# Probabilistic Proofs and NP-completeness
# (A Course on the PCP Theorem and its Consequences)

Jiří Sgall*

October 23, 2002

These notes are course notes in the proper sense: my notes for teaching the class. The goal is not to provide a comprehensive coverage. Rather it aims at highlighting the important ideas, give pointers to the literature, clarify some items which were difficult for me, and otherwise help the reader of the covered papers. Each section lists references on which it is based. Additional information can be found on the WWW page of this course is at `http://www.math.cas.cz/~sgall/pcp/`.

## References

[Arora] Sanjeev Arora. Probabilistic Checking of Proofs and Hardness of Approximation Problems, Technical Report CS-TR-476-94, Princeton University, 1994.
Available at: ftp://www.cs.princeton.edu/reports/1994/476.ps.gz
or http://www.cs.princeton.edu/~arora/pubs/thesis.ps
A technical report based on the author's Ph.D thesis. Contains the complete proof of the PCP theorem, basic relations to approximation.

[AroraSafra] Sanjeev Arora, Muli Safra: Probabilistic Checking of Proofs: A New Characterization of NP. Journal of ACM, 45(1):70–122, 1998. An extended abstract appeared in FOCS 1992.
Available at: http://www.cs.princeton.edu/~arora/pubs/as.ps
A journal version of the original paper containing the weaker version of the PCP theorem.

[ALMSS] Sanjeev Arora and Carsten Lund and Rajeev Motwani and Madhu Sudan and Mario Szegedy. Proof verification and the hardness of approximation problems. Journal of ACM, 45(3):501-555, 1998. An extended abstract appeared in FOCS 1992.
Available at: http://www.cs.princeton.edu/~arora/pubs/almss.ps
or Electronic Colloquium on Computational Complexity, Report TR98-008
A journal version of the original paper containing the PCP theorem.

[RaSa] Ran Raz, Shmuel Safra: Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. Manuscript, 1996. An extended abstract appeared in STOC '97.
Available at: http://www.wisdom.weizmann.ac.il/~ranraz/publications/Prsconf.ps
(STOC version, full version not available)
Contains the improved and simplified low-degree test. The actual test is contained in Part II. The STOC version contains the 3-dimensional version (read it first!).

*E-mail: `sgall@math.cas.cz`, `http://www.math.cas.cz/~sgall`, Mathematical Inst., AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic and Institute for Computer Science, Prague (project LN00A056 of MŠMT CR). Partially supported by grant 201/01/1195 of GA ČR and grant A1019901 of GA AV ČR.

[ArSu] Sanjeev Arora, Madhu Sudan. Improved low-degree testing and its applications. An extended abstract appeared in STOC 1997.
Available at: http://www.cs.princeton.edu/~arora/pubs/ldtest.ps or Electronic Colloquium on Computational Complexity, Report TR97-003
An improved analysis of the original low-degree test. The results are similar as in the previous item, but achieved using the means of algebraic geometry.

[BGS] Mihir Bellare, Oded Goldreich, Madhu Sudan. Free bits, PCP and Non-Approximability - Towards tight results. SIAM J. Comput. 27(3): 804-915, 1998. An extended abstract appeared in FOCS 1995.
Available at: Electronic Colloquium on Computational Complexity, Report TR95-024
Introduced the notion of free bits. An excellent survey of the current best inapproximability results (including those based on the successive work, up to summer 1997). Many results from here are now improved (building on this work).

[Raz] Ran Raz. A Parallel repetition theorem. SIAM J. Comput. 27(3):763-803, 1998. An extended abstract appeared in STOC 1995.
Available at: http://www.wisdom.weizmann.ac.il/~ranraz/publications/Pparal1.ps
Proof of the parallel repetition theorem.

[Hastad1] Johan Håstad. Some optimal inapproximability results. Journal of ACM 48:798–859, 2001. An extended abstract appeared in STOC 1996.
Available at: http://www.nada.kth.se/~johanh/optimalinap.ps or Electronic Colloquium on Computational Complexity, Report TR97-037
Optimal inapproximability results for linear equations and satisfiability.

[Hastad2] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. Acta Mathematica 182:105-142, 1999. An extended abstract appeared in FOCS 1996.
Available at: http://www.nada.kth.se/~johanh/cliqueinap.ps or Electronic Colloquium on Computational Complexity, Report TR97-038

[SuTr] Madhu Sudan and Luca Trevisan. Probabilistically Checkable Proofs with Low Amortized Query Complexity. An extended abstract appeared in FOCS 1998.
Available at: http://www.cs.berkeley.edu/ luca/pubs/st98.ps.Z or Electronic Colloquium on Computational Complexity , Report TR98-040

[SaTr] Alex Samorodnitsky and Luca Trevisan. A PCP Characterization of NP with Optimal Amortized Query Complexity. An extended abstract appeared in STOC 2000.
Available at: http://www.cs.berkeley.edu/ luca/pubs/st-full.ps
A simplified version of Hastad inapproximability result for max clique. Depends heavily on the previous paper.

*Some books covering parts of the proof of the PCP theorem or its relation to hardness of approximation are listed below.*

[ACGKMP99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi: Complexity and Approximation. Springer, 1999.

[MPS98] E. W. Mayr, H. J. Proemel, A. Steger (eds.): Lectures on Proof Verification and Approximation Algorithms. LNCS 1367. Springer, 1998.

[H97] D. S. Hochbaum: Approximation Algorithms for NP-hard problems. PWS Publishing Company, 1997.

[MR95] R. Motwani, P. Raghavan: Randomized algorithms. Cambridge Univ. Press, 1995.

# 1 Background – complexity classes [MR95]

**Definition 1**

- **RP** *is the set of languages $L$ for which there exists a randomized polynomial time algorithm $A$ such that*
  - *If $x \in L$ then $\mathbf{Pr}[A(x)$ accepts$] \geq 1/2$.*
  - *If $x \notin L$ then $\mathbf{Pr}[A(x)$ accepts$] = 0$.*
- $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$.
- **BPP** *is the set of languages $L$ for which there exists a randomized polynomial time algorithm $A$ such that*
  - *If $x \in L$ then $\mathbf{Pr}[A(x)$ accepts$] \geq 2/3$.*
  - *If $x \notin L$ then $\mathbf{Pr}[A(x)$ accepts$] \leq 1/3$.*
- **PP** *is the set of languages $L$ for which there exists a randomized polynomial time algorithm $A$ such that*
  - *If $x \in L$ then $\mathbf{Pr}[A(x)$ accepts$] \geq 1/2$.*
  - *If $x \notin L$ then $\mathbf{Pr}[A(x)$ accepts$] < 1/2$.*

**Fact 2** *Known relations among the complexity classes include:*

$$\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{NP} \subseteq \mathbf{PP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXPTIME}$$

$$\mathbf{RP} \subseteq \mathbf{BPP} \subseteq \mathbf{PP}$$

*No relation between* **BPP** *and* **NP** *is known; it is unlikely that* $\mathbf{NP} \subseteq \mathbf{BPP}$. *It is assumed but unproven that* $\mathbf{ZPP} \neq \mathbf{EXPTIME}$.

# 2 Definitions, PCP theorem statement [Arora, ACGKMP99, MPS98, H97]

**Definition 3** $\mathbf{PCP}[r(n), q(n)]$ *is the set of languages $L$ for which there exists a randomized polynomial time algorithm $A(x, \Pi)$ such that*

- *On any input of length $n$, $A$ uses at most $O(r(n))$ random bits and accesses (non-adaptively) at most $O(q(n))$ bits of the proof $\Pi$.*
- *If $x \in L$ then there is a proof $\Pi$ such that $\mathbf{Pr}[A(x, \Pi)$ accepts$] = 1$.*
- *If $x \notin L$ then for any proof $\Pi$, $\mathbf{Pr}[A(x)$ accepts$] \leq 1/2$.*

*We define* $\mathbf{PCP} = \mathbf{PCP}[\log n, 1]$.

**Fact 4**

$$\begin{aligned}
&\mathbf{P} = \mathbf{PCP}[0, 0] \\
&\mathbf{NP} = \mathbf{PCP}[0, n^{O(1)}] \\
&\mathbf{coRP} = \mathbf{PCP}[n^{O(1)}, 0] \\
&\mathbf{PCP} \subseteq \mathbf{PCP}[\log n, n^{O(1)}] \subseteq \mathbf{NP} \\
&\mathbf{NP} \subseteq \mathbf{PCP}[o(logn), o(logn)] \rightarrow \mathbf{P} = \mathbf{NP}
\end{aligned}$$

**Theorem 5 (PCP Theorem)**

$$\mathbf{NP} = \mathbf{PCP}$$

Note that it is non-trivial to show even that $\mathbf{NP} \subseteq \mathbf{PCP}[n^{O(1)}, 1]$.

Now we give another equivalent definition of **PCP** and another formulation of the **PCP** theorem.

