Probabilistically Checkable Proofs and their Consequences for Approximation Algorithms

S. Hougardy H.J. Prömel A. Steger

Institut für Informatik, Humboldt-Universität zu Berlin, Germany

Abstract. The aim of this paper is to present a self-contained proof of the spectacular recent achievement that $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$. We include, as consequences, results concerning non-approximability of the clique number, as well as of the chromatic number of graphs, and of MAX-SNP hard problems.

Contents

This paper appeared in *Trends in Discrete Mathematics* (W. Deuber, H.J. Prömel, B. Voigt, eds.), Topics in Discrete Mathematics **9**, North Holland, 1995, pp. 175–223.

1 Introduction

On April 7, 1992, The New York Times published an article entitled "New Short Cut Found for Long Math Proofs". The aim of this article was to popularize a new characterization of the class $\mathcal{NP}$, obtained by S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, which can formally be phrased as $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$. This is indeed an amazing result — with far-reaching consequences in discrete mathematics.

In order to roughly explain this result, let us first recall the definition of the class $\mathcal{NP}$. A language (decision problem) is in the class $\mathcal{NP}$ if, for every input x which belongs to this language, there exists a membership proof, say π_x, which can be checked in polynomial time by some Turing machine. Typical decision problems in $\mathcal{NP}$ are the satisfiability problem, the Hamiltonian cycle problem and the 3-coloring problem. Membership proofs for problems in $\mathcal{NP}$ are usually concise and simple. They just consist, for example, of a satisfying assignment, a Hamiltonian cycle, or a 3-coloring, respectively. Surely, to distinguish a proper 3-coloring from a coloring that is proper on all but one of the vertices, one really has to read the color of *every* vertex, i.e., the whole proof!

This is not the case with *probabilistically checkable proofs*. Here we give just an intuitive idea of this notion; a precise definition is given in the next section. Probabilistically checkable proofs are inspected by *verifiers* (polynomial time Turing machines) which proceed as follows. After reading the input x and a string τ of random bits, they decide which bits (positions) of the proof they want to read. Subsequently, they either accept the input x or reject it — only on the knowledge of the (few) queried bits! A language (a decision problem) is said to have a probabilistically checkable proof if, for all x in the language, there exists a proof π_x which the verifier accepts for *all* random strings τ, while, for all x not in the language, the verifier rejects *all* proofs for a majority of the random strings.

As we will see, it is fairly easy to construct a probabilistically checkable proof for problems in $\mathcal{NP}$ which can be checked by reading only a constant number of bits — if we allow the verifier to use polynomially many random bits. But, as a consequence, these proofs may be of exponential length. Highly non-trivial and surprising, however, is the fact that every problem in $\mathcal{NP}$ has even a proof of polynomial length with the same property. More precisely, every input x of length n admits a proof of polynomial length which can be checked probabilistically by reading only a constant number of bits from it, using random strings τ of length at most $\mathcal{O}(\log n)$. This is, roughly, the essence of the "$\mathcal{NP} = \mathcal{PCP}(\log n, 1)$" result.

Just as amazing as this result itself are the consequences in the seemingly unrelated area of approximation algorithms — resolving several long-standing open problems. Here, we will just mention two of them. First, consider the problem of finding a maximum clique in a given graph. The corresponding decision problem, namely to decide, for a given graph G and a number k, whether the clique number of G is at least k, is one of the classical $\mathcal{NP}$-complete problems. Until the $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ result was proved, no non-trivial lower bound for the approximation guarantee of a polynomial time algorithm for the clique-number in a given graph was known. As a consequence of their characterization result for the class $\mathcal{NP}$, Arora, Lund, Motwani, Sudan and Szegedy deduced from a result of Feige et al. (1991) that there exists a constant $\epsilon > 0$ such that no polynomial time approximation algorithm for the clique number of a graph on n vertices can be guaranteed to come within a factor of n^ϵ of the right answer - unless $\mathcal{P} = \mathcal{NP}$.

Second, we take a look at the problem of finding the chromatic number of a given graph. This problem is also known to be $\mathcal{NP}$-hard. Even more, an old result of of GAREY and JOHNSON (1976) states that for any $\epsilon > 0$, no polynomial time algorithm can approximate the chromatic number within a factor of $2 - \epsilon$, unless $\mathcal{P} = \mathcal{NP}$. Now, using an appropriate transformation from the clique-problem, LUND and YANNAKAKIS (1993) have been able to show that the chromatic number of a graph is just as hard to approximate as the clique number. More precisely, there exists a constant $\epsilon > 0$ such that no polynomial time approximation algorithm for the chromatic number of a graph on n vertices can have a performance guarantee that is n^ϵ — unless $\mathcal{P} = \mathcal{NP}$. For constant chromatic number, KHANNA, LINIAL and SAFRA (1993) proved that it is even $\mathcal{NP}$-hard to color a 3-colorable graph with four colors.

The $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ result, but even more its consequences on approximation algorithms, have astonished many people working in discrete mathematics and have had considerable impact on their work. The methods for proving this result have been developed in computer science during the last few years. The idea of writing this paper was to collect and explain the ingredients (some of them not being easily accessible), to present a self-contained proof of the $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ result, and to explore some of its applications. The paper should be, we hope, a readable guide to these results for people who are not experts in this field, but rather view this exciting development more from the angle of a discrete mathematician.

2 Probabilistically checkable proofs

A *verifier* V is a polynomial-time Turing machine with access to an input x and a string τ of random bits. Furthermore the verifier has access to a proof π via an oracle, which takes as input a position of the proof the verifier wants to query and outputs the corresponding bit of the proof π (cf. Figure 1).

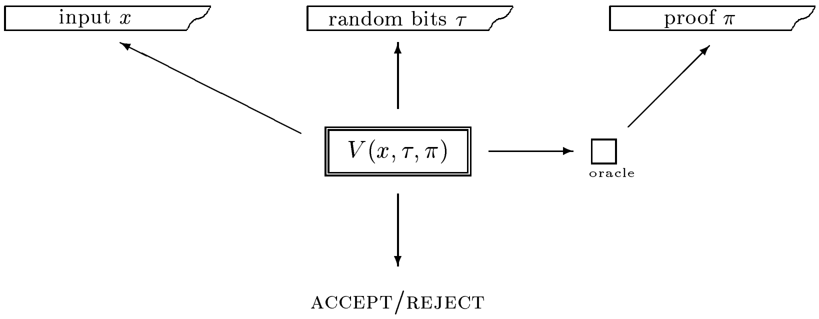

Figure 1: A verifier for probabilistically checkable proofs

The result of V's computation, usually denoted by $V(x, \tau, \pi)$, is either ACCEPT or REJECT. For clarity let us explicitly state, that we always assume verifiers to be non-adaptive, that is we assume that the bits a verifier queries solely depend on the input x and the random string τ, but not on the outcome of any previously queried bits.

An $(r(n), q(n))$-*restricted verifier* is a verifier that for inputs x of length n uses at most $\hat{r}(n)$ random bits and queries at most $\hat{q}(n)$ bits from π, where $\hat{r}(n)$ and $\hat{q}(n)$ are integral functions such that $\hat{r}(n) = \mathcal{O}(r(n))$ and $\hat{q}(n) = \mathcal{O}(q(n))$.

Definition 2.1 *A language L is in $\mathcal{PCP}(r(n), q(n))$ iff there exists an $(r(n), q(n))$-restricted verifier V such that:*

(i) For all $x \in L$ there exists a proof π_x such that

$$Prob_\tau[V(x, \tau, \pi_x) = \text{ACCEPT}] \;=\; 1,$$

(ii) while for all $x \notin L$ every proof π satisfies

$$Prob_\tau[V(x, \tau, \pi) = \text{ACCEPT}] \;<\; \tfrac{1}{4}.$$

Here the notation $\text{Prob}_\tau[\ldots]$ means that the probability is taken over all random strings the verifier may read (that is, over all 0-1 strings of length $\hat{r}(|x|)$), where every string is equally likely. In other words, the probability is computed with respect to the uniform distribution on $\{0,1\}^{\hat{r}(|x|)}$.

Note: In slight abuse of notation we will allow the functions $r(n)$ and $q(n)$ to be of the form $poly(n)$, $poly(\log(n))$ and so on. (Here we assume that, for example, every polynomial function $p(n)$ satisfies $p(n) = \mathcal{O}(poly(n))$.)

The reader is invited to observe that the constant 1/4 in Definition 2.1 may be replaced by any constant α between 0 and 1.

With these definitions in hand we are now able to interpret the new characterization: every language in $\mathcal{NP}$ has a membership proof π which can be checked probabilistically by using $\mathcal{O}(\log n)$ random bits and querying only $\mathcal{O}(1)$ bits of the proof.

Theorem 2.2 [Arora, Lund, Motwani, Sudan, Szegedy, 1992]

$$\mathcal{NP} = \mathcal{PCP}(\log n, 1).$$

Note that one inclusion of Theorem 2.2 is trivial. Namely, the inclusion $\mathcal{PCP}(\log n, 1) \subseteq \mathcal{NP}$ follows immediately from the fact that there exist only polynomially many different random strings of length $\mathcal{O}(\log n)$. The other inclusion is proven in Section 4.

3 Consequences in combinatorial optimization

Our main motivation for being interested in the $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ result are its startling consequences in combinatorial optimization. In this section we will state the three most important consequences: the non-approximability of the clique number, of the chromatic number and of MAX-SNP-hard problems.

The quality of an approximation algorithm is measured by its *performance guarantee* which is defined as follows. Let A be an approximation algorithm and I be an instance for the algorithm. By $OPT(I)$ we denote the value of an optimal solution and by $A(I)$ the value of the solution found by the approximation algorithm A. Then the *performance ratio* of A on input I is defined as the quotient $A(I)/OPT(I)$ resp. $OPT(I)/A(I)$ whatever of the two values is larger. Now the performance guarantee of the algorithm A is the supremum of all performance ratios for instances I with $OPT(I) > n_0$ for some integer n_0.

Until recently the three above mentioned optimization problems shared the same status: the best known polynomial time approximation algorithms — even though they are quite intricate — had very poor approximation ratios compared to the best known lower bounds. For example the best known polynomial time approximation algorithm for the chromatic number of a graph on n vertices has a performance guarantee of $\mathcal{O}(n(\log\log n)^2/\log^3 n)$ [21]. On the other hand the best known lower bound is $2-\epsilon$, i.e., no polynomial time algorithm for approximating the chromatic number can have a performance guarantee better than $2-\epsilon$, unless $\mathcal{P}=\mathcal{NP}$ [17].

The usual way to prove the non-approximability of an optimization problem P is to reduce the instances of some $\mathcal{NP}$-complete language L to instances of the problem P with a large gap in their cost functions. That is, for elements of L one has to construct instances of P that have a value of at least, say, c in the cost function of P, while for all other instances the value of the cost function is at most some constant fraction of c. The difficulty in constructing such a transformation is that it is possible that two strings x and y differ in only one bit even though x is an element of L and y is not. A main feature of the $\mathcal{NP}=\mathcal{PCP}(\log n, 1)$ result is that it provides a robust way to compute instances with the desired gap.

3.1 Non-approximability of the clique number

A *clique* of a graph G is a set of pairwise adjacent vertices in G. The *clique number* of G is defined as the size of a largest clique contained in G and is denoted by $\omega(G)$. The problem CLIQUE is to decide for a graph G and a number k whether the clique number of G is at least k. This problem is one of the classical $\mathcal{NP}$-complete problems [24].

The $\mathcal{NP}$-completeness of CLIQUE leads naturally to the question whether the underlying *optimization* problem – finding a maximum size clique – has at least a "good" polynomial time *approximation* algorithm. For the clique number, the best known performance guarantee of a polynomial time approximation algorithm is achieved by an algorithm due to Boppana and Halldórsson. It has a performance guarantee of $\mathcal{O}(n/\log^2 n)$ [13].

Before the $\mathcal{NP}=\mathcal{PCP}(\log n, 1)$ result (Theorem 2.2) was proved, no non-trivial lower bound for the performance guarantee of a polynomial time approximation algorithm for the clique number of a graph was known. The only result in this direction, due to Garey and Johnson, is that the existence of a polynomial time approximation algorithm for the clique number of a graph with a constant performance guarantee implies the existence of a polynomial time approximation scheme (PTAS for short; a PTAS is a family of algorithms, one for each $\epsilon>0$, which are polynomial in time, and achieve an approximation ratio of $1+\epsilon$) for the clique number [18].

As a first step we will show in Section 3.1.1 how the $\mathcal{NP}=\mathcal{PCP}(\log n, 1)$ result together with a result of Feige, Goldwasser, Lovász, Safra and Szegedy [15] implies that, unless $\mathcal{P}=\mathcal{NP}$, no polynomial time approximation algorithm for the clique number problem can achieve a constant performance guarantee. Further results of Feige, Goldwasser, Lovász, Safra and Szegedy, using random walk techniques of Ajtai, Komlós and Szemerédi [1] resp. Impagliazzo and Zuckerman [22], give in combination with the $\mathcal{NP}=\mathcal{PCP}(\log n, 1)$ result a much stronger statement: there exists a constant ϵ such that no polynomial time approximation algorithm for the clique number can have a performance guarantee that is n^ϵ – unless $\mathcal{P}=\mathcal{NP}$. This will be shown in Section 3.1.2.

3.1.1 Non-approximability up to any constant factor

Feige, Goldwasser, Lovász, Safra and Szegedy [15] were the first who used results in the theory of interactive proofs to obtain some non-approximability results for the clique number. They showed that $\mathcal{NP} \subseteq \mathcal{PCP}(\log n \log\log n, \log n \log\log n)$ and used this result to prove that the clique number of a graph cannot be approximated in polynomial time up to any constant factor unless $\mathcal{NP} = \mathcal{DTIME}(n^{\mathcal{O}(\log\log n)})$. Their proof can immediately be applied to the $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ result to get the following theorem.

Theorem 3.1 *Unless $\mathcal{P} = \mathcal{NP}$, the clique number of a graph cannot be approximated in polynomial time up to a factor of 2.*

Proof. Based on the existence of a $(\log n, 1)$-restricted verifier (Theorem 2.2) for any language in $\mathcal{NP}$ we will construct for a given language $L \in \mathcal{NP}$ and any input x of length n a graph G_x with the following property:

$$\begin{array}{lclcl} x \in L & \Rightarrow & \omega(G_x) & = & f(n) \\ x \notin L & \Rightarrow & \omega(G_x) & < & \frac{1}{4} f(n), \end{array}$$

where the function f will be specified later. Thus a polynomial time algorithm that approximates the clique number of a graph up to a factor of 2 could be used to recognize any language in $\mathcal{NP}$ in polynomial time.

We now describe the construction of the graph G_x. Let V be the $(\log n, 1)$-restricted verifier for the language L and $r(n) = \mathcal{O}(\log n)$ resp. $c(n) = \mathcal{O}(1)$ be the maximum number of random bits resp. query bits used by V on inputs of length n. The vertex set of G_x will consist of all accepting runs of V on input x. Each of these can be described by a tuple $\langle \tau, a_1, a_2, \ldots, a_{c(n)} \rangle$ where τ is the random string of length $r(n)$ used for the computation and a_i is the answer to the ith queried bit of π. The length of the whole tuple is $\mathcal{O}(\log n)$ which implies that the size of G_x is polynomially bounded. To decide whether a given tuple is a vertex of G_x one just has to verify that V accepts the input x with random string τ and answers a_i.

Two different vertices $\langle \tau, a_1, \ldots, a_{c(n)} \rangle$ and $\langle \tilde{\tau}, \tilde{a}_1, \ldots, \tilde{a}_{c(n)} \rangle$ of G_x are adjacent if there exists at least one proof π that is consistent with both tuples, i.e. if there is a position p of the proof string that was queried in both runs then the returned bits, say a_i and $\tilde{a}_j$, have to have the same value. Obviously for any pair of vertices in G_x this can be checked in polynomial time. Observe that G_x is a $2^{r(n)}$ partite graph.

For a fixed proof π any two vertices of G_x that are consistent with π (i.e. the returned bits a_i equal the corresponding bits of π) are adjacent. Thus for all proofs π we have:

$$\begin{array}{rcl} \omega(G_x) & \geq & \text{number of accepting runs of } V \text{ with respect to } \pi \\ & = & 2^{r(n)} \cdot \mathrm{Prob}_\tau[V(x, \tau, \pi) = \text{ACCEPT}] \end{array}$$

On the other hand, if C is a clique in G_x then all vertices in C that query a position p must get the same answer a and thus there exists one proof π that is consistent with *all* vertices of

C. Since this is especially true for a clique of size $\omega(G_x)$ we know that there exists some π_0 such that:

$$\begin{aligned}\omega(G_x) &\leq \text{number of accepting runs of } V \text{ with respect to } \pi_0 \\ &= 2^{r(n)} \cdot \text{Prob}_\tau[V(x,\tau,\pi_0) = \text{ACCEPT}]\end{aligned}$$

Combining the two inequalities we get:

$$\omega(G_x) = 2^{r(n)} \cdot \max_{\pi} \text{ Prob}_\tau[V(x,\tau,\pi) = \text{ACCEPT}].$$

By the definition of a restricted verifier the value of $\max_\pi \text{Prob}_\tau[V(x,\tau,\pi) = \text{ACCEPT}]$ is either 1 or less than $1/4$ depending on whether $x \in L$ or $x \notin L$. This proves the above-stated property of the graph G_x. $\square$

The reader is invited to observe that we did not make use of the fact that the verifier reads only a constant number of bits from π. The same proof works if the verifier would be allowed to read $\mathcal{O}(\log n)$ bits.

As already remarked in Section 2 the constant $1/4$ in the definition of the class $\mathcal{PCP}(.,.)$ may be replaced by any other constant between 0 and 1. If we choose instead of $1/4$ the constant ϵ, then in the above theorem we have to replace the number 2 by $1/\sqrt{\epsilon}$. We therefore get:

Corollary 3.2 *Unless $\mathcal{P} = \mathcal{NP}$, the clique number of a graph cannot be approximated in polynomial time up to any constant factor.*

3.1.2 Non-approximability up to n^ϵ

Let $\mathcal{PCP}_\epsilon(.,.)$ denote the class of languages defined in the same way as $\mathcal{PCP}(.,.)$ except that the constant $1/4$ in Definition 2.1 is replaced by ϵ. As observed already above we have $\mathcal{PCP}_\epsilon(.,.) = \mathcal{PCP}(.,.)$ for any constant ϵ. This follows from the fact that by repeating the run of the restricted verifier a (suitable) constant number of times, the probability of getting a wrong answer can be made arbitrarily (but constantly) small.

If we want to use the proof of Theorem 3.1 to show that the clique number cannot be approximated up to a factor of n^ϵ we would need to prove that $\mathcal{PCP}(\log n, 1)$ equals $\mathcal{PCP}_{n^{-\delta}}(\log n, 1)$ for some δ depending on ϵ. Unfortunately, to show this one had to rerun the restricted verifier k times with k satisfying $(1/2)^k \leq n^{-\delta}$. This means $k \geq \delta \log n$ and thus we would need $\mathcal{O}(\log^2 n)$ random bits and $\mathcal{O}(\log n)$ query bits which gives only $\mathcal{NP} \subseteq \mathcal{PCP}_{n^{-\delta}}(\log^2 n, \log n)$. However, relying on a method of Ajtai, Komlós and Szemerédi [1] resp. Impagliazzo and Zuckerman [22] that makes use of random walks on expanders it can be shown that in fact $\mathcal{O}(\log n)$ random bits are sufficient (i.e., one can show $\mathcal{NP} \subseteq \mathcal{PCP}_{n^{-\delta}}(\log n, \log n)$).

