_k = m_{k-1}$ different pigeons $i \in OpenPigeons(u')$ such that $u' \in Stage_i^{k,l}$. By the definition of $Stage_i^{k,l}$, each one of these pigeons satisfies $|Queries_i^{k,l}(u')| \geq b_{3k-1,3l-1} - 1$, and hence by Claim 3.2,

$$|Queries_i(u')| > n^{2\epsilon} \cdot (b_{3k-1,3l-1} - \hat{n}) \geq n^{2\epsilon} \cdot (1/3) \cdot n^{1-3\epsilon} \cdot (3k-2) > (k-1) \cdot n^{1-\epsilon} = n_{k-1}.$$

Hence, u' is a pigeon-axiom of order $k-1$ and cannot be a non-leaf node. □

We are now ready to put everything together to prove our main claim. $Bad^{k,l}(p)$ defined below identifies those variables $x_{i,j}$ of type (k,l) such that $x_{i,j}$ is in the shadow of something set to 1; that it, $x_{i,j}$ has the potential to be forced at this stage. $Good^{k,l}(p)$ are those variables $x_{i,j}$ of type (k,l) that are queried and not forced– in other words, with nonnegligible probability, we follow the 1-path. The overall reasoning is as follows. Suppose for sake of contradiction that too many (more than $.81\hat{m}\hat{n}$) variables are forced along path p. If $x_{i,j}$ is forced, then it is either in $Tough_i^{k,l}(p)$ or in $Bad_i^{k,l}(p)$, and it is not in $Good_i^{k,l}(p)$. (This is Claim 3.11.) The number in $Tough$ is bounded by $2n^\epsilon$ by Claim 3.9. Therefore, the number in $Bad_i^{k,l}(p)$ is at most $.81\hat{m}\hat{n} - 2n^\epsilon \geq .8\hat{m}\hat{n}$. But on the other hand, $B \leq \sum_{r=1}^{G} r_i$, where $r_1, \ldots r_G$ are independent random variables taking on value 0 with probability $1 - 1/2m_k$ and value $2m_k$ with probability $1/2m_k$. This inequality holds because each bad variable is caused by previously setting a good variable to 1; when a good variable is set to 1 (which happens with probability $1/2m_k$), it creates at most $2m_k$ bad variables, and when a good variable is set to 0, it creates no bad variables. (This argument is proven implicitly in Claim 3.13.) Finally using standard Chernoff bounds for $\sum_{r=1}^{G} r_i$, it can be shown that with very high probability, $B < .8\hat{m}\hat{n}$ (Claim 3.16), and thus we have reached a contradiction.

Let p be a path starting at the root. Define:

$\mathbf{Bad^{k,l}(p)}$ = the set of variables $x_{i,j}$ of type (k,l) such that all the following are satisfied:

1. For some $u' \in p$, $x_{i,j} \in Shadow(p, u')$.
2. The variable $x_{i',j} = Label(u')$ was set to 1 along the path p.

$\mathbf{Good^{k,l}(p)}$ = the set of variables $x_{i,j}$ of type (k,l) such that all the following are satisfied:

1. For some $u \in p$, $x_{i,j} = Label(u)$ (i.e., $x_{i,j}$ was queried at u).
2. u was $Relevant$ for p and $x_{i,j}$ was not $Forced$ to 0 at u (hence, $Strategy(p,u)$ answered the query $x_{i,j}$ by 1 with probability $1/2m_k$).

Claim 3.11 *For any i, j, k, l, p, if $x_{i,j} \in Forced^{k,l}(p)$ then:*

1.
$$x_{i,j} \in Tough_i^{k,l}(p) \bigcup Bad^{k,l}(p).$$

2.
$$x_{i,j} \notin Good^{k,l}(p).$$

Proof:
Part (2) is obvious by the definition of $Good^{k,l}(p)$. We will prove part (1).

If $x_{i,j} \in Forced^{k,l}(p)$ then $x_{i,j}$ was $Forced$ to 0 at some node u of the path p, by some variable $x_{i',j}$. The variable $x_{i',j}$ was queried at some node $u' < u$ of the path p and was set to 1 (along the path p). If $u' \in Before_i^{k,l}$ then $x_{i,j} \in Tough_i^{k,l}(p)$ (by the definition of $Tough_i^{k,l}(p)$) and the claim follows. Otherwise, $u' \notin Before_i^{k,l}$, and since u is $Relevant$ for p, we also know that $u \in Stage_i^{k,l}$ and hence that $u' \notin Over_i^{k,l}$. Hence, $u' \in Stage_i^{k,l}$. We will show that in that case $x_{i,j} \in Shadow(p, u')$ and hence $x_{i,j} \in Bad^{k,l}(p)$.