**Definition 6** **PCP** *is the set of languages $L$ such that there exists a constant $k$ and for each input $x$ there exist functions $f_1(\Pi), \ldots, f_{p(|x|)}(\Pi)$ that can be constructed in polynomial time from $x$ and satisfy*

- *Every $f_i$ depends on only $k$ input bits.*
- *If $x \in L$ then there is a proof $\Pi$ such that for all $i$, $f_i(x, \Pi) = 1$.*
- *If $x \notin L$ then for any proof $\Pi$, $\mathbf{Pr}_i[f_i(x, \Pi) = 1] \leq 1/2$.*

**Theorem 7** *There exists a polynomial time transformation $T$ from 3-CNF to 3-CNF and a constant $\varepsilon > 0$ such that*

- *If $\psi$ is satisfiable, then $T(\psi)$ is satisfiable.*
- *If $\psi$ is not satisfiable, then at most $1 - \varepsilon$ fraction of the clauses of $T(\psi)$ are simultaneously satisfiable.*

# 3 Optimization problems and hardness of approximation [Arora, ACGKMP99, MPS98, H97]

**Definition 8** Max-3-SAT *is the following optimization problem. Given a 3-CNF formula $\psi$, find an assignment which satisfies as many clauses of $\psi$ as possible.*

As an immediate consequence of Theorem 7 we get:

**Theorem 9** *There exists a constant $\varepsilon > 0$ such that it is NP-hard to distinguish satisfiable formulas from those with at most $1 - \varepsilon$ fraction of clauses satisfiable. Therefore, it is NP-hard to $(1 + \varepsilon')$-approximate* Max-3-SAT *for some $\varepsilon' > 0$.*

Now we prove that it is NP-hard to approximate Max-Clique up to any constant factor. Given $\varepsilon$ and a 3-CNF formula $T(\psi)$ with $m$ clauses from Theorem 7, form a graph $G$ as follows. The vertices are indexed by pairs $(r, a)$ where $r$ is a sequence of any $c$ clauses (there are $m^c$ possible values of $r$) and $a$ is a string of $3c$ bits, interpreted as the assignments to (at most) $3c$ variables in the clauses $r$; in addition this assignment is required to satisfy all the clauses. Two vertices $(r, a)$ and $(r', a')$ are connected by an edge if the assignments $a$ and $a'$ are consistent. If the formula $\psi$ is satisfiable, the maximal clique has $m^c$ vertices. If not, the maximal size of a clique is at most $((1-\varepsilon)m)^c$. Choosing $c$ sufficiently large (but constant) makes the gap arbitrarily large constant. A slightly modified construction yields:

**Theorem 10** *There exists a constant $\varepsilon > 0$ such that if there exists a polynomial time algorithm which approximates the size of maximal clique in a graph on $n$ vertices up to a factor of $n^\varepsilon$, then* $\mathbf{NP} = \mathbf{coNP} = \mathbf{ZPP}$.

# 4 A verifier for 3-SAT in $\mathbf{PCP}[n^3, 1]$ [Arora, ACGKMP99, MPS98, MR95]

All computations in this section are done in $GF(2)$ (i.e., mod 2).

First, suppose that we have a vector $\vec{a}$ of $n$ bits (e.g. an assignment to the variables of a 3-CNF). Define a function $A$ on $n$-bit vectors (in fact, we just interpret $\vec{a}$ as coefficients of a linear function) as follows:

$$A(\vec{x}) = \vec{a} \cdot \vec{x} = \sum a_i x_i.$$

Suppose we have a table of all values of $A$ (later it will become a part of the PCP proof). Note that it is exponentially large. We want to design two procedures. Both should read only $O(1)$ bits of the table and achieve an arbitrarily small error $\varepsilon$.

**linearity test** Test that at least 90% of the values in the table are values of a single linear function. Call this function $\tilde{A}$.

**linear self-correcting** Read a value of $\tilde{A}(\vec{a})$.

Note that these are reasonable goals. We cannot expect to verify that $A$ really is a linear function without reading it all. On the other hand, if it is close to some linear $\tilde{A}$ we may pretend that it is this $\tilde{A}$ — however for this we need to read the values of $\tilde{A}$ rather than those of $A$. We do the procedures as follows.

**linearity test** Pick random $\vec{x}$, $\vec{y}$. Verify that $A(\vec{x}) + A(\vec{y}) = A(\vec{x} + \vec{y})$. Repeat several times.

**linear self-correcting** Pick a random $\vec{x}$. Output $A(\vec{a} + \vec{x}) + A(\vec{x})$. Repeat several times.

**Theorem 11** *Let $\delta < 1/6$. Let $A$ have the property*

$$\mathbf{Pr}_{\vec{x},\vec{y}}[A(\vec{x}) + A(\vec{y}) = A(\vec{x} + \vec{y})] \geq 1 - \delta.$$

*Then there exists a linear function $\tilde{A}$ such that $\tilde{A}$ and $A$ are $3\delta$ close (i.e., they agree on all except possibly $3\delta$ fractions of the inputs). If $\delta < 1/12$, such $\tilde{A}$ is unique.*

**Proof**. To prove this theorem takes several steps:

**Lemma 12** *For all $\vec{a}$ there exists $b$ such that $\mathbf{Pr}_{\vec{x}}[A(\vec{a} + \vec{x}) + A(\vec{x}) = b] \geq 1 - 2\delta$.*

If $\delta < 1/4$, this $b$ is unique. We define $\tilde{A}(\vec{a}) = b$ for this $b$.

**Lemma 13** *$\tilde{A}$ is linear and $3\delta$ close to $A$.*

For uniqueness, no linear functions are $< 1/2$ close. □

**Theorem 14** 3-SAT $\in \mathbf{PCP}[n^3, 1]$.

**Proof**. Let $\vec{a}$ be an assignment to $n$ variables occurring in the input formula, represented as a column vector. Let $\vec{b}$ be the vector with $n^2$ entries defined by $b_{ij} = a_i a_j$. Let $\vec{c}$ be the vector with $n^3$ entries defined by $c_{ijk} = a_i a_j a_k$. The vector $\vec{c}$ contains the vectors $\vec{a}$ and $\vec{b}$ since $a_i = c_{iii}$ and $b_{ij} = a_{ijj}$. Now let $M$ be the matrix with $n$ rows and $n^2$ columns with entries defined by $M_{i,jk} = c_{ijk}$. Then $M$ is equal to the tensor product $\vec{a} \otimes \vec{b} = \vec{a}\vec{b}^T$. Moreover, given any vector $\vec{c}$ and the corresponding $M$, if $M = \vec{a}\vec{b}^T$, then the vector $\vec{c}$ is indeed obtained from its subvector $\vec{a}$ as described above.

On an input $\psi$ the verifier expects the proof to contain the table $C$ of size $2^{n^3}$ which is a table of the linear function with coefficients $\vec{c}$ obtained from a satisfying assignment to $\psi$. It does the following:

- Verifies that $C$ is close to a linear function (see Theorem 11). Let $\vec{c}$ be the coefficients of the unique linear function $\tilde{C}$ close to $C$. Let $\vec{a}$ and $\vec{b}$ be its subvectors and $M$ be the rearrangement of $\vec{c}$ into a matrix as described above.

- Verifies that $M = \vec{a}\vec{b}^T$ as follows. Pick a random $n$-bit vector $\vec{r}$ and a random $n^2$-bit vector $\vec{s}$. If $M \neq \vec{a}\vec{b}^T$ then with probability $1/4$ $\vec{r}^T M \vec{s} \neq \vec{r}^T \vec{a}\vec{b}^T \vec{s}$. We can evaluate the last equation with three queries to $\tilde{C}$. Repeat, to boost the probability.

- Verifies that $\vec{a}$ is a satisfying assignment. To do this, pick a random subset of clauses; if the assignment is not satisfying, with probability $1/2$ their sum modulo 2 is non-zero. Represent the sum of these clauses as a polynomial. It is a polynomial of degree 3, hence it may be evaluated reading one value of $\tilde{C}$. □

## 5 Proof composition, a refined notion of a verifier [Arora, ACGKMP99, MPS98]

Suppose we have a $\mathbf{PCP}[\log n, \log n]$ verifier. It reads some random $\log n$ bits of the proof, and then decides whether they are acceptable. Suppose that, instead of deterministically deciding whether to accept this string of $\log n$ bits, we use another verifier to check that they are acceptable. If we use the same verifier and the original decision time is small, this could need less queries, since the string is much smaller. We could even use the $\mathbf{PCP}[n^3, 1]$ verifier for the "inner verifier".

This is somewhat naive, since in this way we would not have time to read the string of $\log n$ bits, and we could be cheated by the inner verifier that would proof that a different string is acceptable. Therefore, we need to define a more refined notion of a verifier.

**Definition 15** *Suppose we have a verifier for* 3-SAT *which reads a proof over some alphabet $\Sigma$, which possibly depends on $n$ (i.e., a query involves reading a symbol of $\Sigma$ instead of a bit). We say that it can* check split assignments *if for every integer $p$ it behaves as follows. There is an encoding $\sigma : \{0,1\}^{n/p} \to \Sigma^*$, and:*

- *If $a_1, \ldots, a_p$ is a satisfying assignment then there exists $\pi$ such that*

$$\mathbf{Pr}[\sigma(a_1), \ldots, \sigma(a_p), \pi \textit{ is accepted}] = 1$$

- *If $\sigma(a_1), \ldots, \sigma(a_p)$ are the closest codes to $z_1, \ldots, z_p$ and $a_1, \ldots, a_p$ is not a satisfying assignment then for any $\pi$,*

$$\mathbf{Pr}[z_1, \ldots, z_p, \pi \textit{ is accepted}] < 1/2$$

For $p = 1$, this is just the normal notion of a verifier.

**Definition 16** *A verifier is* $(r(n), q(n), t(n))$-restricted *if it checks split assignments, uses $O(r(n))$ random bits, reads $O(p \cdot q(n))$ symbols (from $\Sigma$) of the proof (non-adaptively), and after reading them it runs in time $t(n)^{O(1)}$. ($p$ in the number of queries is the number of split assignments.)*

Note that we do not restrict the size of the alphabet $\Sigma$, but the restriction of the running time implies that a symbol of $\Sigma$ can be read in time $t(n)^{O(1)}$.