The idea behind this technique is as follows. Instead of using truly random bits one generates *pseudo-random bits* by taking a special d-regular graph (d is a constant) that has a vertex for every possible 0-1 string of length $r(n) = \mathcal{O}(\log n)$ and chooses an arbitrary vertex of this graph as a starting point of a random walk where each of the d edges incident to a vertex is chosen with probability $1/d$. Every c-th step of the random walk (c is a constant) one uses the string that is associated to the just reached vertex as a pseudo-random string. Obviously, in this way only a constant number of random bits are needed to generate a pseudo-random string of length $r(n)$. Thus the total number of random bits used to get $\mathcal{O}(\log n)$ pseudo-random strings of length $r(n)$ is $\mathcal{O}(\log n)$. The following lemma gives the theoretical background for this method.

Lemma 3.3 *Let $\mathcal{G}_\varrho$ be an infinite family of d-regular graphs with the following property: If $G = (V, E)$ is a member of $\mathcal{G}_\varrho$ and A denotes its adjacency matrix multiplied by $1/d$ then all but the largest eigenvalue of A are less than $1 - \varrho$ and positive.*
Then for every subset C of V with $|C| \leq |V|/16$ there exists a constant c such that the probability that a random walk on G of length $k \cdot c$ arrives in every c-th step in a vertex of C is at most 2^{-k}.

Proof. Let $1 = \lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n \geq 0$ be the eigenvalues of A. Note that since G is d-regular the largest eigenvalue of A is 1. Since λ_2 is bounded away from 1 by a constant there exists a constant c such that $(\lambda_2)^c \leq 1/4$. Let p be a vector with a component for every vertex of G containing the probability of being at that vertex. By p_0 we denote the vector whose entries are all $1/|V|$. The vector Ap gives for each vertex of G the probability of being at that vertex after one single step of the random walk. Similarly $A^c p$ gives these probabilities after c steps of the random walk. Let N be the matrix whose entries are all zero except for the diagonal, where N_{ii} is 1 if $i \in C$. Then $NA^c p$ gives the probabilities of being in a vertex of C after c steps of the random walk. Let $|.|$ denote the L_1-norm. Then $|\left(NA^c\right)^k p|$ is the probability that every c-th step the random walk is in a vertex of C. To estimate this value, we first compute an upper bound for the L_2-norm $||.||$ of $NA^c p$:

$$||NA^c p|| \leq \frac{1}{2}||p||. \tag{1}$$

To prove this inequality we first decompose p into the sum of two vectors v and w where v is a multiple of p_0 and w is a vector orthogonal to v. Since v is an eigenvector of A belonging to the eigenvalue 1 we have

$$||NA^c v|| = ||Nv|| = \sqrt{\sum_{N_{ii}=1} v_i^2} \leq \sqrt{\frac{1}{16}\sum v_i^2} = \frac{1}{4}||v||.$$

Similarly we get for the vector w using that it is orthogonal to v:

$$||NA^c w|| \leq ||A^c w|| \leq (\lambda_2)^c ||w|| \leq \frac{1}{4}||w||.$$

Using the triangle inequality and the fact that the sum of the L_2-norms of two orthogonal vectors is bounded by $\sqrt{2}$ times the L_2-norm of the sum of these vectors we get by combining these two inequalities:

$$||NA^c p|| \leq ||NA^c v|| + ||NA^c w|| \leq \frac{1}{4}\left(||v|| + ||w||\right) \leq \frac{\sqrt{2}}{4}||p|| < \frac{1}{2}||p||$$

and thus have proved (1).

Now making use of a well-known inequality between the L_1- and the L_2-norm and applying (1) k times we get:

$$\begin{aligned} |(NA^c)^k p_0| &\leq \sqrt{|V|} \cdot ||(NA^c)^k p_0|| \\ &\leq \sqrt{|V|} \cdot (\tfrac{1}{2})^k \cdot ||p_0|| \\ &= 2^{-k}. \end{aligned}$$

□

The proof of the existence of families $\mathcal{G}_\varrho$ satisfying the requirements of the above lemma is based on the existence of constant degree *expanders*. An (n,d,c)-expander is a d-regular bipartite graph $G = (A, B, E)$ with $|A| = |B| = n$ such that for every set $X \subset A$ with $|X| \leq n/2$ its neighborhood has size at least $|X|(1 + c(1 - |X|/n))$. A *family* of (n_i, d, c)-expanders is an infinite set of graphs G_i that are (n_i, d, c)-expanders with $n_i \to \infty$ and $n_{i+1}/n_i \to 1$.

The explicit construction of families of expanders was first achieved by MARGULIS [29]. He constructed a class of 5-regular expanders as follows:

Let m be an integer and $Z_m := Z/mZ$ be the ring of residues modulo m. The vertex set of the expander is partitioned into sets A_m and B_m both being $Z_m \times Z_m$. Each vertex (x, y) of A_m is connected to the vertices $(x, y), (x + 1, y), (x, y + 1), (x + y, y)$ and $(-y, x)$ of B_m. The proof that the graphs constructed in this way are indeed expanders can be found in MARGULIS [29]. Another construction is given by GABBER and GALIL [16].

If G is a d-regular expander then let A be its adjacency matrix multiplied by $1/d$. ALON [2] has shown that for every family of d-regular expanders there exists a constant ϱ such that for any member of this family all but the largest eigenvalue of A have a value of less than $1 - \varrho$.

Let G be a d-regular expander belonging to some family $\mathcal{G}_\varrho$ and let G' denote the graph that is obtained from G by adding d loops to every vertex of G. Let A resp. A' be the adjacency matrices of G resp. G' multiplied by $1/d$ resp. $1/2d$. If λ is an eigenvalue of A then $(1 + \lambda)/2$ will be an eigenvalue of A'. Since the eigenvalues of G lie all between -1 and 1, we know that the eigenvalues of G' are between 0 and 1. Thus the family $\mathcal{G}'_\varrho$ consisting of all the graphs G' satisfies the requirements of the above lemma. Obviously the graphs G' are constructable in polynomial time.

We can therefore apply Lemma 3.3 to derive the following corollary:

Corollary 3.4 $\mathcal{NP} \subseteq \mathcal{PCP}_{n^{-\epsilon}}(\log n, \log n)$

Proof. Let $\mathcal{G}_\varrho$ be a family of 5-regular expanders satisfying the requirements of Lemma 3.3. Let G be a member of $\mathcal{G}_\varrho$ that has a vertex for every possible 0-1-string of length $\mathcal{O}(\log n)$.

By performing a random walk on G one takes the 0-1-string associated to every c-th vertex as a pseudo random string for a $(\log n, 1)$-restricted verifier with error probability less than 1/16. Lemma 3.3 shows that the probability that this verifier gives k times a wrong answer is less than 2^{-k}. Choosing $k \geq \epsilon \log n$ proves the corollary. □

Now by plugging the $(\log n, \log n)$-restricted verifier with error probability $n^{-\epsilon}$ into the proof of Theorem 3.1 one gets:

Theorem 3.5 *Unless $\mathcal{P} = \mathcal{NP}$, there exists a constant $\delta > 0$ such that the clique number of a graph cannot be approximated in polynomial time up to n^δ.* □

3.2 Non-approximability of the chromatic number

A *coloring* of a graph is an assignment of colors to the vertices of the graph such that no two adjacent vertices get the same color. The *chromatic number* of a graph G is the minimum number of colors needed in a coloring of G. It is denoted by $\chi(G)$.

The problem COLORING is to decide for a graph G and a number k whether the chromatic number of G is at most k. Like CLIQUE, this problem was shown to be $\mathcal{NP}$-complete in the famous paper of Karp [24]. In contrast to CLIQUE, the problem COLORING remains $\mathcal{NP}$-complete even for any constant $k \geq 3$.

The $\mathcal{NP}$-completeness of COLORING leads to the question of the best possible performance guarantee of an approximation algorithm for the chromatic number. The best algorithm known is due to Halldórsson and has a performance guarantee of $\mathcal{O}(n(\log\log n)^2/\log^3 n)$ [21].

On the other hand it has been shown by Garey and Johnson [18] that no polynomial time algorithm for approximating the chromatic number can have a performance guarantee better than $2 - \epsilon$, unless $\mathcal{P} = \mathcal{NP}$.

Like for CLIQUE the $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$ result can also be used to prove some non-approximability results for the chromatic number. Lund and Yannakakis [28] have shown that it is $\mathcal{NP}$-hard to approximate the chromatic number of a graph up to a factor of n^δ, for some constant $\delta > 0$. Their proof is based on the corresponding result for the clique number (Theorem 3.5). Khanna, Linial and Safra have simplified this proof in a recent paper [25]. Moreover they have shown that even 4-coloring a 3-chromatic graph is $\mathcal{NP}$-hard. The next two sections are devoted to these results.

3.2.1 Non-approximability for arbitrary chromatic numbers

The proof of Lund and Yannakakis for the non-approximability of the chromatic number up to a factor of n^δ is based on the graph G_x used in the proof of Theorem 3.5. Recall that the graph G_x is an s-partite graph with $s = 2^{r(n)}$ and classes of size q where both s and q are polynomial in n, the length of x. The special property of the graph G_x was that its clique number is either s or at most s/n^ϵ. Starting from this graph G_x, Lund and Yannakakis constructed a new graph H_x having the property that the large gap in the possible values of the clique number of G_x is transformed into a large gap of the clique covering number of H_x. Since the clique covering number of H_x equals the chromatic number of its complement this finishes the proof.

Khanna, Linial and Safra have used the same ideas for constructing the graph H_x but found a simpler transformation. We are now going to describe their proof.

Theorem 3.6 *There exists an $\epsilon > 0$ such that the chromatic number of a graph cannot be approximated in polynomial time up to a factor of n^ϵ, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. The proof of Theorem 3.5 shows that it is $\mathcal{NP}$-hard given an s-partite graph G with classes of size $q = \mathcal{O}(poly(s))$ to distinguish between the cases $\omega(G) = s$ and $\omega(G) < s^{1-\delta}$. We will now construct an s-partite graph H with classes of size $q' := (sq)^5$ such that $\omega(H) = \omega(G)$. Moreover if the clique number of G is s then the graph H can be covered by q' cliques of size s. The clique covering number of a graph is at least as large as its number of vertices divided by the size of a largest clique. Therefore the clique covering number of H is either q' or at least $q's^{\delta}$. Since q' is polynomial in s this means that approximating the clique covering number of a graph up to a factor of n^{ϵ} for some ϵ depending on δ is $\mathcal{NP}$-hard. Since the clique covering number of H equals the chromatic number of its complement this finishes the proof of the theorem.

We are now going to describe the construction of H. Let the vertices of the graph G be arranged in s rows each consisting of the q elements of one of the s classes of G. Thus each row is a stable set and if G has a clique of size s then this clique has a representative in each row.

The graph H is constructed in the following way: For each row of G there is a row in H consisting of a stable set of q' vertices. The vertices in each row are arranged into columns numbered $0, 1, \ldots, q' - 1$. Later we will describe a mapping between the vertices of one row of G to the vertices of the corresponding row of H. For every edge (u, v) from G we add an edge (u', v') in H where u' resp. v' are the images of u resp. v under this mapping.

Furthermore the graph H will be made invariant under rotation, i.e. if the i-th vertex in row a is connected to the j-th vertex in row b then we also add the $q' - 1$ edges connecting the $(i + k)$-th vertex of row a with the $(j + k)$-th vertex of row b for $k = 1, 2, \ldots, q' - 1$, where the sums are taken modulo q'.

This property of H implies that if we find a clique of size s in H then there are $q' - 1$ other cliques in H arising from the first clique under rotation, such that all these cliques cover H. Therefore if the clique number of H is s then its clique covering number is q'.

We will now describe how the vertices of each row of G are mapped to the vertices of a corresponding row of H. This mapping is based on the existence of a special function T.

Lemma 3.7 *For every integer n there exists an injective mapping $T : \{0, 1, \ldots, n - 1\} \to \{0, 1, \ldots, m - 1\}$ with $m = n^5$ such that the sums $T(a)+T(b)+T(c)$ are all distinct modulo m for different multisets $\{a, b, c\}$ with $a, b, c \in \{0, 1, \ldots, n - 1\}$.* □

The proof of this lemma is straightforward. If T is defined for all values up to $n - 2$ then one just has to define the value of $T(n - 1)$ appropriately. Note that the mapping T has also the property that for all distinct multisets $\{a, b\}$ of size two the sums $T(a) + T(b)$ are distinct.

Using the mapping T we define a map between the vertices of G and H by just mapping the i-th vertex of the j-th row of G to the $T(jq + i)$-th vertex of the j-th row of H.

We claim that $\omega(G)$ equals $\omega(H)$, which concludes the proof as outlined above. To see this, observe first that by construction every edge in H has at least one "origin" in G, namely the one which created it. Using the definition of T it follows immediately that in fact this origin is uniquely defined and that consequently every clique in H has an origin in G which is also a clique. This shows $\omega(H) \leq \omega(G)$. The reverse inequality follows trivially from the construction of H. □

3.2.2 Non-approximability for constant chromatic numbers

LUND and YANNAKAKIS also proved some non-approximability results for graphs with constant chromatic number. Namely, they showed that for every constant h there exists a constant c_h such that it is $\mathcal{NP}$-hard to color c_h-colorable graphs with less than $h \cdot c_h$ colors. Unfortunately the dependence of c_h on h did not allow them to get any implication for "small" constant chromatic numbers, like for example 3-colorable graphs.

KHANNA, LINIAL and SAFRA [25] however where able to extend the proof in such a way that they could derive non-approximability results also for such small constant chromatic numbers. Especially for 3-chromatic graphs, they where able to prove the following result which we state here without proof.

Theorem 3.8 *Unless $\mathcal{P} = \mathcal{NP}$, it is not possible to color a 3-colorable graph with 4 colors in polynomial time.* □

By substituting every vertex of a clique of size k by a 3-colorable graph, one immediately shows that a $3k$-colorable graph cannot be colored with $5k-1$ colors in polynomial time, unless $\mathcal{P} = \mathcal{NP}$. Thus we obtain as a corollary, for any constant chromatic number k:

Corollary 3.9 *Unless $\mathcal{P} = \mathcal{NP}$, it is not possible to color a k-colorable graph with $k+2\lceil \frac{k}{3} \rceil - 1$ colors in polynomial time.* □

Even though Theorem 3.8 is considered a breakthrough, it is, for example, still unknown whether it is possible to 5-color 3-colorable graphs in polynomial time. This seems to be very unlikely since the best performance ratio in coloring 3-colorable graphs is due to an algorithm of BLUM [11] that achieves a ratio of $n^{3/8} \log^{5/8} n$. Thus it is probably the case that coloring 3-colorable graphs is $\mathcal{NP}$-hard for any constant number of colors. It might even be true that coloring a 3-colorable graph with n^{ϵ} colors is $\mathcal{NP}$-hard for some $\epsilon > 0$.

3.3 Non-approximability of MAX-SNP hard problems

The development of the notion of $\mathcal{NP}$-completeness was mainly motivated by the study of apparently intractable *optimization* problems [24] [18]. Nevertheless by definition only *decision* problems may belong to the class $\mathcal{NP}$. Certainly every optimization problem can easily be converted into a decision problem by just imposing some bound on its cost function. But the particular property of being an optimization problem is not covered by the notion of $\mathcal{NP}$-completeness.

As a consequence the polynomial time reductions used for defining the completeness of a problem in $\mathcal{NP}$ do not reflect intrinsic properties of optimization problems like the value of the cost function or its approximability. While by definition all $\mathcal{NP}$-complete problems are equivalent under polynomial time reductions, the difficulty of the underlying optimization problems may vary tremendeously with regard to their approximability. According to present knowledge the optimization problems corresponding to $\mathcal{NP}$-complete problems fall into three classes:

1) Problems that can be approximated in polynomial time up to *any* desired constant (e.g. BIN-PACKING).
2) Problems that can be approximated in polynomial time up to *some* constant factor (e.g. euclidean TSP).
3) Problems for which no polynomial time constant factor approximation algorithm can exist, unless $\mathcal{P} = \mathcal{NP}$ (e.g. CLIQUE).

Until recently only a few singular results were known about separating the second class from the first. That is, ruling out the existence of a polynomial time approximation scheme (PTAS). One famous example is the graph coloring where GAREY and JOHNSON have shown that unless $\mathcal{P} = \mathcal{NP}$, no polynomial time approximation algorithm can achieve a factor of $2-\epsilon$ [17].

To overcome this situation PAPADIMITRIOU and YANNAKAKIS [30] introduced the class MAX-SNP together with an approximation preserving reduction, called *L-reduction*. This reduction has the property that if a problem A is L-reducible to a problem B for which a polynomial time constant factor approximation algorithm is known then such an algorithm also exists for problem A with some other constant. A problem in MAX-SNP (cf. [30] for a precise definition) that is complete for this class relative to L-reductions is called MAX-SNP complete. If any MAX-SNP complete problem would allow a polynomial time approximation scheme (PTAS) then by definition of L-reductions every problem in MAX-SNP would have a PTAS. On the other hand if one could prove for some MAX-SNP complete problem that it cannot have a PTAS then no MAX-SNP complete problem can have one (unless $\mathcal{P} = \mathcal{NP}$).

PAPADIMITRIOU and YANNAKAKIS have shown that several well-known approximation problems are MAX-SNP complete:

MAX-SAT Given a SAT instance find a truth assignment that satisfies as many clauses as possible. This problem remains MAX-SNP complete even if every clause is allowed to contain at most two variables.

STABLE-SET-B Given a graph whose maximum degree is bounded by some constant B find a maximum stable set.

NODE-COVER-B Given a graph whose maximum degree is bounded by a constant B find a minimum node cover.

DOMINATING-SET-B Given a graph whose maximum degree is bounded by a constant B find a minimum dominating set (i.e. a set of nodes that is adjacent to all other nodes).

MAX-CUT Partition the nodes of a graph into two sets A and B such that the number of edges between A and B is maximized.

EUCLIDEAN TSP For a set of points in the plane find a shortest tour that visits all the points.

While all of the above problems can be approximated in polynomial time up to some constant factor, no PTAS for any of these problems was known. The following theorem shows that such a PTAS cannot exist (modulo $\mathcal{P} \neq \mathcal{NP}$):

Theorem 3.10 (Arora, Lund, Motwani, Sudan, Szegedy 1992) *Unless $\mathcal{P} = \mathcal{NP}$, no MAX-SNP complete problem has a PTAS.*

Proof. We will show that the existence of a PTAS for MAX-3SAT implies $\mathcal{P} = \mathcal{NP}$. Since MAX-3SAT is MAX-SNP complete this proves the theorem.

Let L be an arbitrary language from $\mathcal{NP}$ and let V be the $(\log n, 1)$-restricted verifier for L (whose existence is guaranteed by Theorem 2.2).

For any $x \in \Sigma^*$ we will construct a 3SAT instance S_x such that S_x is satisfiable if and only if x is an element of L. Moreover if x does not belong to L then at most some constant fraction of the clauses in S_x can simultaneously be satisfied. Therefore a PTAS for MAX-3SAT could be used to recognize the language L in polynomial time, i.e. we would have $\mathcal{P} = \mathcal{NP}$.

For every position in a proof π we introduce a variable whose values True and False correspond to the values 1 and 0 of the bit at this position. By using these variables the 3SAT instance S_x is obtained as follows:

- For any possible random string τ let S_τ denote the Boolean formula that expresses which proofs π are accepted by V on input x. Since V queries only a constant number of bits from a proof, the formulas S_τ have each constant size.
- Let S'_τ be the formula S_τ written as a 3SAT formula. Let k denote the maximum number of clauses that appear in a S'_τ. Note that k is a constant.
- Now let S_x be the conjunction of all the S'_τ.

If x is an element of L then by definition of a restricted verifier there exists a proof π_x such that V accepts x for every random string τ. This means that this proof π_x is a satisfying assignment of S_x.