1. Since u is $Relevant$ for p, we know that $i \in OpenPigeons(u)$. Hence by Claim 3.1, $i \in OpenPigeons(u')$.

2. Since $u, u' \in Stage_i^{k,l}$, we know by Claim 3.4 that $u, u' \in Over_i^{k,l-1}$. Since u is $Relevant$ for p, we know that $i \in Stubborn^{k,l-1}(p, u)$. Hence by Claim 3.6, $i \in Stubborn^{k,l-1}(p, u')$.

3. We already showed, $u' \in Stage_i^{k,l}$.

Hence, $x_{i,j} \in Shadow(p, u')$. □

Denote $B = |Bad^{k,l}(Path)|$. Denote $G = |Good^{k,l}(Path)|$. Denote $D = B - G$. Denote the variables in $Good^{k,l}(Path)$ by $g_1, ..., g_G$, where the ordering is according to the order of these queries in $Path$. Note that $B, G, D, g_1, ..., g_G$ are random variables. Denote

$$\hat{m} = (0.9)^{l-1} \cdot m.$$

Denote

$$m' = \min[2m_k, \hat{m}].$$

For $1 \leq r \leq G$, denote by y_r a random variable that gets the value (-1) if g_r was set to 0 along $Path$, and the value $(m'-1)$ if g_r was set to 1 along $Path$. For $r > G$ define y_r to be an (independently chosen) random variable that gets the value (-1) with probability $1-(1/2m_k)$ and the value $(m'-1)$ with probability $1/2m_k$. Note that $B, G, D, g_1, ..., g_G, \hat{m}, m', y_1, ..., y_G$ depend on (k, l). For simplicity of the notations we do not use here the superscript ^{k,l}.

Claim 3.12 *$y_1, y_2, ...$ are independent random variables, and each one of them gets the value (-1) with probability $1-(1/2m_k)$ and the value $(m'-1)$ with probability $1/2m_k$.*

Proof:
Follows immediately by the definition of the adversary strategy, by the definition of the set $Good^{k,l}(p)$ and by the definition of the random variables $y_1, y_2, ...$. □

Claim 3.13 *Assume that $|Stubborn^{k,l-1}(Path, Leaf)| \leq \hat{m}$ (the inductive assumption). Then*

$$D \leq \sum_{r=1}^{G} y_r.$$

Proof:
Since $D = B - G$, it is enough to prove that

$$B \leq G + \sum_{r=1}^{G} y_r = \sum_{r=1}^{G} (y_r + 1).$$

First note that by Claim 3.10 and Claim 3.5, for any $u' \in Path$ of type (k, l),

$$|Shadow(Path, u')| \leq m'.$$

Every $x_{i,j} \in Bad^{k,l}(Path)$ is in $Shadow(Path, u')$ for some $u' \in Path$ (of type (k,l)) such that the variable $x_{i',j} = Label(u')$ was set to 1 along $Path$. Note, however, that the only variables of type (k,l) that were queried along $Path$ and that could be set to 1 are the variables $g_1, ..., g_G$ (these are the only queries for which the answer 1 was given with non-zero probability). Denote by $u'_1, ..., u'_G$ the nodes on $Path$ where $g_1, ..., g_G$ were queried. Then, $x_{i',j} = g_r$ for some g_r that was set to 1 along $Path$, and hence, $x_{i,j} \in Shadow(Path, u'_r)$ for some u'_r such that g_r was set to 1 along $Path$ (we will say in this case that g_r is responsible for $x_{i,j} \in Bad^{k,l}(Path)$). The total number of variables $x_{i,j}$ that g_r is responsible for is at most m' if g_r was set to 1 (along $Path$), and is 0 if g_r was not set to 1 (along $Path$). That is, the total number of variables $x_{i,j}$ that g_r is responsible for is bounded by $(y_r + 1)$. Hence,

$$|Bad^{k,l}(Path)| \leq \sum_{r=1}^{G} (y_r + 1).$$

□

Claim 3.14 *Assume that $|Stubborn^{k,l-1}(Path, Leaf)| \leq \hat{m}$ (the inductive assumption) and assume (for a contradiction) that $|Stubborn^{k,l}(Path, Leaf)| > 0.9 \cdot \hat{m}$. Then all the following are satisfied:*