**Lemma 17** *There exists $(n^3, 1, 1)$ restricted verifier for* 3-SAT.

**Proof**. In addition to the verifier from Theorem 14 we now require the proof to contain the linear function corresponding to each block of the assignment ($f_1(x_1, \ldots, x_{n/p}) = \sum_{i=1}^{p/n} a_i x_i$, etc.); these functions are the encoding $\sigma$. Test that they are linear in time $O(p)$ (over boolean alphabet). Test that they are consistent with the linear part of the big table, again in time $O(p)$. Finally, continue as the old verifier. □

**Lemma 18 (Composition Lemma)** *Suppose we have verifiers $V_1$ and $V_2$ for* 3-SAT *that are $(R(n), 1, T(n))$ and $(r(n), 1, t(n))$ restricted. Then, for some $T'(n) = T(n)^{O(1)}$, there is a verifier which is $(R(n) + r(T'(n)), 1, t(T'(n)))$ restricted.*

**Proof**. Modify both $V_1$ and $V_2$ so that the probability of error is at most $1/4$. Modify $V_1$ so that on a random string $r$ it just checks that the symbols read from the proof are a satisfying assignment of a formula $\phi_r$ computed by $V_1$ based on the input only.

Given a proof $\Pi$ for $V_1$, the new proof contains the symbols of $\Pi$ and for each $r$ an additional string $\rho_r$, all encoded by $\sigma$ from the $V_2$ procedure for split assignments.

The new verifier $V$ computes the locations accessed by $V_1$ with the random string $r$, and using the procedure of $V_2$ for checking assignments split in $Q$ parts on the encoding of the symbols read by $V_1$ together with $\rho_r$, it verifies that $\phi_r$ is satisfied by the symbols read by $V_1$.

It is easy to verify that for a satisfiable $\phi$ we can construct a perfect proof. The complexity is as required.

Suppose we are given an unsatisfiable formula and some proof $(\Pi', \Gamma)$ for $V$. Let $\Pi$ be the sequence of the best decodings of the symbols of $\Pi'$. Since $V_1$ rejects, $\phi_r$ is not satisfiable, for all but 1/4 of random bits $r$. For any such $r$, $V_2$ accepts with probability at most 1/4. Hence $V$ accepts with probability at most 1/2. □

# 6 Coding by polynomials of low degree [Arora, ACGKMP99, MPS98]

We have used coding a bit-string by a table of an associated linear function to obtain a constant number of bits queried. Now we want to do something similar using low degree polynomials.

This will have several advantages: (i) the code of a bit-string will have polynomial size, and this will enable us to use logarithmic randomness for each query; (ii) we will be able to reconstruct many values of the encoded string with only constantly many queries; (iii) we will be able to decrease the error probability exponentially in the length of the symbols of the alphabet, while maintaining the constant number of queries.

Of course, we have to pay for this: (1) The size of the field and hence also the size of the alphabet will be non-constant, thus we are able to achieve a constant number of queries, but each of them will use logarithmic size of symbols. To achieve a constant number of query bit we will need to compose with the inner verifier using the linear function encoding. (2) The proof will be considerably harder, esp. to achieve (iii).

## Setup and parameters

$F$ — a finite field $GF(q)$ for some prime $q$. ($q = (\log n)^{\Theta(1)}$.)
$H \subseteq F$, $h = |H|$ — $H$ is a subset of the field containing 0 and 1. ($h = \Theta(\log n)$.)
$m$ — number of variables. ($m = \Theta(\log n / \log\log n)$, hence $|F|^m = n^{O(1)}$.)
$d$ — degree of the polynomials we use; always the total degree in all variables. ($d = \Theta(mh) = (\log n)^{\Theta(1)}$.)
$F_d[x_1, \dots, x_m]$ — the set of $m$-variate polynomials of total degree $d$. Each polynomial is represented by its table (size $|F|^m \log |F| = n^{O(1)}$).
$\delta$ — a constant which measures the quality of coding and the probability of an error.
We choose the parameters so that $|F| > 2dm$, $|F| > \omega(d^3)$, $|F| > \omega(mh)$.

**Lemma 19** *Any two distinct $m$-variate polynomials of degree $d$ agree on at most $d/|F|$ fraction of points in $F^m$.*

**Definition 20** *Given a function $f : F^m \to F$, let $\tilde{f}$ be the polynomial of degree $d$ which agrees with it on most points. We say that $f$ is* $\delta$-close *if it disagrees with $\tilde{f}$ on at most $\delta$ fraction of points.*

**Lemma 21** *For every function $s : H^m \to F$ there exists a polynomial in $F_{mh}[x_1, \dots, x_m]$ which agrees with $s$ on all points of $H^m$. Moreover, it can be constructed efficiently.*

**Definition 22** *Call a polynomial satisfying the conditions of the previous lemma* a low-degree extension *of $s$ and denote it by $\hat{s}$.*

# 7 Procedures [Arora, ACGKMP99, MPS98]

We want to design two procedures. Both should read only $O(1)$ symbols of the table and achieve an arbitrarily small but constant error $\varepsilon$.

**low-degree test** Given: $f : F^m \to F$, an auxiliary table $T$, and a number $d = o(|F|^{1/c})$, where $c$ is some sufficiently large absolute constant (defined in the test procedure).

Goal: If $f$ is $\delta$ close, accept with probability 1 for some table $T$. If not, reject with probability at least $1/2$ for any $T$.

Complexity: Use $O(m \log |F|)$ random bits. Read $O(1)$ values from $f$ and $T$. Run in time polynomial in $m$, $d$, and $|F|$. All constants involve $1/\delta$, of course.

**low-degree polynomial reconstruction** Given: A $\delta$ close function $f : F^m \to F$, a sequence $z_1, \ldots, z_k \in F^m$ for $kd \leq o(|F|)$, and an auxiliary table $T$ (which may be different for different values of $\vec{z}$.)

Goal: If $f$ is a polynomial of degree $d$, always output $f(z_1), \ldots, f(z_k)$, for some $T$. If not, with probability at least $1 - 2\sqrt{\delta} - o(1)$ either reject or still output the correct value of $\tilde{f}(z_1), \ldots, \tilde{f}(z_k)$, for every $T$.

Complexity: Use $O(m \log |F|)$ random bits. Read $O(1)$ values from $f$ and $T$. Run in time polynomial in $m$, $d$, $k$, and $|F|$. Here the constants do not depend on $\delta$.

Unlike in the procedures for linear functions, here we need to use auxiliary tables $T$. The main reason for that is that we need the probability and the number of queries to be a constant independent of $d$, $m$, and even the number of reconstructed values $c$.

Now we show how to do the reconstruction; we shall discuss low-degree test later.

**Definition 23** *A* curve $C$ of degree $k$ *is given by $m$ polynomials $C_i : F \to F$, each of degree $C$. The* $i$-th point of curve $C$ *is the point in $F^m$ given by $C(i) = (C_1(i), \ldots, C_m(i))$.*

*Given a function $f : F^m \to F$ its* restriction along the curve $C$ *is the function $f|_C : F \to F$ given by $f|_C(i) = f(C(i))$.*

A few observations. First, given any $k+1$ points and their order, there exists a unique curve of degree (at most) $k$ passing through these points. E.g., there is a unique curve of degree 2 with 1st point $(0,3)$, 2nd point $(2,2)$ and 67th point $(0,3)$. This follows from univariate polynomial interpolation.

Second, if $f$ is a polynomial of degree $d$ and $C$ is a curve of degree $k$, then $f|_C$ is a univariate polynomial of degree $kd$. In particular, it can be represented by $O(kd \log |F|)$ bits

Now we are ready to describe the low-degree reconstruction and prove its correctness. For the rest of this section assume that the points $z_1, \ldots, z_c \in F^m$ (where we want to reconstruct $\tilde{f}$) are given. Let $\mathcal{C}$ denote the set of all degree $c$ curves with first $c$ points $z_1, \ldots, z_c$.

**low-degree polynomial reconstruction** Given: A table of a $\delta$ close function $f : F^m \to F$, and an auxiliary table $T$ which for each $C \in \mathcal{C}$ contains a univariate polynomial $g_C : F \to F$ of degree at most $kd$ (considered as a single symbol of the alphabet).

Procedure: Choose uniformly at random element $C \in \mathcal{C}$, more precisely choose $C(0)$ randomly from $F^m$ and interpolate the $k+1$ points we have now to obtain C. Choose a random number $i \in F - \{1, \ldots, k\}$, again with a uniform distribution. Verify that $g_C(i) = f(C(i))$. Output $g_C(1), \ldots, g_C(k)$.

**Lemma 24** *The low-degree polynomial reconstruction procedure behaves as required.*

**Proof**. The complexity is as required, as the number of random bits needed to choose $C$ and $i$ is $O(m \log |F|)$, we ask only 2 queries, the information read has size $O(mkd \log |F|)$ and the tests are polynomial in the size of the query answers.

If $f$ is a polynomial of degree $d$, construct a table $T$ so that $g_C = f|_C$ for every $C$. This always produces the correct answer.

Otherwise $f$ is $\delta$-close to $\tilde{f}$ and we need to prove that the probability of a wrong answer is at most $2\sqrt{\delta} + o(1)$. We do it in two steps.

First, we claim that

$$Pr_{C\in\mathcal{C}}[Pr_{i\in F-\{1,\ldots,k\}}[f|_C(i)\neq \tilde{f}|_C(i)]>\sqrt{\delta}]\leq\sqrt{\delta}.$$

For every $i\in F-\{1,\ldots,k\}$, $C(i)$ is a random point in $F^m$. Hence $\mathbf{Pr}_{i,C}[f|_C(i)\neq\tilde{f}|_C(i)]\leq\delta$, using the fact that $f$ is $\delta$-close to $\tilde{f}$. The claim follows by the Markov inequality (i.e., simply count that if there are more $C$ with many bad $i$, the probability of a bad pair $i, C$ is too high).