If x is not an element of L then for every proof the verifier V accepts x for at most 1/4th of all possible random strings τ. This means that at most 1/4th of the formulas S'_τ are simultaneously satisfiable. Since every S'_τ consists of at most k clauses we get that at most $1 - \frac{3}{4}\frac{1}{k}$ of the clauses of S_x are simultaneously satisfiable.

The existence of a PTAS for MAX-3SAT would therefore allow to distinguish between these two cases and thus it would be possible to recognize every language in $\mathcal{NP}$ in polynomial time. □

4 A proof of $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$

The second part of our paper is devoted to a proof of Theorem 2.2. The proof which we present here is self-contained and – despite of its inherit algebraic nature – we try to formulate it in a "combinatorial" language. We hope that this formulation makes the pioneering new characterization of $\mathcal{NP}$ more easily accessible to all interested discrete mathematicians, even if they have not followed the new developments in theoretical computer science during the last few years.

4.1 Overview and structure of the proof

The proof of Theorem 2.2 combines several recent developments in theoretical computer science. Most noteably, these are the theory of interactive proofs, the arithmetization of Boolean formulas, and the area of self-testing/self-correcting of computer programs.

The theory of interactive proofs originates in the work of GOLDWASSER, MICALI, and RACKOFF [20] and of BABAI [6]. Two of its major achievements are the characterizations $\mathcal{IP} = \mathcal{PSPACE}$ (cf. SHAMIR [34]) and $\mathcal{MIP} = \mathcal{NEXPTIME}$ (cf. BABAI, FORTNOW, LUND [10]). Both of these results are based on the work of LUND, FORTNOW, KARLOFF, and NISAN [27], who showed that $\mathcal{IP}$, the class of languages recognizable by polynomial-time interactive proof systems, contains the polynomial hierarchy. Due to lack of space we will not give a detailed account on the history of interactive proofs in this paper. We even omit a precise definition of interactive proofs and the class $\mathcal{IP}$ as we will not need them. The only fact from the theory of interactive proofs that we directly apply in this paper is a proof system from LUND, FORTNOW, KARLOFF, and NISAN [27]. For sake of completeness this is included in Appendix A. For more information on interactive proofs the interested reader is referred to BABAI [7] for an amusing introduction to this topic. A comprehensive survey together with some applications may also be found in JOHNSON [23].

A key ingredient of probabilistically checkable proofs is the following simple and well-known fact: if two polynomials of degree at most d coincide in at least $d+1$ points, then they are identical. In order to apply this idea and related algebraic concepts one needs to place "combinatorial" problems in an "algebraic" setting. This is, for example, achieved by the arithmetization of Boolean formulas. A brief introduction into this area is given in Section 4.3.

Another simple but important property of polynomials is that they are very robust: Even if, say, 1% percent of all values of a (low degree) polynomial are erroneous, it is not difficult to reconstruct the correct values. Self-testing and self-correcting plays a major role in the proof of Theorem 2.2. In particular, the proof relies on efficient testers for linear functions and low-degree polynomials. A more detailed introduction to this area is given throughout the subsequent sections, in particular in Section 4.5. In addition, Appendix B contains the theoretical background for the existence of efficient testers for low-degree polynomials.

Constructing a probabilistically checkable proof which can be checked by reading only a constant number of bits is not too difficult if we allow the proof to be of exponential length. We will do this in Section 4.4. In Section 4.6 on the other hand we develop a probabilistically checkable proof of polynomial size, which can be checked by reading only a constant number of "blocks" from the proof, but where every such block contains polylogarithmic many bits.

The existence of a probabilistically checkable proof which can be verified by reading only a constant number of bits follows from these two proof systems by applying them recursively. Roughly speaking this is done by also using these proof systems to encode the "blocks" of such proofs. More precisely, we proceed as follows. Using the second proof system recursively we first obtain a new probabilistically checkable proof, which can be verified by reading a constant number of blocks, but this time each block consists of only polydoublylogarithmic many bits. Subsequently, we use the first proof system to encode the blocks of this new proof system. This then gives the desired probabilistically checkable proof showing that $\mathcal{NP} \subseteq \mathcal{PCP}(\log n, 1)$. The other inclusion follows easily from the fact that there exist only polynomially many random strings of length $\mathcal{O}(\log n)$.

It is worthwhile to observe that the definition of the class $\mathcal{PCP}(r(n), q(n))$ or, more generally, that of probabilistically checkable proofs, very nicely reflects an important property of $\mathcal{NP}$. The definition of $\mathcal{NP}$ by nondeterministic Turing machines requires the *existence* of a "proof" (or solution) which can be verified in polynomial time, but places absolutely no restriction on how such a proof can be found. Similarly, the verifier is willing to accept an input, if it is convinced that there exists a solution – even without having an idea what the solution looks like.

Often, however, just knowing of the existence of a solution does not suffice. In particular, for the recursive application of proof systems indicated above, we need more. There the verifier has to check, with help of a specified proof, that a given string y is a solution – and it should do that without reading the solution completely. That is, the solution is subject to the same restrictions as the proof: the verifier may only query a few bits from it.

Having in mind the example of a 3-coloring and an almost 3-coloring mentioned in the introduction, we easily conclude that such a clever verifier cannot exist. Something slightly weaker, however, turns out to be true. If instead of the string y we give the verifier a string y' which supposedly corresponds to the encoding of y according to some fixed, predetermined encoding scheme (or function) E, then the verifier can decide whether y' is close to $E(y)$ for some solution y – by probing y' as well as the proof at only a few places.

In Section 4.2 we give the formal definition of this idea. There we introduce the classes $\mathcal{PCS}(r(n),q(n),b(n))$ which are defined similarly to the sets $\mathcal{PCP}(r(n), q(n))$, the main differences being that the new classes contain p-relations together with encoding schemes for the solutions, instead of simply languages, and have a third parameter indicating the sizes of the blocks read from the proof.

Figure 2 contains an overview of the proof of Theorem 2.2 together with an indication of where the various tools like arithmetization, low-degree and LFKN-tests enter the proof.

We would like to mention that our proof of Theorem 2.2 is based on the papers of Arora, Safra [5], Arora, Lund, Motwani, Sudan, Szegedy [3], [4], Phillips, Safra [31], and Sudan [35]. We combine their ideas into a, we hope, streamlined and self-contained proof.

4.2 Probabilistically checkable solutions

While complexity classes such as $\mathcal{NP}$ are usually defined for languages $L \subseteq \Sigma^*$, a notation that is closer to intuition is that of relations $R \subseteq \Sigma^* \times \Sigma^*$ which associate with every problem instance x a finite set $R(x)$ of "solutions". As an example consider

$$\begin{aligned} \mathrm{R}_{3\,\mathrm{SAT}} = \{ (x,y) \mid\ & x \in \Sigma^* \text{ encodes a Boolean formula } F \text{ in conjunctive} \\ & \text{normal form with exactly three literals per clause,} \\ & y \in \Sigma^* \text{ encodes a satisfying assignment of } F \ \}. \end{aligned}$$

(Here and in the following we assume without loss of generality $\Sigma = \{0,1\}$.) A relation $R \subseteq \Sigma^* \times \Sigma^*$ is called a *p-relation* iff

(i) There exists a polynomial p such that $|y| \leq p(|x|)$ for all $(x,y) \in R$.

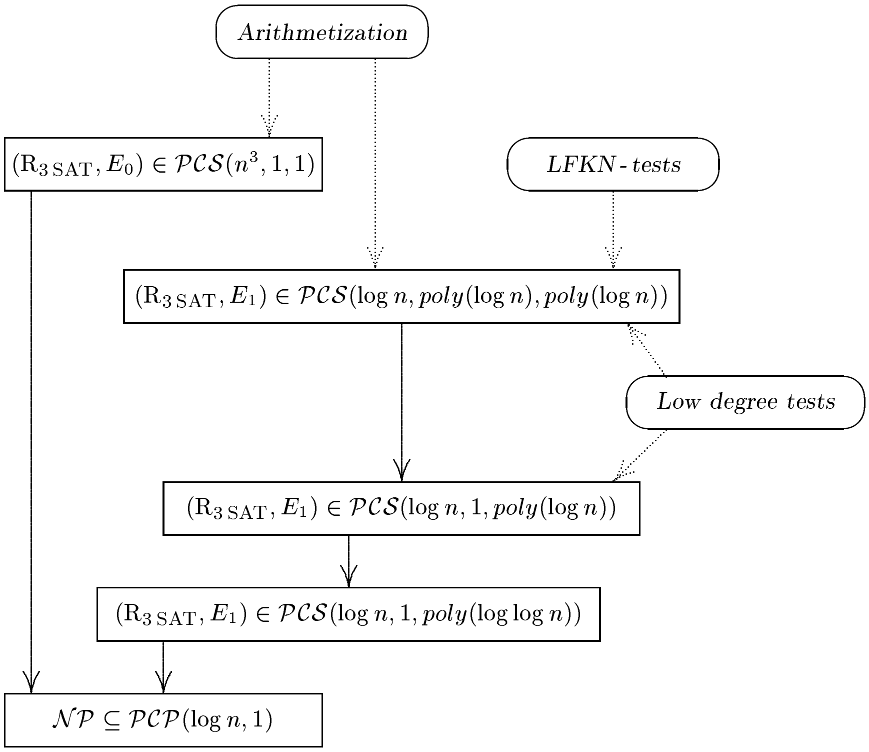

Figure 2: Structure of the proof of Theorem 2.2

(ii) The predicate $(x, y) \in R$ can be tested in time polynomial in $|x| + |y|$.

It is well-known, that the class of existence problems associated with p-relations may be identified with the class $\mathcal{NP}$. In particular, the relation $\mathrm{R}_{3\,\mathrm{SAT}}$ introduced above is a p-relation.

Let $x, y \in \Sigma^*$ be two strings such that $|x| = |y|$. Then x and y are called δ-close if and only if the fraction of bits on which they differ is less than δ. An *encoding scheme* is a function $E : \Sigma^* \to \Sigma^*$ such that for all $x, x' \in \Sigma^*$, $x \neq x'$ with $|x| = |x'|$ the encodings $E(x)$ and $E(x')$ have the same length and coincide in at most $\frac{1}{2}$ of their bits, that is they are not $\frac{1}{2}$-close.

A *solution verifier* V is a verifier which in addition has access to a solution s, which it can query via an oracle in the same way as the membership proof π (cf. Figure 3).

An $(r(n), q(n), b(n))$*-restricted solution verifier* is a solution verifier which for inputs x of length n uses at most $\hat{r}(n)$ random bits and queries at most $\hat{q}(n)$ blocks of length $\hat{b}(n)$ from s <u>and</u> π, where the starting positions of such blocks are all congruent one modulo $\hat{b}(n)$[1] and $\hat{r}(n)$, $\hat{q}(n)$, and $\hat{b}(n)$ are integral functions such that $\hat{r}(n) = \mathcal{O}(r(n))$, $\hat{q}(n) = \mathcal{O}(q(n))$, and $\hat{b}(n) = \mathcal{O}(b(n))$.

Definition 4.1 *Let R be a p-relation and E be an encoding-scheme. Then (R, E) is in $\mathcal{PCS}(r(n), q(n), b(n))$ iff there exists an $(r(n), q(n), b(n))$-restricted solution verifier V such that:*

[1]That is, we assume that the membership proof π and the solution s are partitioned into blocks of length $\hat{b}(n)$ and the verifier may read at most $\hat{q}(n)$ of these blocks.

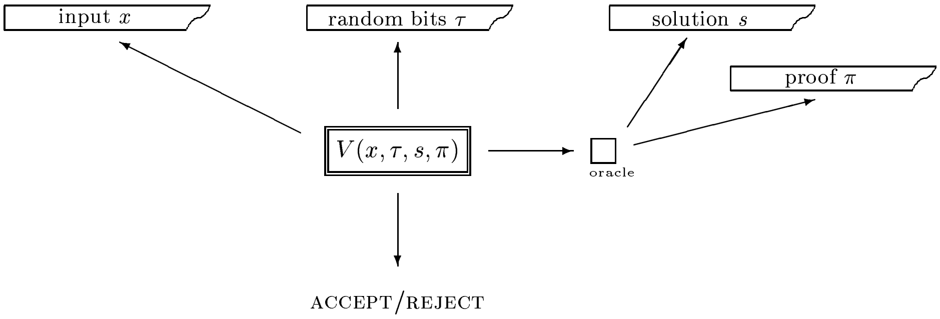

Figure 3: A solution verifier for p-relations

(i) For all $x, y \in \Sigma^*$ with $(x,y) \in R$ there exists a proof $\pi_{x,y}$ such that

$$Prob_\tau[V(x,\tau,E(y),\pi_{x,y}) = \text{ACCEPT}] \;=\; 1,$$

(ii) For all $x, s \in \Sigma^*$ such that s is not $\frac{1}{4}$-close to the encoding $E(y)$ of a solution $y \in R(x)$ every proof π satisfies

$$Prob_\tau[V(x,\tau,s,\pi) = \text{ACCEPT}] \;<\; \tfrac{1}{4}.$$

The next proposition establishes the intuitive idea that solution verifiers are at least as powerful than ordinary verifiers.

Proposition 4.2 *Let R be a p-relation and E be an encoding-scheme, and let L denote the language defined by $x \in L$ if and only if $R(x) \neq \emptyset$. Then $(R,E) \in \mathcal{PCS}(r(n), q(n), b(n))$ implies $L \in \mathcal{PCP}(r(n), q(n) \cdot b(n))$.*

□

4.3 Arithmetization

The concept of arithmetization of the intrinsically Boolean process of computation by using multivariate polynomials was introduced simultaneously and independently by Babai and Fortnow [8] and Shamir [34]. It has been a key tool in determining the power of interactive proof systems culminating in the results $\mathcal{IP} = \mathcal{PSPACE}$ [34] and $\mathcal{MIP} = \mathcal{NEXPTIME}$ [10]. One reason for the dramatic success of arithmetization is that it opened the way for the application of a variety of algebraic concepts and methods, such as the degree of polynomials, interpolation and field extensions within complexity theory.

In [8] Babai and Fortnow describe the technique of arithmetization in a very general setting. For our purposes, however, it suffices to restrict our attention to the arithmetization of Boolean formulas, in fact even to the arithmetization of conjunctions.

A *Boolean formula* is an expression built from variables x_i and their negations $\bar{x}_i$ using the operations $\vee$ and $\wedge$. A *conjunction* (*disjunction*) is a Boolean formula using only the operation $\wedge$ ($\vee$). A *Boolean formula in conjunctive normal form*, finally, is obtained by joining several disjunctions by the operator $\wedge$.

An *arithmetic formula* is an expression built from the constants 0, 1 and variables x_i using the operations $+$, $-$ and $\cdot$. An arithmetic formula represents a multivariate polynomial function over any field in the obvious way.

The arithmetization of a Boolean formula is obtained by replacing every negated variable $\bar{x}_i$ by $1-x_i$, every conjunction $\alpha \wedge \beta$ by $\alpha \cdot \beta$, and every disjunction $\alpha \vee \beta$ by $1-(1-\alpha)(1-\beta)$. One easily checks that a Boolean formula B has a satisfying assignment (an assignment such that the formula evaluates to true), if and only if its arithmetization $A(B)$ is not identically zero. Even more is true. Considered as a polynomial over $\mathbb{F}_2$, the value of $A(B)$ coincides with the value of B (identifying 0 with *false* and 1 with *true*).

Recall that the input of a 3SAT-problem is a Boolean formula in conjunctive normal form, in which every disjunction (usually called a *clause*) contains exactly three (potentially negated) variables. In the following we always assume that the input of such a satisfiability problem contains exactly n clauses $C_1, \ldots, C_n$ using m variables $x_1, \ldots, x_m$, where without loss of generality (by adding dummy variables) we may also assume that $n = m$.

The arithmetization $\mathcal{C}(x) = (\hat{C}_1(x), \ldots, \hat{C}_n(x))$ of a satisfiability problem $C_1 \wedge \ldots \wedge C_n$ is obtained by letting $\hat{C}_i$ be the arithmetization of the complement of the ith clause. Then the following observation is immediate.

Observation 4.3 *A vector $a \in \mathbb{F}_2^n$ corresponds to a satisfying assignment of $C_1 \wedge \ldots \wedge C_n$ if and only if $\mathcal{C}(a) = (\hat{C}_1(a), \ldots, \hat{C}_n(a))$ is identically zero.*

Despite its simplicity Observation 4.3 forms the basis of the proof of $\mathcal{NP} = \mathcal{PCP}(\log n, 1)$. In particular, it enters the proof at two places. In the next section we will use it to develop a $(n^3, 1)$-restricted verifier for 3SAT, while in Section 4.6 it is used to show that 3SAT is contained in $\mathcal{PCP}(\log n, \mathrm{poly}(\log n))$.

4.4 $\mathcal{NP} \subseteq \mathcal{PCP}(\mathbf{poly}(n), 1)$

In this section we will show that 3SAT has an $(n^3, 1, 1)$-restricted solution verifier, thereby establishing that $\mathcal{NP}$ is a subset of $\mathcal{PCP}(\mathrm{poly}(n), 1)$.

While at first sight the existence of any polynomial time verifier for 3SAT reading only a constant number of bits from the proof seems rather surprising (even if we allow access to an arbitrary number of random bits), such a verifier can in fact be quite easily constructed from the arithmetization of the previous section. The key idea here is that testing whether a given vector $x \in \mathbb{F}_2^n$ is identically zero can easily be done by choosing a random vector $r \in \mathbb{F}_2^n$ and considering the product $x^T r$. While this product is always zero if $x = 0$, it is nonzero with probability $\frac{1}{2}$, whenever $x \neq 0$.

The only other observation we need is that the product of the arithmetization $(\hat{C}_1(x), \ldots, \hat{C}_n(x))$ of a satisfiability problem with a vector $r \in \mathbb{F}_2^n$ can be written as

$$\sum_{i=1}^{n} r_i \hat{C}_i(x) \quad = \quad c(r) + \sum_{i \in S_1(r)} x_i + \sum_{(i,j) \in S_2(r)} x_i x_j + \sum_{(i,j,k) \in S_3(r)} x_i x_j x_k,$$

where the sets $S_1(r)$, $S_2(r)$, $S_3(r)$ and the constant $c(r)$ depend only on the given 3SAT formula and the vector r, but *not* on the assignment x. In particular, this shows that if for some fixed assignment $a \in \mathbb{F}_2^n$ the verifier would have some "magical" access to the sums $\sum_{i \in S_1} a_i$, $\sum_{(i,j) \in S_2} a_i a_j$ and $\sum_{(i,j,k) \in S_3} a_i a_j a_k$ for given sets S_1, S_2 and S_3, resp., it would indeed need to make only a constant number of enquiries to decide whether a is a satisfying assignment or not.

The rest of this section is devoted to turning these rough ideas into a precise description of an $(n^3, 1)$-restricted verifier. We start with some notations. For a vector $a \in \mathbb{F}_2^n$ we define three linear functions as follows:

$$\begin{aligned} A &: \mathbb{F}_2^n \to \mathbb{F}_2, & A(x) &:= \sum_{i=1}^n a_i\, x_i, \\ B &: \mathbb{F}_2^{n^2} \to \mathbb{F}_2, & B(y) &:= \sum_{i=1}^n \sum_{j=1}^n a_i\, a_j\, y_{ij}, \qquad \text{and} \\ C &: \mathbb{F}_2^{n^3} \to \mathbb{F}_2, & C(z) &:= \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n a_i\, a_j\, a_k\, z_{ijk}. \end{aligned}$$

The verifier interprets every proof π as $\pi = \tilde{A}\tilde{B}\tilde{C}$, where $\tilde{A}$ has length 2^n and is considered as a function $\tilde{A} : \mathbb{F}_2^n \to \mathbb{F}_2$. Similarly, $\tilde{B}$ and $\tilde{C}$ have length 2^{n^2} and 2^{n^3}, respectively, and are interpreted as functions $\tilde{B} : \mathbb{F}_2^{n^2} \to \mathbb{F}_2$ and $\tilde{C} : \mathbb{F}_2^{n^3} \to \mathbb{F}_2$. Ideally, $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ correspond to the functions A, B and C from above, defined with respect to some vector $a \in \mathbb{F}_2^n$ corresponding to a satisfying assignment.