1. $$G < 0.2 \cdot \hat{m} \cdot \hat{n}.$$

2. *If Path is not Cheating then*
$$B > 0.8 \cdot \hat{m} \cdot \hat{n}.$$

3. *If Path is not Cheating then*
$$D > 0.6 \cdot \hat{m} \cdot \hat{n}.$$

Proof:
First note that by the assumption $|Stubborn^{k,l-1}(Path, Leaf)| \leq \hat{m}$ and by Claim 3.5 and by the fact that along any path the total number of nodes in $Stage_i^{k,l}$ is at most $\hat{n}$, we can conclude that the total number of nodes $u \in Path$ of type (k,l) that are *Relevant* for *Path* is at most $\hat{m} \cdot \hat{n}$.

Denote $F = |Forced^{k,l}(Path)|$. Since $|Stubborn^{k,l}(Path, Leaf)| > 0.9 \cdot \hat{m}$, and since for each $i \in Stubborn^{k,l}(Path, Leaf)$, at least $0.9 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were *Forced* to 0 at some node of *Path* (by the definition of $Stubborn^{k,l}(p,u)$), we can conclude that $F > 0.81 \cdot \hat{m} \cdot \hat{n}$.

Now, part (1) follows by Claim 3.11. Part (2) follows by Claim 3.11, Claim 3.9 and by the fact that $2n^{\epsilon} < 0.01 \cdot \hat{n}$. Part (3) follows from part (1) and part (2). □

Claim 3.15 *Assume (for a contradiction) that $|Stubborn^{k,l}(Path, Leaf)| > 0.9 \cdot \hat{m}$. Then at least one of the following events occurs:*

1. $|Stubborn^{k,l-1}(Path, Leaf)| > \hat{m}$.

2. *Path is Cheating.*

3. *For some* $s < 0.2 \cdot \hat{m} \cdot \hat{n}$, *we have* $\sum_{r=1}^{s} y_r \geq 0.6 \cdot \hat{m} \cdot \hat{n}$.

Proof:
Assume that (1) and (2) do not occur. Then by part (1) of Claim 3.14, $G < 0.2 \cdot \hat{m} \cdot \hat{n}$, and by part (3) of Claim 3.14, $D > 0.6 \cdot \hat{m} \cdot \hat{n}$. By Claim 3.13, $D \leq \sum_{r=1}^{G} y_r$. Hence, for some $s < 0.2 \cdot \hat{m} \cdot \hat{n}$,

$$\sum_{r=1}^{s} y_r \geq D > 0.6 \cdot \hat{m} \cdot \hat{n}.$$

□

Claim 3.16 *With probability exponentially close to 1, for every* $s < 0.2 \cdot \hat{m} \cdot \hat{n}$,

$$\sum_{r=1}^{s} y_r < 0.6 \cdot \hat{m} \cdot \hat{n}.$$

Proof:
This follows from Claim 3.12, by the standard Chernoff bounds (see for example [ASE92], Appendix A, Corollary A.10). For any specific s, the probability that $\sum_{r=1}^{s} y_r \geq 0.6 \cdot \hat{m} \cdot \hat{n}$ is at most $2^{-c \cdot \hat{n}}$, where c is a small constant (say $c = 1/5$). Hence, the probability that this happens for some $s < 0.2 \cdot \hat{m} \cdot \hat{n}$ is at most $0.2 \cdot \hat{m} \cdot \hat{n} \cdot 2^{-c \cdot \hat{n}}$, which is smaller than $2^{-c' \cdot \hat{n}}$, where c' is a small constant (say $c' = 1/10$). □

