Clique is hard to approximate within $n^{1-\epsilon}$

Johan Håstad
Royal Institute of Technology
Sweden
email:johanh@nada.kth.se

February 18, 2002

Abstract

We prove that, unless any problem in NP can be solved in probabilistic polynomial time, for any $\epsilon > 0$, the size of the largest clique in a graph with n nodes is hard to approximate in polynomial time within a factor $n^{1-\epsilon}$. This is done by constructing, for any $\delta > 0$, a probabilistically checkable proof for NP which uses logarithmic randomness and δ amortized free bits.

1 Introduction

The basic entity in complexity theory is a computational problem which, from a mathematical point of view, is simply a function F from finite binary strings to finite binary strings. To make some functions more intuitive these finite binary strings should sometimes be interpreted as integers, graphs, or descriptions of polynomials. An important special case is given by decision problems where the range consists of only two strings usually taken to be 0 or 1.

A function F should be realized by an algorithm and there are many ways to mathematically formalize the notion of an algorithm. One of the first formalizations which is still heavily used is that of a Turing Machine. However, since we in this paper do not deal with the fine details of the definition, the reader might as well think of a standard computer with a standard programming language. The only idealization needed is that the computer

contains an infinite number of words of memory each of which remain of bounded size. The algorithm has some means of reading information from the external world and also some mechanism to write the result. Time is measured as the number of elementary steps.

A finite binary string x in the domain is simply called the input, while the output is the final result of the computation delivered to the outside world. An algorithm solves the computational problem F if it when presented x on its input device, it produces output $F(x)$. A parameter that is important to measure the performance of the algorithm is the length of the input which simply is the number of binary symbols in x.

In complexity theory the basic notion of efficiently computable is defined as computable in time polynomial in the input-length. The class of polynomial time solvable decision problems is denoted by P. Establishing that a problem cannot be solved efficiently can sometimes be done but for most naturally occurring computational problems of combinatorial nature, no such bounds are known. Many such problems fall into the class NP; problems were positive answers have proofs that can be verified efficiently. Standard problems in NP are satisfiability (given a formula φ over Boolean variables, is it possible to assign truth values to the variables to make φ true) and the clique problem (given a graph G and an integer k, are there k nodes all of which are connected in G). These problems are traditionally, in computer science, denoted by SAT and CLIQUE, respectively. It is still unknown whether NP=P, although it is widely believed that this is not the case. It is even the case that much work in complexity theory, and indeed even this paper, would have to be reevaluated if NP=P.

There is a group of problem in NP, called the *NP-complete* problems and introduced by Cook [14], which have the property that they belong to P if and only if NP=P. Thus being NP-complete is strong evidence that a problem is computationally intractable and literally thousands of natural computational problems are today known to be NP-complete (for an outdated but still large list of hundreds of natural problems see [21]). SAT and CLIQUE are two of the most well known NP-complete problems.

Many combinatorial optimization problems have a corresponding decision problem which is NP-complete. For instance, consider the optimization problem that given a graph G to determine the size of the largest set of nodes which are all pairwise connected in G. Since solving this problem implies solving CLIQUE, a polynomial time algorithm always giving the correct optimum would imply that NP=P. Optimization problem with this property are called *NP-hard* (not NP-complete as they do not fall into the class NP as they are not decision problems). Solving NP-hard optimization

problems exactly is thus hard, but in many practical circumstances it is almost as good to get an approximation of the optimum. Different NP-hard optimization problems behave very differently with respect to efficient approximation algorithms and this set of questions form a research area in its own.

In this paper we study the possible performance of a polynomial time approximation algorithm for the optimization version of CLIQUE, traditionally denoted Max-Clique but here we use the abbreviation MC. We demand that the algorithm, on input a graph G with n vertices, outputs a number that is always at most the size of the largest clique in G. We say that we have an $f(n)$ approximation algorithm if this number is always at least the size of the largest clique divided by $f(n)$. The best polynomial time approximation algorithm for MC achieves an approximation ratio of $O(\frac{n}{(\log n)^2})$ [12], and thus it is of the form $n^{1-o(1)}$. This is not an easy result but note that an approximation factor of n is trivial since the clique cannot contain more than all n nodes and any set of a single node is a clique. On the negative side, there has been a sequence of papers [11, 17, 2, 1, 8, 18, 9, 7] giving stronger and stronger inapproximability results based on very plausible complexity theoretic assumptions. The strongest lower bound is by Bellare, Goldreich and Sudan [7] who prove (under the assumption that NP$\neq$ZPP[1]) that, for any $\epsilon > 0$, MC cannot be efficiently approximated within $n^{1/3-\epsilon}$. We strengthen these results to prove that, under the same assumption, for any $\epsilon > 0$, MC cannot be efficiently approximated within $n^{1-\epsilon}$. Thus, MC is indeed very difficult to approximate. As in previous papers we use the connection, discovered by Feige et al. in their seminal paper [17], between multiprover interactive proofs and inapproximability results for MC. Let us briefly describe this connection.

NP can be viewed as a proof system where a single infinitely powerful prover P tries to convince a polynomial time verifier V that a statement is true. For concreteness let us assume that the statement is that a formula φ is satisfiable. In this case, P displays a satisfying assignment and V can easily check that it is a correct proof. This proof system is complete since every satisfiable φ admits a correct proof, and it is sound since V can never be made to accept an incorrect statement.

If φ contains n variables, V reads n bits in the above proof. An interesting question is whether we could restrict V to read fewer bits, the fewest

[1]ZPP is that class of problems that can be solved in expected polynomial time by a probabilistic algorithm that never makes an error, i.e. only the running time is stochastic. The faith in the hypothesis NP$\neq$ZPP is almost as strong as in NP$\neq$P.

possible being a number of bits which is independent of the the number of variables in φ. It is not hard to see that this is impossible unless we relax the requirements of the proof. The proof remains a finite binary string, but we allow the verifier to make random choices. This means that given φ we can now speak of the probability that V accepts a certain proof π. When V was restricted to be deterministic this probability was either 0 or 1 while now it is a number in the interval $[0, 1]$. In this paper we assume that if φ is satisfiable then there is a proof that makes V accept with probability 1 while when φ is not satisfiable then there is some constant $s < 1$ such that for any proof π the probability that V accepts is bounded by s. The parameter s is called the soundness of the proof and a 0-sound proof is a proof in the original sense of the word. Note that this soundness probability is only taken over V's internal random choices and is true for any nonsatisfiable φ and any attempted proof π. This implies that we can decrease this false acceptance probability to s^k by using a verifier $V^{(k)}$ that runs the original verifier k times using independent random choices.

It is an amazing fact, proved by Arora et al [1], that any NP-statement has a proof of the above type, usually called probabilistically checkable proof or simply PCP, where V only reads a constant, independent of the size of the statement being verified, number of bits of the proof and achieves soundness $s = 1/2$. Apart from being an amazing proof system this gives a connection to approximation of optimization problems as follows (for details on the connection to MC we refer to [17]).

Fix a formula φ and consider the PCP by Arora et al. We have a well defined function $acc(\pi)$, the probability that V accepts a certain proof π. Consider $\max_\pi acc(\pi)$. If φ is satisfiable this optimum is 1, while if φ is not satisfiable then the optimum is at most s. Thus, even computing this optimum approximately would enable us to decide an NP-complete question. It turns out that by choosing a suitable coding one can make $\max_\pi acc(\pi)$ be proportional to the size of the maximal clique in a graph G_φ. It follows that approximating MC within a factor $1/s$ implies solving an NP-complete problem and hence the former must be NP-hard.

Since we are aiming at rather exact quantitative results, all parts of the above argument have to be carried out in detail and optimized to identify the crucial parameters to obtain the best possible results. This has already been done and, somewhat surprisingly, for clique the situation is not very complicated.

The construction of G_φ uses all the possible random choices made by V and hence it is essential that this number is polynomial, or, in other words, that V only flips a logarithmic number of binary coins. Apart from

this requirement, the only parameter that matters is the *amortized free-bit complexity*. To explain this concept let us give a small example. A common "subroutine" in a PCP is to check that a function g given by a table is a low degree polynomial; in the simplest case a linear function. To be specific, assume that the proof requires that g is a linear function over $GF[2]$ from $\{0,1\}^n$ to $\{0,1\}$ and is thus given by 2^n bits. To check this, V can generate two random points x and y and check that $g(x)+g(y)=g(x+y)$. Thus V reads the bits $g(x)$ and $g(y)$ recording their values and then checks that $g(x+y)$ has the correct value. One can prove (see [6]) that any g that passes this test with high probability is close to a linear function. This essentially means that one can assume that g is a linear function and this can be used to prove correctness of the overall PCP.

Now consider a general PCP. During the verification procedure V looks at a number of bits. Sometimes V has no idea what the value of the bit should be (as when reading $g(x)$ and $g(y)$ in the example) while some other times it is in a "checking mode" (as when reading $g(x+y)$ above) and knows what to expect, and when the value is not as expected V rejects the input. The number of questions of the first type is the number of free bits and if we denote this number by f, the number of amortized free bits is $f/\log_2 s^{-1}$ where s is the soundness. One indication that this is a natural parameter can be seen from replacing V by $V^{(k)}$ as discussed above, i.e. running V k times with independent random choices. In this case f is replaced by kf while s is replaced by s^k. Thus the number of amortized free bits is preserved.

The connection between inapproximability and amortized free bits is now the following [17, 18, 9, 34]; suppose any NP-statement admits a PCP which has a polynomial time verifier which uses logarithmic randomness and uses k amortized free bits. Then for any $\epsilon > 0$, unless NP=ZPP, MC cannot be approximated with $n^{\frac{1}{k+1}-\epsilon}$ in polynomial time.

Bellare, Goldreich and Sudan [7] proved that in fact we essentially have an equivalence, in that if it is NP-hard to approximate MC within a factor $n^{\frac{1}{k+1}}$ then NP has a proof-system with essentially k amortized free bits.

In this paper, for any $\delta > 0$, we give a proof-system which uses δ amortized free bits. As discussed above, this gives an inapproximability factor for MC of $n^{1-\epsilon}$ for any $\epsilon > 0$.

This paper is the final version of the results announced in [24] and [25].

Related results. The framework of PCPs has lead to a number of strong inapproximability results. For a good survey of the results we refer to [7], and let us here only mention a couple of the strongest results on some problems

of general interest. Feige and Kilian [19] has used the results of this paper to derive the same strong, i.e. factor $n^{1-\epsilon}$, inapproximability results for chromatic number. Chromatic number is the problem to color the nodes of graph G with the minimal number of colors such that no two adjacent nodes have the same color.

Set cover is another central problem and an instance of this problem is given by a family of sets S_i contained in a universe X. The task is to find the minimal size subcollection $(S_{i_j})_{j=1}^{k}$ that covers X, i.e. such that each $x \in X$ is contained in some S_{i_j}. A greedy strategy approximates this number within approximately $\ln n$ (see, for instance [27]) where n is the cardinality of X. This was proved by Feige [15] to be the best possible performance for an efficient approximation algorithm.

There are many NP-hard optimization problems which can be efficiently approximated within some constant c_1 but such that there is another constant c_2 for which the approximation problem is NP-hard. For some problems, the gap between c_1 and c_2 can be made arbitrarily small. The latter is true for Max-E3-SAT (the problem of satisfying the maximal number of clauses in a CNF formula where each clause is of length exactly 3) and Max-Lin-2 (satisfying the maximal number of equations in a linear system of equations mod 2) where the c_1 and c_2 both are essentially 8/7 and 2, respectively [26]. There are other problems where a gap remains between the two constants. Examples of such problems are Max-Cut (given a graph, partition the nodes into two sets such that a maximal number of edges go between the two sets) and Max-E2-SAT (analogous to Max-E3-SAT with the difference that clauses are of length 2). For Max-Cut $c_1 \approx 1.074$ [22] while $c_2 = 22/21 - \epsilon \approx 1.047$ [26] while for Max-E2-SAT, $c_1 \approx 1.138$ [16] and $c_2 = 17/16 - \epsilon \approx 1.0625$ [26], both for an arbitrary $\epsilon > 0$.

Organization of the paper. In Section 2 we give some basic definitions and statements of prior works that are essential to us. In Section 3 we take the first steps towards the desired PCP and recall the long code introduced in [7]. It is important to test the property that a given string is a correctly formed long code and in Section 4 we give such a test. It turns out to be essential that this test can take into account side conditions and thus the main result of this section is given in Theorem 4.17. In Section 5 we show how to use the constructed test as a subroutine to get the desired PCP for an arbitrary NP-statement.