The verifier needs to achieve two tasks:

(A) Verify that $\tilde{A}$, $\tilde{B}$, $\tilde{C}$ are what they are supposed to be, namely linear functions defined with respect to the *same* vector $a \in \mathbb{F}_2^n$, and

(B) verify that this vector a corresponds to a satisfying assignment.

We first consider task (A). By reading only a constant number of bits from the proof π it is clearly impossible to verify that $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ *are* linear functions. (Assume for example $\tilde{A}$ differs from a linear function in just one bit. So the verifier can detect this with probability one only if it reads all bits from $\tilde{A}$, and with high probability only by reading a substantial fraction of all bits.) So the best we can hope for is to assure that $\tilde{A}$ corresponds to a linear function at all but a constant fraction of $\mathbb{F}_2^n$. We make this precise as follows. Let F and G be two arbitrary finite fields. Two functions $f, g : F \to G$ are called *δ-close*, iff the number of vectors $x \in F$ for which $f(x) = g(x)$ is at least $(1-\delta)|F|$. Using the language of probability theory, the latter condition can also be written as

$$\mathrm{Prob}_x[f(x) = g(x)] \geq 1 - \delta.$$

We will henceforth use this notation quite often. (Note that correctly we should write $\mathrm{Prob}_{x \in_{\mathrm{R}} F}[\cdots]$ instead of $\mathrm{Prob}_x[\cdots]$, where $x \in_{\mathrm{R}} F$ denotes a random element chosen uniformly from F. We use the short notation for conciseness whenever there is no risk of confusion.)

During the last few years the problem of detecting whether a given function is δ-close to a polynomial of some given degree has been intensively studied. In Appendix B we give an account of the results obtained. There we also prove the following lemma.

Lemma B.1 *Let $\delta < \frac{1}{3}$ be a constant and let $\tilde{g} : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function such that*

$$Prob_{x,y}[\tilde{g}(x) + \tilde{g}(y) \neq \tilde{g}(x+y)] \leq \frac{\delta}{2}.$$

Then there exists $h \in \mathbb{F}_2^n$ so that the functions $g(x) = h^T x$ and $\tilde{g}$ are δ-close.

With Lemma B.1 at hand, a linearity test is easily designed.

<u>LINEARITY TEST</u>
Pick $x, x' \in_{\mathrm{R}} \mathbb{F}_2^n$,
    verify that $\tilde{A}(x) + \tilde{A}(x') = \tilde{A}(x + x')$.
Pick $y, y' \in_{\mathrm{R}} \mathbb{F}_2^{n^2}$,
    verify that $\tilde{B}(y) + \tilde{B}(y') = \tilde{B}(y + y')$.
Pick $z, z' \in_{\mathrm{R}} \mathbb{F}_2^{n^3}$,
    verify that $\tilde{C}(z) + \tilde{C}(z') = \tilde{C}(z + z')$.

Observe that Lemma B.1 immediately implies that if at least one of the functions $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$ is *not* δ-close to a homogeneous linear function, then LINEARITY TEST fails with probability at least $\frac{\delta}{2}$. Repeating this test a constant number of times we can therefore push the failure probability arbitrarily close to one.

Corollary 4.4 *Let $0 < \delta < \frac{1}{3}$ be a fixed constant. Then there exists a constant $k = k(\delta)$ such that, if there do not exist vectors $a \in \mathbb{F}_2^n$, $b \in \mathbb{F}_2^{n^2}$, and $c \in \mathbb{F}_2^{n^3}$ so that $\tilde{A}$ is δ-close to A, $\tilde{B}$ is δ-close to B, and $\tilde{C}$ is δ-close to C, where A, B and C are the functions given by $A(x) = a^T x$, $B(y) = b^T y$, and $C(z) = c^T z$, resp., then with probability at least $1 - \delta$ at least one of k calls of* LINEARITY TEST *fails.* □

To conclude task (A), it remains to assure that the vectors $a = (a_i)$, $b = (b_{ij})$, and $c = (c_{ijk})$ of Corollary 4.4 are consistent, i.e. satisfy $b_{ij} = a_i a_j$ and $c_{ijk} = a_i a_j a_k$. Observe that if for $x, x' \in \mathbb{F}_2^n$ we let $x \circ x'$ denote the vector $y \in \mathbb{F}_2^{n^2}$ given by $y_{ij} = x_i \cdot x_j$, then the functions $A(x) = a^T x$ and $B(y) = b^T y$ satisfy

$$A(x) \cdot A(x') = B(x \circ x') \qquad \text{for all } x, x' \in \mathbb{F}_2^n \tag{2}$$

if and only if $b = a \circ a$. Similarly, if for $x \in \mathbb{F}_2^n$ and $y \in \mathbb{F}_2^{n^2}$ we let $x \circ y$ denote the vector $z \in \mathbb{F}_2^{n^3}$ given by $z_{ijk} = x_i \cdot y_{jk}$, then $A(x) = a^T x$, $B(y) = b^T y$, and $C(z) = c^T z$ satisfy

$$A(x) \cdot B(y) = C(x \circ y) \qquad \text{for all } x \in \mathbb{F}_2^n \text{ and } y \in \mathbb{F}_2^{n^2} \tag{3}$$

if and only if $c = a \circ b$.

In principle, (2) and (3) are natural candidates for a test whether the functions $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ are consistent. In the analysis of such a test one problem occurs, however. For $x, x' \in_{\mathrm{R}} \mathbb{F}_2^n$, for example, the vector $x \circ x'$ is *not* a random element from $\mathbb{F}_2^{n^2}$. So offhand we can *not* use the δ-closeness of $\tilde{B}$ and B to bound $\mathrm{Prob}_{x,x'}[\tilde{B}(x \circ x') = B(x \circ x')]$ by $1 - \delta$.

We resolve this problem by using so called *self-correcting functions*. (This notion was introduced independently by LIPTON [26] and BLUM, LUBY, and RUBINFELD [12]. For more information on this topic we refer the interested reader to these articles.)

SELF-CORRECTING FUNCTIONS

SC-$\tilde{A}(x)$:	Pick $r \in_{\mathrm{R}} \mathbb{F}_2^n$,	return $\tilde{A}(r+x) - \tilde{A}(r)$.
SC-$\tilde{B}(y)$:	Pick $r \in_{\mathrm{R}} \mathbb{F}_2^{n^2}$,	return $\tilde{B}(r+y) - \tilde{B}(r)$.
SC-$\tilde{C}(z)$:	Pick $r \in_{\mathrm{R}} \mathbb{F}_2^{n^3}$,	return $\tilde{C}(r+z) - \tilde{C}(r)$.

Note that SC-$\tilde{A}(x)$, SC-$\tilde{B}(x)$ and SC-$\tilde{C}(x)$ are random functions which is not reflected in the notation.

Observation 4.5 *If $\tilde{A}$ is δ-close to a linear function $A : \mathbb{F}_2^n \to \mathbb{F}_2$, then*

$$Prob[SC\text{-}\tilde{A}(x) = A(x)] \geq 1 - 2\delta \qquad \text{for every } x \in \mathbb{F}_2^n .$$

Analogous results hold for $\tilde{B}$ and $\tilde{C}$. □

Combining the ideas from above with the self-correcting functions we can now state a consistency test for $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$.

CONSISTENCY TEST

Pick $x, x' \in_{\mathrm{R}} \mathbb{F}_2^n$,
    verify that SC-$\tilde{A}(x) \cdot$ SC-$\tilde{A}(x')$ = SC-$\tilde{B}(x \circ x')$.
Pick $x \in_{\mathrm{R}} \mathbb{F}_2^n$, $y \in_{\mathrm{R}} \mathbb{F}_2^{n^2}$,
    verify that SC-$\tilde{A}(x) \cdot$ SC-$\tilde{B}(y)$ = SC-$\tilde{C}(x \circ y)$.

Lemma 4.6 *Let $0 < \delta < \frac{1}{24}$ be a fixed constant. Then there exists a constant $k = k(\delta)$ such that, if there does not exist a vector $a \in \mathbb{F}_2^n$ so that $\tilde{A}$ is δ-close to $A(x) = a^T x$, $\tilde{B}$ is δ-close to $B(y) = (a \circ a)^T y$, and $\tilde{C}$ is δ-close to $C(z) = (a \circ a \circ a)^T z$, then with probability at least $1 - \delta$ at least one of k calls of* LINEARITY TEST *and* CONSISTENCY TEST *fails.*

Proof. By Corollary 4.4 we may assume that $\tilde{A}$, $\tilde{B}$, and $\tilde{C}$ are δ-close to linear functions $A(x) = a^T x$, $B(y) = b^T y$ and $C(z) = c^T z$. If $b = a \circ a$ and $c = a \circ b$, there is nothing to show. So we assume without loss of generality that $b \neq a \circ a$. (The case that $c \neq a \circ b$ is treated similarly.) Recall that for vectors $\alpha \neq \hat{\alpha} \in \mathbb{F}_2^n$ one has

$$\mathrm{Prob}_{x \in_{\mathrm{R}} \mathbb{F}_2^n}[\alpha^T x \neq \hat{\alpha}^T x] = \frac{1}{2}. \tag{4}$$

In particular, this implies that matrices $\beta \neq \hat{\beta} \in \mathbb{F}_2^{n^2}$ satisfy

$$\mathrm{Prob}_{x \in_{\mathrm{R}} \mathbb{F}_2^n}[\beta^T x \neq \hat{\beta}^T x] \geq \frac{1}{2}. \tag{5}$$

Combining (4) and (5) we deduce, that if $b \neq a \circ a$ then (considering b as a matrix)

$$\mathrm{Prob}_{x, x' \in_{\mathrm{R}} \mathbb{F}_2^n}[x^T (a \circ a)^T x' \neq x^T b x'] \geq \frac{1}{4}.$$

As $x^T (a \circ a)^T x' = A(x) \cdot A(x')$ and $x^T b x' = B(x \circ x')$, this together with the δ-closeness and Observation 4.5 implies

$$\mathrm{Prob}_{x, x' \in_{\mathrm{R}} \mathbb{F}_2^n}[\text{SC-}\tilde{A}(x) \cdot \text{SC-}\tilde{A}(x') \neq \text{SC-}\tilde{B}(x \circ x')] \geq \frac{1}{4} - 6\delta \; > \; 0,$$

concluding the proof of Lemma 4.6. □

It remains to design a procedure which enables the verifier to achieve task (B). This procedure, however, is an immediate consequence of Observation 4.3 and the remarks at the beginning of this section.

<u>SATISFIABILITY TEST</u>

Pick $r \in_{\mathrm{R}} \mathbb{F}_2^n$, compute $c = c(r) \in \mathbb{F}_2$, $S_1 = S_1(r) \in \mathbb{F}_2^n$, $S_2 = S_2(r) \in \mathbb{F}_2^{n^2}$, and $S_3 = S_3(r) \in \mathbb{F}_2^{n^3}$, verify that $c + \tilde{A}(S_1) + \tilde{B}(S_2) + \tilde{C}(S_3) = 0$.

Lemma 4.7 *Let $\delta > 0$ be a fixed constant and assume that $\tilde{A}$ is δ-close to $A(x) = a^T x$, $\tilde{B}$ is δ-close to $B(y) = (a \circ a)^T y$, and $\tilde{C}$ is δ-close to $C(z) = (a \circ a \circ a)^T z$. Then there exists a constant $k = k(\delta)$ such that with probability at least $1 - \delta$ at least one of k calls of* SATISFIABILITY TEST *fails if the vector a does not correspond to a satisfying assignment.* □

Combining Lemma 4.6 and 4.7 concludes the proof of the desired result.

Theorem 4.8 *There exists a constant k such that repeating* LINEARITY TEST, CONSISTENCY TEST *and* SATISFIABILITY TEST *k times and rejecting whenever one of the tests fails, forms a $(n^3, 1)$-restricted verifier for 3SAT.* □

Corollary 4.9 $\mathcal{NP} \subseteq \mathcal{PCP}(poly(n), 1)$. □

If we consider the function $\tilde{A}$ not as part of the proof, but as an (encoded) solution, Theorem 4.8 also implies the existence of a $(n^3, 1, 1)$-restricted solution verifier.

Corollary 4.10 *Let E_0 denote the encoding scheme given by $E_0 : y \mapsto \{y^T z\}_{z \in \mathbb{F}_2^{|y|}}$. Then* $(R_{3\,\mathrm{SAT}}, E_0) \in \mathcal{PCS}(n^3, 1, 1)$. □

4.5 Low degree tests and low degree extensions

Despite its relative simplicity the proof of $\mathcal{NP} = \mathcal{PCP}(poly(n), 1)$ of the previous section is not only just an example of a proof systems which can be checked probabilistically by reading only a constant number of bits, it also contains already the major ideas required in the remaining (more technical) part of the proof of Theorem 2.2. Before we continue with this proof we will elaborate these ideas more clearly.

Traditional membership proofs for an $\mathcal{NP}$-problem usually consist just of a 3-coloring, a satisfying assignment, etc. While these proofs are very concise they are at the same time also "unstructured" in the sense that every single bit matters. Therefore, in order to check the proof, one really has to read the whole proof. As we have seen in the previous section, the picture changes if the proof contains instead of simply, say, a satisfying assignment a, all values of the function $x \mapsto a^T x$. Not only does the "structure" inherent to such a proof allows to check its correctness (more precisely, the correctness of all but a δ-fraction of the bits) by querying only a constant number of bits, but at the same time such a proof contains 2^n bits of (useful) information (which can be used to check that the vector a *is* a satisfying assignment).

A main drawback, however, of encoding a vector a by the homogeneous linear function $a^T x$ is that the obtained proofs are of exponential size and therefore require $poly(n)$ many random bits for checking it. The aim of this section is to introduce a better encoding scheme. This is based on polynomials, whose degree is allowed to depend on the length of the encoded vector.

The first problem which arises is that every straightforward generalization of the linearity test of the previous section to polynomials whose total degree depends on n would have to read more than constantly many bits – as every polynomial of degree d is determined only by $d+1$ points. So, in order to define a test procedure for arbitrary polynomials one has to add a new idea. In fact, one which was also contained in the proof of the previous section works here as well.

A straightforward way of testing whether a vector a is identically zero would be to read every bit and check whether it is zero. By adding additional information (the values of the function $x \mapsto a^T x$), however, we were able to avoid reading every bit of a. To construct a tester for polynomials whose degree is small compared to the size of the field (in the sequel such polynomials are simply called *low degree polynomials*) we proceed similarly. To formally state such a test procedure, the following theorem, whose proof is contained in Appendix B, is extremely useful.

Theorem B.2 *Let $0 < \delta < \frac{1}{10512}$ and $d, m \in I\!N$ be constants and $p \geq 64d^3$ a prime. Let $\tilde{g} : \mathbb{F}_p^m \to \mathbb{F}_p$ be a function, and let $T : \mathbb{F}_p^m \times \mathbb{F}_p^m \to \mathbb{F}_p^{d+1}$ be a function such that the degree d polynomials $\tilde{P}_{x,h}$ over $\mathbb{F}_p$ given by $\tilde{P}_{x,h}(t) = \sum_{i=1}^{d+1} T(x,h)_i \cdot t^{i-1}$ satisfy*

$$Prob_{x,h,t}[\tilde{P}_{x,h}(t) = \tilde{g}(x+th)] \geq 1 - \frac{\delta}{8}.$$

Then there exists a (unique) polynomial $g : \mathbb{F}_p^m \to \mathbb{F}_p$ of total degree d so that g and $\tilde{g}$ are δ-close.

With Theorem B.2 at hand, it is now straightforward to design a test procedure — if we provide it with access to a table containing all values of the function T.

<u>Low Degree Test</u>
Repeat $2/\log(1+\delta/8)$ times:
· Pick $x, h \in_{\mathrm{R}} \mathbb{F}_p^m$ and $t \in_{\mathrm{R}} \mathbb{F}_p$,
· *if* $\sum_{i=1}^{d+1} T(x,h)_i \cdot t^{i-1} \neq \tilde{g}(x+th)$ *then* REJECT.

The following theorem shows that the procedure Low Degree Test has the desired properties.

Theorem 4.11 *Let $0 < \delta < \frac{1}{10512}$ and $d, m \in I\!N$ be constants and $p \geq 64d^3$ a prime and suppose the procedure* Low Degree Test *has access to the values of a function $\tilde{g} : \mathbb{F}_p^m \to \mathbb{F}_p$ and to a table T containing $|\mathbb{F}_p|^{2m}$ entries of size $(d+1)\lceil \log |\mathbb{F}_p| \rceil$. Then the following is true:*

· *If $\tilde{g} : \mathbb{F}_p^m \to \mathbb{F}_p$ is a polynomial of total degree d then there exists a table $T = T_g$ such that the procedure* Low Degree Test *always accepts.*

- *If on the other hand $\tilde{g}$ is not δ-close to such a polynomial then* LOW DEGREE TEST *rejects with probability at least $3/4$, for all tables T.*
- LOW DEGREE TEST *queries only $\mathcal{O}(1)$ values from $\tilde{g}$, reads $\mathcal{O}(1)$ entries from T, and uses $\mathcal{O}(m \log |\mathbb{F}_p|)$ random bits.*

□

While the procedure LOW DEGREE TEST provides us with a test of whether a given function is δ-close to a low degree polynomial, in order to apply it we still have to describe a way to transform an arbitrary vector into a low degree polynomial. This is done by the so called *low degree extensions*. Let F be a finite field, H be an arbitrary subset of F and $f : H^m \to \{0,1\}$ be a function. Then there exists a unique polynomial $f' : F^m \to F$ of degree at most $|H|$ in each variable that agrees with f on H^m. Namely, the one given by the Lagrange polynomial

$$f'(x_1, \ldots, x_m) = \sum_{(h_1,\ldots,h_m)\in H^m} \prod_{i=1}^{m} \prod_{\substack{y\neq h_i \\ y\in H}} \frac{x_i - y}{h_i - y} \cdot f(h_1, \ldots, h_m).$$

To exhibit the usefulness of these low degree extensions more clearly, assume that a is an arbitrary string of bits of length n. Let $p = \Theta(poly(\log n))$ be a prime number, let $H \subseteq \mathbb{F}_p$ be an arbitrary subset of size $|H| = \lceil \log n \rceil$, and let $m = \lceil \log n / \log\log n \rceil$. Then $|H|^m \geq n$ and we can therefore interpret a as a function f_a from H^m to $\{0,1\}$. If we now require a proof π to contain all values of the low degree extension f'_a (instead of simply the string a) the proof has length $poly(n)$ (instead of just n), but now contains the information a in a structured form, whose "correctness" can be checked by the LOW DEGREE TEST from Theorem 4.11 by using $\mathcal{O}(m \log |\mathbb{F}_p|) = \mathcal{O}(\log n)$ random bits and querying only $\mathcal{O}(1)$ values of size $\log |\mathbb{F}_p| = \mathcal{O}(poly(\log n))$ of this proof and $\mathcal{O}(1)$ values of size $m|H| \log |\mathbb{F}_p| = \mathcal{O}(poly(\log n))$ of an additional table also of size $\mathcal{O}(poly(n))$. This approach will be used heavily in the following section.