We can now complete the proof of Claim 3.7. By Claim 3.15, if $|Stubborn^{k,l}(Path, Leaf)| > 0.9 \cdot \hat{m}$ then one of 3 events occurs. The probability for the first event is exponentially small (by the inductive assumption for $l - 1$). The probability for the second event is bounded by $2^{-n^{\epsilon}}$ (by Claim 3.8). The probability for the third event is bounded by $2^{-c' \cdot \hat{n}}$ (by Claim 3.16). Note that since the number of possible k, l is relatively small (and hence also the number of induction steps is small), the probabilities add to an exponentially small probability. Hence, the event $|Stubborn^{k,l}(Path, Leaf)| > 0.9 \cdot \hat{m}$ occurs with an exponentially small probability. □

3.6 The lower bound

Recall that we assume that the size of our BP is lower than $2^{n^{\epsilon}/10}$. We will show that such a BP cannot solve the weak pigeon hole principle. Denote by $\mathcal{E}$ the following event:

$$\mathcal{E} = \text{the event: for every } k, l, \ |Stubborn^{k,l}(Path, Leaf)| \leq (0.9)^l \cdot m.$$

By Claim 3.7,

$$\mathbf{Pr}[\mathcal{E}] > 1/2.$$

Let u be a pigeon-axiom of order k. We will show that given the event $\mathcal{E}$, the probability that $Leaf = u$ is exponentially small. Denote,

$\mathcal{E}_{\mathbf{u}}$ = the event: $Leaf = u$.

Claim 3.17 *For any pigeon-axiom u of order k,*

$$\mathbf{Pr}[\mathcal{E}_u|\mathcal{E}] < 2 \cdot e^{-\hat{n}/20}.$$

Proof:
Let u be a pigeon-axiom of order k. Then, there is a set

$$A_u \subset OpenPigeons(u),$$

such that $|A_u| = m_k$ and for every $i \in A_u$, $|Queries_i(u)| \geq n_k$. By Claim 3.2 and by the definition of $Over_i^{k,l}$, for every $i \in A_u$ and every l,

$$u \in Over_i^{k,l}.$$

Let p be any path from the root to u, such that for every l,

$$|Stubborn^{k,l}(p,u)| \leq (0.9)^l \cdot m$$

(we assume this because otherwise we would have had $p \neq Path$, if the event $\mathcal{E}$ occurs). In particular, for $l = n^\epsilon$,

$$Stubborn^{k,n^\epsilon}(p,u) = \emptyset$$

(since we assume $m < 2^{n^\epsilon/10}$). At the other hand,

$$Stubborn^{k,0}(p,u) = Pigeons.$$

Therefore, for any pigeon i there exists $l = l_{p,u}(i)$, such that $i \in Stubborn^{k,l-1}(p,u)$ and $i \notin Stubborn^{k,l}(p,u)$. In particular, this applies for any pigeon $i \in A_u$. Thus, for every $i \in A_u$ we have (for $l = l_{p,u}(i)$):

1. $i \in Stubborn^{k,l-1}(p,u)$.

2. $u \in Over_i^{k,l}$.

3. $i \notin Stubborn^{k,l}(p,u)$.

By the definition of $Stubborn^{k,l}(p,u)$, this implies that no more than $0.9 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were *Forced* to 0 at some node of p (for $l = l_{p,u}(i)$). Recall that this is true for any $i \in A_u$.

At the other hand, for every $i \in A_u$, since $u \in Over_i^{k,l}$, there are exactly $\hat{n}$ nodes $u_1, ..., u_{\hat{n}} \in p$ of type (k,l), such that for every u_r, $Label(u_r) \in Pigeon_i$, and such that $u_1, ..., u_{\hat{n}}$ are in $Stage_i^{k,l}$. For each of these nodes u_r we have (for $l = l_{p,u}(i)$):

1. $i \in OpenPigeons(u)$, and hence by Claim 3.1, $i \in OpenPigeons(u_r)$.

2. Since $u_r \in Stage_i^{k,l}$, we know by Claim 3.4 that $u_r \in Over_i^{k,l-1}$, and hence also that $u \in Over_i^{k,l-1}$. Since $i \in Stubborn^{k,l-1}(p,u)$, we have (by Claim 3.6), $i \in Stubborn^{k,l-1}(p,u_r)$.

3. $u_r \in Stage_i^{k,l}$.

Hence, each u_r is *Relevant* for the path p. Since no more than $0.9 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were *Forced* to 0 at nodes of p, we conclude that for any $i \in A_u$ (and for $l = l_{p,u}(i)$) at least $0.1 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were *Relevant* for the path p and were not *Forced* to 0 (when queried).