2 Definitions and formal statements of prior work

We consider binary strings and the length of a string x is denoted by $|x|$ and we also use absolute values to denote the size of other natural objects. In particular $|T|$ is the cardinality of a set T. The notation $O(f(n))$ denotes any function which is bounded by $cf(n)$ for some absolute constant c and all sufficiently large values of n.

An assignment on a set U is an element of $\{0,1\}^U$. For two sets $U \subseteq W$ and an assignment y on W we let $y|_U$ be the induced assignment on U. For a set β of assignments of W we let $\pi_U(\beta)$ be the set of assignments on U that contain exactly $y|_U$ for all $y \in \beta$. A function f defined for assignments on U is automatically extended to assignments on W by letting $f(y) = f(y|_U)$.

A Boolean formula is a CNF-formula if it is a conjunction of disjunctions of literals, where a literal is a variable or a negated variable. Such a disjunction is also called a clause and a formula is a 3-CNF-formula if each clause is of length at most 3.

We introduce some more notation as needed later but right now we turn to some basic definitions.

2.1 Complexity classes

To define complexity classes we need to fix one formal model of computation. We let this be the Turing machine (for a definition see [30]), although any other formal model would do as well. Time is measured as the number of elementary steps of the machine. A *language* is simply a set of finite binary strings. An example is the set of satisfiable Boolean formulas under some suitable encoding. When speaking of Turing machines in connection with languages we say that a Turing machine M accepts an input x iff it outputs 1 on this input and otherwise we say that it rejects. We say that M accepts a language L if it accepts exactly the elements of L. The most basic complexity class is P.

Definition 2.1 *A language L belongs to P iff there is some constant c such that there is a deterministic Turing machine M that on every input x runs in time $O(|x|^c)$ and accepts L.*

Other classes of interest in this paper are NP and ZPP. A probabilistic Turing machine has the ability to select a random bit. This has become known as "flipping a random coin" and we use this terminology. The output of a probabilistic machine is in general a random variable. For ZPP, however, the output is determined by the input only.

Definition 2.2 *A language L belongs to ZPP iff there is some constant c such that there is a probabilistic Turing machine M that on input x runs in expected time $O(|x|^c)$ and outputs 1 iff $x \in L$.*

Thus for ZPP the answer of the machine is always correct while the running time is relaxed to be true only in the expected sense. The class NP is usually defined in terms of nondeterministic Turing machines. Since we do not need the definition of nondeterministic computation while proof systems play an essential role here, we choose to define NP in terms of proof systems and hence we need a detour.

A proof system is defined through a verifier V. It is an efficient algorithm and thus a, possibly probabilistic, polynomial time Turing machine. It needs some mechanism to access the proof and we allow V to have access to one or more oracles. An oracle π can be simply be thought of as a bit string and the question "i ?" is simply answered by the i'th bit of π. When we want to emphasize the fact that V uses a particular oracle π we write V^π. Once the input x and π are fixed we get a well-defined probability that V^π accepts the input x. If V is deterministic then it is either 0 or 1 while if V is probabilistic it is a number in the interval $[0, 1]$.

Definition 2.3 *A language L belongs to NP iff there is a deterministic polynomial time verifier V such that for some constant c the following is true. On each input $x \in L$ there is a proof π such that V^π accepts x in time at most $O(|x|^c)$. If $x \notin L$ there is no π such that V^π accepts x.*

It is not difficult to see that P$\subseteq$ZPP$\subseteq$NP and it is not known whether any of the inclusions is proper. It is strongly believed that ZPP$\subset$NP, while the relation between P and ZPP is more open to speculation. For a more complete discussion of complexity classes and related concepts we refer to [30].

2.2 Probabilistic proof systems

As discussed in the introduction we are interested in proof systems where the verifier is probabilistic. The simplest variant is a probabilistically checkable proof.

Definition 2.4 *A* Probabilistically Checkable Proof (PCP) *with soundness s for a language L is given by a verifier V with the following properties*

- *For $x \in L$ there is a proof π such that V^π outputs 1 on input x with probability 1.*

- *For $x \notin L$ and all π the probability that V^π outputs 1 on input x is bounded by s.*

We are interested in efficient PCPs and hence we assume that V runs in worst case polynomial time. There are many other parameters of V of interest. We here only discuss the parameters relevant to our paper and for a more complete discussion we refer to [7].

Definition 2.5 *The verifier V uses* logarithmic randomness *if on each input x and proof π, V^π flips $O(\log |x|)$ random coins.*

When discussing the acceptance probability of V as a combinatorial problem it is natural to discuss all possible executions of V. The only parts of V that are not predictable are the random coins and the answers that V gets from the oracle. These parameters completely determine whether or not V accepts. We call a sequence of oracle answers and random coins a *pattern*. We are only interested in patterns that cause V to accept. The key concept we need is that of amortized free bits but first we need to define free bits.

Definition 2.6 *A PCP uses $\leq f$* free bits *if for each sequence of random coins of V, there are at most 2^f different sets of oracle answers that complete an accepting pattern.*

In our protocols, the number of free bits is in fact rather simple to calculate. When V reads bits in the oracle, either he has no idea what to expect or he is checking that a certain bit has a given value. The number of free bits is then simply the number of read bits of the first kind. We now proceed to define amortized free bits.

Definition 2.7 *The* amortized free bit complexity *of a PCP with soundness s is defined as*

$$\frac{f}{\log(1/s)},$$

where f is the number of free bits.

As mentioned in the introduction amortized free bits is the key to inapproximability of clique. The basic construction is from [17] while the current statement of the connection is from [7][2] but at least Theorem 2.8 can be extracted from [11, 34].

[2] In [7], only the conclusion that NP⊆coRP is given. It is not difficult, as in [34], to get the conclusion NP=ZPP.

Theorem 2.8 *[7] Suppose any language in NP admits a PCP with a probabilistic polynomial time verifier that uses logarithmic randomness and f amortized free bit. Then, unless NP=ZPP, for any $\epsilon > 0$, Max Clique cannot be approximated within $n^{1/(1+f+\epsilon)}$ in polynomial time.*

If one is only willing to believe that NP≠P then we have the following variant.

Theorem 2.9 *[7] Suppose any language in NP admits a PCP with a probabilistic polynomial time verifier that uses logarithmic randomness and f amortized free bit. Then, unless NP=P, for any $\epsilon > 0$, Max Clique cannot be approximated within $n^{1/(2+f+\epsilon)}$ in polynomial time.*

We also need what is generally called a two-prover one-round interactive proof. Such a verifier has two oracles but has the limitation that it can only ask one question to each oracle and that both questions have to be produced before either of them is answered. We do not limit the answer sizes of the oracles which we denote by P_1 and P_2.

Definition 2.10 *A probabilistic polynomial time Turing machine V is a verifier in a* two-prover one-round proof system *with soundness s for a language L if on input x it produces two strings $q_1(x)$ and $q_2(x)$, such that*

- *For $x \in L$ there are two oracles P_1 and P_2 such that the probability that V accepts $(x, P_1(q_1(x)), P_2(q_2(x)))$ is 1.*
- *For $x \notin L$, for any two oracles P_1 and P_2 the probability that V accepts $(x, P_1(q_1(x)), P_2(q_2(x)))$ is bounded by s.*

Let us be more specific on the order of the quantifiers. The provers P_1 and P_2 depend on x but must be fixed before $q_1(x)$ and $q_2(x)$ are produced and hence the answer $P_1(q_1(x))$ only depends on x and $q_1(x)$ and in particular it is independent of $q_2(x)$. The similar statement is true for $P_2(q_2(x))$.

Brief history. The notion of PCP was introduced by Arora and Safra [2]. It was a variation of randomized oracle machines discussed by Fortnow, Rompel and Sipser [20] and transparent proofs by Babai et al [4]. Multi-prover interactive proofs were introduced by Ben-Or et al [10], and all these systems are variants of interactive proofs as introduced by Goldwasser, Micali, and Rackoff [23] and Babai [3].

2.3 Essential previous work

The surprising power of interactive proofs was first established in the case of one prover [28], [33] and then for many provers [5]. After the fundamental connection with approximation was discovered [17] the parameters of the proofs improved, culminating in the following result [2, 1].

Theorem 2.11 *[1] For any integer $k \geq 3$ there is a constant $c_k < 1$ such that any language in NP admits a PCP with soundness c_k and a probabilistic polynomial time verifier V that uses logarithmic randomness and reads at most k bits of the proof.*

Note that the soundness of any V can be improved by making d independent runs. This implies, in particular, that $c_{dk} \leq c_k^d$ and hence the constant c_k can be made to go to 0 when k increases. The number of bits read cannot, unless P=NP, be decreased to 2 preserving the property that V always accepts a correct proof of a correct NP-statement. This follows from the fact that one can decide whether a 2-CNF formula is satisfiable in polynomial time (for a formal proof see [7]).

Properties described by reading 3 bits of a proof can be coded by a 3-CNF formula where the variables in the formula correspond to the bits of the proof. The acceptance probability of a proof is closely related to the number of clauses satisfied by the corresponding assignment and in this case Theorem 2.11 can be rephrased.

Theorem 2.12 *[1] There is a universal constant $c < 1$ such that, given an arbitrary NP-statement, we can, in polynomial time, construct a 3-CNF formula φ such that if the NP-statement is true then φ is satisfiable while if the NP-statement is false, no assignment satisfies a fraction larger then c of the clauses.*

Subsequent work, [26], has shown that this universal constant can be set to any constant larger than 7/8; however, we do not use this fact here. On the other hand it is convenient to work with a very uniform looking formula φ. The following extension, based on results in [31], is found as Proposition 1 in [15].

Theorem 2.13 *[15] There is a universal constant $c < 1$ such that, given an arbitrary NP-statement, we can, in polynomial time, construct a 3-CNF formula φ in which each clause is of length exactly 3 and such that each variable appears exactly 5 times, such that if the NP-statement is true then*

φ is satisfiable while if the NP-statement is false, no assignment satisfies a fraction larger then c of the clauses.

Let us now turn to two-prover interactive proofs. Given a one-round protocol with soundness s we can repeat it k times in sequence improving the soundness to s^k. This creates many round protocols, whereas we need our protocols to remain one-round. This can be done by what has become known as parallel repetition and this simply means that V repeats his random choices to choose k pairs of questions $(q_1^{(i)}, q_2^{(i)})_{i=1}^k$ and sends $(q_1^{(i)})_{i=1}^k$ to P_1 and $(q_2^{(i)})_{i=1}^k$ to P_2 all at once. V then receives k answers from each prover and accepts if it would have accepted in all k protocols given the individual answers. The soundness of such a protocol can be greater than s^k, but Raz [32] proved that, when the answer size is small, the soundness is exponentially decreasing with k.

Theorem 2.14 *[32] For all integers d and $s < 1$, there exists $c_{d,s} < 1$ such that given a two-prover one-round proof system with soundness s and answer sizes bounded by d, then, for all integers k, the soundness of k protocols run in parallel is bounded above by $c_{d,s}^k$.*

In fact it is sufficient for our main theorem that the acceptance probability, for fixed s and d, tends to 0, arbitrarily slowly, when k increases. We do not know of a simple proof of this fact and hence we might as well use the powerful Theorem 2.14.

Finally, we need standard Chernoff bounds to estimate the probability that we have large deviations. Constants are of no great concern and we use Theorem 4.2 and Theorem 4.3 of [29].

Theorem 2.15 *Let $p \leq \frac{1}{2}$ and let $X_1, X_2 \ldots X_n$ be independent Bernoulli random variables with $Prob[X_i = 1] = p$ for each i. Then for all δ, $0 \leq \delta \leq p$ we have*

$$Prob\left[\left|\frac{1}{n}\sum_{i=1}^{n} X_i - p\right| \geq \delta\right] \leq 2e^{-\frac{\delta^2 n}{4p}}$$

3 First steps towards a good proof system

We want to construct a proof system for an arbitrary language in NP. The basic steps in constructing such a proof system are rather simple and let us give an overview.

We start by a simple two-prover one-round protocol which is obtained more or less immediately from Theorem 2.13. We improve the soundness of this protocol by running several copies of it in parallel and using Theorem 2.14. It is possible to transform this improved two-prover protocol to a PCP simply by writing down prover answers of P_1 and P_2 to all possible questions. The answers are, however rather long and since the key quantity we want to keep small is the number of (free) bits read, we write the answers in a more useful form by asking the prover to supply the value of all Boolean functions of these answers. This is the long code of the answers as defined in [7]. This enables V to access complicated information in a single bit. The fact that we allow the proof to contain the answers of the provers in expanded form puts the extra burden on V to check that the these parts of the proof are indeed a correct code of something. Once this is established, though in a very weak sense, we prove that this something would have enabled P_1 and P_2 to convince the verifier in the parallelized two-prover protocol with a substantial probability.