Second, let $C$ be such that

$$Pr_{i\in F-\{1,\ldots,k\}}[f|_C(i)\neq\tilde{f}|_C(i)]\leq\sqrt{\delta}.$$

Then with probability at least $1-\sqrt{\delta}-o(1)$ we either output the right values or reject. If $g_C$ in $T$ is equal to $\tilde{f}|_C$, we output the right values. If not, then $g_C$ agrees with $\tilde{f}|_C$ on at most $kd$ points, and by the assumption, it agrees with $f|_C$ on at most $kd+\sqrt{\delta}|F|$ points. The probability of not rejecting is at most $(k+kd+\sqrt{\delta}|F|)/|F|=\sqrt{\delta}+o(1)$. The extra $k$ comes from the fact that we do not choose $i$ from the whole $F$; this cannot be worse than sampling over whole $F$ and making an error on the $k$ excluded points.

The bound on the total error now follows. □

# 8 A verifier using logarithmic randomness [Arora, ACGKMP99, MPS98]

We first show how to convert any verifier so that it reads only constant number of symbols, with only a small increase in the alphabet. Of course, the polynomial reconstruction is essential for this.

**Lemma 25** *If there is a* $(\log n, Q(n), \log n)$ *restricted verifier for* 3-SAT *for any* $Q(n)$*, then there is a* $(\log n, 1, \log n)$ *restricted verifier for* 3-SAT.

**Proof**. Let $R$, $Q$, and $T$ be the number of random bits, queries, and the running time of the verifier $V$. Clearly $Q\leq T$. The proof for the verifier $V$ has length at most $N=2^R\cdot T=2^{O(\log n)}$. Choose the parameters as above with $N$ in place of $n$, with the additional condition that $|F|=\omega(Qd)$. The size of the field is still polynomial in $\log n$, which ensures that the complexity of $V'$ will be as required.

The verifier $V'$ expects as a proof a low-degree extension of a proof $\Pi$ for $V$, and auxiliary tables $T$, $T_1$, …, $T_{2^R}$. First it checks that $\Pi$ is 1% close to a low degree polynomial. Then it computes which bits $V$ reads on the given random string $r$, uses the low-degree polynomial reconstruction with the table $T_r$ to read them, and simulates $V$ on their values.

Given an unsatisfiable formula and a string $\Pi$, let $\Pi'$ be the string whose low-degree extension $\tilde{\Pi}$ is. If $\Pi$ is not 1% close, $V'$ rejects. Otherwise, with probability at least $1-2\sqrt{0.01}$ it behaves just like $V$ on $\Pi'$, and hence it rejects with probability at least $1/2-\sqrt{0.01}$. □

Consequently by the composition lemma we "only" need to construct a $(\log n, q(n), \log n)$ restricted verifier. This uses three ingredients. First, we show that to test some polynomial sums to 0 over $H^m$; for the linear code this was trivial, now this needs significantly more work. Then we show how to use this to check that some polynomial is identically 0 on $H^m$; for the linear code we took a sum of the function on a random subset, here we need to be more careful. And, finally, we show that $A$ is a low-degree extension of a satisfying assignment, it is sufficient to check that some polynomial $g_A$ is identically 0 on $H^m$; this is easier than for linear code, since there we had to deal with higher degree polynomials by using the tensor product testing.

**Lemma 26** *Let* $dm=o(|F|)$*. There exists a verifier which on an input* $B\in F_d[x_1,\ldots,x_m]$*, a value* $b$*, and a proof* $T$ *verifies that the sum of* $B$ *on* $H^m$ *is* $b$*. It reads one random value of* $B$*, some additional information from* $T$*, and runs in time polynomial in* $l$*,* $d$*, and* $|F|$*. It needs only* $m\log|F|$ *random bits.*

**Proof**. For $i = 1, \ldots, m$ and $a \in F^m$ define

$$B^i_{a_1,\ldots,a_{i-1}}(y_i) = \sum_{y_{i+1},\ldots,y_m \in H} B(a_1, \ldots, a_{i-1}, y_i, \ldots, y_m).$$

Note that each $B^i_{a_1,\ldots,a_{i-1}}$ is a univariate polynomial of degree at most $d$. Moreover, if $b$ is the correct sum then, for any $a$,

$$\begin{aligned} b &= \sum_{a_1 \in H} B^1(a_1), \\ B^i_{a_1,\ldots,a_{i-1}}(a_i) &= \sum_{a_{i+1} \in H} B^{i+1}_{a_1,\ldots,a_i}(a_{i+1}) \quad \text{for all } i < m, \textit{and} \\ B^m_{a_1,\ldots,a_{m-1}}(a_m) &= B(a). \end{aligned}$$

The verifier expects as the auxiliary table $T$ which for each $a \in F^m$ and $i = 1, \ldots, m$ contains a univariate polynomial $g^i_a : F \to F$ of degree at most $d$ (considered as a single symbol of the alphabet). It chooses uniformly at random element $a \in F^m$, reads all polynomials $g^i_a$, and verifies that

$$\begin{aligned} b &= \sum_{a_1 \in H} g^1(a_1), \\ g^i_{a_1,\ldots,a_{i-1}}(a_i) &= \sum_{a_{i+1} \in H} g^{i+1}_{a_1,\ldots,a_i}(a_{i+1}) \quad \text{for all } i < m, \textit{and} \\ g^m_{a_1,\ldots,a_{m-1}}(a_m) &= B(a). \end{aligned}$$

The complexity is as required. If $b$ is correct, then for each $a$ and $i$ we let $g^i_{a_1,\ldots,a_{i-1}} = B^i_{a_1,\ldots,a_{i-1}}$.

Suppose that $b$ is not correct. If $b \neq \sum_{a_1 \in H} g^1(a_1)$, we reject. Hence $g^1$ is a different polynomial than $B^1$ and thus the probability over $a_1$ that $g^1(a_1) = B^1(a_1)$ is at most $d/|F|$. For any $a_1$ such that $g^1(a_1) \neq B^1(a_1)$, if the condition for $g^1$ is accepted, then the polynomials $g^2_{a_1}$ and $B^2_{a_1}$ are different. We continue inductively, and obtain that, for a random $a$, with probability at least $1 - md/|F|$ we either reject one of the conditions for $i < m$ or we have $g^m_{a_1,\ldots,a_{m-1}}(a_m) \neq B^m_{a_1,\ldots,a_{m-1}}(a_m) = B(a)$, and hence we reject the last condition. □

**Lemma 27** *There exists a family of $|F|^{O(m)}$ polynomials $\mathcal{P} \subseteq F_{mh}[x_1, \ldots, x_m]$, efficiently constructible, such that for any function $f : H^m \to F$ not identically zero, for at most $o(1)$ fraction of the polynomials $P \in \mathcal{P}$,*

$$\sum_{y \in H^m} P(y) f(y) = 0$$

**Proof**. Given $b \in F$, let $P_b(z)$ be a univariate polynomial of degree at most $h$ such that $P_b(a) = b^a$ for any $a \in H$. Now the family $\mathcal{P}$ contains polynomials $P_{\vec{b}} = \prod_{i=1}^m P_{b_i}(y_i)$ for all $\vec{b} \in F^m$. Now

$$\sum_{y \in H^m} P_{\vec{b}}(y) f(y) = \sum_{y \in H^m} f(y) \prod_{i=1}^m b_i^{y_i}.$$

Consider this as a polynomial in variables $\vec{b}$. Clearly, this polynomial is the zero polynomial if and only if $f$ is identically zero on $H^m$. It is a polynomial of degree $mh$, and hence the probability that it is 0 for a random $\vec{b} \in F^m$ is at most $mh/|F| = o(1)$. □

**Lemma 28** *For any $A \in F_{mh}[x_1, \ldots, x_m]$ there exists a polynomial $g_A \in F_{7mh}[x_1, \ldots, x_{4m}]$ such that $A$ is a low-degree extension of a satisfying assignment iff $g_A$ is identically 0 on $H^{4m}$. Moreover, $g_A$ is given efficiently and to evaluate $g_A$ on any point it is sufficient to evaluate $A$ on three points.*

**Proof**. The polynomial $g_A$ takes four variables $\vec{z}, \vec{v}_1, \vec{v}_2, \vec{v}_3 \in F^m$ which (if they are in $H^m$) are interpreted as a number of clause and the numbers of three variables.

Let $g(\vec{z}, \vec{v}_1, \vec{v}_2, \vec{v}_3) = 1$ on $H^{4m}$ if the clause $\vec{z}$ contains the variables $\vec{v}_1, \vec{v}_2, \vec{v}_3$, in this order, and let $g$ be 0 elsewhere on $H^{4m}$. Let $s_j(\vec{z}) = 0$ on $F^m$ if $j$th variable in $\vec{z}$ is negated, and 1 otherwise. Now define

$$g_A(\vec{z}, \vec{v}_1, \vec{v}_2, \vec{v}_3) = \hat{g}(\vec{z}, \vec{v}_1, \vec{v}_2, \vec{v}_3) \cdot (\hat{s}_1(\vec{z}) - A(\vec{v}_1)) \cdot (\hat{s}_2(\vec{z}) - A(\vec{v}_2)) \cdot (\hat{s}_3(\vec{z}) - A(\vec{v}_3)),$$

where $\hat{g}$ and $\hat{s}_j$ are the low-degree extensions of $g$ and $s_j$. $\square$

**Theorem 29** *There exists a* $(\log n, q(n), \log n)$*-restricted verifier for* 3-SAT.

**Proof**. The verifier expects a low-degree extension of the satisfying assignment $A$, together with the tables necessary to verify for every $P \in \mathcal{P}$ that the sum of $P(y)g_A(y)$ over $H^{6m}$ is zero, the tables necessary for the low-degree test, and tables for low-degree polynomial reconstruction of $\tilde{A}$ at any three points.