We will now consider the path *Path*. Assume that the event $\mathcal{E}$ occurs. That is, for every l, $|Stubborn^{k,l}(Path, Leaf)| \leq (0.9)^l \cdot m$. Intuitively, in order to have $Leaf = u$, we must have for every l, $|Stubborn^{k,l}(Path,u)| \leq (0.9)^l \cdot m$, and by the above argument we conclude that for every $i \in A_u$, there exists l such that at least $0.1 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were *Relevant* for *Path* and were not *Forced* to 0 (when queried). For each one of these queries the adversary strategy answers by 0 with probability $(1 - 1/2m_k)$ (by the definition of *Strategy*(p,u)). Since $A_u \subset OpenPigeons(u)$, in order to have $Leaf = u$ the adversary strategy must answer all these queries by 0. Since $|A_u| = m_k$, this happens with probability of at most

$$(1 - 1/2m_k)^{0.1 \cdot \hat{n} \cdot m_k} < e^{-\hat{n}/20},$$

if we ignore the condition that $\mathcal{E}$ occurs. Since we need the conditional probability under the assumption that $\mathcal{E}$ occurs, we have an additional factor of 2 (using the bound that $\mathcal{E}$ occurs with probability of at least 1/2).

More formally, we need to define,

> $\mathbf{\Gamma_u}$ = the event: for every $i \in A_u$ there exists $l = l(i)$, such that at least $0.1 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were *Relevant* for *Path* and were not *Forced* to 0 (when queried).

Assuming that the event $\mathcal{E}$ occurs, we know that if Γ_u does not occur then $Leaf \neq u$. Therefore,

$$\mathbf{Pr}[\mathcal{E}_u|\mathcal{E}] = \mathbf{Pr}[\mathcal{E}_u|\Gamma_u, \mathcal{E}] \cdot \mathbf{Pr}[\Gamma_u|\mathcal{E}].$$

If $\Gamma_u, \mathcal{E}$ occur then for every $i \in A_u$ and for $l = l(i)$, at least $0.1 \cdot \hat{n}$ variables $x_{i,j}$ of type (k,l) were *Relevant* for *Path* and were not *Forced* to 0 (when queried). For each one of these queries the adversary strategy answers by 0 with probability $(1 - 1/2m_k)$, and in order to have $Leaf = u$ the adversary strategy must answer all these queries by 0. Hence,

$$\mathbf{Pr}[\mathcal{E}_u, \Gamma_u, \mathcal{E}] \leq (1 - 1/2m_k)^{0.1 \cdot \hat{n} \cdot m_k} < e^{-\hat{n}/20},$$

that is,

$$\mathbf{Pr}[\mathcal{E}_u|\Gamma_u, \mathcal{E}] < \frac{e^{-\hat{n}/20}}{\mathbf{Pr}[\Gamma_u, \mathcal{E}]}.$$

Therefore,

$$\mathbf{Pr}[\mathcal{E}_u|\mathcal{E}] = \mathbf{Pr}[\mathcal{E}_u|\Gamma_u, \mathcal{E}] \cdot \mathbf{Pr}[\Gamma_u|\mathcal{E}] < \frac{e^{-\hat{n}/20}}{\mathbf{Pr}[\Gamma_u, \mathcal{E}]} \cdot \mathbf{Pr}[\Gamma_u|\mathcal{E}] = \frac{e^{-\hat{n}/20}}{\mathbf{Pr}[\mathcal{E}]} < 2 \cdot e^{-\hat{n}/20}.$$

□

Our lower bound is now proved in the following way: By the definition of $Strategy(p, u)$ the answer given is always 0 if $x_{i,j}$ is $Forced$ to 0. Therefore, $Path$ never reach a hole-axiom, and hence the probability that $Leaf$ is a hole-axiom is 0. At the other hand, by Claim 3.17 the conditional probability that $Leaf$ is a pigeon-axiom, given that the event $\mathcal{E}$ occurs, is at most

$$2 \cdot e^{-\hat{n}/20} \cdot m < 2 \cdot e^{-\hat{n}/20} \cdot 2^{n^\epsilon/10} < 1/2 < \mathbf{Pr}[\mathcal{E}]$$

(since the total number of leaves u is at most m). Therefore, there is a positive probability that $Leaf$ is not a pigeon-axiom, and hence is not an axiom at all. We conclude that a read once BP of size $< 2^{n^\epsilon/10}$ cannot solve the weak pigeon hole principle.