We follow the above outline and start by describing the two-prover protocols. We are thus given an arbitrary NP-statement.

We translate, using Theorem 2.13, the NP-statement to a 3-CNF formula φ with the properties given in that theorem. Assume that the resulting formula has n variables and hence $m = \frac{5n}{3}$ clauses each of length exactly 3. Suppose $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where C_j contains the variables x_{a_j}, x_{b_j} and x_{c_j}. Consider the following one-round two-prover interactive proof.

Simple two-prover protocol

1. V chooses $j \in [m]$ and $k \in \{a_j, b_j, c_j\}$ both uniformly at random. V sends j to P_1 and k to P_2.

2. V receives values for x_{a_j}, x_{b_j} and x_{c_j} from P_1 and for x_k from P_2. V accepts iff the two values for x_k agree and C_j is satisfied.

Before we proceed let us make an observation. Since each clause is of length exactly 3 and each variable appears in exactly 5 clauses, if V first chooses a random variable x_k to send to P_2 and next a random clause containing x_k to send to P_1 he generates questions with exactly the same probability distribution.

Proposition 3.1 *If any assignment satisfies at most a fraction c of the clauses of φ, then V accepts in the simple two-prover protocol with probability at most $(2 + c)/3$.*

Proof: The answers by P_2 defines an assignment x^0 to all variables. Since the provers coordinate their strategies, P_1 knows x^0 and it is now not hard to determine the optimal strategy for P_1. Whenever V chooses a clause that is satisfied by x^0, P_1 answers according to x^0. Whenever V chooses a clause not satisfied by x^0, to have any probability of V accepting P_1 should not answer according to x^0 and to have minimal probability of his answer being found inconsistent with the answer of P_2 he should change the value of exactly one variable. The probability of V rejecting in this case is exactly $1/3$, and since x^0 (as well as any other assignment) satisfies at most a fraction c of the clauses, the probability that V rejects is at least $(1-c)/3$. The proposition follows. ∎

We now concentrate on the game consisting of u parallel copies of this basic game which we call the *u-parallel two-prover game*. In this game V picks u variables $(x_{i_k})_{k=1}^u$ and then uniformly at random for each k he picks a clause C_{j_k} that contains x_{i_k}. The $(C_{j_k})_{k=1}^u$ are sent to P_1 while $(x_{i_k})_{k=1}^u$ are sent to P_2. The provers return values for the queried variables and V accepts if the values are consistent and satisfy the chosen clauses. The verifier can again be made to always accept when φ is satisfiable, while the acceptance probability in the case where it is only possible to satisfy a constant fraction of the clauses is, by Theorem 2.14 and Proposition 3.1, bounded by c^u for some suitable constant c. Combining with Theorem 2.13 we get a result that is central to us and hence we state it for later reference.

Theorem 3.2 *There is a universal constant $c < 1$ such that, given an arbitrary NP-statement, we can, in polynomial time, construct a 3-CNF formula φ such that when the u-parallel two-prover game is executed on φ the following is true. If the NP-statement is true then the verifier can be made to always accept while if the NP-statement is false, no strategy of the provers can convince the verifier with probability larger than c^u.*

We reserve for the rest of this paper c to denote the value of the constant in this theorem.

To fix notation, let $U = \{x_{i_1}, x_{i_2} \dots x_{i_u}\}$ be the set of variables chosen by V and sent to P_2, and W the set of variables in $(C_{j_k})_{k=1}^u$ and thus the set of variables to which P_1 is supposed to give a value. Typically, provided no variable is chosen twice, U is of size u and W of size $3u$ and we always have $U \subseteq W$.

As discussed in the introduction to this section we want to replace this two-prover interactive proof by a PCP consisting of the answers of P_1 and P_2 given in a more redundant form.

Definition 3.3 *[7] The* long code *of a string x of length w is of length 2^{2^w}. The coordinates of the codeword are identified with all possible functions $f : \{0,1\}^w \mapsto \{0,1\}$ and the value of coordinate f is $f(x)$.*

The long code is extremely wasteful but in our applications we have $w \leq 3u$, and since u is a constant, the size of the codeword is also a constant. Consequently, the size of proof is just a constant times larger than the size that would have been required to simply write down the answers of P_1 and P_2.

To see how useful this coding can be, let us give a simple test to check the PCP we have constructed. Thus the PCP is supposed to be constructed from a satisfying assignment x^0 and for each U and W as constructed above it contains the long code of x^0 restricted to the set in question. Let us denote by A_T the supposed long code on the set T.

Simple PCP test(ℓ)

1. Choose U by choosing u variables with the uniform distribution. For $i = 1, 2 \ldots \ell$, choose a set W_i by uniformly selecting, for each variable x_{i_k}, a random clause $C_{j_k^i}$ containing x_{i_k} and letting W_i be the set of variables in the u clauses. The constructions of the different W_i are done independently.

2. Choose a random function $f : \{0,1\}^U \mapsto \{0,1\}$ and let $g_i = C_{j_1^i} \wedge C_{j_2^i} \wedge \ldots \wedge C_{j_u^i}$. Accept if $A_U(f) = A_{W_i}(f)$ and $A_{W_i}(g_i) = 1$ for all $i = 1, 2 \ldots, \ell$. Remember that f can be interpreted as a function on W_i by ignoring the coordinates not in U.

It is easy to see that the simple PCP test always accepts a correct proof. Also note that the simple PCP test uses only one free bit, as determined by $A_U(f)$, independent on the value of ℓ. Now suppose that the proof is correctly formed in the sense that, for every T, A_T is the long code of some string x^T, and that the simple PCP test accepts with probability p. We claim that for sufficiently large ℓ this gives strategies for P_1 and P_2 that make the verifier accept with probability at least $p/5$. Since the latter is at most c^u we get that the soundness of the PCP is at most $5c^u$, and thus the number of amortized free bits would be $O(1/u)$ which can be made arbitrarily small.

The mentioned strategy for the provers is almost what one expects. P_1, when asked W, answers x^W. Note that, since by assumption, $1 = A_{W_i}(g_i) = g_i(x^W)$ whenever V accepts we can assume that x^W satisfies the clauses

used to construct W. The strategy for P_2 is the optimal strategy given the strategy of P_1. In other words, given U, P_2 considers all W that could be asked in the same conversation. He knows the answer of P_1 in each case and simply chooses the assignment that maximizes the probability that the verifier accepts. For completeness we analyze this strategy in Appendix A.

The assumption that each A_T is a correct long code is extremely strong and crucial in the above analysis; there are incorrect proofs that do not satisfy this description and are always accepted. One such incorrect proof is the proof where each bit is equal to 1.

We conclude that the property of being a correct code-word is a crucial one. We next design a test to test exactly this property.

4 Testing a supposed long code

Let us first make some minor changes of the notation used so far and also introduce some new notation. We want at analyze a supposed long code $A : \{0,1\}^{2^w} \mapsto \{0,1\}$. We replace $\{0,1\}$ by $\{-1,1\}$ with -1 taking the place of 1. With this correspondence exclusive-or turns into multiplication. Other logical operators, like $\wedge$ remain defined (but note the $\wedge$ is *not* multiplication). Thus from now on, $A : \{-1,1\}^{2^w} \mapsto \{-1,1\}$. The inputs to A are thought of as functions $\{-1,1\}^w \mapsto \{-1,1\}$. A typical function is denoted by f and we also use vectors of functions denoted by $\vec{f} = (f_i)_{i=1}^s$ for some s. To distinguish a function f from the string of length 2^w which we use as an input to A, we denote the latter $\langle f \rangle$. We let A operate on a vector of functions and we let $\vec{A}(\langle \vec{f} \rangle)$ be the vector $(A(\langle f_i \rangle))_{i=1}^s$. We are also interested in functions $B : \{-1,1\}^s \mapsto \{-1,1\}$ which we think of as Boolean predicates. Combining $\vec{A}$ and $\vec{f}$ as above we get $B(\vec{A}(\langle \vec{f} \rangle))$ which is a bit and $B \circ \vec{f}$ which is the function which on input x takes the value $B(f_1(x), f_2(x) \ldots f_s(x)) = B(\vec{f}(x))$.

We are interested in the property that A describes a correct long code, i.e., that $A(\langle f \rangle) = f(x^0)$, for some x_0 and each f. In other words, a long code is simply a point evaluation. When probing a supposed long code we use the terminology "A looks like a point evaluation at x^0" to denote the fact that $A(\langle f \rangle) = f(x^0)$ for all *queried* f. Thus a correct long code always looks like a point evaluation at the input which it codes. Our main test of a long code is now rather straightforward.

The complete Non-Adaptive Test(s)

1. Pick, with the uniform distribution, s random functions $f_i : \{-1,1\}^W \mapsto \{-1,1\}$ and ask for $(A(\langle f_i \rangle))_{i=1}^s$.

2. For all Boolean predicates B of s bits ask for $A(\langle B \circ \vec{f} \rangle)$ and check that
$$A(\langle B \circ \vec{f} \rangle) = B(\vec{A}(\langle \vec{f} \rangle)).$$

First note that for A to have any chance to pass the test it must be correct on the constant functions. This follows, since if B is identically one then $B(\vec{A}(\langle \vec{f} \rangle)) = 1$ and hence $A(\langle B \circ \vec{f} \rangle)$ must also take this value and $B \circ \vec{f}$ is the function which is constant one. Similarly for B being identically -1.

We refer to the test as the CNA-test(s) and we claim that it uses s free bits, as given by the queries for $(A(\langle f_i \rangle))_{i=1}^s$. This follows since the values of $A(\langle B \circ \vec{f} \rangle)$ are known before the corresponding query is asked.

When accessing an oracle a question is called *non-adaptive* if the decision to ask the question is independent of previous answers. In the CNA-test, the verifier asks all possible non-adaptive questions to which it knows the answer given the information $\vec{A}(\langle \vec{f} \rangle)$. Note, however, that there are other questions one might ask since if $A(\langle f \rangle) = 1$ then we should have $A(\langle f \wedge f' \rangle) = 1$ for any function f'. These questions are, however, adaptive and seems harder to analyze and we do not know how to use them to simplify the current analysis.

Let us first show that if the test accepts then the outcome looks like some point evaluation.

Lemma 4.1 *Suppose the CNA-test(s) accepts using a specific set of random choices $\vec{f}$. Then there is an input x such that A looks like a point evaluation at x. In other words, $A(\langle f \rangle) = f(x)$ for all tested functions f.*

Proof: Since, in case of accept, all values are determined by $\vec{A}(\langle \vec{f} \rangle)$, it is sufficient to find an x such that $f_i(x) = A(\langle f_i \rangle)$ for $i = 1, 2 \ldots s$. Suppose there is no such x. Let $\sigma_i = A(\langle f_i \rangle)$ and consider the Boolean predicate $B_{\vec{\sigma}}(z) \triangleq (\wedge_{i=1}^s (z_i = \sigma_i))$. Then $B_{\vec{\sigma}} \circ \vec{f}$ is a function which is identically false. This follows since, by our hypothesis, for each x there is an i such that $f_i(x) \neq A(\langle f_i \rangle)$ and this causes the i'th term in the expression for $B(\vec{f}(x))$ to be false. $B_{\vec{\sigma}}(\vec{A}(\langle \vec{f} \rangle))$ is, however, true and hence $A(\langle B_{\vec{\sigma}} \circ \vec{f} \rangle) = 1$ while $B_{\vec{\sigma}}(\vec{A}(\langle \vec{f} \rangle)) = -1$ and the test rejects. This is a contradiction. ∎

Before going into the analysis consider the following example.

Example: For any three specific assignments x^0, x^1 and x^2 let

$$A(\langle f \rangle) = f(x^0) \oplus f(x^1) \oplus f(x^2).$$

This is not a correct long code and in fact it is not difficult to see that any correct long code takes the same value as A for exactly half of the possible f. Now consider what happens when we do the CNA-test. Suppose that $f_i(x^0) = f_i(x^1)$ for $i = 1, 2, \ldots, s$. Then this is also true for any $B \circ \vec{f}$ and hence $A(\langle f \rangle) = f(x^2)$ for all queried functions f. Thus, with probability at least 2^{-s} the test accepts and the result looks like the long code for x^2. Similarly it is possible to get results that look like the long codes for x^0 or x^1, respectively.

Since we want an arbitrarily small number of amortized free bits we cannot afford a failure probability of 2^{-s} when we are using s free bits (since this gives at least one amortized free bit). Thus, we modify the acceptance criteria by allowing the supposed long code to look like a small number of different correct long codes. The important property is that this set S of possible long codes is small and that it can be specified in advance before performing the test. In the above example we have $S = \{x^0, x^1, x^2\}$.