4.6 $(\mathbf{R}_{3\,\mathrm{SAT}}, E_1) \in \mathcal{PCS}(\log n, 1, poly(\log n))$

In this section we will show that every language in $\mathcal{NP}$ has a solution verifier that uses only $\mathcal{O}(\log n)$ random bits and queries only $\mathcal{O}(1)$ blocks of size $poly(\log n)$ from the proof π and the solution s.

A preliminary result (querying $poly(\log n)$ instead of $\mathcal{O}(1)$ blocks) was first proved by BABAI, FORTNOW, LEVIN and SZEGEDY [9], cf. Section 4.6.1. The main tools used for proving this result are an extended version of a test designed by LUND, FORTNOW, KARLOFF and NISAN (henceforth called the LFKN-test), which is described in more details in Appendix A, and the low degree extension of functions, introduced in the previous section.

4.6.1 $\mathcal{NP} \subseteq \mathcal{PCP}(\log n, poly(\log n))$

We will show that 3SAT is in $\mathcal{PCS}(\log n, poly(\log n), poly(\log n))$ which yields the desired result for $\mathcal{NP}$ because of the $\mathcal{NP}$-completeness of 3SAT.

Let $S = C_1 \wedge C_2 \wedge \ldots \wedge C_n$ be a 3SAT instance with variables $\mathcal{V}$ and let $\mathcal{C} = (\hat{C}_1, \ldots, \hat{C}_n)$ denote the arithmetization of S as described in Section 4.3. Observe that $\mathcal{C}$ contains at most four different types of polynomials. Namely, $P_1 := xyz, P_2 := xy(1-z), P_3 := x(1-y)(1-z)$ and $P_4 := (1-x)(1-y)(1-z)$. According to these four polynomials we partition the set of all clauses into sets $\mathcal{C}_j, j = 1,2,3,4$ such that $C \in \mathcal{C}_j$ if and only if $\hat{C}$ is of type P_j. Then – as already shown in Observation 4.3 – a function $W : \mathcal{V} \to \{0,1\}$ will be a satisfying assignment of S if and only if for all $j = 1,2,3,4$:

(6) $P_j(W(x), W(y), W(z)) = 0,$ whenever there exists a $C \in \mathcal{C}_j$ with $\hat{C} = P_j(x,y,z)$.

For $j = 1,2,3,4$ we define a function $\chi_j : \mathcal{V}^3 \to \{0,1\}$ with $\chi_j(x,y,z) = 1$ if there exists a clause $C \in \mathcal{C}_j$ with variables x, y and z such that $\hat{C}$ is of type P_j. Otherwise χ_j has value 0. Using the functions χ_j, we can reformulate condition (6) as follows:

$$\chi_j(x,y,z) \cdot P_j(W(x), W(y), W(z)) = 0 \qquad \text{for all } (x,y,z) \in \mathcal{V}^3. \tag{7}$$

Thus, the problem of verifying that W is a satisfying assignment for S is reduced to the task of checking that a certain function is identical zero. At this point Babai, Fortnow, Levin and Szegedy extended a test of Lund, Fortnow, Karloff and Nisan (which tests whether a certain sum is zero). This extension (as well as the original LFKN-test) is described in detail in the Appendix A. Here we only state the result.

Theorem A.2 *Let $f : F^m \to F$ be a polynomial of degree at most d in every variable with F being a finite field such that $|F| \geq 4m(d + |H|)$, where $H \subseteq F$ is an arbitrary subset of F. Then there exists a procedure* Extended Lfkn-Test, *which has access to f and an additional table T containing $\mathcal{O}(d|F|^{2m})$ values each of length $\lceil (m+1)\log|F| \rceil$, that has the following properties:*

- *If f satisfies the equation*

  $$f(u) = 0 \qquad \textit{for all } u \in H^m \tag{8}$$

  then there exists a table $T = T_f$ such that Extended Lfkn-Test *always accepts.*

- *If f does not satisfy equation* (8) *then* Extended Lfkn-Test *rejects with probability at least* $3/4$ *for all tables T.*

- Extended Lfkn-Test *queries f at only five points, randomly chosen from F^m, reads $\mathcal{O}(md)$ entries from T, and uses $\mathcal{O}(m \log|F|)$ random bits.*

In order to apply Theorem A.2 we interpret a truth assignment no longer as a vector of length $n = |\mathcal{V}|$ over $\mathbb{F}_2$, but instead we identify $\mathcal{V}$ with H^m and encode the low degree extension of a satisfying assignment – as indicated at the end of the previous section. More precisely, let p be the smallest prime such that $p \geq (\log n)^3$, let $F = \mathbb{F}_p$ and let $H \subseteq F$ be an arbitrary subset of order $|H| = \lceil \log n \rceil$. (Note that p can be computed in polynomial time and that $(\log n)^3 \leq p \leq 2(\log n)^3$.) Furthermore, let $m = \lceil \log n / \log\log n \rceil$ and observe that $|H|^m \geq n$. So we may identify $\mathcal{V}$ with H^m, adding some dummy variables, if necessary.

We then extend every truth assignment $W : H^m \to \{0,1\}$ and the functions $\chi_j : H^{3m} \to \{0,1\}$ introduced above to polynomials $W' : F^m \to F$ and $\chi'_j : F^{3m} \to F$ of degree at most $|H|$ in each variable. For every $j = 1,2,3,4$ the function $f_j(x,y,z) := \chi'_j(x,y,z) \cdot P_j(W'(x), W'(y), W'(z))$ is then a polynomial over F^{3m} of degree at most $4|H| = \mathcal{O}(\log n)$ in each variable.

With the help of Theorem A.2 the construction of the desired $(\log n, poly(\log n), poly(\log n))$-restricted solution verifier is now easily completed. The verifier interprets the solution s as $s = \widetilde{W}$, where $\widetilde{W}$ is considered as a function $\widetilde{W} : F^{3m} \to F$. Ideally, $\widetilde{W}$ corresponds to the low degree extension W' of a truth assignment $W : H^m \to \{0,1\}$. Using the Low Degree Test of the previous section the verifier checks first that $\widetilde{W}$ is at least δ-close to a polynomial of degree at most $|H|$ in every variable. Recall that in order to perform this test, the verifier needs to read only $\mathcal{O}(1)$ entries from $\widetilde{W}$, but requires access to an additional table T.

The verifier therefore interprets every proof π as $\pi = \widetilde{A}\widetilde{B}_1\widetilde{B}_2\widetilde{B}_3\widetilde{B}_4$, where $\widetilde{A}$ and $\widetilde{B}_i$ are the tables of Theorem 4.11 (with respect to the function $\widetilde{W}$) and of Theorem A.2 (with respect to the function f_i), respectively.

The verifier needs to check two things:

(A) that $\widetilde{W}$ is δ-close to a polynomial of degree at most $|H|$ in each variable and

(B) that condition (7) is satisfied.

Task (A) is achieved by the procedure Low Degree Test of the previous section (which queries $\widetilde{W}$ and $\widetilde{A}$). Task (B) on the other hand is achieved by using the procedure Extended LFKN-Test of Theorem A.2, which queries the functions $\widetilde{f}_j$ and the tables $\widetilde{B}_j$. The functions χ'_j depend only on the 3SAT instance S. Therefore these functions can be computed by the verifier. The five queries to the function $\widetilde{f}_j = \chi_j \cdot P_j(\widetilde{W}(\cdot), \widetilde{W}(\cdot), \widetilde{W}(\cdot))$ needed for the procedure Extended LFKN-Test can be replaced by fifteen queries to the function $\widetilde{W}$. By choosing the constant δ sufficiently small then if test (A) passes we know with high probability – say at least $3/4$ – that all these fifteen values of $\widetilde{W}$ and therefore also the five values of $\widetilde{f}_j$ are correct. Since the procedure Extended LFKN-Test finds an error in equation (7) with probability at least $3/4$ in total we have that the error probability of the solution verifier is at most $1/2$. By repeating the whole process twice this error probability becomes at most $1/4$ as required.

We still have to compute the resources consumed by the verifier. For applying the procedure Low Degree Test a constant number of times the verifier needs to read $\mathcal{O}(1)$ values from $\widetilde{W}$ of length $\mathcal{O}(\log|F|)$ and $\mathcal{O}(1)$ values of length $\mathcal{O}(m|H|\log|F|) = \mathcal{O}(poly(\log n))$ from the table $\widetilde{A}$. The total number of random bits needed by the verifier for the Low Degree Test is $\mathcal{O}(m \log|F|) = \mathcal{O}(\log n)$. For the Extended LFKN-Test the verifier has to read $m|H| = \mathcal{O}(poly(\log n))$ entries of length $\mathcal{O}(m\log|F|) = \mathcal{O}(\log n)$ from the tables $\widetilde{B}_j$. The number of random bits used for this test is again $\mathcal{O}(m\log|F|) = \mathcal{O}(\log n)$.

Therefore we have the following theorem.

Theorem 4.12 *Let E_1 denote the encoding scheme defined as follows. Given a vector $x \in \mathbb{F}_2^n$, interpret x as a function $x : H^m \to \mathbb{F}_p$, where p, m, and H are defined as above, and let*

$E_1(x) : \mathbb{F}_p^m \to \mathbb{F}_p$ be the low degree extension of x. Then

$$(R_{3\,\mathrm{SAT}}, E_1) \in \mathcal{PCS}(\log n, poly(\log n), poly(\log n)).$$

□

Corollary 4.13 $\mathcal{NP} \subseteq \mathcal{PCP}(\log n, poly(\log n))$.

In fact we have shown even more. Namely, recall that the procedures LOW DEGREE TEST and EXTENDED LFKN-TEST read only $\mathcal{O}(1)$ values from $\widetilde{W}$. So we can formulate the following slightly stronger version of Theorem A.2, which we will need in the next section.

Corollary 4.14 *Let E_1 denote the encoding scheme from Theorem 4.12. Then there exists a $(\log n, poly(\log n), poly(\log n))$-restricted solution verifier for $(R_{3\,\mathrm{SAT}}, E_1)$, which queries the solution string s only $\mathcal{O}(1)$ times.*

□

4.6.2 $(\mathbf{R}_{3\,\mathrm{SAT}}, E_1) \in \mathcal{PCS}(\log n, 1, poly(\log n))$

We will now improve the result of the last section as follows: the verifier is still allowed to read $poly(\log n)$ bits from a proof π but these bits are now required to be consecutive bits of π. The idea behind the proof of this result is quite simple: using the proof π that the $(\log n, poly(\log n), poly(\log n))$-restricted solution verifier V of Corollary 4.14 would read, we construct a new verifier $\hat{V}$ that uses a proof $\hat{\pi}$ which contains for every possible random string τ the sequence of $poly(\log n)$ bits that V would read from π on input x and random string τ. For the verification process the verifier $\hat{V}$ reads a consecutive sequence of $poly(\log n)$ bits from $\hat{\pi}$ that depend on τ and uses these bits to determine what the verifier V would have answered if it had received these bits as answers for his queries to π.

Theorem 4.15 *Let E_1 denote the encoding scheme as defined in Theorem 4.12. Then*

$$(R_{3\,\mathrm{SAT}}, E_1) \in \mathcal{PCS}(\log n, 1, poly(\log n)).$$

Proof. Let V be the $(\log n, poly(\log n), poly(\log n))$-restricted solution verifier from Corollary 4.14 that queries bits in a proof π of length $l = \mathcal{O}(poly(n))$. Let r be the smallest prime larger than $\log^2 l$ and let G be the field with r elements. We may assume that $\log l$ and $\log\log l$ are integers (otherwise elongate π by a suitable number of bits). Set $m := \lceil \log l / \log\log l \rceil$. Let I be a subset of G of size $\log l$. Then we may interpret the proof π as a function $\pi : I^m \to \{0,1\}$. Let $\pi' : G^m \to G$ denote the low degree extension of π which has degree at most $|I|$ in each variable. Let $q(n) = \mathcal{O}(poly(\log n))$ denote the number of bits that are queried by the verifier V from π. Then for $a_0, a_1, \ldots, a_{q(n)} \in G^m$ we define $P_{a_0,a_1,\ldots,a_{q(n)}} : G \to G^m$ to be the unique polynomial of degree $q(n)$ that interpolates the points $\{(i, a_i)\}_{i=0}^{q(n)}$.

The verifier $\hat{V}$ now assumes that the proof $\hat{\pi}$ consists of the low degree extension π', a table T needed for the LOW DEGREE TEST of π' and a table of the coefficients of all the polynomials

$\pi'(P_{a_0,\ldots,a_{q(n)}})$ where a_0 is an arbitrary value of G^m and $a_1,\ldots,a_{q(n)}$ are positions that can be queried by V.

The verifier $\hat{V}$ proceeds as follows: first it uses the procedure LOW DEGREE TEST to make sure that the tabulated values $\widetilde{\pi}'$ are δ-close to a polynomial of total degree $m|I|$. If this test passes then it determines which positions of π the verifier V would have queried for a random string τ. Let $a_1,\ldots,a_{q(n)}$ denote these positions. The verifier now chooses a random position a_0 and computes the polynomial $P_{a_0,a_1,\ldots,a_{q(n)}}$. It then queries the coefficients of the polynomial $p := \pi'(P_{a_0,\ldots,a_{q(n)}})$ from the proof $\hat{\pi}$ and checks that V would have accepted if the answers to its queries $a_1,\ldots,a_{q(n)}$ had been $p(1),\ldots,p(q(n))$. Finally, V chooses a random point $t \in G - \{0,\ldots,q(n)\}$ and tests whether $p(t) = \widetilde{\pi}(P_{a_0,\ldots,a_{q(n)}}(t))$.

We now show that this verification process enables the verifier $\hat{V}$ to detect whether $\widetilde{W}$ is a a proper encoding of a satisfying assignment of S with the desired probability.

If $\widetilde{W}$ is a proper encoding of a satisfying assignment of S then there exists a proof π such that the verifier V accepts with probability 1. Thus if $\hat{\pi}$ consists of π', the table T needed for the LOW DEGREE TEST and the correctly tabulated coefficients of the polynomials $\pi'(P_{a_0,\ldots,a_{q(n)}})$ then the verifier $\hat{V}$ accepts with probability 1.

Let us now suppose that $\widetilde{W}$ is not an proper encoding of a satisfying assignment of S. Then for an arbitrary proof π the verifier V accepts with probability at most $1/4$. The verifier $\hat{V}$ therefore only needs to detect that the values $p(1),\ldots,p(a_{q(n)})$ are wrong with probability at least $1/2$. Then for an arbitrary proof $\hat{\pi}$ the verifier $\hat{V}$ gives the correct answer with probability at least $1/4$ and by repeating the whole verification procedure five times this probability can be increased to $3/4$ as desired.

The final test $p(t) = \widetilde{\pi}'(P_{a_0,\ldots,a_{q(n)}}(t))$ assures that if π' is correctly tabulated then p equals $\pi'(P_{a_0,\ldots,a_{q(n)}})$ with probability $\geq 1 - m|I|/|G| \geq 3/4$. The point $P_{a_0,\ldots,a_{q(n)}}(t)$ is uniformly distributed over $|G|$ since a_0 and t are randomly chosen points from G. The low degree test of $\widetilde{\pi}'$ has shown that the tabulated values are δ-close to the low-degree extension π'. This implies that the right hand side of the final test will be correctly evaluated with probability $\geq 1/4$. Therefore if the values $p(1),\ldots,p(a_{q(n)})$ are not correct, this will be detected by the verifier with probability $\geq 1/2$ and, as already shown above, this suffices to detect with probability $\geq 3/4$ that $\widetilde{W}$ is not a proper encoding of a satisfying assignment of S.

We now show that the consumed resources are as stated in the theorem. For the application of the procedure LOW DEGREE TEST to the function $\widetilde{\pi}'$ the verifier needs to make $\mathcal{O}(1)$ queries to $\widetilde{\pi}'$ of size $\log|G| = \mathcal{O}(poly(\log n))$ and $\mathcal{O}(1)$ queries to the table T of size $\mathcal{O}(m|I|\log|G|) = \mathcal{O}(poly(\log n))$. Moreover the number of random bits needed for the LOW DEGREE TEST is $\mathcal{O}(m\log|G|) = \mathcal{O}(\log n)$. For generating the value $a_0 \in G^m$ the verifier needs $\mathcal{O}(m\log|G|) = \mathcal{O}(\log n)$ random bits. To query the coefficients of the polynomial p one query is made of length $\mathcal{O}(q(n)\log|G|) = \mathcal{O}(poly(\log n))$. Finally to generate the value $t \in G$ the verifier needs $\mathcal{O}(\log|G|) = \mathcal{O}(\log n)$ random bits and makes two additional queries to the proof $\hat{\pi}$ for the final test.

Thus in total $\mathcal{O}(1)$ queries of length at most $\mathcal{O}(poly(\log n))$ are made to the proof $\hat{\pi}$. The number of random bits needed for the whole verification process is $\mathcal{O}(\log n)$. □

4.7 Composing verifiers: recursive proof checking

In this section we conclude the proof of Theorem 2.2. A major tool for doing that is Lemma 4.18 which shows how two verifiers can be composed to form a new verifier which queries fewer bits.

In order to state and prove this lemma we need some technical prerequisites. Recall that so far we have used just two different encoding schemes. Namely, $E_0 : y \mapsto \{y^T z\}_{z \in \mathbb{F}_2^{|y|}}$ and the encoding E_1 of Theorem 4.12. For the composition of proof systems it will be more convenient to use slight modifications of these encodings. Informally speaking these are given by partitioning a given string y first into a (constant) number of substrings and then using the original encoding scheme to encode each of these substrings. For a formal definition let E be an arbitrary but fixed encoding scheme, $d \in \mathbb{N}$ a constant. We define a new encoding E' with respect to d by

$$E'(y) = (E(y_1), \ldots, E(y_d)), \tag{9}$$

where $y = y_1 \cdots y_d$ and $|y_1| = \ldots = |y_d|$.

It is not difficult to show that for E_0 and E_1 the dashed encoding schemes E_0' and E_1' behave like the original ones. (Note that formally, the encodings E_0' and E_1' are only defined for values of n divisible by d. As we may always assume that $\mathrm{R}_{3\,\mathrm{SAT}}$ consists only of those formulas whose number of variables satisfies such a condition this is only a technical restriction which we omit for ease of notation.)

Corollary 4.16 *For all constants $d \in \mathbb{N}$ one has*

$$(R_{3\,\mathrm{SAT}}, E_0') \in \mathcal{PCS}(n^3, 1, 1) \qquad \textit{and} \qquad (R_{3\,\mathrm{SAT}}, E_1') \in \mathcal{PCS}(\log n, 1, \mathit{poly}(\log n)).$$

Proof. We first show that $(\mathrm{R}_{3\,\mathrm{SAT}}, E_0') \in \mathcal{PCS}(n^3, 1, 1)$. Let V_0 be the $(n^3, 1, 1)$-restricted solution verifier from Corollary 4.10. (Recall, that with slight abuse of notation we assumed that n denotes the number of variables *and* the number of clauses.) We construct a new verifier V_0' as follows. V_0' interprets the solution as $s = (s_1, \ldots, s_d)$ and the proof as (e, π), where the s_i and e are viewed as functions $s_i : \mathbb{F}_2^{n/d} \to \mathbb{F}_2$ and $e : \mathbb{F}_2^n \to \mathbb{F}_2$. The new verifier V_0' first uses the old verifier V_0 to check whether e is close to an encoding $E_0(y')$ of a string y' and, if so, whether π is a proof that y' is a solution for x. If V_0 rejects then V_0' also rejects. If on the other hand V_0 accepts, then V_0' proceeds by checking whether s is close to an encoding $E_0'(y)$ for a string y and, if so, whether also $y = y'$.

These checks are performed by first using the tester LINEARITY TEST of Section 4.4 to verify that $s_1, \ldots, s_d$ are all close to homogeneous linear functions, and then calling the procedure CONSISTENCY TEST I, which verifies that $y = y'$.