Corollary 3.1 *Any read once Branching Program for the weak pigeon hole principle is of size at least* $2^{n^\epsilon/10}$.

4 Independence results for Regular Resolution

It has been shown by Razborov [Raz] that certain formulations of the statement that nonuniform P does not equal NP cannot be proven with polynomial-size proofs in the Polynomial Calculus. In this section, we will show that the basic idea can be carried out in regular resoltuion, and from this derive as a corollary that (a certain formulation of) nonuniform circuit lower bounds for SAT cannot be proven with subexponential size regular resolution proofs.

Let $3SAT(n)$ denote the satisfiability problem, where all clauses have length 3, and the number of variables is n. The input to a $3SAT(n)$ instance will be a boolean vector of length $n' = (2n)^3$. Each bit of the input will correspond to the presence or absence of each of the $n' = (2n)^3$ distinct 3-clauses (with possible repetition of literals) in the instance. Other natural encodings of $3SAT$ are also possible, but we will stick with this specific one for clarity. For each n, we will let N denote the number of yes instances of $3SAT(n)$.

The basic idea is to show that from short regular resolution refutations of the propositional statement asserting that $3SAT(n)$ is computable by size T boolean circuits, one can obtain short regular resolution refutations of the weak pigeonhole principle, with N underlying pigeons and T holes.

For each n, $3SAT(n)$ can clearly be computed by a formula of size $O(N)$, consisting of the disjunction of all size n' full terms where $3SAT(n)$ evaluates to 1. This will clearly have size polynomial in N, which is exponential in n'. Now assume for sake of contradiction that there is a one-to-one function from N to T, $T << N$. Using this function, we can get a smaller formula (in disjunctive normal form), consisting of only T terms. We need to show that this reduction, from the clauses expressing $\neg PHP_T^N$ to the clauses representing the negation of the circuit lower bound, can be carried out efficiently

We will now proceed with the formal argument. In what follows, we will consider boolean circuits with binary AND gates, binary OR gates, and input gates. The allowable input gates are: (i) the constant 0; (ii) the constant 1; (iii) any of the $2n'$ literals; (iv) any full term, where a full term is a conjunction of n' literals (where the underlying variables of the literals are distinct). Note that fullterms are in one-to-one correspondence with the set of all $2^{n'}$ truth assignments. Slightly abusing notation, $a \in [2^{n'}]$ will sometimes denote the fullterm and other times denote the associated truth assignment, depending on the context. The *size* of a boolean circuit in this representation will be the number of internal AND and OR gates. The notation $[k]$ will denote the set $\{1, \ldots k\}$.

$Circuit_T(n, P)$ will denote the propositional formula stating that there is a boolean circuit with T internal nodes computing $3SAT(n)$. P denotes the following vector of propositional variables that will aid us in the encoding.

- For each node $v \in [T]$, $w \in \{constant1, constant0, fullterm, literal, and, or\}$, $Type(v, w)$ will be a boolean formula specifying what type of node v is. If $Type(v, constant0) = 1$, then v is an input node that always takes on the value 0. If $Type(v, constant1) = 1$, then v is an input node that always takes on the value 1. If $Type(v, fullterm) = 1$, then v is an input node that is the conjunction of n' literals, or v is 0. If $Type(v, literal) = 1$, then v is an input node that is equal to one of the $2n'$ literals. Finally, if $Type(v, and) = 1$, then v is an internal AND-gate, and if $Type(v, or) = 1$ then v is an internal OR-gate.

- For each $v \in [T]$, $i \in [n']$, $w \in \{0,1\}$, $Literal(v, i, w)$ is a boolean variable specifying which literal is associated with v, whenever $Type(v, literal) = 1$. More precisely, $Literal(v, i, 0)$ takes on the value 1 if the literal associated with node v is $\neg x_i$, and $Literal(v, i, 1)$ takes on the value 1 if the literal associated with node v is x_i.

- For each distinct triple of nodes $v, v', v'' \in [T]$, where $v', v'' \leq v$, $Children(v, v', v'')$ is a boolean variable that will be true if v is an internal node, with children v' and v''.

- For each $v \in [T]$, fullterm $a \in [2^{n'}]$, $Value(v, a)$ is a boolean variable specifying that node v computes the fullterm a, whenever $Type(v, fullterm) = 1$.