Let us return to the main path. In the following theorem, $C_{\epsilon,k}$ (resp., $D_{\epsilon,k,s}$) is a constant depending on only ϵ and k (resp. ϵ, k, and ϵ).

Theorem 4.2 *For any $\epsilon > 0$ and positive integer k, for $s \geq C_{\epsilon,k}$ and $w \geq D_{\epsilon,k,s}$ the following is true. For any $A : \{-1,1\}^{2^w} \mapsto \{-1,1\}$ there is a set S containing at most $2^{\epsilon s}$ points in $\{-1,1\}^w$ such that when the CNA-test(s) is performed, except with probability 2^{-ks}, the test either rejects or the outcome is consistent with being a point evaluation at an element $x \in S$.*

The probability is taken over the random choices of the verifier performing the test, i.e. over the choice of random functions f_i.

Proof: Decreasing ϵ only strengthens the conclusion since it decreases the allowed size for S, and hence we can assume that $\epsilon \leq 1/2$. We do not only assume that s is sufficiently large compared to k and ϵ but also compared to constants l and m to be introduced later. This can be done since the latter constants are made to depend only on k and ϵ. A constant denoted by $c_{l,m}$ depends in some way on parameters m and l but not on other parameters. The value of $c_{l,m}$ might change from line to line.

The proof relies on Fourier transforms and we assume the reader is familiar with the basic concepts. In our setting,[3] the Fourier coefficients are defined by

$$\hat{A}_\alpha = 2^{-2^w} \sum_{f:\{-1,1\}^w \mapsto \{-1,1\}} A(\langle f \rangle) \prod_{x \in \alpha} f(x) \tag{1}$$

where $\alpha \subseteq \{-1,1\}^w$ and we also have Fourier inversion given by

$$A(\langle f \rangle) = \sum_{\alpha \subseteq \{-1,1\}^w} \hat{A}_\alpha \prod_{x \in \alpha} f(x).$$

By Parseval's identity and $|A(\langle f \rangle)| = 1$ for all f, we know that $\sum_\alpha \hat{A}_\alpha^2 = 1$.

First we note that whenever the test asks A about a function f it also asks about the function $-f$. This implies that it is optimal for the adversary to have $A(\langle f \rangle) = -A(\langle -f \rangle)$ since any violation of this causes immediate rejection. From this point on we assume[4] that indeed $A(\langle f \rangle) = -A(\langle -f \rangle)$ is true for all f. This implies that $\hat{A}_\alpha = 0$ for all α with $|\alpha|$ even. This follows since the terms for f and $-f$ cancel each other in the defining sum (1).

The set S is taken to be the points that are elements in Fourier coefficients that have a large absolute value and correspond to small sets. To be more exact

$$S = \{x \mid \exists \alpha, \alpha \ni x \text{ such that } |\alpha| \leq l \wedge \hat{A}_\alpha^2 \geq l 2^{-\epsilon s}\}. \tag{2}$$

where l is a parameter (depending only on k and ϵ) to be specified later. Since $\sum_\alpha \hat{A}_\alpha^2 = 1$ at most $2^{\epsilon s} l^{-1}$ different α have $\hat{A}_\alpha^2 \geq l 2^{-\epsilon s}$ and since each α contributes at most l points to S, it follows that S contains at most $2^{\epsilon s}$ points as required by the theorem.

[3] The setup might look a little bit unfamiliar. Normally, we deal with function $f : \{-1,1\}^n \mapsto \{-1,1\}$ and then

$$\hat{f}_\alpha = 2^{-n} \sum_x f(x) \prod_{i \in \alpha} x_i \quad \text{and} \quad f(x) = \sum_{\alpha \subseteq [n]} \hat{f}_\alpha \prod_{i \in \alpha} x_i.$$

In this familiar case we can view x as a function from $[n]$ to $\{-1,1\}$. In the present situation, however, the argument to A is a function from $\{-1,1\}^w$ to $\{-1,1\}$ and thus it is natural that $\{-1,1\}^w$ takes the place of $[n]$ in the definition of the Fourier transform.

[4] Technically this is justified by modifying A to satisfy this property and then working with the modified A.

4.1 Concentrating on a specific point evaluation

We want to analyze the probability that the CNA-test accepts and is not consistent with a point evaluation at any point of S. Recall that whenever the test accepts it is consistent with *some* point y (cf. Lemma 4.1). It is easier to analyze the probability that for a fixed point $y \notin S$ the test is consistent with a point evaluation at y but not at any point of S.

Whenever the outcome of the test is consistent with a point evaluation at x it is also consistent with a point evaluation at any point x' such that $f_i(x) = f_i(x')$ for all i's. Since the f_i's are random functions we expect about 2^{w-s} such x'. Thus, arguing informally, if we have probability p of the test accepting and not being consistent with a point evaluation at any point S there should be a point y such that the probability of being consistent with a point evaluation at y but not at any point in S should be around $p2^{-s}$. Since p anyway is of the form 2^{-ks} for an arbitrary k we lose little by replacing 2^{-ks} with $2^{-(k+1)s}$ while the advantage of working with a specific y is significant. We make this argument formal.

Lemma 4.3 *Let $\vec{f} = f_1, f_2, \ldots, f_s$ be uniformly and independently selected random functions. The probability that there exists a vector $\vec{b} = (b_1, \ldots b_s)$ such that*

$$|\{x : \vec{f}(x) = \vec{b}\}| < 2^{w-(s+1)}$$

is bounded by $2^{s+1}2^{-2^{w-(s+4)}}$.

Proof: Fix any value of $\vec{b}$. The probability that x satisfies $\vec{f}(x) = \vec{b}$ is 2^{-s} and it is independent for different x. Thus we can apply the Chernoff bound (cf. Theorem 2.15) with $n = 2^w$, $p = 2^{-s}$ and $\delta = 2^{-(s+1)}$. Summing over all possible $\vec{b}$, the result follows. ∎

Now assume that we have probability p of the CNA-test(s) accepting while not being consistent with any point in S. Then if $p > 2^{-ks}$, by Lemma 4.3 we can conclude that, for sufficiently large w (e.g. $w > s + 6 + \log(ks)$), we have probability $p/2$ of the test accepting, not being consistent with any point in S *and* being consistent with $2^{w-(s+1)}$ different points. Now for any point $y \notin S$ let P_y be the probability that the CNF-test is consistent with y but not any point in S. It follows that

$$\sum_y P_y \geq (p/2) \cdot 2^{w-(s+1)}$$

as each accept event described above is counted in at least $2^{w-(s+1)}$ different P_y's and it happens with probability at least $p/2$. Hence for some y we have

must have

$$P_y \geq p2^{-(s+2)} \geq 2^{-(k+2)s}.$$

We state this for future reference.

Lemma 4.4 *Assume that $w \geq w_{s,k}$. If for any fixed $y \notin S$, the probability that the outcome of the CNA-test(s) is consistent with a point evaluation at y but not with a point evaluation at any point in S is at most $2^{-(k+2)s}$, then Theorem 4.2 follows.*

Fixing $y \notin S$, we proceed to estimate the probability that the CNA-test(s) is consistent with y but not with any point of S. We represent the first event (i.e. the consistency of the test with point evaluation at y) by

$$\sum_B I_y(\langle B \circ \vec{f} \rangle) = 0 \tag{3}$$

where I_y is an indicator function defined by $I_y(\langle f \rangle) = 1$ if $A(\langle f \rangle) \neq f(y)$ and 0 otherwise. For technical reasons we only sum over those B which are unbiased i.e. take the value one at exactly 2^{s-1} points. This makes the sum only smaller and corresponds to allowing only unbiased B in the CNA-test. Note that for such a B, the function $B \circ \vec{f}$ is a random function with uniform distribution. Of course, different B do not give independent random functions, but each function in itself is random. Denote by Y the random variable defined by the sum (3). Using

$$I_y(\langle f \rangle) = \frac{1 - A(\langle f \rangle) f(y)}{2},$$

we show (below) that the Fourier coefficients $\hat{I}_{\alpha,y}$ of I_y satisfy

$$\hat{I}_{\emptyset,y} = \frac{1}{2}(1 - \hat{A}_{\{y\}}) \tag{4}$$

$$\hat{I}_{\alpha,y} = -\frac{1}{2}\hat{A}_{\alpha \Delta \{y\}}, \text{ for any } \alpha \neq \emptyset, \tag{5}$$

where Δ denotes symmetric difference of sets. A basic fact that we will use many times is that for a uniformly chosen function f, the $f(x)$'s are identical and independent random variables, each uniformly distributed in $\{-1, 1\}$. Thus, $\sum_f \prod_{x \in \alpha} f(x) = 0$ for every nonempty set α, and (4) and (5) follow. For example, for $\alpha \neq \emptyset$,

$$\begin{aligned} \hat{I}_{\alpha,y} &= 2^{-2^w} \cdot \frac{1}{2} \sum_{f:\{-1,1\}^w \mapsto \{-1,1\}} (1 - A(\langle f \rangle) f(y)) \prod_{x \in \alpha} f(x) \\ &= -2^{-2^w} \cdot \frac{1}{2} \sum_{f:\{-1,1\}^w \mapsto \{-1,1\}} A(\langle f \rangle) \prod_{x \in \alpha \Delta \{y\}} f(x) = -\frac{1}{2}\hat{A}_{\alpha \Delta \{y\}}. \end{aligned}$$

Now

$$\begin{aligned} Y &= \sum_B I_y(B \circ \vec{f}) = \sum_B \sum_\alpha \hat{I}_{\alpha,y} \prod_{x\in\alpha} B(\vec{f}(x)) \\ &= \frac{1}{2}\binom{2^s}{2^{s-1}} - \frac{1}{2}\sum_B \sum_\alpha \hat{A}_{\alpha\Delta\{y\}} \prod_{x\in\alpha} B(\vec{f}(x)). \end{aligned}$$

We divide this last sum into three pieces. The first sum, Σ^1, is over those α with $|\alpha| \geq l$, the second, Σ^2, is over those α with $|\alpha| < l$ and $\hat{A}^2_{\alpha\Delta\{y\}} \geq l2^{-\epsilon s}$ and Σ^3 is over the rest, i.e. over α with $|\alpha| < l$ and $\hat{A}^2_{\alpha\Delta\{y\}} < l2^{-\epsilon s}$. The random variable Y can now be written as

$$Y = \frac{1}{2}\binom{2^s}{2^{s-1}} - \frac{1}{2}(Y_1 + Y_2 + Y_3),$$

where Y_i corresponds to the sum Σ^i. In order for y to be a possible point of evaluation we need $Y = 0$, and hence $Y_i \geq \frac{1}{3}\binom{2^s}{2^{s-1}}$ for $i = 1$, 2 or 3. Recall that we are actually interested in the intersection of the event $Y = 0$ and the event that the test is *not* consistent with any point evaluation in S. Thus, we may analyze the partial sums, Y_i's, assuming that the latter event holds (i.e. inconsistency with S). Actually, we take advantage of this liberty only in the analysis of Y_2. Specifically, we use the fact that in this case for every $x \in S$ there exists a query f_i of the test so that $f_i(x) \neq f_i(y)$. In other word for $\vec{f}$ selected by the test we have $\vec{f}(x) \neq \vec{f}(y)$, for every $x \in S$.

4.2 Estimating Y_1.

It is not difficult to bound

$$Y_1 \triangleq \sum_B \sum_{\alpha \mid |\alpha|\geq l} \hat{A}_{\alpha\Delta\{y\}} \prod_{x\in\alpha} B(\vec{f}(x))$$

since we can compute the second moment almost immediately.

Lemma 4.5

$$E(Y_1^2) \leq \left(2^{-(ls)/4} + 2e^{-2^{(s/2)-4}}\right) \cdot \binom{2^s}{2^{s-1}}^2.$$

Before we prove this lemma let us state the immediate corollary we really need.