It chooses $P \in \mathcal{P}$ at random, verifies that $A$ is $\delta$-close for $\delta = 0.01$, and verifies that the sum of $P(y)g_{\tilde{A}}(y)$ over $H^{6m}$ is zero. To do this, we need to evaluate $g_{\tilde{A}}$ at a single point, which is done by reading $\tilde{A}$ at three points. The complexity is as required.

If $\psi$ is satisfiable, we can construct the table so that the verifier always accepts. Suppose $\psi$ is not satisfiable. If $A$ is not $\delta$-close, we reject with probability $1/2$. Hence we may assume that $A$ is $\delta$-close. Then for almost all $P \in \mathcal{P}$ the sum is not zero. In such a case we almost always reject, except if we reconstruct wrong values of $\tilde{A}$. However, this happens with probability at most $0.2 + o(1)$. Thus the total probability of error is strictly less than 1.

We need to design the verifier so that it can check split assignments. To do this, simply take a low-degree extension $A_1, \ldots, A_p$ for each part of the assignment separately. Similar procedures work, using only $O(1)$ queries from each piece. $\square$

Now the only missing piece in the proof of the PCP theorem is the low-degree test.

# 9 Low degree test [RaSa]

**low-degree test** Given: A table of a function $f : F^m \to F$, and an auxiliary table $T$ which for each plane $P \subseteq F^m$ contains a bivariate polynomial $g_P : F^2 \to F$ of degree at most $d$ (considered as a single symbol of the alphabet).

Procedure: Choose uniformly at random two planes $P$ and $P'$ which intersect in a line and a point $x \in P \cap P'$. Check that $g_P$ and $g_{P'}$ agree on their intersection and that on the point corresponding to $x$ they are equal to $f(x)$. Repeat, to decrease the error.

**Lemma 30** *The complexity of the low-degree test is as required. A low-degree $f$ always passes the test.*

**Proof**. The complexity is as required, as the number of random bits needed to choose $P$, $P'$ and $x$ is $O(m \log |F|)$, we ask only 3 queries, the information read has size $O(mcd \log |F|)$ and the tests are polynomial in the size of the query answers.

If $f$ is a polynomial of degree $d$, construct a table $T$ so that $g_P = f|_P$ for every $C$. This always produces the correct answer. $\square$

The rest of this section is devoted to the proof of the correctness of the low-degree test.

## 9.1 Notation, the statement

$\mathcal{S}^M$ is the set of all linear (affine) subspaces of $F^m$ of dimension $M \leq m$.
$S$ denotes a subspace of dimension $m - 1$.
$P$ denotes a plane, i.e., an element of $\mathcal{S}^2$.

$\mathcal{P}$ the set of pairs of planes that intersect in a line.
$\alpha$ denotes $(d+1)/|F|$.

**Definition 31** *A function $A : \mathcal{S}^2 \to F_d[x_1, \ldots, x_m]$ is called an extension if for each $P \in S^2$, $g_P = A(P)|_P$. The set of all extensions of dimension $m$ is denoted $E_m$.*

**Definition 32** *The system $g_P$ is $\gamma$-consistent if for a $\gamma$ fraction of pairs $(P, P') \in \mathcal{P}$, $g_P|_{P \cap P'} = g_{P'}|_{P \cap P'}$.*

I.e., if the low-degree test accepts with probability $\gamma$, then $g_P$ is $\gamma$-consistent.

**Definition 33** *The system $g_P$ is $\gamma$-close if there exists an extension $A$ such that for a $\gamma$ fraction of pairs $(P, P') \in \mathcal{P}$, $A(P) = A(P')$.*

In our case, i.e., if $\gamma$ is a constant sufficiently close to 1 ($\gamma = 0.99$ is sufficient), then there is a value of $A(P)$ assigned to a majority of planes $P$, and this is the unique low-degree polynomial close to $f$. To prove this, choose a value for each point $x \in F^m$ as the most popular value $A(P)$ among all planes containing $x$. Prove that for 90% of points $x$ the value chosen for $x$ is equal to the extensions of the planes in 90% of pairs in $\mathcal{P}$ with $x$ on the intersecting line, and thus the value is the same for 80% of all planes containing $x$. Now prove that for most "good" points the value is the same.

**Theorem 34** *There exists an absolute constant $c \geq 2$ (independent of the degree $d$ and the dimension $m$) such that for $\varepsilon = c\alpha^{1/c}$, if $g_P$ is $\gamma$-consistent then it is $(\gamma - 2\varepsilon m)$-close.*

Note that $\varepsilon m = o(1)$ if we choose a sufficiently large field $F$ (like $|F| = (\log n)^c$), and hence the theorem implies the correctness of the low-degree test.

**Definition 35** *The consistency graph is a weighted graph (undirected, with loops) defined as follows. The set of vertices is the set of $\mathcal{S}^{m-1} \times F_d[x_1, \ldots, x_{m-1}]$. For a vertex $(S, f)$, the polynomial $f$ is interpreted as a mapping from $S$ to $F$.*

*Two vertices $(S, f)$ and $(S', f')$ are connected if either $S$ and $S'$ are parallel (or equal), or they intersect in $S' \in \mathcal{S}^{m-2}$ and $f$ and $f'$ agree on $S'$.*

*Suppose that for each $S \in \mathcal{S}^{m-1}$, $A_S$ is the extension closest to the system $g_P$ (in the sense of the definition of $\gamma$-close). Define the weight of a vertex $(S, f)$ to be*

$$\psi_S(f) = \mathbf{Pr}_{P \subseteq S}[A_S(P) = f].$$

*Define the probability of a vertex to be*

$$\psi(S, f) = \frac{\psi_S(f)}{|\mathcal{S}^{m-1}|}$$

Note that for $m = 3$ we can ignore the $f$ in $(S, f)$, as $S$ ranges over the planes and only $f = g_P$ has a non-zero weight.

We prove Theorem 34 by induction on $m$. The general idea is to prove that the consistency graph is dense (this uses the induction assumption), then to prove that it contains a large clique, and to conclude that this implies that there exists a close low-degree polynomial.

## 9.2 The combinatorial lemma and the case $m = 3$

**Definition 36** *Given a graph $G = (V, E)$ with a probability $\psi$ on vertices, and $v, v' \in V$, let $\beta(v, v')$ be the total probability of $v''$ such that $(v, v''), (v'', v') \in G$.*

*Let $\beta(G)$ be the maximum of $\beta(v, v')$ over $v, v' \in V$ distinct and non-adjacent.*

*For $E' \subseteq V^2$, let $\psi(E') = \mathbf{Pr}_{v,v'}[(v, v') \in E']$, where the probability is taken according to the distribution $\psi$, independently for $v$ and $v'$. Define $\psi(G) = \psi(E)$.*

**Lemma 37** *Let $G = (V, E)$ be a graph probability $\psi$. Then there exists a union of cliques $E' \subseteq E$ which satisfies*

$$\psi(E') \geq \psi(E) - O(\sqrt{\beta(G)}).$$

**Proof**. If there is any vertex with the total probability of its neighborhood at most $\sqrt{\beta(G)}$, remove it. If not, find a vertex $v$ non-adjacent but connected to some $v'$ and disconnect the neighborhood of $v$ from the rest of the graph. Repeat. If you cannot continue, you have a union of cliques. □

**Lemma 38** *For any consistency graph $G$, $\beta(G) \leq \alpha$.*

**Proof**. By straightforward counting and algebra. □

**Lemma 39** *Let $T$ be a clique in the consistency graph such that $|T| \geq \alpha|\mathcal{S}^{m-1}|$. Then there exists a unique $h \in F_d[x_1, \ldots, x_m]$ which for every $(S, f) \in T$ agrees with $f$ on $S$.*

**Proof**. If we take any $\alpha|F|$ parallel $S \in \mathcal{S}^{m-1}$ spaces and a degree $d$ polynomial on each of them, it gives a unique degree $2d$ polynomial on the whole $F^m$. Since $T$ is large, there exist two such sets of different directions and we obtain $h$ of degree $2d$ (we need two sets of parallel subspaces since we need to deal with the possible extra planes parallel to the first set). It is possible to show that this $h$ actually has degree $d$. □

Now we are ready to prove Theorem 34 for the case of $m = 3$. If $g_P$ are $\gamma$-consistent then $\psi(G) \geq \gamma$ for the consistency graph, by its definition. By Lemma 37 and 38 we can remove $O(\sqrt{\alpha})$ fraction of edges and obtain a union of cliques. Remove all cliques of less than $\alpha$ fraction of vertices; this is at most $\alpha$ fraction of edges. Define an extension consistently on each clique according to Lemma 39. This proves that $g_P$ is $(\gamma - \varepsilon)$-close, since we removed only $O(\sqrt{\alpha}) = o(1)$ fraction of edges.

## 9.3 The general case

For the induction step, we need to prove that the consistency graph is dense. If $\gamma = 1 - \gamma'$, it is not hard to prove that its density is $1 - c\gamma'$ for some constant $c$. However, this is not good enough since we do not want the constant in the probability to depend on $m$. To overcome this, we need to use the induction assumption of Theorem 34 for small values of $\gamma$.