- For each truth assignment $a \in \{0,1\}^{n'}$ and each node $v \in [T]$, $p_{a,v}$ is a boolean variable specifying the value computed by node v of the circuit on input a.

The statement $Circuit_T(n, P)$ will be the CNF formula equivalent to the conjunction of the following axioms.

1. For every $v \in [T]$, we have clauses stating that there is exactly one w such that $Type(v, w)$.

2. For every $v \in [T]$, we have clauses stating that there is exactly one $i \in [n]$ and one $w \in \{0,1\}$ such that $Literal(v, i, w) = 1$.

3. For every $v \in [T]$, there is at most one fullterm $a \in [2^{n'}]$ such that $Value(v, a) = 1$. (Note that if v is of type fullterm, and there are no fullterms a such that $Value(v, a) = 1$, then v will output the constant 0.)

4. For every $v \in [T]$, we have clauses stating that if $Type(v, and)$ or $Type(v, or)$, then v has exactly two children, v' and v'' such that v' and v'' are both less than v.

5. For every $v \in [T]$ and truth assignment $a \in [2^{n'}]$, we have: $Type(v, constant0) \to p_{a,v} = 0$.

6. For every $v \in [T]$ and truth assignment $a \in [2^{n'}]$, we have: $Type(v, constant1) \to p_{a,v} = 1$.

7. For every $v \in [T]$, $a \in [2^{n'}]$, $Type(v, fullterm) \wedge Value(v, a) \to p_{a,v}$.

8. For every $v \in [T]$, $a \in [2^{n'}]$, $Type(v, fullterm) \wedge \neg Value(v, a) \to \neg p_{a,v}$.

9. For every $v \in [T]$, truth assignment $a \in [2^{n'}]$, variable $i \in [n]$ and value $w \in \{0, 1\}$ we have: $Type(v, literal) \wedge Literal(v, i, w) \to p_{a,v} = a_i^w$, where a_i^q equals a_i if $w = 1$, and equals $\neg a_i$ if $w = 0$.

10. For every distinct triple of nodes v, v', $v'' \in [T]$, and truth assignment $a \in [2^{n'}]$, $Type(v, and) \wedge Children(v, v', v'') \to p_{a,v} = p_{a,v'} \wedge p_{a,v''}$.

11. For every distinct triple of nodes v, v', $v'' \in [T]$ and truth assignment $a \in [2^{n'}]$, $Type(v, or) \wedge Children(v, v', v'') \to p_{a,v} = p_{a,v'} \vee p_{a,v''}$.

12. For every truth assignment $a \in [2^{n'}]$, $p_{a,T} = 3SAT(n)[a]$. That is, the final node of the circuit, T, computes the same function as $3SAT(n)$ on input a.

Note that the number of variables in the above formula is polynomial in $2^{n'}$ and T, and the size of the formula is also polynomial in $2^{n'}$ and T.

Corollary 4.1 *The propositional statement $Circuit_T(n, P)$ requires exponential-size (in T) regular Resolution refutations.*

Proof:
Let N be the number of 1's of $3SAT(n)$. For $i \in [N]$, let $ta(i) = a$ if a is the lexicographically i^{th} truth assignment (on n' variables) such that $SAT(a) = 1$. Let $ta^{-1}(a)$ be the inverse function: if $SAT(a) = 1$, then $ta^{-1}(a) = i$ if and only if a is the lexicographically i^{th} truth assignment, and otherwise $ta^{-1}(a) = 0$. We want to show how to efficiently derive the clauses of $Circuit_T(n, P)$ from the clauses of $\neg PHP_T^N$ in regular Resolution. The small circuit computing $3SAT(n)$ on n variables will consist of internal $\vee$ nodes $w_1, \ldots w_{T-1}$, and input nodes $v_1, \ldots v_T$. That is, the circuit will be the disjunction of terms $v_1, \ldots v_T$, where each term is either a fullterm or the constant 0.

Let $\neg PHP_T^N$ be the weak pigeonhole principle, with underlying variables $x_{i,j}$, $i \leq N$, $j \leq T$. We will associate $i \in [N]$ with the lexicographically i^{th} truth assignment, $ta(i)$, where SAT evaluates to 1.