<u>CONSISTENCY TEST I</u>

Pick $w_i \in_{\mathrm{R}} \mathbb{F}_2^{n/d}$ and let $w = w_1 \ldots w_d$,
verify that $e(w) = s_1(w_1) + \ldots + s_d(w_d)$.

By repeating LINEARITY TEST a constant number of times without failure we may assume (cf. Lemma 4.4), that with probability $1 - \frac{1}{8}$ each of the functions s_i is $\frac{1}{8d}$-close to homogeneous linear functions $y_i^T w$, for vectors $y_1, \ldots, y_d \in \mathbb{F}_2^{n/d}$. Let $y = y_1 \cdots y_d$. If $y' \neq y$, then $y'^T w \neq$

$y^T w$ for half of the vectors $w \in \mathbb{F}_2^n$. So in this case CONSISTENCY TEST I fails with probability at least $\frac{1}{2} - \frac{1}{4} - d \cdot \frac{1}{8d} = \frac{1}{8}$ (the probability of choosing a w so that $y'^T w \neq y^T w$ minus the probability that $e(w) \neq y^T w$ or at least one of the values $s_i(w_i)$ is different from that of the corresponding linear function). By calling CONSISTENCY TEST I a constant number of times the probability that no failure is reported even if y and y' are different can be reduced to less than $\frac{1}{8}$.

Combining these facts we observe that the probability that the verifier V_0' accepts when it shouldn't is bounded by $\frac{1}{2}$ (the sum of the probabilities that V_0 failed to reject a wrong solution or proof, that LINEARITY TEST failed to reject an erroneous function, and that CONSISTENCY TEST I failed to reject two unequal strings), so repeating V_0' twice gives the desired result.

To show that $(\mathrm{R}_{3\,\mathrm{SAT}}, E_1') \in \mathcal{PCS}(\log n, 1, \mathrm{poly}(\log n))$ we assume without loss of generality that n is such that $|H|^{m-1} = n/d$ and that the low degree extension a' of a satisfying assignment a is obtained by appending a sufficiently many zeros. If we let $\chi_j(x)$ denote the (unique) polynomial of degree $|H|$ such that $\chi_j(j) = 1$, while $\chi_j(h) = 0$ for all $h \in H \setminus j$ and write a as $a = a_1 \dots a_d$, then

$$a'(x, y) = \sum_{j=1}^{d} \chi_j(x) \cdot a_j'(y) \qquad \text{for all } x \in \mathbb{F}_p, y \in \mathbb{F}_p^{m-1} .$$

The rest of the proof follows the lines of the one above. The only difference being that the linearity tester LINEARITY TEST has to be replaced by the low degree tester LOW DEGREE TEST of Section 4.5, and that the consistency test between two strings y and y' is performed by the following procedure CONSISTENCY TEST II.

<u>CONSISTENCY TEST II</u>
Pick $x \in \mathbb{F}_p$ and $y \in \mathbb{F}_p^{m-1}$,
verify that $e(x, y) = \sum_{j=1}^{d} \chi_j(x) \cdot s_j(y)$.

The correctness follows similarly as above. (Use the fact that by the Lemma of SCHWARTZ two different (multivariate) polynomials of total degree d can coincide only in $d|\mathbb{F}_p|^{m-1}$ points.) We omit the details.

□

The key idea of recursive proof checking is to encode the verification procedure of one verfier as a Boolean formula. This will be done by using the following theorem of Cook, which he used in his famous proof of the $\mathcal{NP}$-completeness of satisfiability.

Theorem 4.17 (Cook [14]) *For every polynomial time Turing machine M there exist polynomials $p(n)$ and $q(n)$ and a Turing machine M' such that the following is true. For every natural number n the machine M' constructs in time $p(n)$ a 3SAT-formula F_M containing variables $y_1, \dots, y_n$ and $q(n)$ additional variables z_i so that there exists a satisfying assignment $y_1, \dots, y_n, z_1, \dots, z_{q(n)}$ for F_M with $y_i = x_i$, $i = 1, \dots, n$, if and only if M accepts the input $x = x_1 \cdots x_n$.*

□

Lemma 4.18 *Let $r_i(n), b_i(n)$ be positive functions. If $(R_{3\,\mathrm{SAT}}, \hat{E}) \in \mathcal{PCS}(r_1(n), 1, b_1(n))$ for some encoding scheme $\hat{E}$ and E is an encoding scheme such that for any constant $d \in \mathbb{N}$ the* *encoding E' given by equation (9) satisfies $(R_{3\,\mathrm{SAT}}, E') \in \mathcal{PCS}(r_2(n), 1, b_2(n))$, then*

$$(R_{3\,\mathrm{SAT}}, \hat{E}) \in \mathcal{PCS}(r_1(n) + r_2(poly(b_1(n))),\ 1,\ b_2(poly(b_1(n)))).$$

Proof. Let $L \in \mathcal{NP}$ be an arbitrary but fixed language, and let V_1 be an $(r_1(n), 1, b_1(n))$-restricted solution verifier showing that $(\mathrm{R}_{3\,\mathrm{SAT}}, \hat{E}) \in \mathcal{PCS}(r_1(n), 1, b_1(n))$, which uses $\hat{r}_1(n) = \mathcal{O}(r_1(n))$ many random bits and queries $d_1 \in \mathbb{N}$ many blocks of size $\hat{b}_1(n) = \mathcal{O}(b_1(n))$ from the proof. Every proof π for V_1 consists of, say, $\nu_1(n)$ many substrings of length $\hat{b}_1(n)$, that is $\pi = (\pi_1, \ldots, \pi_{\nu_1(n)})$, where without loss of generality $\nu_1(n) = d_1 \cdot 2^{\hat{r}_1(n)}$.

By Theorem 4.17 we know that for every fixed input x of length n and every fixed string τ of length $\hat{r}_1(n)$ there exists a 3SAT formula $F_{x,\tau}$ containing variables $q_{11}, \ldots, q_{d_1\,\hat{b}_1(n)}$ and $z_1, \ldots, z_{q(n)}$ such that V_1 accepts for input x, random string τ and queried blocks $q_1, \ldots, q_{d_1}$, where $q_i = q_{i1} \cdots q_{i\,\hat{b}_1(n)}$, if and only if there exists an assignment of $z_1, \ldots, z_{q(n)}$ such that $q_{11}, \ldots, q_{d_1\,\hat{b}_1(n)}, z_1, \ldots, z_{q(n)}$ is a satisfying assignment for $F_{x,\tau}$. Without loss of generality we assume in the following that $q(n) = \hat{b}_1(n)$.

By the second assumption there exists an $(r_2(n), 1, b_2(n))$-restricted solution verifier V_2 which, given the 3SAT formula $F_{x,\tau}$ as input, checks whether a solution given as $(s_1, \ldots, s_{d_1}, s)$ is $\frac{1}{4}$-close to an encoding

$$(E(q_{11} \cdots q_{1\,\hat{b}_1(n)}), \ldots, E(q_{d_1 1} \cdots q_{d_1\,\hat{b}_1(n)}), E(z_1 \cdots z_{\hat{b}_1(n)}))$$

of a satisfying assignment for $F_{x,\tau}$.

With these notations at hand, we are now able to define a new verifier V_{12} (which will be an $(r_1(n) + r_2(poly(b_1(n))),\ 1,\ b_2(poly(b_1(n))))$-restricted) as follows. V_{12} interprets a proof as

$$(e_1, \ldots, e_{\nu_1(n)}, f_1, \rho_1, f_2, \rho_2, \ldots).$$

Upon reading an input x of length n, the verifier V_{12} reads a string τ of random bits, $|\tau| = \hat{r}_1(n)$, and computes $F_{x,\tau}$. Then V_{12} computes the positions $r_1, \ldots, r_{d_1}$ which the verifier V_1 would have read on input x and random string τ, and calls V_2 with input $F_{x,\tau}$, solution $(e_{r_1}, \ldots, e_{r_{d_1}}, f_\tau)$ and proof ρ_τ. If V_2 outputs ACCEPT then V_{12} accepts x, otherwise V_{12} rejects.

To see the correctness of V_{12} assume first that $x \in L$. By assumption there exists a proof $\pi_x = (\pi_1^x, \ldots, \pi_{\nu_1(n)}^x)$ so that the verifier V_1 accepts for every random string τ. In particular this implies that for all such τ there exists an assignment of $z_1, \ldots, z_{\hat{b}_1(n)}$ and a proof ρ_τ so that verifier V_2 accepts the solution $(E(\pi_{r_1}^x), \ldots, E(\pi_{r_{d_1}}^x), E(z_1 \cdots z_{\hat{b}_1(n)}))$, if $r_1, \ldots, r_{d_1}$ denote the positions which V_1 reads in π_x for the random string τ. By letting

$$e_i = E(\pi_i^x), i = 1, \ldots, \nu_1(n), \quad \text{and} \quad f_\tau = E(z_1 \cdots z_{\hat{b}_1(n)}), \tau \in \mathbb{F}_2^{\hat{r}_1(n)}$$

we have thus constructed a proof for which V_{12} accepts with probability one.

Finally assume that $x \notin L$. The probability that for a given proof π V_{12} fails to reject, is obviously bounded by the sum of the probabilities that V_1 fails to reject plus the probability that V_2 fails to reject an unsatisfiable formula and/or an unsatisfying assignment. Hence,

$$\begin{aligned}
&\max_\pi \mathrm{Prob}_{\tau,\tau'}[V_{12}(x, \tau\,\tau', \pi) = \text{ACCEPT}] \\
&\leq\ \max_\pi \mathrm{Prob}_\tau[V_1(x, \tau, \pi) = \text{ACCEPT}] + \max_{x' : R_{3SAT}(x') = \emptyset}\ \max_{\pi'} \mathrm{Prob}_{\tau'}[V_2(x', \tau', \pi') = \text{ACCEPT}] \\
&<\ \tfrac{1}{4} + \tfrac{1}{4}\ =\ \tfrac{1}{2}.
\end{aligned}$$

□

Theorem 2.2 is now an easy consequence of Corollary 4.16 and Lemma 4.18.

Proof of Theorem 2.2. From Corollary 4.16 we observe that the requirements of Lemma 4.18 are satisfied with respect to the encoding scheme $\hat{E} = E = E_1$ and the functions $r_1(n) = r_2(n) = \log n$ and $b_1(n) = b_2(n) = poly(\log n)$. We therefore deduce that $(\mathrm{R}_{3\,\mathrm{SAT}}, E_1) \in \mathcal{PCS}(\log n, 1, poly(\log\log n))$.

Applying Lemma 4.18 once more, this time with respect to the encoding schemes $\hat{E} = E_1$ and $E = E_0$ the functions $r_1(n) = \log n$, $b_1(n) = poly(\log\log n)$, $r_2(n) = n^3$, and $b_2(n) = 1$, we obtain the desired result $\mathcal{NP} \in \mathcal{PCP}(\log n, 1)$.

□

4.8 A corollary: How to verify a theorem without even reading it

In this final section we state an immediate corollary of the proof presented above. Even though no consequences of this result are (yet?) known, it is quite interesting and surprising. Essentially it states that all languages in $\mathcal{NP}$ can be recognized by verifiers which only read a constant number of bits of their input! That is, the modification from the definition of $\mathcal{NP}$ by polynomial Turing machines to verifiers is fully symmetrized as indicated in Figure 4.

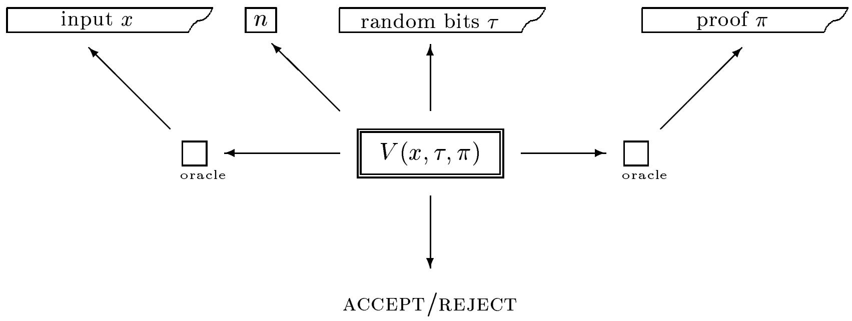

Figure 4: A verifier for probabilistically checkable languages

More formally, a *language verifier* is a polynomial-time Turing machine with access to a string τ of random bits, and a tape containing the length n of the input x. Furthermore the verifier has access to the input x and a membership proof π via oracles.

Definition 4.19 *A language $L \subseteq \Sigma^*$ is in $\mathcal{PCL}$ iff there exists an encoding scheme E, a language verifier V that uses at most $\mathcal{O}(\log n)$ random bits, and queries only $\mathcal{O}(1)$ bits from the input x and the proof π such that*

(i) For all $x \in L$ there exists a proof π_x such that

$$Prob_\tau[V(E(x), \tau, \pi_x) = \text{ACCEPT}] \;=\; 1,$$

(ii) for all $s \in \Sigma^*$ that are not $\frac{1}{4}$-close to the encoding $E(x)$ of an element $x \in L$ every

proof π satisfies

$$Prob_\tau[V(s,\tau,\pi) = \text{ACCEPT}] < \tfrac{1}{4}.$$

The idea of considering classes of languages which can be checked by reading only a small portion of an (encoded) input and of an appropriate membership proof goes back to Babai, Fortnow, Levin, and Szegedy [9]. They used the phrase probabilistic proof systems with *theorem-candidates* and *transparent proofs* for such an approach. Phillips and Safra [31] use the term *input-efficient verifier.* The following theorem is essentially contained in Arora, Lund, Motwani, Sudan, and Szegedy [4] and in Sudan [35].

Theorem 4.20 *The class of languages which can be recognized by verifiers by querying only a constant number of bits from an (encoded) input and a (polynomial sized) membership proof equals the class of languages recognizable by nondeterministic Turing machines. That is*

$$\mathcal{PCL} = \mathcal{NP}.$$

The proof of Theorem 4.20 hinges on the fact that in the proof of Theorem 2.2 we have in fact shown the following slightly stronger result.

Corollary 4.21 *There exists an encoding scheme E such that $(R_{3\,\mathrm{SAT}}, E) \in \mathcal{PCS}(\log n, 1, 1)$.*

□

Proof of Theorem 4.20. Let L be an arbitrary but fixed language in $\mathcal{NP}$. According to Theorem 4.17 there exists for every n a 3SAT formula F_n with variables $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_{poly(n)})$ such that for all $x \in \{0,1\}^n$:

$$x \in L \text{ if and only if there exits a } y \in \{0,1\}^{poly(n)} \text{ such that } F_n(x,y) \text{ is true.}$$

Now consider the encoding scheme E and the solution verifier V of Corollary 4.21. Considering the first part of the solution as input and the remaining part together with the old membership proof as a new membership proof obviously concludes the proof of Theorem 4.20.

□

A Appendix: LFKN-type tests

The aim of this section is to describe a method for verifying the truth of certain equations involving low degree polynomials. This method – called LFKN-test – was invented by LUND, FORTNOW, KARLOFF and NISAN to prove that any language in the polynomial hierarchy has an interactive proof system [27]. Their test also played a fundamental role in proving $\mathcal{IP} = \mathcal{PSPACE}$ [34] and $\mathcal{MIP} = \mathcal{NEXPTIME}$ [10]. In the first part we will describe this test in the form as it was used by LUND, FORTNOW, KARLOFF and NISAN to prove that the permanent of a square 0-1-matrix has an interactive proof system.

The second part is devoted to the description of an extended version of this test. The need of such an extension arose first in [10] and it was further extended in [9]. This extended version is an essential ingredient of the $\mathcal{NP} \subseteq \mathcal{PCS}(\log n, 1, \mathrm{polylog}\; n)$ proof of Section 4.6.

A.1 The LFKN-test for verifying large sums

The LFKN-test is based on the simple property that two different degree d polynomials defined over a domain F can agree in at most d points. Thus for a randomly chosen point $x \in F$ the probability that both polynomials agree at this point is at most $d/|F|$. By a lemma of SCHWARTZ [33] the same reasoning is also valid for multivariate polynomials: two different m-variate polynomials of (total) degree d over a field F agree in a randomly chosen point of F^m with probability $d/|F|$. Based on this property of low degree polynomials, the LFKN-test allows one to verify with high probability that a certain sum of values of a polynomial is zero. More precisely, let $f : F^m \to G$ be a polynomial of degree at most d in every variable, with F being a finite field and G a field extension of F. Furthermore, let $H \subseteq F$ be an arbitrary subset of F, and suppose we want to check that

$$\sum_{u \in H^m} f(u) = 0. \tag{10}$$

While this, of course, could easily be done in $\mathcal{O}(|H|^m)$ time by simply computing the sum, our goal here is to design a test procedure which evaluates f at only one (!) point.

To achieve this any such test obviously requires access to some additional information. Let $g_i : F^i \to G$ denote the following partial sums of the sum in (10):

$$\begin{array}{lcrl}
g_1(x_1) & = & \sum_{y_2\in H}\sum_{y_3\in H}\cdots\sum_{y_m\in H} & f(x_1, y_2, y_3, \ldots, y_m) \\
g_2(x_1, x_2) & = & \sum_{y_3\in H}\cdots\sum_{y_m\in H} & f(x_1, x_2, y_3, \ldots, y_m) \\
\vdots & & & \vdots \\
g_{m-1}(x_1, x_2, \ldots, x_{m-1}) & = & \sum_{y_m\in H} & f(x_1, x_2, \ldots, x_{m-1}, y_m) \\
g_m(x_1, x_2, \ldots, x_m) & = & & f(x_1, x_2, \ldots, x_m)
\end{array}$$

Note that g_i and g_{i+1} are related by the following equation:

$$\sum_{x \in H} g_{i+1}(x_1, x_2, \ldots, x_i, x) = g_i(x_1, x_2, \ldots, x_i) \qquad \text{for all } i = 1, \ldots, m-1. \tag{11}$$

In particular, the sum we are interested in satisfies:

$$\sum_{x \in H^m} f(x) = \sum_{x_1 \in H} g_1(x_1). \tag{12}$$

So, with the help of a function $\tilde{g}_1$ that is supposed to equal g_1 the task of checking equation (10) could be reduced to just $\mathcal{O}(|H|)$ queries to the function $\tilde{g}_1$ (instead of $\mathcal{O}(|H|^m)$ queries to the function f) — leaving open, however, how to verify that $\tilde{g}_1$ indeed equals g_1. At this point the fact that f is a polynomial of degree d in each variable turns out to be extremely useful.

Suppose the tester has access to functions $\tilde{g}_i : F^i \to G$ which are supposed to be the polynomials g_i defined above and assume for the moment that the functions $\tilde{g}_i$ are in fact polynomials of degree d in every variable. The consistency check then would be straightforward. For example, checking whether g_m and f are consistent (that is, identical) can be done by verifying that $f(r_1, r_2, \ldots, r_m) = \hat{g}_m(r_1, r_2, \ldots, r_m)$ at a single point $(r_1, \ldots, r_m)$ randomly chosen from F^m, the error probability being at most $md/|F|$, which is small if the cardinality of the field F is sufficiently large compared to the degree and the number of variables of f. Similarly, checking whether $\tilde{g}_i$ and $\tilde{g}_{i-1}$ are consistent (that is, satisfy relation (11)) can be done by evaluating $\tilde{g}_{i-1}$ at a randomly chosen point in F^{i-1} and comparing it to the corresponding sum of $|H|$ values of $\tilde{g}_i$.