**Lemma 40** *There exists a constant $c$ such that for any $g_P$ $\gamma$-consistent for $\gamma \geq \varepsilon(2m-1)$, the consistency graph has $\phi(G) \geq (\gamma - \varepsilon(2m-2))^c/c$.*

**Proof**. Let $\Delta = \gamma - \varepsilon(2m-2) \geq \varepsilon$. The average consistency restricted to a random $S \in \mathcal{S}^{d-1}$ is $\gamma$, since every pair of intersecting planes lies in the same number of subspaces. It follows that at least $\Delta/2$ subspaces are $(\gamma - \Delta/2)$-consistent and thus by the induction assumption they are $\Delta/2$-close. It is possible to prove that in each such space there exists a value of the best extension which has probability at least $\Delta/4$. We consider only the vertices in the consistency graph corresponding to these "good" spaces and values. This already reduces the density of vertices to about $\Delta^2$ (and the density of edges to $\Delta^4$). Further averaging arguments show that a sufficient fraction of these vertices is connected by an edge. □

The next lemma follows by Lemmas 37, 38, and 39, similarly as the case $m = 3$.

**Lemma 41** *Given any $g_P$ $\gamma$-consistent for $\gamma \geq \varepsilon(2m-1)$, there exists $f \in F_d[x_1, \ldots, x_m]$ consistent with $g_P$ for a $\sqrt{\alpha}$ fraction of planes $P$.*

Now we are ready to prove the theorem. Construct a maximal sequence $f_1, f_2, \ldots, f_k$ of distinct polynomials in $F_d[x_1, \ldots, x_m]$, such that $f_i$ is consistent with $g_P$ on some set of planes $\Sigma_i$ which is at least a $\sqrt{\alpha}$ fraction of all planes and in addition all sets $\Sigma_i$ are pairwise disjoint. Let $W$ be the set of remaining planes.

Define an extension to be $f_i$ on all planes in $\Sigma_i$, and arbitrary on $W$. We need to bound the number of consistent pairs from $\mathcal{P}$ in which the planes do not belong to the same $\Sigma_i$. By the previous lemma, the probability that a random pair from $\mathcal{P}$ is consistent and belongs to $W^2$ is at most $\varepsilon(2m-1)$; otherwise we could construct an additional $f_i$ consistent on many planes of $W$. The fraction of pairs from $\mathcal{P}$ that are consistent and in $\Sigma_i \times (\mathcal{S}^2 - \Sigma_i)$ for any fixed $i$ is at most $\alpha$. Clearly $k \leq 1/\sqrt{\alpha}$, and hence the total fraction of such edges is at most $\alpha/\sqrt{\alpha} = \sqrt{\alpha} \leq \varepsilon$. Hence, the extension defined above proves that $g_P$ are $(\gamma - 2\varepsilon m)$-close.

# 10 Free bits and non-approximability of clique [BGS, MPS98]

In Theorem 10 we have constructed a graph with vertices of the form $(r, a)$ where $r$ is a random seed and $a$ is an accepting answer. We argued that there are $2^q$ possible answers, where $q$ is the number of query bits, and derived a non-approximability result from that. But $q$ is not the most important parameter here, it is better to use the number of possible *accepting* answers, whose logarithm is called *free bits*. To get the best result, we need to use the amortized number of free bits and a randomized reduction which has better parameters than the deterministic. Later we show how to obtain a proof system in which the amortized number of free bits is $\varepsilon$ for arbitrary $\varepsilon > 0$.

## 10.1 Promise problems and reductions

For precise statement of the non-approximability results we deal with so-called promise problems.

**Definition 42** *A promise problem is a pair of disjoint languages $(A, B)$. An algorithm for a promise problem has to accept on an input from $A$ and reject on any input from $B$; on other input it can behave arbitrarily. Randomized algorithms are defined similarly (on an input outside of $A$ and $B$, there is no restriction, so even in a* **BPP** *algorithm the probability of accepting them may be $1/2$). A language $L$ is identified with the promise problem $(L, \bar{L})$.*

Next we define reductions between promise problems; we consider Karp (i.e., many-one) reductions, both deterministic and randomized.

**Definition 43** *A deterministic reduction from $(A, B)$ to $(A', B')$ function $T$ computable in polynomial time such that for any $x \in A$, $f(x) \in A'$ and for any $x \in B$, $f(x) \in B'$.*

**Definition 44** *A randomized reduction with perfect completeness from $(A, B)$ to $(A', B')$ is a function $T$ computable in probabilistic polynomial time such that:*

- *If $x \in A$ then* $\mathbf{Pr}[T(x) \in A'] = 1$.
- *If $x \in B$ then* $\mathbf{Pr}[T(x) \in B'] = 1 - 2^{-k}$,

*where $k$ is an arbitrary constant (more precisely, $k$ is given to $T$ in unary and $T$ has to run in time polynomial in $k$).*

A general randomized reduction would allow an error also for $x \in A$.

**Definition 45** Gap-Max-Clique$_{c,s}$ *is the promise problem $(A, B)$ where $A$ are all graphs on $N$ vertices with a clique of size $c$ and $B$ are all graphs on $N$ vertices with no clique of size $s$.*

**Theorem 46** *Suppose that* Max-Clique *has a polynomial $\alpha$-approximation algorithm for some $\alpha < c/s$ and* **NP** *reduces to* Gap-Max-Clique$_{c,s}$ *via a randomized reduction with perfect completeness. Then* **NP** = **coNP** = **ZPP**.

*Suppose that* Max-Clique *has a polynomial $\alpha$-approximation algorithm for some $\alpha < c/s$ and* **NP** *reduces to* Gap-Max-Clique$_{c,s}$ *via a deterministic reduction. Then $P = NP$.*

## 10.2 Refined PCP classes

We introduce new parameters in the definition of the class **PCP**. Also, since we are interested in the exact constants, we no longer allow a constant factor in the definitions.

**Definition 47** *A promise problem $(A, B)$ is in $\mathbf{PCP}_{c,s}[r, q]$ if there exists a verier $V$*

- *On any input of length $n$, $V$ uses at most $r(n)$ random bits and accesses at most $q(n)$ bits of the proof $\Pi$.*
- *If $x \in A$ then there is a proof $\Pi$ such that $\mathbf{Pr}[V(x, \Pi)$ accepts$] \geq c$.*
- *If $x \notin A$ then for any proof $\Pi$, $\mathbf{Pr}[A(x)$ accepts$] \leq s$.*

*The parameters $c$ and $s$ are called completeness and sourness, their ratio $g = c/s$ is called the gap. We omit $c$ in the notation if it is $1$.*

**Definition 48** *A verifier has free bit complexity $f$ if for each random string $r$ there exist at most $2^f$ views of the query bits on which it accepts; in addition these accepting views must be generated in time polynomial in the size of the input of the verifier.*

*A verifier with gap $g$ has amortized free bit complexity $\bar{f}$ if it has free bit complexity $f \cdot \log_2 g$.*

*$\mathbf{FPCP}_{c,s}[r, f]$ is the class of all promise problems with a verifier in $\mathbf{PCP}_{c,s}[r, q]$ (for some $q$) with free bit complexity $f$.*

*$\overline{\mathbf{FPCP}}_c[r, f]$ is the class of all promise problems verifier in $\mathbf{PCP}_{c,s}[r, q]$ (for some $q$ and $s$) with amortized free bit complexity $f$.*

Note that in the definition of $\overline{\mathbf{FPCP}}$ the soundness is arbitrary, only bounded implicitly by $f$. The key reduction is stated in the following theorem.

**Theorem 49** *For any constant $\varepsilon > 0$, any problem in $\overline{\mathbf{FPCP}}[O(\log n), f]$ reduces to the promise problem* Gap-Max-Clique$_{c,s}$ *with gap $c/s \geq N^{1/(1+f+\varepsilon)}$ via a randomized reduction with perfect completeness.*

Thus, to obtain the desired result on non-approximability it is now sufficient to find verifiers for **NP** with arbitrarily small free bit complexity. (It is interesting that the reverse implication also holds, i.e., a strong non-approximability result for Max-Clique also implies existence of such verifiers.)

The reduction to Max-Clique we already know gives the following lemma.

**Lemma 50** *Any problem in $\mathbf{FPCP}_{2^{-t}}[r, f]$ reduces to* Gap-Max-Clique$_{c,s}$ *with gap $c/s = N^{t/(r+f)}$ via a deterministic reduction.*

So, to finish the proof of the theorem, we need to decrease the number of random bits close to the logarithm of the gap. We first increase the gap to be $\Theta(\log n)$ by simply repeating the verifier several times, and then reduce the number of random bits needed by the verifier. The second part is done by proving that polynomially many random seeds are likely to be good for all possible proofs; this is similar to proving that e.g. **BPP** has polynomial circuits. Choosing the seeds can be viewed as a part of the reduction, and the new problem then needs only a few random bits to choose one of the seeds. Thus, in some sense, we are moving the randomness from the verifier into the reduction.

**Lemma 51** *For any constant $\varepsilon > 0$, any problem in $\overline{\mathbf{FPCP}}[O(\log n), f]$ reduces to some problem in $\mathbf{FPCP}_{(1+\varepsilon)2^{-t}}[(1 + \varepsilon) \cdot t, f \cdot t]$ for some $t = O(\log n)$ via a randomized reduction with perfect completeness.*

If we want to base the hardness result on $\mathbf{P} \neq \mathbf{NP}$, we need to use deterministic reductions. This is done by derandomizing the previous reduction using random walks on expanders.