Corollary 4.6 *For each integer k there is a constant s_k such that for $l > 4k+9$, and $s \geq s_k$ we have*

$$Prob\left[Y_1 \geq \frac{1}{3} \cdot \binom{2^s}{2^{s-1}}\right] \leq 2^{-((k+2)s+1)}.$$

Proof: (Of Corollary 4.6) By Markov's inequality (applied to Y_1^2), the probability that $Y_1 \geq X$ is at most $X^{-2}E(Y_1^2)$. Substituting $X = \frac{1}{3} \cdot \binom{2^s}{2^{s-1}}$ and the bound for $E(Y_1^2)$ given by Lemma 4.5 and doing a calculation establishes the corollary. ∎

Proof: (Of Lemma 4.5)

$$E(Y_1^2) = E\left(\left(\sum_B \sum_{\alpha,|\alpha|\geq l} \hat{A}_{\alpha\Delta y} \prod_{x\in\alpha} B(\vec{f}(x))\right)^2\right) =$$

(for notational simplicity we skip the condition $|\alpha| \geq l$ in the calculations)

$$= E\left(\sum_{B_1,B_2} \sum_{\alpha_1,\alpha_2} \hat{A}_{\alpha_1\Delta\{y\}}\hat{A}_{\alpha_2\Delta\{y\}} \prod_{x\in\alpha_1} B_1(\vec{f}(x)) \prod_{x\in\alpha_2} B_2(\vec{f}(x))\right)$$

$$= \sum_{\alpha_1,\alpha_2} \hat{A}_{\alpha_1\Delta\{y\}}\hat{A}_{\alpha_2\Delta\{y\}} \sum_{B_1,B_2} E\left(\prod_{x\in\alpha_1} B_1(\vec{f}(x)) \prod_{x\in\alpha_2} B_2(\vec{f}(x))\right) \quad (6)$$

We claim that whenever $\alpha_1 \neq \alpha_2$ the inner expected value is 0. To see this assume that $x^0 \in \alpha_1$ and $x^0 \notin \alpha_2$. Then $B_1(\vec{f}(x^0))$ has expected value 0 and is independent of all other variables that influence the product. The case $x^0 \notin \alpha_1$ and $x^0 \in \alpha_2$ is of course symmetric. The remaining terms are

$$\sum_{\alpha} \hat{A}^2_{\alpha\Delta\{y\}} \sum_{B_1,B_2} E\left(\prod_{x\in\alpha} B_1(\vec{f}(x))B_2(\vec{f}(x))\right). \quad (7)$$

To estimate this we first establish.

Lemma 4.7 *For any function G that maps $\{-1,1\}^s$ to $\{-1,1\}$, the probability, over a random unbiased predicate B, that*

$$\left|\sum_{z\in\{-1,1\}^s} G(z)B(z)\right| \geq k \quad (8)$$

is bounded by $2e^{-k^2 2^{-(s+4)}}$.

Proof: We opt for a simple proof rather than the best bounds. Think of choosing an unbiased predicate B as first pairing the elements of $\{-1,1\}^s$ into 2^{s-1} disjoint pairs and then giving the value 1 to exactly one element in each pair. If both the pairing and the choice which variable gets the value 1 in each pair are done with the uniform distribution we select a random B with the uniform distribution. Now fix any pairing and analyze the event in the lemma using only the randomness of the choice within the pairs. For a pair (x^1, x^2) the contribution to the sum in (8) is always 0 when $G(x^1) = G(x^2)$, and otherwise it is either 2 or -2 depending on the choice within the pair. Thus, the sum in (8) is the sum of t random variables taking the values 2 and -2 with equal probability, where t is the number of pairs (x^1, x^2) with $G(x^1) = -G(x^2)$. The lemma now follows from Chernoff bounds (Theorem 2.15). Specifically, letting $X_i' \in \{\pm 2\}$ denote the value of the i'th pair, and using $X_i = (X_i' + 2)/4$, we have

$$Prof\left[\left|\sum_{i=1}^{t} X_i'\right| \geq k\right] = Prof\left[\left|\frac{1}{t}\sum_{i=1}^{t} X_i - \frac{1}{2}\right| \geq \frac{k}{4t}\right]$$

$$< 2 \cdot e^{-\frac{(\frac{k}{4t})^2 \cdot t}{4 \cdot \frac{1}{2}}} \leq 2 \cdot e^{-k^2 2^{-(s+4)}},$$

where the last inequality uses $t \leq 2^{s-1}$. ∎

Let us resume the analysis of (7). Assume that for some (fixed) B_1 and B_2

$$\left|\sum_{z \in \{-1,1\}^s} B_1(z)B_2(z)\right| \leq 2^{3s/4}. \tag{9}$$

Then this is just another way of saying that for each $x \in \alpha$

$$|E(B_1(\vec{f}(x))B_2(\vec{f}(x)))| \leq 2^{-s/4}$$

and since we have independence for different x we have

$$\left|E\left(\prod_{x \in \alpha} B_1(\vec{f}(x))B_2(\vec{f}(x))\right)\right| \leq 2^{-|\alpha|s/4}.$$

For fixed B_2, the fraction of B_1 violating (9) is, by Lemma 4.7, bounded by $2e^{-2^{s/2-4}}$. Since $|\alpha| \geq l$ for any term in (7) we get

$$E(Y_1^2) \leq \sum_{\alpha} \hat{A}^2_{\alpha\Delta\{y\}} \cdot (2^{-ls/4} + 2e^{-2^{s/2-4}})\binom{2^s}{2^{s-1}}^2,$$

since $\binom{2^s}{2^{s-1}}$ is the number of unbiased B's and each term in (7) is bounded above by 1. Using $\sum_\alpha \hat{A}_\alpha^2 \leq 1$, Lemma 4.5 follows. ∎

4.3 Estimating Y_2

Our second term Y_2, which sums over α's of size at most l which correspond to large Fourier coefficients, is estimated by a worst case estimate, using the hypothesis that any element $x \in S$ we have $\vec{f}(y) \neq \vec{f}(x)$ (see above). The key fact is that since we only use Fourier coefficients that contain elements from S (defined explicitly to contain only the α's considered here), the summation over B creates a lot of cancellation.

Lemma 4.8 *For any integer l and $\epsilon > 0$, there is a constant $s_{l,\epsilon}$ such that for $s > s_{l,\epsilon}$ and any choice of $\vec{f}$ such that $\vec{f}(x) \neq \vec{f}(y)$ for any $x \in S$ we have $|Y_2| \leq 2^{1+\epsilon s - s}\binom{2^s}{2^{s-1}} < \frac{1}{3}\binom{2^s}{2^{s-1}}$.*

Proof: Remember that

$$Y_2 = \sum_{\alpha, |\alpha|<l, \hat{A}^2_{\alpha\Delta\{y\}} \geq l2^{-\epsilon s}} \hat{A}_{\alpha\Delta\{y\}} \sum_B \prod_{x\in\alpha} B(\vec{f}(x)) \tag{10}$$

Since $y \notin S$, by the definition of S, for all α in the above sum $y \notin \alpha\Delta\{y\}$ and hence $y \in \alpha$. We now have the following lemma

Lemma 4.9 *Let $z_1, z_2 \ldots z_r \in \{-1, 1\}^s$ be any values such that $z_i \neq z_j$ for $i \neq j$. Then if we sum over all unbiased predicates*

$$\sum_B \prod_{i=1}^{r} B(z_i) = (-1)^{r/2} \prod_{i=1}^{r/2} \frac{r+1-2i}{2^s+1-2i} \binom{2^s}{2^{s-1}}. \tag{11}$$

if r is even and otherwise the sum is 0.

Proof: The statement for odd r is obvious since the terms for the predicates B and $-B$ cancel each other. For even r, think of the sum in (11) as an expected value over a random unbiased B. Again pick a random B as in the proof of Lemma 4.7 by first randomly picking a pairing and then randomly giving one element in each pair the value 1 and the other the value -1. If the r elements do not pair up, the expected value over the second random choice is 0 while if they do pair up it is $(-1)^{r/2}$. To analyze the probability that the elements pair up we put the elements z_i into pairs one by one. The element z_1 goes into some pair. The probability that its mate is one of the

z_i is $(r-1)/(2^s-1)$. This follows since there are 2^s-1 possible partners of which $r-1$ are allowed. Assume that z_1 did pair up with z_j and consider any remaining z_i. The probability that it pairs up is by a similar reasoning $(r-3)/(2^s-3)$ and we continue in this way until all elements are paired up. The probability this happens is

$$\prod_{i=1}^{r/2} \frac{r+1-2i}{2^s+1-2i}$$

and the lemma follows. ∎

Now, for any α in the sum (10) consider the set of values $\vec{f}(x)$ for $x \in \alpha$. Make these values pairwise distinct by simply erasing both element of pairs that are equal (this does not affect the product) resulting in a set $\alpha' \subseteq \alpha$. Since $\vec{f}(y) \neq \vec{f}(x)$ for any $x \in \alpha$ where $x \neq y$ and $y \in \alpha$, we get $|\alpha'| \geq 2$ (remember that $|\alpha \Delta \{y\}|$ is odd since otherwise $\hat{A}_{\alpha\Delta\{y\}} = 0$) and clearly $|\alpha'| \leq l$. Using Lemma 4.9 we see that the maximal value (assuming $l \leq s$) of $|\sum_B \prod_{x\in\alpha} B(\vec{f}(x))|$ is obtained $\alpha' = 2$ and we get

$$\left| \sum_B \prod_{x\in\alpha} B(\vec{f}(x)) \right| \leq \frac{1}{2^s-1} \cdot \binom{2^s}{2^{s-1}} < 2^{1-s} \cdot \binom{2^s}{2^{s-1}}.$$

Substituting this in (10) gives

$$\begin{aligned} |Y_2| &\leq \sum_{\alpha, \hat{A}_\alpha^2 \geq l 2^{-\epsilon s}} |\hat{A}_\alpha| \cdot 2^{1-s} \binom{2^s}{2^{s-1}} < \sum_\alpha \hat{A}_\alpha^2 \cdot 2^{1+\epsilon s - s} \binom{2^s}{2^{s-1}} \\ &= 2^{1+\epsilon s - s} \binom{2^s}{2^{s-1}} \end{aligned}$$

and Lemma 4.8 follows. ∎

4.4 Estimating Y_3

To estimate Y_3 we calculate a high order moment. As when calculating the second moment of Y_1, many terms in the Fourier expansion do vanish due to the expected value being 0. The remaining sum is somewhat nontrivial and we use the properties of functions with the Fourier support concentrated on small sets. (Recall that Y_3 is the sum over α's of small size which correspond to small coefficients.)

Lemma 4.10 *Let m be an even integer then for any integer l there is a constant $c_{l,m}$ such that*

$$E(Y_3^m) \leq c_{l,m} \cdot \left(e^{-2^{s/2-4}} + 2^{-\epsilon m s/4} + 2^{-ms/16} \right) \cdot \binom{2^s}{2^{s-1}}^m.$$

Again we have an immediate corollary that gives us what we really want.

Corollary 4.11 *For any integers k and l and $\epsilon > 0$ there is constant $s_{k,l,\epsilon}$ such that for $s \geq s_{k,l,\epsilon}$*

$$Prob\left[Y_3 \geq \frac{1}{3}\binom{2^s}{2^{s-1}} \right] \leq 2^{-((k+2)s+1)}.$$

Proof: (Of Corollary 4.11) Analogously to the proof of Corollary 4.6, for any X, the probability that $Y_3 \geq X$ is bounded by $X^{-m}E(Y_3^m)$ for any even integer m. Now set $X = \frac{1}{3}\binom{2^s}{2^{s-1}}$ and $m > \max(4 \cdot (k+3) \cdot \epsilon^{-1}, 16 \cdot (k+3))$, apply Lemma 4.10, set s sufficiently large and make a calculation. ∎

Proof: (Of Lemma 4.10) We have

$$E(Y_3^m) = \sum_{\alpha_1, \alpha_2 \ldots \alpha_m} \left(\prod_{i=1}^m \hat{A}_{\alpha_i \Delta \{y\}} \right) \sum_{B_1, B_2 \ldots B_m} E\left(\prod_{i=1}^m \prod_{x \in \alpha_i} B_i(\vec{f}(x)) \right) \quad (12)$$

where the sum ranges over all α_i satisfying $|\alpha_i| < l$ and $\hat{A}^2_{\alpha_i \Delta \{y\}} \leq l 2^{-\epsilon s}$.

To analyze a generic term, set $T = \cup_{i=1}^m \alpha_i$. We first claim that if there is some $x \in T$ that belongs to exactly one α_i then the inner expected value is 0. This follows since in this case $B_i(\vec{f}(x))$ is an unbiased random variable that is independent of all other factors in the product. Thus, we are interested in sets α_i that form a *double cover* (of T). By this we mean that each $x \in T$ appears in at least two different α_i's. From now on we only sum over α_i's which form a double cover.