While in principle one could use the low degree tester from Section 4.5 to assure that the functions $\tilde{g}_i$ are δ-close to polynomials, here a much easier trick works. As we will argue in more detail below, the consistency test outlined above can be modified such that it uses only that the functions $\tilde{g}_i$ are polynomials of degree at most d in the *last* variable. So in order to avoid a low degree test, we simply assume that the tester has access to a table T containing $\sum_{i=1}^{m}(d+1)|F|^{i-1}$ entries of size $\lceil \log |G| \rceil$, which is supposed to contain the coefficients of all degree d polynomials $g_i(x_1, \ldots, x_{i-1}, \cdot)$ for all $x_1, \ldots, x_{i-1} \in F$ and all $1 \leq i \leq m$. With this notation the LFKN-test can be formally stated as follows:

<u>LFKN-TEST</u>
Choose $r_1, \ldots, r_m \in_{\mathrm{R}} F$ and let $\hat{g}_0 := 0$.
for $i := 1$ to m do
  · Let $\hat{g}_i$ denote the degree d polynomial defined by the $d+1$ values contained in T as the coefficients of $g_i(r_1, \ldots, r_{i-1}, \cdot)$.
  · if $\sum_{x \in H} \hat{g}_i(x) \neq \hat{g}_{i-1}(r_{i-1})$ then REJECT
if $f(r_1, \ldots, r_m) \neq \hat{g}_m(r_m)$ then REJECT

The following theorem shows that the procedure LFKN-TEST does indeed have the desired properties.

Theorem A.1 *Let $f : F^m \to G$ be a polynomial of degree at most d in every variable with F and G being finite fields such that $|F| \geq 4md$ and G is a field extension of F. Furthermore, let $H \subseteq F$ denote an arbitrary subset of F and let T be a table containing $\sum_{i=1}^{m}(d+1)|F|^{i-1}$ values of length $\lceil \log |G| \rceil$. Then the procedure* LFKN-TEST *has the following properties:*

- *If f satisfies equation* (10) *then there exists a table $T = T_f$ such that the procedure* LFKN-TEST *always accepts.*
- *If f does not satisfy equation* (10) *then* LFKN-TEST *rejects with probability at least* 3/4 *for all tables T.*

- LFKN-TEST *queries f at only one point, distributed uniformly over F^m, reads $m(d+1)$ entries from T, and uses $\mathcal{O}(m \log |F|)$ random bits.*

Proof. Suppose first that equation (10) holds and assume that T_f contains the coefficients of the functions $g_i(x_1, \ldots, x_{i-1}, \cdot)$ defined above. Then $\hat{g}_i(x)$ equals $g_i(r_1, r_2, \ldots, r_{i-1}, x)$ and therefore equation (11) guarantees that no error will be detected.

Now suppose that equation (10) does not hold. If $g_1 = \hat{g}_1$ then the test $\sum_{x \in H} \hat{g}_1(x) = \hat{g}_0$ detects the error in equation (10). So assume $g_1 \neq \hat{g}_1$. Then $g_1(r_1) \neq \hat{g}_1(r_1)$ with probability $\geq 1 - d/|F|$, as two different degree d polynomials can agree in at most d points. If $g_1(r_1) \neq \hat{g}_1(r_1)$ and $g_2(r_1, \cdot) = \hat{g}_2(\cdot)$ then $\sum_{x \in H} \hat{g}_2(x) = g_1(r_1)$ and the test $\sum_{x \in H} \hat{g}_2(x) = \hat{g}_1(r_1)$ reveals an error. So we may assume that $g_2(r_1, \cdot) \neq \hat{g}_2(\cdot)$ for all r_1 such that $g_1(r_1) \neq \hat{g}_1(r_1)$.

Iterating this argument another $m-2$ times we observe that with probability at least $(1 - d/|F|)^{m-1}$ we may assume that either one of the tests $\sum_{x \in H} \hat{g}_i(x) = \hat{g}_{i-1}(r_{i-1})$ detects an error or $\hat{g}_m(\cdot) \neq g_m(r_1, \ldots, r_{m-1}, \cdot)$. But in the latter case the final test $f(r_1, \ldots, r_m) \neq \hat{g}_m(r_m)$ fails with probability $1 - d/|F|$. Summarizing, this shows that the procedure LFKN-TEST rejects with probability at least $(1 - d/|F|)^m \geq 1 - md/|F|$. As by assumption $|F| \geq 4md$, this probability is $\geq 3/4$ as claimed.

Finally, we have to ensure that the resources consumed by the procedure are as stated in the theorem. For generating the random points $r_1, r_2, \ldots, r_m \in F$ one needs $\mathcal{O}(m \log |F|)$ random bits. Each iteration of the for-loop requires $d+1$ queries to the table T, while the function f itself is evaluated only in the final test at exactly one point distributed uniformly over F^m.

□

A.2 The LFKN-test for everywhere vanishing functions

Let $f : F^m \to F$ be a polynomial of degree at most d in every variable, where F again denotes a finite field. Instead of checking a sum to be zero we are now interested in knowing that the function f is identically zero at a certain part of its domain:

$$f(u) = 0 \qquad \text{for all } u \in H^m \tag{13}$$

with H being an arbitrary subset of F. If we were working over a field like $\mathbb{R}$ then (13) would be equivalent to checking whether the sum $\sum_{u \in H^m} f(u)^2$ is zero. Thus (13) could be reduced to an application of Theorem A.1 for the polynomial f^2. For finite fields, however, this simple approach does not work. With some more effort it is nevertheless possible to reduce the test of (13) to a constant number of applications of Theorem A.1.

Theorem A.2 *Let $f : F^m \to F$ be a polynomial of degree at most d in every variable with F being a finite field such that $|F| \geq 4m(d + |H|)$, where $H \subseteq F$ is an arbitrary subset of F. Then there exists a procedure* EXTENDED LFKN-TEST, *which has access to f and an additional table T containing $\mathcal{O}(d|F|^{2m})$ values of length $\lceil (m+1) \log |F| \rceil$, that has the following properties:*

- *If f satisfies equation* (13) *then there exists a table $T = T_f$ such that* EXTENDED LFKN-TEST *always accepts.*

- *If f does not satisfy equation* (13) *then* Extended LFKN-Test *rejects with probability at least* $3/4$ *for all tables* T.
- Extended LFKN-Test *queries f at only five points, randomly chosen from* F^m, *reads* $\mathcal{O}(md)$ *entries from* T, *and uses* $\mathcal{O}(m \log |F|)$ *random bits.*

Proof. Let K be a field extension of F satisfying $2|H|^m \leq |K| \leq 2|H|^m|F|$ and let $\rho : H \rightarrow \{0, 1, \ldots, |H| - 1\}$ be an arbitrary bijection. Then for $u = (u_0, \ldots, u_{m-1}) \in H^m$ we define a mapping $\sigma : H^m \rightarrow \{0, 1, \ldots, |H|^m - 1\}$ by setting $\sigma(u) := \sum_{i=0}^{m-1} |H|^i \cdot \rho(u_i)$. Note that σ again is a bijection.

Now the function $g : K \rightarrow K$ defined as $g(t) := \sum_{u \in H^m} f(u) \cdot t^{\sigma(u)}$ is a polynomial of degree at most $|H|^m$. Thus either (13) holds or g has at most $|H|^m$ zeroes. As $|K| \geq 2|H|^m$, we therefore have:

$$\text{(14)} \qquad \text{if} \quad \text{(13) does not hold} \quad \text{then} \quad \mathrm{Prob}_{t \in K}[g(t) = 0] \leq \frac{1}{2}.$$

In particular, if g is different from the zero-function on K then the probability that the function g is zero at five random points from K is at most $(1/2)^5 = 1/32$. Even if g can be evaluated correctly only with probability $3/4$ then this probability is still less than $(3/4)^5 < 1/4$. Thus the probability that f is different from the zero-function and all the five queries of g yield the value zero is less than $1/4$ and a so-constructed tester would detect an error with the required probability.

We want to use Theorem A.1 to test whether g evaluated at a random point $t \in K$ is zero. Therefore we have to check that all the requirements of Theorem A.1 are fulfilled. First we show that the function $f(u)t^{\sigma(u)}$ is a polynomial in u. This can be seen as follows for all $u \in H^m$.

$$\begin{aligned} f(u) \cdot t^{\sigma(u)} &= f(u) \cdot \prod_{i=0}^{m-1} t^{|H|^i \rho(u_i)} \\ &= f(u) \cdot \prod_{i=0}^{m-1} \left(\sum_{h \in H} t^{|H|^i \rho(h)} \cdot L_h(u_i) \right) \end{aligned}$$

where $L_h(x)$ is defined to have value 1 if $x = h$ and 0 otherwise. Thus viewing $f(u)t^{\sigma(u)}$ as a function from F^m to K this shows that it is a polynomial of degree $d + |H|$ in every variable u_i.

By assumption the field F has size at least $4m(d + |H|)$ and thus satisfies the condition from Theorem A.1. The value of the function $f(u)t^{\sigma(u)}$ needed in Theorem A.1 can easily be computed from the function f. Thus we may use the procedure LFKN-Test to verify that the function g is zero at five randomly chosen points $\in K$. As already shown above this procedure will find out whether (13) holds with probability at least $1/4$.

The table T has to contain the tables from Theorem A.1 for the functions $f(u) \cdot t^{\sigma(u)}$ for every $t \in K$. As $|K| \leq 2|H|^m|F|$, it therefore contains at most $2|H|^m|F|m(d+1)|F|^{m-1} = \mathcal{O}(d|F|^{2m})$ many entries of size, say, $\lceil (m+1) \log |F| \rceil$.

Now we still have to count the resources used by the so-constructed procedure EXTENDED LFKN-TEST. For generating the five random points from K the verifier needs $\mathcal{O}(\log |K|) = \mathcal{O}(m \log |F|)$ random bits. All other resources are consumed by the procedure LFKN-TEST which is called five times. Thus in total $\mathcal{O}(m \log |F|)$ random bits are used and $m(d + |H|)$ entries are queried from T.

$\square$

B Appendix: Low degree tests

In this appendix we describe an efficient test for checking whether a given function is δ-close to a polynomial of degree at most d. We start by illustrating the main ideas of such testers at the simplified problem of checking whether a given function is δ-close to a homogeneous linear function.

B.1 The linear case

Lemma B.1 *Let $\delta < \frac{1}{3}$ be a constant and let $\tilde{g} : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function such that*

$$\mathit{Prob}_{x,y}[\tilde{g}(x) + \tilde{g}(y) \neq \tilde{g}(x+y)] \leq \frac{\delta}{2}.$$

Then there exists $h \in \mathbb{F}_2^n$ so that the functions $g(x) = h^T x$ and $\tilde{g}$ are δ-close.

Proof. The proof is by construction. We will show that the function g given by

$$g(x) := \text{majority}_y\{\tilde{g}(x+y) - \tilde{g}(y)\}$$

(solve ties arbitrarily) has the desired properties. First we show that g satisfies

$$p_a := \text{Prob}_x[g(a) = \tilde{g}(a+x) - \tilde{g}(x)] \geq 1 - \delta \qquad \text{for all } a \in \mathbb{F}_2^n\ . \tag{15}$$

Fix $a \in \mathbb{F}_2^n$. By assumption

$$\begin{aligned} \text{Prob}_{x,y}[\tilde{g}(x+a) + \tilde{g}(y) \neq \tilde{g}(x+a+y)] &\leq \frac{\delta}{2}, \qquad \text{and} \\ \text{Prob}_{x,y}[\tilde{g}(x) + \tilde{g}(y+a) \neq \tilde{g}(x+y+a)] &\leq \frac{\delta}{2}. \end{aligned}$$

Recalling that by definition $p_a \geq \frac{1}{2}$, the above inequalities easily imply (15):

$$\begin{aligned} 1-\delta &\leq \text{Prob}_{x,y}[\tilde{g}(x+a) + \tilde{g}(y) = \tilde{g}(x) + \tilde{g}(y+a)] = \sum_{z \in \mathbb{F}_2} (\text{Prob}_x[\tilde{g}(x+a) - \tilde{g}(x) = z])^2 \\ &= p_a^2 + (1-p_a)^2 \leq p_a(p_a + (1-p_a)) = p_a. \end{aligned}$$

Next we show that g and $\tilde{g}$ are δ-close. Assume not. Then $\text{Prob}_x[\tilde{g}(x) \neq g(x)] > \delta$. By the definition of g we also have $\text{Prob}_y[g(x) = \tilde{g}(x+y) - \tilde{g}(y)] \geq \frac{1}{2}$. As both events are independent this implies $\text{Prob}_{x,y}[\tilde{g}(x) \neq \tilde{g}(x+y) - \tilde{g}(y)] > \frac{\delta}{2}$, contradicting the assumption of the lemma.

To show that g satisfies the desired linearity condition, fix $a, b \in \mathbb{F}_2^n$ and apply (15) three times to obtain

$$\begin{aligned} &\text{Prob}_x[g(a) + g(b) + \tilde{g}(x) \neq \tilde{g}(a+x) + g(b)] \leq \delta, \\ &\text{Prob}_x[g(b) + \tilde{g}(a+x) \neq \tilde{g}(b+a+x)] \leq \delta, \qquad \text{and} \\ &\text{Prob}_x[\tilde{g}(a+b+x) \neq g(a+b) + \tilde{g}(x)] \leq \delta. \end{aligned}$$

Hence,

(16) $$\text{Prob}_x[g(a) + g(b) + \tilde{g}(x) = g(a+b) + \tilde{g}(x)] \geq 1 - 3\delta > 0.$$

As the condition on the left hand side of (16) is independent of x, the probability is either 0 or 1. The positivity therefore implies that $g(a) + g(b) = g(a+b)$ for all $a, b \in \mathbb{F}_2^n$, from which the existence of the desired vector $h \in \mathbb{F}_2^n$ follows immediately. □

B.2 The general case

The reader is invited to observe that the proof of Lemma B.1 easily generalizes to arbitrary finite fields F instead of $\mathbb{F}_2$. For $F = \mathbb{F}_p$, p a prime, this approach can even be generalized to arbitrary, multivariate degree d polynomials. For this one needs the well known fact (cf. e.g. VAN DER WAERDEN [36]) that a function $f : \mathbb{F}_p^m \to \mathbb{F}_p$ is a polynomial of total degree d if and only if for all $x, h \in \mathbb{F}_p^m$ the function satisfies $\sum_{t=0}^{d+1} (-1)^{t+1} \binom{d+1}{t} f(x+th) = 0$. Such a tester was first described in GEMMELL, LIPTON, RUBINFELD, SUDAN, and WIGDERSON [19]. It required $O(d^2)$ tests of the form "*Pick* $x, h \in \mathbb{F}_p^m$ *at random and verify that* $\sum_{t=0}^{d+1} (-1)^{t+1} \binom{d+1}{t} f(x+th) = 0$", adding up to a total of $O(d^3)$ evaluations of f. This bound was later improved to $O(d^2)$ by RUBINFELD and SUDAN [32] by a slight modification of the tester and an improved analysis.

While obviously $O(d)$ evaluations are a natural lower bound for any such tester, a breakthrough occurred with the results of ARORA and SAFRA [5] who showed that in fact $O(1)$ probes of f suffice, if the tester has also access to an appropriate additional function, which it may probe $O(d)$ times.

In this section we describe a strengthening of the ARORA, SAFRA tester due to ARORA, LUND, MOTWANI, SUDAN, and SZEGEDY [4] which requires only $O(1)$ probes of f *and* the additional function.

Theorem B.2 *Let* $0 < \delta < \frac{1}{10512}$ *and* $m \in \mathbb{N}$ *be constants and* p *a prime. Let* $\tilde{g} : \mathbb{F}_p^m \to \mathbb{F}_p$ *be a function, and let* $T : \mathbb{F}_p^m \times \mathbb{F}_p^m \to \mathbb{F}_p^{d+1}$ *be a function such that the degree* d *polynomials* $\tilde{P}_{x,h}$ *over* $\mathbb{F}_p$ *given by* $\tilde{P}_{x,h}(t) = \sum_{i=1}^{d+1} T(x,h)_i \cdot t^{i-1}$ *satisfy*

(17) $$Prob_{x,h,t}[\tilde{P}_{x,h}(t) = \tilde{g}(x+th)] \geq 1 - \frac{\delta}{8}.$$

Then there exists a (unique) polynomial $g : \mathbb{F}_p^m \to \mathbb{F}_p$ *of total degree* d *so that* g *and* $\tilde{g}$ *are* δ*-close.*

To the best of our knowledge there does not yet exist a published proof of Theorem B.2 in the literature. The proof we are presenting here is based on the one in ARORA, LUND, MOTWANI, SUDAN, SZEGEDY and SUDAN [35] [3] and SUDAN [35]. However, here we also include the details left out in that version and correct several minor errors.

The structure of the proof of Theorem B.2 is similar to the one of Lemma B.1. That is, we define the function g explicitly, and show subsequently that g is well-defined, is δ-close to $\tilde{g}$, and is a polynomial of total degree at most d. An essential tool in the proof is the following technical lemma from [5], known as the *Matrix Transposition Lemma.*

Lemma B.3 *Let $0 < \epsilon < \frac{1}{8}$ and $d \in I\!N$ be constants and let F be a finite field such that $|F| \geq 80\,d^3$. If r_s, $s \in F$, and c_t, $t \in F$, are polynomials over F of degree at most d and $A = (a_{st})$ is an $|F| \times |F|$ matrix such that*

$$Prob_{s,t}[a_{st} = r_s(t) \quad and \quad a_{st} = c_t(s)] \geq 1 - \epsilon.$$

Then there exists a bivariate polynomial $Q : F^2 \to F$ of degree at most d in each variable such that the sets $S = \{s \in F \mid r_s(\cdot) = Q(s,\cdot)\}$ and $T = \{t \in F \mid c_t(\cdot) = Q(\cdot,t)\}$ satisfy

$$|S| \geq (1-5\epsilon)|F| \qquad and \qquad |T| \geq (1-5\epsilon)|F|.$$

Proof. Call a pair (s,t) an *error-point*, if $r_s(t) \neq a_{st}$ or $c_t(s) \neq a_{st}$. By our assumption we know that there exist at most $\epsilon|F|^2$ error-points. A straightforward average argument shows that this implies the existence of sets $C \subseteq F$ and $R \subseteq F$ of size $|C| = 4d$ and $|R| = 4 \cdot \lceil \frac{1}{8}|F| \rceil$ such that for each $s \in R$ there exist at most d many t's in C for which (s,t) is an error-point. (Indeed, assume such sets C and R would not exist. Then for every set $C \subseteq F$ of size $4d$ the columns indexed by C would contain at least $\frac{1}{2}|F|(d+1)$ error points. As every error-point is counted at most $\binom{|F|}{4d-1}$ times, this would imply $\binom{|F|}{4d} \cdot \frac{1}{2}|F|(d+1) \leq \binom{|F|}{4d-1} \cdot \epsilon|F|^2$, contradicting the assumptions of the lemma.)

Assume for the moment, that $e_s \not\equiv 0$ and f_s are polynomials of degree at most d and $2d$, respectively, satisfying

$$c_t(s) \cdot e_s(t) = f_s(t) \qquad \text{for all } s \in R,\ t \in C. \tag{18}$$

We claim that this implies

$$c_t(s) \cdot e_s(t) = r_s(t) \cdot e_s(t) \qquad \text{for all } s \in R,\ t \in C.$$

Indeed, by choice of C and R there exist univariate polynomials $e'_s(t) \not\equiv 0$, $s \in R$ of degree at most d such that

$$c_t(s) \cdot e'_s(t) = r_s(t) \cdot e'_s(t) \qquad \text{for all } s \in R,\ t \in C.$$

For all $t \in C$ such that $e'_s(t) \neq 0$ we easily obtain $f_s(t) = e_s(t) \cdot r_s(t)$. As both sides are polynomials of degree at most $2d$, this implies that both sides are in fact identical for all $t \in C$.