**Theorem 52** *For any constant $\varepsilon > 0$, any problem in $\overline{\mathbf{FPCP}}[O(\log n), f]$ reduces to the promise problem* Gap-Max-Clique$_{c,s}$ *with gap $c/s \geq N^{1/(2+f+\varepsilon)}$ via a deterministic reduction.*

# 11 Improved inapproximability results for Max-3-SAT [BGS, MPS98]

In the future proofs it is useful to start with a verifier for formulas in which all the variables appear the same number of times. We define a problem 3-SAT($k$) (resp. 3-SAT(E$k$)) as a satisfiability problem for all 3-CNF formulas in which each variable appears at most $k$ times (resp. exactly $k$ times). We define similarly E3-SAT(E$k$), requiring also that all clauses have exactly 3 different variables, and we define the maximization and gap problems for all classes of formulas.

**Lemma 53** *There exists a constant $s < 1$ such that both promise problems* Gap-Max-3-SAT(E3)$_{1,s}$ *and* Gap-Max-E3-SAT(E5)$_{1,s}$ *are NP-complete.*

**Proof**. First we reduce Gap-Max-3-SAT$_{1,s}$ to Gap-Max-3-SAT$(k)_{1,s'}$, for some constant $k$ and soundness $s' < 1$. We call an undirected graph expander if for every partition of the vertices into $S$ and $T$, $|S| \leq |T|$, there are at least $|S|$ edges between $S$ and $T$. Given $m$, it is possible to construct an expander of some constant degree $d$ and $m + o(m)$ vertices. (A random graph is certainly an expander; that gives us a randomized reduction even without the construction of explicit expanders.) Suppose that the variable $x_i$ has $m_i$ occurrences in the given formula. Construct an expander with (about) $m_i$ vertices, assign a new variable $x_{i,j}$ to each vertex and modify the formula as follows. Change all the old occurrences of $x_i$ into distinct $x_{i,j}$, and for each edge $(j, j')$ in the expander add clauses $\neg x_{i,j} \vee x_{i,j'}$ and $x_{i,j} \vee \neg x_{i,j'}$. Do this for all variables. Now there exists an optimal assignment for the new formula which for every $i$ assigns to all $x_{i,j}$ the same value, and thus is an assignment for the old formula. To see this, modify any assignment to the new formula so that all the vertices of an expander are assigned the majority value on this expander. By the expansion property, the number of old clauses that may become unsatisfied is at most the number of new clauses that become satisfied. Since the number of clauses increased only by a constant factor, and a satisfiable formula is mapped to a satisfiable formula, the reduction follows.

Next we guarantee that the formula has three variables in each clause and at least two occurrences of each variable. First remove all variables that occur only single time. Instead of $x \vee y$ include two new clauses $x \vee y \vee z$ and $x \vee y \vee \neg z$, similarly for clauses of length 1 and 0. This changes the error and the number of occurrences of variables by a constant factor only.

To reduce to 3-SAT(E3) now use a similar transformation as before, only instead of connecting the multiple occurrences by an expander, we connect them by a directed cycle, and including only one new clause $x_{i,j} \vee x_{i,j'}$ for each edge. Since the number of occurrences of each variable is bounded by a constant, the new soundness is a constant. Each variable now appears in one clause of length 3 and two clauses of length 2. To reduce further to E3-SAT(E5), expand the clauses with length 2 using new variables as above. The new variables $z_i$ now have only two occurrences, so add more new variables $t_i$ and some clauses of the form $z_i \vee z_j \vee t_i$ (or variants with more $t_i$ variables). These can always be satisfied, since $t_i$ occurs only positively, and it is easy to arrange that all the new variables occur exactly 5 times, while their number is at most linear. □

# 12 Two-prover systems and parallel repetition [Raz, MPS98]

The parallel repetition theorem for two prover systems is a powerful tool to decrease the error at a reasonable cost. We define these systems only with perfect completeness.

**Definition 54** *Two-prover single-round proof system consists of two oracles and a randomized verifier which makes a single query to each oracle. The system accepts a gap problem $(A, B)$ if for every $x \in A$ there exist oracles so that the verifier always accepts, but for $x \in B$ and any oracles it accepts with probability at most $s < 1$.*

Clearly, any two-prover verifier can be also viewed as a **PCP** verifier (with the appropriate parameters). The proof would now contain two separate tables for the two oracles and access each part once. The other direction of the transformation is also easy. If a **PCP** verifier asks $q$ queries, then let the first oracle contain all the possible $q$-tuples of answers, and the other oracle contain all the individual answers. The verifier would ask the first oracle the desired $q$-tuple and the other oracle a random question of the $q$-tuple; if the answer to this question is not identical, it would reject. So, if the $q$-tuple answer is not consistent with the other oracle, the verifier rejects with probability at least $1/q$. Otherwise the system behaves as a **PCP** system, thus the soundness is bounded away from 1. (More exactly, it is $1-(1-s)/q$, where $s$ is the original soundness.)

Since we are restricted to a single query to each prover, it is not obvious how to reduce soundness. We define a parallel repetition as follows. The verifier chooses $u$ independent instances of pairs of question in the original system, sends to each prover $u$-tuple of its questions, and gets a $u$-tuple of answers from each; it accepts only if all $u$ instances accept. One could expect that this reduces soundness from $\varepsilon$ to $\varepsilon^u$, but the following example shows that this is not true.

Suppose that (for an input not in the language) the verifier sends to each prover one random bit, say $a_1$ and $a_2$. Each prover answers by a pair $(i, c)$, and the verifier accepts if the answers are identical and moreover $a_i = c$. It is easy to see that in the single repetition, the error is at least $1/2$. We describe a strategy for 2 repetitions which allows the provers to achieve an error of $1/2$ instead of $1/4$. The first verifier upon receiving $(a_1, b_1)$ answers $((1, a_1), (2, a_1))$; the second prover upon receiving $(a_2, b_2)$ answers $((1, b_2), (2, b_2))$. It is easy to check that the verifier rejects iff $a_1 \neq b_2$, which happens with probability $1/2$.

However, the following nontrivial theorem holds, and thus we know that the soundness of parallel repetition decreases exponentially with the number of repetitions. In fact, the constant depends only on the soundness of the original system and the number of possible answers from the provers.

**Theorem 55** *For any two-prover single-round proof system there exists a constant $s$ such that the soundness of the $u$ parallel repetitions is at most $s^u$. The constant $s$ depends only on the soundness of the original system and the size of the answers obtained from the oracles (in the single repetition); $s < 1$ whenever the original soundness it less than 1.*

## 13 The long code [BGS, MPS98]

Now we know that there exists a constant $s$ such that for any $u$ the following verifier has soundness $c^u$. Given a E3-CNF with exactly 5 occurrences of each variable, pick $u$ random clauses, call the set of all variables in them $W$, and one random variable from each clause, call this set of variables $U$. Ask one prover for the assignment to all variables in $W$ and the other prover for the assignment to $U$. Accept if the answers are consistent and the $u$ clauses are satisfied by this assignment. Note that $U$ has a uniform distribution among all sets with $|U| = u$, since each variable appears the same number of times.

This verifier is not good enough as it asks too many queries and even needs too many free bits. We modify it by introducing a new, very wasteful code, called the long code.

**Definition 56** *Given a string of $u$ bits $x$, the* long code *of $x$ is the string $A$ of $2^{2^u}$ bits indexed by all functions $f : \{0,1\}^u \to \{0,1\}$ defined by $A(f) = f(x)$.*

In the **PCP** systems we will study the perfect proof consists of the long code of the satisfying assignment on $U$, for each set of $u$ variables $U$, and the long code of the same satisfying assignment on $W$, for each set of $3u$ variables $W$. Since $k$ is a constant, we may assume that $|W| = 3u$ (the probability that this is not true is $O(u^2/n)$, so if we always accept in such a case, the soundness does not increase significantly). The proof systems will differ in the way they test the long codes.

## 14 Approximating systems of linear equations [Hastad1]

In this section we look at the problem Max-E3-Lin defined as follows. Given a multiset of linear equations over $GF(2)$ such that each equation contains exactly three variables, find an assignment such that it satisfies as many equations as possible. Note that a random assignment satisfies 1/2 of the equations, and that it is easy to decide whether it is possible to satisfy all the equations. We will prove that no better algorithms exist. We do it using the following theorem which we prove later.

**Theorem 57** *For any $\varepsilon > 0$, there exists an* **NP**-*complete problem with a proof system in* $\mathbf{PCP}_{1-\varepsilon,1/2+\varepsilon}[O(\log n), 3]$ *such that for any queries its acceptance depends only on the parity of the three bits read from the proof.*

**Theorem 58** *For any $\varepsilon > 0$,* Gap-Max-E3-Lin$_{1-\varepsilon,1/2+\varepsilon}$ *is NP-complete.*

**Proof**. Consider the bits of the oracle as unknowns, and the possible queries as equations. □

An easy corrolary is a hardness result for Max-E3-SAT . (Modifying the techniques used it is even possible to obtain perfect completeness.)

**Theorem 59** *For any $\varepsilon > 0$,* Gap-Max-E3-SAT$_{1-\varepsilon,7/8+\varepsilon}$ *is NP-complete.*

Now we will prove Theorem 57. We have the two prover system for Gap-Max-E3-SAT$(5)_{1,s}$ for some $s < 1$ in which the verifier picks $k$ clauses at random, denote their variables $W$, and the set $U$ of $u$ variables, one chosen randomly form each clause, sends to the first prover $W$, to the second prover $U$, receives the assignment to these variables, checks that they are consistent and satisfy all the clauses of $W$. For any $\delta > 0$ there exist $u$ (not depending on the size of the formula!) such that this system has soundness $\delta$.