Lemma 4.12 *If α_i form a double cover of T, then*

$$\sum_{B_1, B_2 \ldots B_m} E\left(\prod_{i=1}^m \prod_{x \in \alpha_i} B_i(\vec{f}(x)) \right) \leq \left(2^{-|T|s/4} + 2^{m+1} e^{-2^{s/2-4}} \right) \binom{2^s}{2^{s-1}}^m.$$

Proof: This is again based on the fact that for almost all $B_1, B_2 \ldots B_m$, $E(\prod_{i|x \in \alpha_i} B_i(\vec{f}(x)))$ is exponentially small and that it is independent for

different $x \in T$. Consider any nonempty subset C of $\{1, 2, \ldots m\}$. The probability (over random unbiased $B_1, B_2 \ldots B_m$) that

$$\left| \sum_{z \in \{-1,1\}^s} \prod_{i \in C} B_i(z) \right| \geq 2^{3s/4} \tag{13}$$

is bounded, by Lemma 4.7, by $2e^{-2^{s/2-4}}$. Thus the number of B_i such that (13) is violated for any C is bounded by

$$2^{m+1} e^{-2^{s/2-4}} \binom{2^s}{2^{s-1}}^m.$$

For any other sequences of B_i's we have

$$|E(\prod_{i|x \in \alpha_i} B_i(\vec{f}(x)))| \leq 2^{-s/4}$$

for any $x \in T$. Since the values of f at different points are independent we have

$$\left| E\left(\prod_{i=1}^{m} \prod_{x \in \alpha_i} B_i(\vec{f}(x)) \right) \right| = \prod_{x \in T} \left| E\left(\prod_{i|x \in \alpha_i} B_i(\vec{f}(x)) \right) \right| \leq 2^{-|T|s/4}.$$

The lemma follows by summing the two terms. ∎

We proceed to bound (12). We first estimate that sum over *even covers*, i.e. those collections of α_i such that each x is a member of an even number of α_i. For an arbitrary function $F : \{-1, 1\}^{2^w} \mapsto \mathbf{R}$ denote the sum

$$\sum_{\alpha_1, \alpha_2 \ldots \alpha_m} \left| \prod_{i=1}^{m} \hat{F}_{\alpha_i} \right|$$

by $dc_m(F)$ when we sum over double covers and by $ec_m(F)$ when we sum over even covers. We later apply the estimates with $F(\langle f \rangle) = f(y)A(\langle f \rangle)$ which has $\hat{F}_\alpha = \hat{A}_{\alpha \Delta \{y\}}$.

Lemma 4.13 *For any function F such that $\hat{F}_\alpha = 0$ for $|\alpha| > l$ we have $ec_m(F) \leq c_{m,l} \|F\|_2^m$.*

Proof: Let F' be the function with Fourier coefficients $|\hat{F}_\alpha|$. Since

$$\sum_f \prod_{i=1}^{m} \prod_{x \in \alpha_i} f(x) = 2^{2^w}$$

when α_i form an even cover and this sum is 0 otherwise, we have

$$\begin{aligned} ec_m(F) &= 2^{-2^w} \sum_f \sum_{\alpha_1,\alpha_2\ldots\alpha_m} \prod_{i=1}^{m} (|\hat{F}_{\alpha_i}| \cdot \prod_{x\in\alpha_i} f(x)) \\ &= 2^{-2^w} \sum_f \left(\sum_\alpha |\hat{F}_\alpha| \prod_{x\in\alpha} f(x) \right)^m \\ &= 2^{-2^w} \sum_f F'(\langle f \rangle)^m = \|F'\|_m^m \end{aligned}$$

where we are using the standard L_m-norm. However, when considering functions whose Fourier support are on sets of constant size, the various L_p norms are all related (this is Proposition 3 in [13]):

Lemma 4.14 *[13] For every constants l and m there is a constant $c_{l,m}$, such that all function F such that $\hat{F}_\alpha = 0$ for $|\alpha| > l$ we have*

$$\|F\|_m \leq c_{l,m} \cdot \|F\|_2.$$

Lemma 4.13 now follows from Lemma 4.14, $\|F'\|_2 = \|F\|_2$, and the above reasoning. ∎

Next we do a similar estimate for double covers (remember that constants $c_{m,l}$ might change their value):

Lemma 4.15 *For any function F such that $\hat{F}_\alpha = 0$ for $|\alpha| > l$ we have $dc_m(F) \leq c_{m,l}\|F\|_2^m$.*

Proof: From the function F we probabilistically construct a different function F' such that each term in the sum for $dc_m(F)$ has a constant (depending on l and m) probability of occurring in the sum for $ec_m(F')$. We make sure that $\|F\|_2 = \|F'\|_2$ while the size of the nonzero Fourier coefficients only increase by a factor of 2.

The construction is as follows: Replace each $x \in \{-1,1\}^w$ by two inputs x^1 and x^2. For each α we construct α' by, randomly and independently for each $x \in \alpha$, letting α' contain only x^1, only x^2, or both with probability 1/3 each. For example $\alpha = \{x, y\}$ may be replaced to either of the nine sets $\{x^1, y^1\}, \{x^2, y^1\}, \{x^1, x^2, y^1\}, \{x^1, y^2\}, \{x^2, y^2\}, \{x^1, x^2, y^2\}, \{x^1, y^1, y^2\}, \{x^2, y^1, y^2\}$, or $\{x^1, x^2, y^1, y^2\}$. The mapping $\alpha \mapsto \alpha'$ is injective and we define the function F' by its Fourier coefficients. We have

$$\hat{F}'_\beta = \begin{cases} |\hat{F}_\alpha| & \text{if } \beta = \alpha' \\ 0 & \text{otherwise} \end{cases}$$

and in particular $\hat{F}'_\beta = 0$ for every β of size greater than $2l$. We claim that each term in the sum for $dc_m(F)$ has a probability at least $(2/9)^{ml/2}$ to appear in the sum for $ec_m(F')$.

Namely suppose that for a term $\prod_{i=1}^m \hat{F}_{\alpha_i}$, $(\alpha_i)_{i=1}^m$ form a double cover of T. Clearly $|T| \leq ml/2$. Let T' be obtained from T by replacing each element x by the two elements x^1 and x^2. We claim that probability that $(\alpha'_i)_{i=1}^m$ form an even cover of T' is at least $(2/9)^{ml/2}$. Since replacement is done independently for each x, we just need to establish that the probability that both x^1 and x^2 are covered an even number of times is at least $2/9$. Take any two sets that contain x (assume that these are α_1 and α_2 and let $\alpha_3, \alpha_4, \ldots \alpha_r$ be all other sets containing x). Now there are four cases to consider and let us for brevity only consider one, the other cases being similar. Suppose α'_i for $i \geq 3$ contain x^1 an odd number of times, and x^2 an even number of times. Then if α'_1 contains both elements and α'_2 only x^2 (or the other way around) both x^1 and x^2 are covered an even number of times. This happens with probability $2/9$.

To wrap up the proof, note that, by Lemma 4.13, for any F, $ec_m(F') \leq c_{m,2l}\|F'\|_2^m = c_{m,2l}\|F\|_2^m$ and by the above argument

$$dc_m(F) \leq (9/2)^{ml/2} \cdot E(ec_m(F'))$$

and thus adjusting the value of the constant $c_{l,m}$, the lemma follows. ∎

Lemma 4.15 and Lemma 4.12 can be used to bound the part of the sum (12) when $|T|$ is large (see below). Next we address the case when $|T|$ is small.

Lemma 4.16 *We have*

$$\sum_{\alpha_1,\alpha_2\ldots\alpha_m} \prod_{i=1}^m |\hat{A}_{\alpha_i \Delta \{y\}}| \leq c_{m,l,t} 2^{-(m-2t)\epsilon s/2} \tag{14}$$

where the sum is over all $(\alpha_i)_{i=1}^m$ that form a double cover of some set of size t and such that $|\alpha_i| < l$ and $\hat{A}^2_{\alpha_i \Delta \{y\}} < l2^{-\epsilon s}$ for $i = 1, 2, \ldots, m$.

Proof: The key to this proof is that there are few such collections of sets $(\alpha_i)_{i=1}^m$ while $\hat{A}_{\alpha_i \Delta \{y\}}$ are small and as there are many factors in each product this makes the total sum very small.

Consider the sum

$$\sum_{\alpha_1,\alpha_2\ldots\alpha_{2t}} \prod_{i=1}^{2t} |\hat{A}_{\alpha_i \Delta \{y\}}|$$

where the sum is over all $(\alpha_i)_{i=1}^m$ that form a double cover of some set of size t. By Lemma 4.15, with $F(\langle f \rangle) = f(y)A(\langle f \rangle)$, this is bounded by a constant $c_{t,l}$. Now for each double cover with m elements choose a double sub-cover of the same T with $2t$ elements. This is always possible. Since each $\hat{A}_{\alpha\Delta\{y\}}$ that we are considering is bounded by $(l2^{-\epsilon s})^{1/2}$, the original double cover has weight (i.e. the value of the corresponding product) which is at most $(l2^{-\epsilon s})^{(m-2t)/2}$ times the weight of the double sub-cover. Each sub-cover of size $2t$ can occur for at most $(\sum_{i=0}^{l} \binom{t}{i})^{m-2t}$ original covers of size m. This follows since there are at most $\sum_{i=0}^{l} \binom{t}{i}$ subsets of T of size at most l and hence at most that many choices for each α_i. Hence we get the total bound for the sum in (14)

$$(l2^{-\epsilon s})^{(m-2t)/2} \cdot \left(\sum_{i=0}^{l} \binom{t}{i} \right)^{m-2t} c_{t,l} < c_{t,l,m} 2^{-\epsilon s(m-2t)/2}$$

and the lemma follows. ∎

Let us now conclude the proof of Lemma 4.10. We divide the sum (12) according to the size of $T = \cup_{i=1}^{m} \alpha_i$. Summing over T with $|T| \geq m/4$, we use Lemma 4.12 and Lemma 4.15, again with $F(\langle f \rangle) = f(y)A(\langle f \rangle)$, (and $\sum \hat{A}^2_{\alpha\Delta\{y\}} \leq 1$) to get the bound

$$c_{l,m} \left(e^{-2^{s/2-4}} + 2^{-ms/16} \right) \binom{2^s}{2^{s-1}}^m \tag{15}$$

for that part of the sum. Summing over T with $|T| < m/4$, Lemma 4.16 gives, together with the trivial estimate

$$\sum_{B_1, B_2 \ldots B_m} E\left(\prod_{i=1}^{m} \prod_{x \in \alpha_i} B_i(\vec{f}(x)) \right) \leq \binom{2^s}{2^{s-1}}^m,$$

the bound

$$c_{l,m} 2^{-ms\epsilon/4} \binom{2^s}{2^{s-1}}^m. \tag{16}$$

Combining (15) and (16), Lemma 4.10 follows. ∎

4.5 Concluding Theorem 4.2 and extending it

We have already done all the work to prove Theorem 4.2. If $Y = 0$ and the CNF-test is not consistent with any point in S, by Lemma 4.8, we need

either $Y_1 \geq \frac{1}{3}\binom{2^s}{2^{s-1}}$ or $Y_3 \geq \frac{1}{3}\binom{2^s}{2^{s-1}}$. The sum of the probabilities of these two events is, by Corollary 4.6 and Corollary 4.11, bounded from above by $2^{-(k+2)s}$. By Lemma 4.4 this is sufficient to prove the theorem. ∎

Theorem 4.2 is a powerful theorem as it stands but we require slightly more. In a protocol, establishing that a table describes a correct long code is just testing a syntactic property and what we really care about is to establish that the input for which we have this long code satisfies some properties of interest. In our case we are interested in establishing that the input satisfies the chosen clauses and that we have consistency between different long codes. It turns out that we can get these extra, and essential, properties by adding some extra probes to the table A. These extra probes are not free (in the technical sense of the word) and hence do not cost us anything.

We formalize the extra property that we want to test as $h(x^0) = -1$ (i.e. $h(x^0)$ is true) for some function h. It turns out that it is sufficient to make sure that $A(\langle f \rangle) = A(\langle g \rangle)$ for all queried f and all g with $g \wedge h = f \wedge h$. We now give the extended test.

CNA-Test(s) **with side condition** h

1. Pick, with the uniform distribution, s random functions $f_i : \{-1,1\}^W \mapsto \{-1,1\}$ and ask for $A(\langle f_i \rangle)$.
2. For all Boolean predicates B of s bits ask for $A(\langle B \circ \vec{f} \rangle)$ and check that
$$A(\langle B \circ \vec{f} \rangle) = B(\vec{A}(\langle \vec{f} \rangle)).$$
3. For any queried function f ask for $A(\langle g \rangle)$ for any g such that $f \wedge h = g \wedge h$. Reject unless $A(\langle f \rangle) = A(\langle g \rangle)$ for all such g.

We now establish that Theorem 4.2 remains true even in the presence of side conditions.

Theorem 4.17 *For any $\epsilon > 0$ and integer k, for $s \geq C_{\epsilon,k}$ and $w \geq D_{\epsilon,k,s}$ the following is true. For any $A : \{-1,1\}^{2^w} \mapsto \{-1,1\}$ there is a set S containing at most $2^{\epsilon s}$ points in $\{-1,1\}^w$ such that, for any h, when the CNA-test(s) with side condition h, is performed, except with probability 2^{-ks}, the test either rejects or the outcome is consistent with being a point evaluation at an element $x \in S$ with the additional property that $h(x)$ is true.*

The probability is taken over the random choices of the verifier performing the test, i.e. over the choice of random functions f_i.