We will now define the variables underlying $Circuit_T(n, P)$ in terms of the variables underlying $WPHP_T^N$.

The propositional variables $Type(*, *)$, $Children(*, *, *)$ and $Literal(*, *, *)$ will all be constant values, either 0 or 1, since the structure of the circuit is fixed and there will be no literals in the circuit. More specifically, the internal nodes $w_1, \ldots w_{T-1}$ will be of type *or* and the input nodes $v_1, \ldots, v_T$ will be of type $fullterm$. All variables $Literal(*, *, *)$ will be 0. $Children(w_1, v_1, v_2) = 1$, and for each node w_i, $i \geq 2$, $Children(w_i, v_{i+1}, w_{i-1}) = 1$; all other $Children(*, *, *)$ are equal to 0.

We will now define the remaining variables, $Value$ and $p_{a,v}$. For $v \in [T]$ and $a \in [2^{n'}]$, if a is a satisfying assignment for $3SAT(n)$, then $Value(v, a) = x_{ta^{-1}(a),v}$; otherwise ($a$ is not a satisfying assignment) $Value(v, a) = 0$.

For each node $v \in [T]$ and truth assignment $a \in [2^{n'}]$, $p_{a,v}$ will be equal to $x_{ta^{-1}(a),v}$ if $3SAT(n)[a] = 1$, and 0 otherwise. For each node w_i, $i \geq 2$ and truth assignment a, p_{a,w_i} will be equal to $p_{a,v_{i+1}} \vee p_{a,w_{i-1}}$. For node w_1 and truth assignment a, p_{a,w_1} will be equal to $p_{a,v_1} \vee p_{a,v_2}$.

It is straightforward to see that the axioms of $Circuit_T(n, P)$ (with these substitutions) can be easily derived in regular Resolution from $\neg PHP_T^N$. All axioms except for axiom 3 and the last axiom translate into true CNF formulas. Axiom 2 follows directly from the hole axioms, stating that no hole can have more than one pigeon mapped to it.

It is left to show that the CNF formula expressing the last axiom can be efficiently derived from the clauses of the weak pigeonhole principle. Fix a particular truth assignment $a \in [2^{n'}]$ such that $3SAT(n)[a] = 1$. If follows that a corresponds to some $i \in [N]$, and therefore, by the pigeon axioms corresponding to i, we know that $x_{i,1} \vee \ldots \vee x_{i,T}$. Using case-by-case reasoning, suppose that $x_{i,j} = 1$. Then $p_{a,v_j} = x_{i,v_j} = 1$ and hence $p_{a,w_{j-1}} = 1$, and hence $p_{a,w_T} = 1$ as desired. This type of simple reasoning can be carried out efficiently in regular resolution. Now consider a truth assignment a such that $3SAT(n)[a] = 0$. In this case, for every $i \leq T$, $p_{a,v_i} = 0$ and hence for every $i \leq T - 1$, $p_{a,w_i} = 0$ as desired. Again, this simple reasoning can be carried out efficiently in regular resolution.

Thus we have efficiently derived (in regular resolution) an instance of the propositional formula $Circuit_T(n, P)$ from the formula $\neg PHP_T^N$. By the exponential lower bound for the weak pigeonhole principle derived in the previous section, the proof is complete.

References

[ASE92] N. Alon, J. Spencer, and P. Erdos. *The Probabilisitic Method.* John Wiley and Sons, Inc., 1992.

[BKPS98] P. Beame, R. Karp, T. Pitassi, and M. Saks. On the complexity of unsatisfiability of random k-CNF formulas. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, Dallas, TX, May 1998.

[BSW] E. Ben Sasson and A. Wigderson. Short proofs are narrow–resolution made simple. Manuscript, October 1998.

[BT88] S. Buss and G. Turan. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, 62:311–317, 1988.

[Hak85] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–305, 1985.

[Kra96] J. Krajicek. *Bounded Arithmetic, Propositional Logic and Complexity Theory.* Cambridge University Press, 1996.

[Raz] A. A. Razborov. Lower bounds for the polynomial calculus. November 1996.

[RWY97] A. A. Razborov, A. Wigderson, and A. C. Yao. Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus. In *Proceedings of the Twenty Ninth Annual ACM Symposium on Theory of Computing*, pages 739–748, El Paso, TX, May 1997.

[Urq87] A. Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, 1987.