Our new proof system will contain slightly modified long codes of the assignments to the variables $U$ and $W$ for each $U$ and $W$. Let $A$ be the long code for $U$, we may assume that for any $f$, $A(\neg f) = \neg(A(f))$. Let $B'$ be the long code for $W$ and let $h(y)$ be the function asserting that $y$ is a satisfying assignment to $W$. Define $B(g) = B'(g \wedge h)$; again we can assume that $B(\neg g) = \neg(B(g))$.

From now on we denote the value true by $-1$ and false by 1. This is useful, since our main tool will be Fourier analysis.

The verifier is defined as follows. After choosing $W$ and $U$, choose independently random functions

$$\begin{aligned} f &: \{0,1\}^U \to \{-1,1\}, \\ g &: \{0,1\}^W \to \{-1,1\}, \text{ and} \\ e &: \{0,1\}^W \to \{-1,1\}. \end{aligned}$$

$f$ and $g$ are chosen uniformly, $e$ is at each point equal to 1 with probability $1-\varepsilon$, the values at different points are independent. Put

$$g'(y) = f(y)g(y)e(y).$$

Accept if $A(f)B(g)B(g') = 1$. The honest proof is accepted with probability $1-\varepsilon$ (it would be always accepted if there is no $e$).

Suppose that on some proof we accept with probability at least $1/2+\varepsilon$. Our goal is to find a strategy for the two provers that causes the original verifier to accept with some probability $p$ which depends only on $\varepsilon$ (and not $u$). Then we are done, by choosing $u$ sufficiently large.

We express $A$ and $B$ in the Fourier basis as

$$\begin{aligned} A(f) &= \sum_{\alpha \subseteq \{0,1\}^U} \hat{A}_\alpha \prod_{x \in \alpha} f(x) \\ B(g) &= \sum_{\beta \subseteq \{0,1\}^W} \hat{B}_\beta \prod_{y \in \beta} f(y). \end{aligned}$$

The modifications of the long code above guarantee that $\hat{A}_\emptyset = 0$, and that if $\hat{B}_\beta \neq 0$ than all elements in $\beta$ are satisfying assignments for $W$ (and $\beta \neq \emptyset$).

Now, since the accepting probability is $1/2+\varepsilon$, we have

$$\begin{aligned} \varepsilon &\leq \mathbf{Exp}[A(f)B(g)B(g')] \\ &= \sum_{\alpha,\beta,\beta'} \hat{A}_\alpha \hat{B}_\beta \hat{B}_{\beta'} \mathbf{Exp}\left[\prod_{x\in\alpha} f(x) \prod_{y\in\beta} g(y) \prod_{y\in\beta'} g'(y)\right]. \end{aligned}$$

From the independence of choices of functions on different points it follows that the only terms where the inner expectation is non-zero are those where $\beta=\beta'$ and $\alpha=\pi(\beta)$, where $\pi(\beta)$ obtains all elements of $\beta$ restricted to $U$ taken with appropriate parity (i.e., a restriction appears in the projection only if there is odd number of elements with that restriction. In these terms the values of $A(f)$, $B(g)$, and $B(g')$ cancel and we get

$$\varepsilon \leq \sum_\beta \hat{A}_{\pi(\beta)} \hat{B}^2_\beta (1-\varepsilon)^{|\beta|}$$

Due to the last factor, coming from the error function $e$, most of the weight is on the coefficients with small $\beta$. Discarding all the other $\beta$ and also $\alpha$ such that $\hat{A}_\alpha < \varepsilon/4$, we are still left with a sum significantly larger than 0.

Now, the strategy for the first prover is to take one of the remaining values of $\beta$, with a probability proportional to $\hat{B}^2_\beta$, choose randomly one of $y\in\beta$, and return this $y$. Similarly, the second prover takes one of the remaining $\alpha$, with a probability proportional to $\hat{A}_\alpha$, a random element of $\alpha$, and returns it. Since the remaining sum is large, with reasonable probability the provers choose $\alpha$ and $\beta$ such that $\alpha=\pi(\beta)$. In such a case they win with probability at least $1/|\beta|$, which is sufficient since the remaining terms have small $\beta$.

## 15 Small number of amortized bits [SuTr, SaTr]

We need to prove that $\mathbf{NP} \subseteq \overline{\mathbf{FPCP}}[O(\log n),\varepsilon]$ for any $\varepsilon>0$, i.e., that some NP-complete problem has verifiers with an arbitrarily small number of amortized free bits.

As in the previous section, we start from a two prover system for Gap-Max-E3-SAT$(5)_{1,s}$ for some $s<1$ in which the verifier picks $k$ clauses at random, denote their variables $W$, and the set $U$ of $u$ variables, one chosen randomly form each clause, sends to the first prover $W$, to the second prover $U$, receives the assignment to these variables, checks that they are consistent and satisfy all the clauses of $W$. For any $\delta>0$ there exist $u$ such that this system has soundness $\delta$.

The honest proof in our proof system will (again) contain long codes of the assignments to the variables $U$ and $W$ for each $U$ and $W$, "folded" so that for any long code $A$ and any function $f$, $A(\neg f)=\neg(A(f))$.

The verifier $V$ proceeds as follows

- Choose a random set $U$ of $u$ variables, choose $k$ random sets $W_j$ of clauses containing these variables.
- Apply the following inner verifier $V^{in}$ to their long codes $A$, $B_1$, …, $B_k$:
  - Choose independently for $i,j=1,\dots,k$ random functions

    $$\begin{aligned} &f_i : \{0,1\}^U \to \{-1,1\}, \\ &g_j : \{0,1\}^W \to \{-1,1\}, \text{ and} \\ &e_{i,j} : \{0,1\}^W \to \{-1,1\}. \end{aligned}$$

    $f_i$ and $g_j$ are chosen uniformly, $e_{i,j}$ is at each point equal to 1 with probability $1-\varepsilon$, the values at different points are independent.
  - Put $g'_{i,j}(y)=f_i(y)g_j(y)e_{i,j}(y)$ and $T_{i,j}=A(f_i)B_j(g_j)B_j(g'_{i,j})$ for all $i$ and $j$.

– Accept if $T_{i,j} = 1$ for all $i$ and $j$.

The honest proof is accepted with probability $(1-\varepsilon)^{k^2}$ (it would be always accepted if there is no $e$). The verifier reads $k^2 + 2k$ bits, however, only $2k$ of them are free bits. It remains to prove that, for a sufficiently large $u$, the soundness is arbitrarily close to $2^{-k^2}$; then it follows that the amortized number of free bits is $O(1/k)$, which can be made arbitrarily small.

Given a long code $A$, let $D(A)$ be a randomized decoder defined as follows. Compute Fourier coefficients $\hat{A}_\alpha$, choose a random $\alpha$ with probability $\hat{A}_\alpha^2$ (note that this is a probability distribution), and choose uniformly random $x \in \alpha$ (note that $\alpha$ is non-empty, as $A$ is folded). For given $A$, $B_1, \ldots, B_k$, we say that $D$ succeeds if at least two of $D(A)$, $D(B_1), \ldots, D(B_k)$ are consistent on the variables from $U$ (the different runs of $D$ make independent random choices). The following lemma is the key technical part of the analysis.

**Lemma 60** *For any $A$, $B_1, \ldots, B_k$, if $V^{in}$ accepts with probability $2^{-k^2} + \delta$ for a $\delta > 0$ then $D$ succeeds with probability at least $\delta' > 0$ which depends only on $\delta$.*

**Proof**. Let $T_S = \prod_{(i,j)\in S} T_{i,j}$ for $S \subseteq [k] \times [k]$. The probability that $V^{in}$ accepts is equal to

$$\prod_{i,j=1}^{k} \frac{1+T_{i,j}}{2} = 2^{-k^2} + 2^{-k^2} \sum_{\emptyset \neq S \subseteq [k]\times[k]} T_S.$$

It remains to prove that if for some nonempty $S$, $T_S > \delta$, then $D$ succeeds with probability $\delta'$. In [SuTr] this is shown for $S = [2] \times [k]$ using Fourier analysis, in [SaTr] the general case is reduced to this one. □

Using $D$ and a proof for a verifier $V$, we define a strategy for the two prover game. Let $A$, $B$ be the long codes from a proof on which $V$ accepts with highest probability. The second prover, upon receiving $W$, generates $D(B)$ (where $B$ is the assumed long code of the satisfying assignment to $W$) . The first prover, upon receiving $U$, with probability $1/k$ returns $D(A)$, otherwise generates a random $W$ containing $U$ and returns (the projection of) $D(B)$.

Consider the following experiment: upon receivin $U$ and $W$, generate $W_1, \ldots, W_k$ randomly (compatible with $U$), and replace a random one of them by $W$. Accept if the decoder $D$ succeeds on the corresponding input $A$, $B_1, \ldots, B_k$. If this experiment succeeds with probability $p$, it is easy to see that the two prover strategy succeeds with probability at least $p/k^2$.

Now assume that the verifier $V$ accepts with probability $2^{-k^2} + 2\delta$. By averaging, for at least $\delta$ fraction of choices $U$, $W_1, \ldots, W_k$, the inner verifier $V^{in}$ accepts with probability at least $2^{-k^2} + \delta$, and thus $D$ succeeds with probability at least $\delta'$. By the previous paragraph, it follows that the previous two prover strategy accepts with probability at least $\delta\delta'/k^2$. For a sufficiently large $u$ it means that the input formula must be satisfiable.