We stress that the set S is independent of the side condition h.

Proof: The proof only requires a rather simple argument on top of the proof of Theorem 4.2. Again define the set S by (2) and define

$$A'(\langle f \rangle) = 2^{-H} \sum_{g \mid g \wedge h = f \wedge h} A(\langle g \rangle) \tag{17}$$

where H is the number of points satisfying $h(x) = 1$ and thus 2^H is the number of functions g appearing in the sum (17). Thus A' maps into $[-1, 1]$ and the verifier rejects whenever it gets a value not of absolute value 1. The CNA-test(s) with side condition h can be viewed as querying the function A'[5]. Thus we can repeat the proof of Theorem 4.2 and conclude that the outcome of is, except with probability 2^{-ks}, either that the test fails or the outcome is consistent with a point evaluation at a point $y \in S'$ where

$$S' = \{x \mid \exists \alpha, \alpha \ni x \text{ such that } |\alpha| \leq l \wedge \hat{A'}_\alpha^2 \geq l 2^{-\epsilon s}\}. \tag{18}$$

From Lemma 4.18 below it follows that $S' \subseteq S \cap \{x \mid h(x) = -1\}$ and this completes the proof of Theorem 4.17. ∎

Lemma 4.18 *The Fourier coefficients of A' are given by $\hat{A}'_\alpha = \hat{A}_\alpha$ if all $x \in \alpha$ satisfy $h(x) = -1$ and $\hat{A}'_\alpha = 0$ otherwise.*

Proof: Using (17), the definition of the Fourier transform and the Fourier inversion formula we have

$$\begin{aligned}
\hat{A}'_{\alpha'} &= 2^{-2^w} \sum_f A'(\langle f \rangle) \prod_{x' \in \alpha'} f(x') \\
&= 2^{-(2^w + H)} \sum_f \sum_{g \mid g \wedge h = f \wedge h} A(\langle g \rangle) \prod_{x' \in \alpha'} f(x') \\
&= 2^{-(2^w + H)} \sum_\alpha \sum_f \sum_{g \mid g \wedge h = f \wedge h} \hat{A}_\alpha \left(\prod_{x \in \alpha} g(x) \right) \left(\prod_{x' \in \alpha'} f(x') \right). \qquad (19)
\end{aligned}$$

Now suppose we have an $x^0 \in \alpha$ with $h(x^0) = 1$. Consider a pairing of the functions g and g' where $g'(x^0) = -g(x^0)$ while $g'(x) = g(x)$ for all $x \neq x^0$.

[5]We have to extend the CNA-test by allowing the values of the function A be in $[-1, 1]$ with the understanding that the test rejects whenever it sees a value which does not have absolute value 1. This changes nothing since being consistent with a point evaluation at the point y is still equivalent to $Y = 0$.

Then either both g and g' belong or both do not belong to the sum (19) and hence their contributions cancel each other and

$$\sum_{g \mid g \wedge h = f \wedge h} \left(\prod_{x \in \alpha} g(x) \right) = 0.$$

We can thus drop the terms with α containing an x such that $h(x) = 1$. If, on the hand $h(x) = -1$ for each $x \in \alpha$ then, since $g(x) = f(x)$ for all such x's

$$\sum_{g \mid g \wedge h = f \wedge h} \prod_{x \in \alpha} g(x) = 2^H \prod_{x \in \alpha} f(x)$$

and the sum (19) reduces to

$$2^{-2^w} \sum_{\alpha} \sum_{f} \hat{A}_\alpha \prod_{x' \in \alpha \Delta \alpha'} f(x')$$

where we only sum over α's with $h(x) = -1$ for each $x \in \alpha$. Now using the fact that

$$2^{-2^w} \sum_{f} \prod_{x' \in \alpha \Delta \alpha'} f(x') = 1$$

if $\alpha' = \alpha$ and 0 otherwise the lemma follows. ∎

5 Main theorem

We want to prove

Theorem 5.1 *For any $\delta > 0$ there is a PCP for NP which uses logarithmic randomness and δ amortized free bits.*

By Theorems 2.8 and 2.9 we have two immediate corollaries.

Theorem 5.2 *For any $\epsilon > 0$, unless $NP = ZPP$, there is no polynomial time algorithm that approximates Max-Clique within a factor $n^{1-\epsilon}$.*

Theorem 5.3 *For any $\epsilon > 0$, unless $NP = P$, there is no polynomial time algorithm that approximates Max-Clique within a factor $n^{1/2-\epsilon}$.*

Proof: (Of Theorem 5.1) We can clearly assume $\delta \leq 1/4$. The PCP follows closely the simple test discussed in Section 3. The modifications needed are that we choose many functions on U and we use CNA-Test(s) with appropriate side conditions to test the supposed long codes on the sets W_i. We call it the FAF-test as in Few Amortized Free bits. The test is applied to a 3-CNF-formula φ, as given by Theorem 2.13, which has exactly 3 variables in each clause and such that each variable appears 5 times. It tries to distinguish the two cases when P_1 and P_2 can convince the verifier in the u-parallel two-prover game with probability 1 and the case when they can only convince the verifier with probability c^u. The formula φ is given by the clauses $(C_i)_{i=1}^m$ and has n variables. The written proof consists of, for each set T of size at most $3u$ a table A_T of size $2^{2^{|T|}}$ which for a correct proof of a satisfiable φ is the long code of the restriction to T of a fixed satisfying assignment. The value of the constant u is specified below.

FAF-Test(δ)

1. Setting of parameter.
   - Set $\ell = \lceil \delta^{-1} \rceil$.
   - Set $k = 40\ell^2$.
   - Set s sufficiently large compared to ℓ and k. In particular $s > C_{1/2,k}$ where $C_{1/2,k}$ is the constant of Theorem 4.17 and $s > s_\ell$ where s_ℓ is the constant from Lemma 5.5 below.
   - Set u sufficiently large compared to ℓ, k and s. In particular we need Theorem 4.17 to be true with $w = 3u$ (so $w \geq D_{1/2,k,s}$), and we also need $c^u < 2^{-30\ell^2 s}$ where c is the constant from Theorem 3.2.

2. Choose U by choosing u variables with the uniform distribution. For $i = 1, 2, \ldots, 10\ell$, choose a set W_i by, for each variable x_{i_k} in U, picking, with uniform probability, a random clause $C_{j_k^i}$ that contains x_{i_k} and letting W_i be the set of all variables in the clauses. The constructions of the different W_i are done independently.

3. Choose $10\ell s$ random functions $g_j : \{-1,1\}^U \mapsto \{-1,1\}$, $j = 1, 2, \ldots, 10\ell s$, with uniform distribution, and read $(A_U(\langle g_j \rangle))_{j=1}^{10\ell s}$.

4. Apply the CNA-test(s) with side conditions, to the supposed long code A_{W_i} on W_i for $i = 1, 2, \ldots, 10\ell$. The side conditions are given by

$(g_j(x) = A_U(\langle g_j \rangle))_{j=1}^{10\ell s}$ and that $(C_{j_k^i})_{k=1}^{u}$ are all true. The functions g_j are extended to W_i by ignoring all coordinates not in U.

5. Accept iff all tests accept.

Note that for a correct NP-statement we can easily construct a correct proof, i.e. fix one satisfying assignment x and for each set U and W simply write down the long code of x restricted to that set. It is not difficult to see that in this case the verifier always accepts. Note also that the amount of randomness used by the verifier is logarithmic. Most of the randomness is used to choose the set U and after this only a constant number of random bits is needed to choose each W_i and the g_j. We next turn to the free bit complexity.

Lemma 5.4 *FAF-Test(δ) uses $20\ell s$ free bits.*

Proof: Reading $(A_U(\langle g_j \rangle))_{j=1}^{10\ell s}$ constitutes $10\ell s$ free bits and the free bits in the 10ℓ applications of the CNA-Test(s) total another $10\ell s$ free bits. ∎

The other crucial (and hard) part of the proof of Theorem 5.1 is the soundness and it is given below.

Lemma 5.5 *For any integer ℓ there is a constant s_ℓ such that for $s > s_\ell$, if FAF-test(δ) accepts with probability at least $2^{-20\ell^2 s}$ then there are strategies for P_1 and P_2 in the u-parallel one-round two-prover protocol that makes the verifier accept with probability at least $2^{-30\ell^2 s}$.*

We first claim that Theorem 5.1 follows by Lemma 5.4 and Lemma 5.5. Note first that, by our choice of u, the soundness error of the u-parallel one-round two-prover protocol is smaller than $2^{-30\ell^2 s}$ and thus the conclusion of Lemma 5.5 implies that φ is satisfiable. Thus it follows that the soundness error of the FAF-test(δ) is at most $2^{-20\ell^2 s}$. Using Lemma 5.4, the amortized number of free bits is at most $20\ell s/(20\ell^2 s) = 1/\ell \leq \delta$ and Theorem 5.1 follows. ∎

Proof: (Of Lemma 5.5) For each W_i we have, by Theorem 4.17 (with $\epsilon = 1/2$ and $k = 40\ell^2$), a set S_{W_i} of assignments on W_i of cardinality at most $2^{s/2}$ such that if the test does not fail then, except with probability $2^{-40\ell^2 s}$, the outcome of the test is consistent with a point evaluation at some $y \in S_{W_i}$ that also satisfies the side conditions i.e. it satisfies the chosen clauses and takes the correct value under the functions $(g_j)_{j=1}^{10\ell s}$. The latter conditions, forcing $A_{W_i}(\langle g_j \rangle) = A_U(\langle g_j \rangle)$ play a key role below. We define a set which measures the amount of coordination among the different S_{W_i}.

Definition 5.6 *For a set U let common(U) be the set*

$$\{x \mid Pr_W[\exists y \in S_W \wedge y|_U = x] \geq 2^{-20\ell s}\}$$

where W is chosen with the probability distribution that is used to pick W_i in the FAF-test(δ). Here we only consider $y \in S_W$ that satisfies the clauses used to construct W.

Before we continue, let us give some intuition. We want to check consistency between the long code on U and the long code on W. As proved in [7], two-way consistency requires one amortized free bit. To get around this lower bound we use here one-many consistency. We have many tables (e.g. the long codes on W_i for $i = 1, 2, \ldots 10\ell$) which should be consistent with some other (i.e. the long code on U). We can now read a few bits ($10\ell s$) in the long code on U and check it against the many tables. If there were no consistency among the many tables, say that they were random long codes, the probability of acceptance would be around $(2^{-10\ell s})^{10\ell} = 2^{-100\ell^2 s}$ which is smaller than we are aiming to prove. Thus, to have a good probability of success, the long codes on W_i should have some common properties and this is what we use.

Note that the long code on U merely produces a reference point and hence plays no essential role in the argument. This is natural since changing it to the long code of a random assignment changes the acceptance probability by at most a factor $2^{-10\ell s}$. Thus, the important parameter is not the behavior on U but rather the properties the long codes on W_i have in common. This reflects the central role of *common*(U). After this detour let us return to the main path.

Lemma 5.7 *Suppose common(U) is empty, then the probability that FAF-test(δ) accepts given that U is chosen is bounded by $c_\ell 2^{-40\ell^2 s}$.*

Proof: Remember that if the test accepts then, except with probability $10\ell 2^{-40\ell^2 s}$, it is compatible with some $y^i \in S_{W_i}$ for all $i = 1, 2, \ldots, 10\ell$. Thus we analyze the probability that this happens and the test accepts given that *common*(U) is empty. In this case there must be a collection $\left(y^i \in S_{W_i}\right)_{i=1}^{10\ell}$ such that for some b_j's

$$g_j(y^i) = b_j \text{ for all } 1 \leq i \leq 10\ell \text{ and } 1 \leq j \leq 10\ell s. \tag{20}$$

We denote the vector of y^i's by $\vec{y}$ and we always assume that $y^i \in S_{W_i}$. We analyze the probability of (20) by first fixing $\vec{y}$ and then analyzing the probability that this particular $\vec{y}$ satisfies (20) for a random choice of functions

g_j. Consider the set

$$K(\vec{y}) \triangleq (y^i|_U)_{i=1}^{10\ell}$$

of projections onto U, keeping only one copy of each assignment if the several y^i give the same projection. The condition (20) says exactly that every chosen g_j is constant on the set $K(\vec{y})$. The probability that this happens for an individual j is $2^{-(|K(\vec{y})|-1)}$ and thus the probability that (20) is true for this $\vec{y}$ is $2^{-10\ell s(|K(\vec{y})|-1)}$. Thus the key is to analyze the size of $K(\vec{y})$.