Our goal is to define the polynomials e_s and f_s satisfying (18) in such a way that the number of pairs $(s,t) \in R \times C$ with $e_s(t) = 0$ is at most $\frac{1}{2}|R|$. Observe that this would imply that at least $\frac{1}{2}|R| = \frac{1}{4}|F|$ of the rows indexed by R contain no error-points within the columns indexed by C. In particular this would imply that for any $d+1$ elements $t_0, \ldots, t_d \in C$

$$Q(s,t) = \sum_{i=0}^{d} L_{t_i}(t) \cdot c_{t_i}(s)$$

forms the desired polynomial, where $L_{t_i}(t)$ denotes the unique degree d polynomial which is 1 if $t = t_i$ and 0 if $t \in \{t_0, \ldots, t_d\} \setminus \{t_i\}$. To see this, observe that by definition of Q one has $Q(\cdot,t) = c_t(\cdot)$ for all $t \in \{t_0, \ldots, t_d\}$. As a degree d polynomial is determined by specifying the values at $d+1$ points, this implies that also $Q(s,\cdot) = r_s(\cdot)$ for all rows $s \in R$ which contain no error-point within the columns indexed by C. This in turn shows that a column t either belongs to the set T or contains at least $\frac{1}{2}|R| - d = \frac{1}{4}|F| - d$ error-points. As the total number of error-points is bounded by $\epsilon|F|^2$ this immediately implies that $|T| \geq (1-5\epsilon)|F|$. By the same reasoning this inequality in turn also implies that $|S| \geq (1-5\epsilon)|F|$.

Let $e_s(t) = \sum_{i=0}^{d} e_{si}t^i$, $f_s(t) = \sum_{i=0}^{2d} f_{si}t^i$ and let $g(s) = (e_{s0}, \ldots, e_{sd}, f_{s0}, \ldots, f_{s\,2d})$. Then (18) can be rewritten as $M(s) \cdot g(s) = 0$ for all $s \in R$, where M is a matrix of dimension $4d \times 3d+2$ whose entries are degree d polynomials in s. Let k be the largest integer for which there exists an $s_0 \in R$ and a $k \times k$ submatrix of $M(s_0)$ which is non-singular. Let M' denote the corresponding submatrix of M and set the e_{si} and f_{si} not corresponding to columns of M' to 1. Then there exist k degree d polynomials $h_1, \ldots, h_k$ such that every solution of $M'(s) \cdot g'(s) = (h_1(s), \ldots, h_k(s))$ can be extended to a solution of $M(s) \cdot g(s) = 0$. Solve the former system by Cramer's rule to obtain that the $g'_i(s)$ are rational functions with common denominator $det(M'(s))$, which is a polynomial of degree at most kd. That is, $M(s) \cdot g(s) = 0$ has a solution in which all the e_{si} are polynomials of degree at most kd. Hence, for every $t \in C$, $e_s(t)$ is zero for at most kd s-values. By assumption $5kd^2 \leq 20d^3 \leq \frac{1}{4}|F| = \frac{1}{2}|R|$. This concludes the proof of the lemma.

□

As a corollary we obtain that for families of matrices (A_i) and polynomials $(r_s(t)_i)$ and $(c_t(s)_i)$, which satisfy the assumptions of Lemma B.3 with high probability, the two degree d polynomials best fitting a fixed row and a fixed column coincide with high probability at their intersection.

Corollary B.4 *Let $\delta > 0$ and $d, m \in \mathbb{N}$ be constants, let F be a finite field such that $|F| \geq 80\,d^3$, and let $s_0, t_0 \in F$ be fixed. Assume that for every pair $h_1, h_2 \in F^m$ there exist matrices $A = (a_{st})$ and degree d polynomials r_s and c_t over F such that the sets*

$$\tilde{S} := \{s \in F \mid \; Prob_{h_1,h_2}[r_s(t) = a_{st}] \geq 1 - \frac{\delta}{2} \quad \text{for all } t \in F\},$$

and

$$\tilde{T} := \{t \in F \mid \; Prob_{h_1,h_2}[c_t(s) = a_{st}] \geq 1 - \frac{\delta}{2} \quad \text{for all } s \in F\}$$

satisfy

$$|\tilde{S}| \geq (1-\delta)|F| \quad \text{and} \quad |\tilde{T}| \geq (1-\delta)|F|.$$

Let P_{row} denote the degree d polynomial best fitting the s_0-th row, i.e. the degree d polynomial that maximizes the number of $t \in F$ for which $P_{row}(t) = a_{s_0 t}$, and similarly let P_{col} denote the degree d polynomial best fitting the t_0-th column. Then

$$Prob_{h_1,h_2}[P_{row}(t_0) = P_{col}(s_0)] \geq 1 - 72\,\delta.$$

(In order to simplify notation, the dependence of A, r_s, c_t, P_{row} and P_{col} on h_1 and h_2 is not reflected in the notation. It will, however, always be clear from the context.)

Proof. Let $\epsilon = \frac{1}{12}$ and observe that by assumption

$$\mathrm{Prob}_{h_1,h_2}[\mathrm{Prob}_{s,t}[r_s(t) = a_{st} \text{ and } c_t(s) = a_{st}] \geq 1 - \epsilon] \geq 1 - \frac{3\delta}{\epsilon}.$$

Thus with probability $1 - \frac{3\delta}{\epsilon}$ the matrix A satisfies the requirements of Lemma B.3. In the following let S, T and Q be as defined in Lemma B.3. (For completeness, if A does not satisfy the requirements of Lemma B.3 let $S = T = \emptyset$ and $Q \equiv 0$.) Then

$$\text{Prob}_{h_1,h_2}[|S| \geq (1-5\epsilon)|F| \text{ and } |T| \geq (1-5\epsilon)|F|] \geq 1 - \frac{3\delta}{\epsilon}. \tag{19}$$

Applying the assumption of the corollary for $t = t_0$ and $s = s_0$, respectively, we obtain

$$\text{Prob}_{h_1,h_2}[r_s(t_0) = a_{st_0}] \geq 1 - \frac{\delta}{2} \qquad \text{for all } s \in \tilde{S}$$

and

$$\text{Prob}_{h_1,h_2}[c_t(s_0) = a_{s_0t}] \geq 1 - \frac{\delta}{2} \qquad \text{for all } t \in \tilde{T}.$$

In particular the sets

$$\hat{S} := \{s \mid r_s(t_0) = a_{st_0}\} \quad \text{and} \quad \hat{T} := \{t \mid c_t(s_0) = a_{s_0t}\}$$

therefore satisfy

$$\text{Prob}_{h_1,h_2}[|\hat{S}| \geq (1-\epsilon)|F| \text{ and } |\hat{T}| \geq (1-\epsilon)|F|] \geq 1 - \frac{3\delta}{\epsilon}. \tag{20}$$

Combining (19) and (20) we deduce that

$$\text{Prob}_{h_1,h_2}[|S \cap \hat{S}| \geq (1-6\epsilon)|F| \text{ and } |T \cap \hat{T}| \geq (1-6\epsilon)|F|] \geq 1 - \frac{6\delta}{\epsilon}.$$

As $|S \cap \hat{S}| \geq \frac{1}{2}|F|$ implies that $Q(\cdot, t_0)$ is the best polynomial fitting the t_0-th column, and similarly $|T \cap \hat{T}| \geq \frac{1}{2}|F|$ implies that $Q(s_0, \cdot)$ is the best polynomial fitting the s_0-th row, this together with the choice of ϵ concludes the proof of Corollary B.4.

□

Proof of Theorem B.2. For every $x, h \in \mathbb{F}_p^m$ let $P_{x,h}$ denote a polynomial of total degree at most d which maximizes the number of points from $\{x + th \mid t \in \mathbb{F}_p\}$ at which $P_{x,h}$ and $\tilde{g}$ agree. The function g is then given by (break ties arbitrarily)

$$g(x) = \text{majority}_h\{P_{x,h}(0)\}.$$

For the proof that g has the desired properties, we first observe that assumption (17) on the polynomials $\tilde{P}_{x,h}$ induces a similar property on the polynomials $P_{x,h}$. More precisely, we claim that

$$\text{Prob}_{x,h}[P_{x,h}(t) = \tilde{g}(x + th)] \geq 1 - \frac{\delta}{2} \qquad \text{for all } t \in \mathbb{F}_p\ . \tag{21}$$

For the proof of (21) fix $t \in \mathbb{F}_p$ and observe that (17) implies that

$$\text{Prob}_{x,h}[\text{Prob}_t[\tilde{P}_{x,h}(t) = \tilde{g}(x + th)] \geq \frac{2}{3}] \geq 1 - \frac{3\delta}{8}$$

As $\text{Prob}_t[\tilde{P}_{x,h}(t) = \tilde{g}(x + th)] \geq \frac{2}{3}$ can only hold if $\tilde{P}_{x,h} = P_{x,h}$, this implies that

$$\text{Prob}_{x,h}[\tilde{P}_{x,h} = P_{x,h}] \geq 1 - \frac{3\delta}{8}$$

and therefore together with (17) also

$$\mathrm{Prob}_{x,h,t'}[P_{x,h}(t+t') = \tilde{P}_{x,h}(t+t') = \tilde{g}(x+(t+t')h)] \geq 1 - \frac{\delta}{2}.$$

By definition, $P_{x,h}(t+t') = P_{x+ht',h}(t)$. This together with the fact that $x + t'h$ is a random element of $\mathbb{F}_p^m$ whenever $x \in_R \mathbb{F}_p^m$ and $t' \in_R \mathbb{F}_p$ concludes the proof of (21).

Next we use Corollary B.4 to show that

$$(22) \mathrm{Prob}_{h_1,h_2}[P_{x+s_0h_1,h_2}(t_0) = P_{x+t_0h_2,h_1}(s_0)] \geq 1 - 72\,\delta \quad \text{for all } x \in \mathbb{F}_p^m \text{ and } s_0, t_0 \in \mathbb{F}_p\,.$$

Let $A = (a_{st})$ denote the matrix given by $a_{s,t} = \tilde{g}(x + sh_1 + th_2)$, and let $r_s(t) = P_{x+sh_1,h_2}(t)$ and $c_t(s) = P_{x+th_2,h_1}(s)$, all defined with respect to $h_1, h_2 \in F^m$. Using (21) we immediately observe that the requirements of Corollary B.4 are satisfied (with $\tilde{S} = \tilde{T} = \mathbb{F}_p \setminus \{0\}$). As by definition $P_{x+s_0h_1,h_2}(t)$ and $P_{x+t_0h_2,h_1}(s)$ are the polynomials best fitting the s_0-th row and the t_0-th column, respectively (22) follows immediately from Corollary B.4.

An immediate consequence of (22) is the following strengthening of (21):

$$(23) \qquad \mathrm{Prob}_h[P_{x,h}(t) = \tilde{g}(x+th)] \geq 1 - 73\,\delta \qquad \text{for all } x \in \mathbb{F}_p^m \text{ and } t \in \mathbb{F}_p, t \neq 0.$$

To see (23) fix $x \in \mathbb{F}_p^m$ and $t \in \mathbb{F}_p \setminus \{0\}$. From (21) we deduce that

$$\mathrm{Prob}_{h_1,h_2}[P_{x+th_2,h_1}(0) = \tilde{g}(x+th_2)] \geq 1 - \frac{\delta}{2}$$

and from (22) we deduce (letting $s_0 = 0$ and $t_0 = t$) that

$$\mathrm{Prob}_{h_1,h_2}[P_{x+th_2,h_1}(0) = P_{x,h_2}(t)] \geq 1 - 72\,\delta.$$

Obviously, combining both inequalities proves (23).

Another consequence of (22) is

$$(24) \qquad \mathrm{Prob}_h[g(x) = P_{x,h}(0)] \geq 1 - 72\,\delta \qquad \text{for all } x \in \mathbb{F}_p^m\,.$$

Indeed, fix an $x \in \mathbb{F}_p^m$ and apply (22) for $s_0 = t_0 = 0$. Then

$$\begin{aligned} 1 - 72\,\delta &\leq \mathrm{Prob}_{h_1,h_2}[P_{x,h_1}(0) = P_{x,h_2}(0)] \\ &= \sum_{a \in \mathbb{F}_p} (\mathrm{Prob}_h[P_{x,h}(0) = a])^2 \\ &\leq \mathrm{Prob}_h[g(x) = P_{x,h}(0)] \cdot \underbrace{\sum_{a \in F} \mathrm{Prob}_h[P_{x,h}(0) = a]}_{=1}. \end{aligned}$$

The δ-closeness of g and $\tilde{g}$ now follows immediately from (21) and (24).

For the proof that g is a degree d polynomial we make use of the following characterization of polynomials:

A function $f : \mathbb{F}_p^m \to \mathbb{F}_p$ is a polynomial of total degree d if and only if for all $x, h \in \mathbb{F}_p^m$ the point $(x, f(x))$ lies on the degree d polynomial containing most points of $\{(x+th, f(x+th)) \mid t \in \mathbb{F}_p\}$.

Let $x, h \in \mathbb{F}_p^m$ be fixed. For $h_1, h_2 \in \mathbb{F}_p^m$ let $A' = (a'_{s,t})$ denote the matrix given by

$$a'_{s,t} = \begin{cases} g(x+sh), & \text{if } t = 0, \\ \tilde{g}(x+sh+t(h_1+sh_2)), & \text{otherwise} \end{cases}$$

and let $r'_s(t) = P_{x+sh,h_1+sh_2}(t)$ and $c'_t(s) = P_{x+th_1,h+th_2}(s)$. Then (21) implies

$$\mathrm{Prob}_{h_1,h_2}[c'_t(s) = a'_{st}] \geq 1 - \delta \qquad \text{for all } s,t \in \mathbb{F}_p, t \neq 0.$$

By (24), we also have

$$\text{(25)} \quad \mathrm{Prob}_{h_1,h_2}[a'_{s0} \equiv g(x+sh) = P_{x+sh,h_1+sh_2}(0) \equiv r'_s(0)] \geq 1 - 72\,\delta \qquad \text{for all } s \in \mathbb{F}_p\,.$$

Finally, we deduce from the δ-closeness and (23) that for all $s,t \in \mathbb{F}_p, t \neq 0$:

$$\mathrm{Prob}_{h_1,h_2}[a'_{st} \equiv \tilde{g}(x+sh+th_1+sth_2) = P_{x+sh,h_1+sh_2}(t) \equiv r'_s(t)] \geq 1 - 73\,\delta.$$

The assumptions of Corollary B.4 are therefore again satisfied, this time with "$\frac{\delta}{2}$" replaced by 79δ. As the first columns of the matrices A' do not depend on h_1 and h_2 and, by definition, $r'_0(t)$ is the best polynomial fitting the first row, Corollary B.4 (applied for $s_0 = t_0 = 0$) together with (25) for $s = 0$ and the above characterization of degree d polynomials concludes the proof of Theorem B.2.

□

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi (1987): Deterministic simulation in LOGSPACE, in: STOC, 132–140, 1987.

[2] N. Alon (1986): Eigenvalues and expanders, *Combinatorica* **6**, 83–96, 1986.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy (1992): *On the intractability of approximation problems (early draft)*, 1992.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy (1992a): Proof verification and the intractability of approximation problems, in: FOCS, 14–23, 1992.

[5] S. Arora and S. Safra (1992): Probabilistic checking of proofs; a new characterization of $\mathcal{NP}$, in: FOCS, 2–13, 1992.

[6] L. Babai (1985): Trading group theory for randomness, in: STOC, 421–429, 1985.

[7] L. Babai (1990): E-mail and the unexpected power of interaction, in: Proc. Structure in Complexit Theory, 30–44, 1990.

[8] L. Babai and L. Fortnow (1991): Arithmetization: a new method in structural complexity theory, *Comput. Complexity* **1**, 41–66, 1991.

[9] L. Babai, L. Fortnow, L. Levin, and M. Szegedy (1991): Checking computations in polylogarithmic time, in: STOC, 21–31, 1991.

[10] L. Babai, L. Fortnow, and C. Lund (1991): Non-deterministic exponential time has two-prover interactive protocols, *Comput. Complexity* **1**, 3–40, 1991.

[11] A. Blum (1990): Some tools for approximate 3-coloring, in: FOCS, 554–562, 1990.

[12] M. Blum, M. Luby, and R. Rubinfeld (1990): Self-testing/correcting with applications to numerical problems, in: STOC, 73–83, 1990.

[13] R. Boppana and M.M. Halldórsson (1992): Approximating maximum independent set by excluding subgraphs, *BIT* **32**, 180–196, 1992.

[14] S.A. Cook (1971): The complexity of theorem-proving procedures, in: STOC, 151–158, 1971.

[15] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy (1991): Approximating clique is almost NP-complete, in: FOCS, 2–12, 1991.

[16] O. Gabber and Z. Galil (1981): Explicit constructions of linear-sized superconcentrators, *Journal of Computer and System Sciences* **22**, 407–420, 1981.

[17] M.R. Garey and D.S. Johnson (1976): The complexity of near-optimal graph coloring, *Journal of the Association for Computing Machinery* **23**, 43–49, 1976.

[18] M.R. Garey and D.S. Johnson (1979): *Computers and Intractablility: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[19] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson (1991): Self-testing/correcting for polynomials and for approximate functions, in: STOC, 32–42, 1991.

[20] S. Goldwasser, S. Micali, and C. Rackoff (1989): The knowledge complexity of interactive proof systems, *SIAM J. Comput.* **18** (1), 186–208, 1989.

[21] M. Halldórsson (1993): A still better performance guarantee for approximate graph coloring, *Information Processing Letters* **45**, 19–23, 1993.

[22] T. Impagliazzo and D. Zuckerman (1989): How to recycle random bits, in: FOCS, 248–253, 1989.

[23] D.S. Johnson (1992): The NP-completeness column: An ongoing guide, *Journal of Algorithms* **13**, 502–524, 1992.

[24] R. Karp (1972): Reducibility among combinatorial problems, in: Complexity of Computer Computations (R.E. Miller, J.W. Thatcher, eds.), Plenum Press, New York, 85–103, 1972.

[25] S. Khanna, N. Linial, and S. Safra (1993): On the hardness of approximating the chromatic number, in: Proceedings of the Israel Symposium on Theoretical Computer Science 1993, 250 –260, 1993.

[26] R. Lipton (1991): New directions in testing, in: Distributed Computing and Cryptography, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 2, American Mathematical Society, 191–202, 1991.

[27] C. Lund, L. Fortnow, H. Karloff, and N. Nisan (1992): Algebraic methods for interactive proof systems, *Journal of the Association for Computing Machinery* **39** (4), 859–868, 1992.
[28] C. Lund and M. Yannakakis (1993): On the hardness of approximating minimization problems, in: STOC, 286–293, 1993.
[29] G.A. Margulis (1973): Explicit construction of concentrators, *Problemy Peredachi Informatsii* **9** (4), 71–80, 1973; English translation in: *Problems of Information Transmission*, Plenum, New York (1975).
[30] Chr. Papadimitriou and M. Yannakakis (1991): Optimization, approximation, and complexity classes, *Journal of Computer and System Sciences* **43**, 425–440, 1991.
[31] S. Phillips and S. Safra (1993): *On efficient probabilistic verification*, manuscript, 1993.
[32] R. Rubinfeld and M. Sudan (1992): Self-testing polynomial functions efficiently and over rational domains, in: SODA, 23–32, 1992.
[33] J.T. Schwartz (1980): Fast probabilistic algorithms for verification of polynomial identities, *Journal of the Association for Computing Machinery* **27**, 701–717, 1980.
[34] A. Shamir (1992): IP = PSPACE, *Journal of the Association for Computing Machinery* **39** (4), 869–877, 1992.
[35] M. Sudan (1992): *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems*, Ph.D. Thesis, University of California at Berkeley, 1992.
[36] B.L. van der Waerden (1971): *Algebra I* (8th edition), Springer-Verlag Berlin, Heidelberg, New York, 1971.