Lemma 5.8 *Suppose common(U) is empty. Then*

$$Prob_{W_1,W_2\ldots W_{10\ell}}[\min_{\vec{y}\in S_{W_1}\times,\ldots S_{W_{10\ell}}} |K(\vec{y})| \leq 5\ell] \leq c_\ell 2^{-95\ell^2 s}.$$

Proof: Let us first analyze the probability that $\pi_U(S_{W_i})$ intersects $\cup_{j=1}^{i-1}\pi_U(S_{W_j})$. The latter contains at most $10\ell 2^{s/2}$ elements (since $|S_{W_j}| \leq 2^{s/2}$). The probability of any single element occurring in $\pi_U(S_{W_i})$ is bounded by $2^{-20\ell s}$ (by the definition of *common*(U) and using that this set is empty) and hence the probability of a nonempty intersection is bounded by $10\ell 2^{-19\ell s}$. For there to exist a $\vec{y} \in S_{W_1}\times,\ldots S_{W_{10\ell}}$ giving at most 5ℓ different projections it must be the case that for 5ℓ different i, $\pi_U(S_{W_i})$ (i.e., the set of possible projections of the ith element in $\vec{y}$) has a nonempty intersection with $\cup_{j=1}^{i-1}\pi_U(S_{W_j})$ (i.e., the possible projections of prior elements). The probability of this event is bounded by $\binom{10\ell}{5\ell} \cdot (c_\ell 2^{-19\ell s})^{5\ell} = c_\ell 2^{-95\ell^2 s}$ and the lemma follows. ∎

Let us return to the proof of Lemma 5.7. There are at most $(2^{s/2})^{10\ell}$ ways of picking $\vec{y} \in S_{W_1}\times,\ldots S_{W_{10\ell}}$. Assuming that $|K(\vec{y})| \geq 5\ell + 1$, the probability that an individual choice is compatible with the functions g_j is bounded, by the reasoning above, by $2^{-10\ell s(5\ell+1-1)} = 2^{-50\ell^2 s}$. Thus the probability that any $\vec{y}$ is compatible with the choice of the g_j's is bounded by $2^{5\ell s}2^{-50\ell^2 s} \leq 2^{-45\ell^2 s}$. Since, by Lemma 5.8, the probability that any $\vec{y}$ satisfies $|K(\vec{y})| \leq 5\ell$ is bounded by $c_\ell 2^{-95\ell^2 s}$, we just add the two probabilities and Lemma 5.7 follows. ∎

To wrap up the proof of Lemma 5.5, let us now define a strategy for the provers in the u-parallel two-prover game.

P_2 simply answers with any element in *common*(U), while P_1 answers with a random element of S_W. If either of these sets is empty the corresponding prover gives up. Assuming FAF-test(δ) accepts with probability $2^{-20\ell^2 s}$ then, in view of Lemma 5.7, *common*(U) is nonempty with probability at least $2^{-20\ell^2 s} - c_\ell 2^{-40\ell^2 s}$ and let us analyze the probability that the verifier accepts when this is the case. Suppose P_2 answers with x^U then, by

the definition of $common(U)$, the probability that S_W contains an element y such that $y|_U = x^U$ and such that y satisfies the clauses C_{j_k} is at least $2^{-20\ell s}$. The probability that P_1 answers with such an element, given that it exists, is at least $|S_W|^{-1} \geq 2^{-s/2}$. Thus in the two-prover game we have an overall success probability that is at least

$$(2^{-20\ell^2 s} - c_\ell 2^{-40\ell^2 s}) \cdot 2^{-20\ell s} \cdot 2^{-s/2} > 2^{-30\ell^2 s}$$

provided $s > s_\ell$ and the proof is complete. ∎

Acknowledgment. The input from Oded Goldreich has benefited this paper in several ways and in particular it has greatly improved the presentation of the paper. I am also grateful to Gunnar Andersson, Mihir Bellare, Muli Safra, and Madhu Sudan for comments on and discussions about this paper.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof verification and the hardness of approximation problems. Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science, Pittsburgh, 1992, pp 14-23. To appear in Journal of the ACM.

[2] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. Journal of the ACM, Vol 45, pp 70-122.

[3] L. Babai. Trading group theory for randomness. Proceedings of the 17th Annual ACM Symposium on Theory of Computation, Providence, 1985, pp 420-429.

[4] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polynomial time. Proceedings of the 23rd Annual ACM Symposium on Theory of Computation, New Orleans, 1991, pp 21-31.

[5] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity, Vol 1, 1991, pp 3-40.

[6] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity Testing in Characteristic Two. IEEE Transactions on Information Theory, Vol 42, No 6, November 1996, pp 1781-1796.

[7] M. Bellare, O. Goldreich and M. Sudan. Free Bits, PCPs and Non-Approximability – Towards tight Results. SIAM J. on Computing, 1998, Vol 27, No 3, pp 804-915.

[8] M. Bellare, S. Goldwasser, C. Lund and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. Proceedings of the 25th Annual ACM Symposium on Theory of Computation, San Diego, 1993, pp 294-304.

[9] M. Bellare and M. Sudan. Improved non-approximability results. Proceedings of the 26th Annual ACM Symposium on Theory of Computation, Montreal, 1994, pp 184-193.

[10] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multiprover interactive proofs. How to remove intractability. Proceedings of the 20th Annual ACM Symposium on Theory of Computation, Chicago, 1988, pp 113-131.

[11] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. Information and Computation, Vol 96, pp 77-94.

[12] R. Boppana and M. Haldórsson. Approximating maximum independent sets by excluding subgraphs. BIT, Vol. 32, No. 2, 1992, pp 180-196.

[13] J. Bourgain. Walsh subspaces of l_p product spaces. In Seminaire D'Analyse Fonctionnelle. Ecole Polytechnique, Centre De Mathematiques, 1979-80, pp 4.1-4.9.

[14] S. A. Cook. The complexity of Theorem Proving Procedure. Proceeding 3rd ACM Symposium on Theory of Computing, 1971, pp 151-158.

[15] U. Feige. A threshold of $\ln n$ for approximating set cover. Proceedings of the 28th Annual ACM Symposium on Theory of Computation, Philadelphia 1996, pp 314-318. Accepted for publication in Journal of the ACM.

[16] U. Feige and M. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. 3rd Israeli Symposium on the theory of Computing and Systems, 1995, pp 182-189.

[17] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. Journal of the ACM, Vol, 43:2, pp 268-292.

[18] U. Feige and J. Kilian. Two prover protocols – Low error at affordable rates. Proceedings of the 26th Annual ACM Symposium on Theory of Computation, Montreal 1994, pp 172-183.

[19] U. Feige and J. Kilian. Zero-Knowledge and the chromatic number. Proceedings of the 11th Annual IEEE conference on Computational Complexity, Philadelphia 1996, pp 278-287.

[20] L. Fortnow, J. Rompel, and M. Sipser. On the power of Multi-Prover Interactive Protocols. Proceedings 3rd IEEE Symposium on Structure in Complexity Theory, pp 156-161, 1988.

[21] M.R. Garey and D.S. Johnson. Computers and intractability; a guide to the theory of NP-completeness. W.H. FREEMAN, 1979.

[22] M. Goemans and D. Williamson. .878-approximation algorithms for Max-Cut and Max-2-SAT. Proceedings of 26th Annual ACM Symposium on Theory of Computation, Montreal, 1994, pp 422-431.

[23] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. SIAM Journal on Computing, Vol 18, pages 186-208, 1989.

[24] J. Håstad Testing of the long code and hardness for clique. Proceedings of the 28th Annual ACM Symposium on Theory of Computation, Philadelphia 1996, pp 11-19.

[25] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science, Burlington, 1996, pp 627-636.

[26] J. Håstad. Some Optimal In-approximability Results. Proceedings 29th Annual ACM Symposium on Theory of Computation, 1997, pp 1-10.

[27] D.S. Johnsson Approximation algorithms for combinatorial problems. J. Computer and System Sciences, 1974, Vol 9, pp 256-278.

[28] C. Lund, L. Fortnow, H. Karloff and N. Nisan. Algebraic methods for interactive proof systems. Journal of the ACM, Vol 39, No 2, pp 859-868.

[29] R. Motwani and P. Raghavan. Randomized algorithms. Cambridge University Press, 1995.

[30] C. Papadimitriou. Computational complexity. Addison-Wesley, 1994.

[31] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. Journal of Computer and System Sciences, Vol 43, 1991, pp 425-440.

[32] R. Raz. A parallel repetition theorem. SIAM Journal on Computing. Vol 27, 1998, pp 763-803.

[33] A. Shamir. IP=PSPACE. Journal of the ACM, Vol 39, No 2, pp 869-877.

[34] D. Zuckerman. On unapproximable versions of NP-complete problems. SIAM Journal on Computing. Vol 25, 1996, pp 1293-1304.

A Analysis of idealized protocol of Section 3

Set $\ell = d \log p^{-1}$ where d is a constant to be determined.

By assumption A_{W_i} is the long code of an assignment x^{W_i} on W_i. We may assume without loss of generality that x^{W_i} satisfies the picked clauses (i.e., $g_i(x^{W_i}) = 1$), or else the test rejects anyhow. Let us consider the set $X_{U,\vec{W}} = (x^{W_i}|_U)_{i=1}^{\ell}$, where $\vec{W} = (W_1, ..., W_\ell)$. The acceptance condition of the test (i.e., $A_{W_i}(f) = A_U(f)$ for $i = 1, ..., \ell$) implies that f is constant on $X_{U,\vec{W}}$. The probability of this happening, for a random f, is $2/2^{|X_{U,\vec{W}}|}$. Thus the probability, p, that the test accepts is at most

$$2 \cdot E_{U,\vec{W}}[2^{-|X_{U,\vec{W}}|}] \tag{21}$$

Now for a fixed U let p_U denote the value

$$\max_x Pr_W[x^W|_U = x] \tag{22}$$

where W is a random set constructed as in the simple test (i.e., a random extension of U to clauses). We claim that $E_U[p_U]$ is a lower bound on the acceptance probability of the two-prover proof system. This is shown by letting P_1 answer according to the x^{W_i}'s, and P_2 answer with the x which gives the maximum in (22). Thus, all that remains is to lower bound $E_U[p_U]$ as a function of p. Towards this end let us analyze the probability of acceptance in the simple test as a function of p_U.

It is natural to study the size of $X_{U,\vec{W}}$, and we analyze this by fixing U and picking the sets W_i at random one by one, investigating how the size of $X_{U,\vec{W}}$ grows. We define

$$X^i_{U,\vec{W}} = (x^{W_j}|_U)_{j=1}^i$$

i.e. the part of $X_{U,\vec{W}}$ obtained from the first i sets W_i. Clearly $|X^{i+1}_{U,\vec{W}}| = |X^i_{U,\vec{W}}| + 1$ unless $X^i_{U,\vec{W}}$ already contains $x^{W_{i+1}}|_U$ in which case the two sets are equal. Since the probability that $x^{W_{i+1}}|_U$ take any fixed value is bounded from above by p_U, the probability that $X^{i+1}_{U,\vec{W}} = X^i_{U,\vec{W}}$ is at most $|X^i_{U,\vec{W}}| \cdot p_U$, which is smaller than $1/2$ when $|X^i_{U,\vec{W}}| \leq \frac{1}{2p_U}$. We claim that

$$Pr_{\vec{W}}[|X_{U,\vec{W}}| \leq \min(\frac{\ell}{4}, \frac{1}{2p_U})] \leq 2^{-c\ell}$$

for some absolute constant c. This follows since for this event not to be true, events of the form $|X^{i+1}_{U,\vec{W}}| = |X^i_{U,\vec{W}}|$, each occurring with probability at most $1/2$, must happen at least $3\ell/4$ times in ℓ tries.

Thus, $E_{\vec{W}}[2^{-|X_{U,\vec{W}}|}] \leq 2^{-c\ell} + \max(2^{-\ell/4}, 2^{-1/2p_U})$. Now, using (21), we have

$$\frac{p}{2} \leq E_{\vec{W}}[2^{-|X_{U,\vec{W}}|}] < 2^{-c\ell} + 2^{-\ell/4} + E_U[2^{-1/2p_U}]$$

and setting the constant d (in the definition of ℓ) sufficiently large, we conclude $E_U[2^{-1/2p_U}] > p/2.5$. Finally, using $x > 2^{-1/x}$ for all $x > 0$, we have

$$E_U[p_U] > \frac{1}{2} \cdot E_U[2^{-1/2p_U}] > p/5,$$

and the proof is complete.