A Sub-Constant Error-Probability PCP Characterization of NP

PART I: The Main Proof
(Preliminary Version)*

Ran Raz
Weizmann Inst. of Science
ISRAEL

Shmuel Safra
Tel-Aviv University
ISRAEL

October 12, 1996

Abstract

This paper establishes a new characterization of NP in terms of PCP, being the first such exact characterization to exhibit a sub-constant error-probability with constant number of accesses.

Specifically, our theorems state the following:

- For any $\beta < 1/4$, membership in any NP language can be verified with a constant number of accesses, each reading $O(\log^\beta n)$ bits, and such that the error-probability is at most exponentially small in the number of bits read, i.e., $2^{-\log^\beta n}$.
- For any $\beta < 1$, membership in any NP language can be verified with a constant number of accesses, each reading $\log^\beta n \cdot \log\log^c n$ bits (for some fixed constant c), and such that the error-probability is at most $2^{-\log^\beta n}$.

This new characterization of NP, almost immediately implies approximating set-cover to within logarithmic factors to be NP-hard.

The main technical lemma utilized for that purpose presents a new consistency-test, implying a new low-degree-test, which is the first to exhibit sub-constant error-probability with constant number of accesses. Part II of this work describes that consistency-test.

Introduction

Characterizing the class NP, by itself or with respect to other computational-complexity classes, is perhaps one of the most fundamental avenues of research in the mathematics of computing. Since the early days, when the classes P and NP were defined, and the question

*To be found at URL http://www.math.tau.ac.il/school/courses/PCP was posed as to whether they are the same or do they differ, many problems were shown to be in NP, while many shown to be NP-hard, thereby increasing the weight on finding stricter characterization for the class NP.

Cook's Characterization

The characterization of NP most commonly applied in order to establish NP-hardness of various problems, is that of Cook (and slight variations of it) which characterizes NP in terms of satisfiability of Boolean-function-system as follows:

Theorem 1 ([Coo71]) *For every $L \in \mathsf{NP}$ there exists a polynomial-time procedure that, on input $I \in \{0,1\}^n$, constructs a set of Boolean-functions*

$$\Phi_{L,I} = \{\varphi_1, ..., \varphi_l\}$$

over Boolean variables $\mathcal{Y} = \{y_1, ..., y_l\}$ — representing the witness — such that

1. *Each $\varphi_i \in \Phi_{L,I}$ depends on a constant number of variables in $\mathcal{Y}$*
2. *$I \in L$ if and only if there exists an assignment $A: \mathcal{Y} \to \{0,1\}$ such that all $\varphi_i \in \Phi_{L,I}$ evaluate to* `true` *on A.*

Hence, in order to establish NP-hardness of a certain problem, it is only necessary to come up with an efficient algorithm that, given such a function-system, computes an instance of the problem whose outcome determines uniquely whether the function-system is satisfiable or not. Many problems were shown, that way, to be NP-hard: the largest-clique of a graph, chromatic-number of a graph or set-cover (shown so by Karp [Kar72]) are just a few to mention.

Approximation Problems

A major source of problems for which neither polynomial-time algorithm was known, nor were they shown to be NP-hard, is *approximation* problems. Once an optimal solution is not likely to be found, the next to best natural replacement is a non-optimal solution. For example, an algorithm that does not color a graph with the smallest number of colors possible, instead partitions the vertexes into a larger number of monochromatic groups, nevertheless, where this larger number is known to deviate from the optimal only by a known factor.

Much effort was invested trying to come up with reasonable (say even sub-exponential) algorithms for many of these problems, while, at the same time, serious effort was focused at coming up with improved translations that would show those approximation problems to be NP-hard.

An avenue of research, initiated in [FGL$^+$91], suggested a new methodology for proving hardness results for approximation problems. The method takes a twist on results of a seemingly unrelated area — that of interactive proofs [GMR89, Bab85, BGKW88, LFKN92, Sha92, BFL91], specifically that of [BFL91] — in order to construct, for any $L \in \mathsf{NP}$ a Boolean-function-system more suitable for proving hardness of approximation problems.

In other words, [FGL$^+$91] proposes that, for the purpose of proving hardness results for approximation problems, one — instead of trying to come up with improved translations taking Cook's characterizations of NP as a starting point — should strengthen the characterization, so that it becomes suitable for proving NP-hardness of such problems. [FGL$^+$91] could not obtain an exact characterization of NP however, as the function-systems constructed there were not polynomial in the input-size, hence, could only claim *almost* NP-hard results.

PCP Characterizations of NP

A hierarchy of complexity classes, referred to as the PCP hierarchy, was later introduced [AS92], making explicit the parameters of importance in [FGL$^+$91]. [AS92] managed to establish an exact characterization of NP in terms of PCP, so as to enable proofs for NP-hardness results for approximation problems.

Characterizations of NP in terms of PCP differ from Cook's characterization only in regards to how many of the Boolean-functions can be simultaneously satisfied in case the input I is not in L (see condition 2). The proof method of [AS92] was improved [ALM$^+$92] to yield the following characterization of NP:

Theorem 2 ([AS92, ALM$^+$92]) *For every $L \in \mathsf{NP}$ there exists a polynomial time procedure that, on input $I \in \{0,1\}^n$, constructs a set of Boolean-functions*

$$\Phi_{L,I} = \{\varphi_1, ..., \varphi_l\}$$

over Boolean variables $\mathcal{Y} = \{y_1, ..., y_l\}$ such that

1. *Each $\varphi_i \in \Phi_{L,I}$ depends on a constant number of variables in $\mathcal{Y}$*
2. *For every input instance I:*
   - *if $I \in L$ then there exists an assignment A to the variables $\mathcal{Y}$ such that all $\varphi_i \in \Phi_{L,I}$ evaluate to* `true` *on A.*
   - *if $I \notin L$ then for any assignment A to the variables $\mathcal{Y}$, more than $\frac{l}{2}$ of $\varphi_i \in \Phi_{L,I}$ evaluate to* `false` *on A.*

In contrast to Cook's characterization, here, if the input I is not in L, at least half of the Boolean-functions are unsatisfied, no matter which assignment is chosen.

One should note that the variation of this characterization of NP from the old one, however syntactically small, is in fact not at all minor. It causes both a great increase in how difficult it is to prove the theorem, and, in return, enables quite stronger implications deduced.

Characterizations of NP in terms of the PCP hierarchy — though have taken around 20 years to be formulated — seem now as the most natural extension of the old characterization [Coo71], if one has in mind proving hardness results for approximation problems: Theorem 2 immediately implies that, given a Boolean-function-system, approximating the maximum number of functions satisfied by any single assignment, to within $\sqrt{2}$, is NP-hard. This is the case since, the outcome of any algorithm that approximates that number to within that factor would determine precisely which of the cases in condition 2 is the one true for the specific input-instance.

Stronger PCP Characterizations of NP

[BGLR93] refined the parameters governing the PCP hierarchy and conjectured a stronger PCP characterization of NP. The possibility of letting variables range over sets containing more than two elements was not considered in [AS92]. Once a larger range is allowed, the number of variables each Boolean-function depends on may be considerably smaller than the total number of bits the function takes as input. Hence, NP can possibly be characterized as in the above except that, while the number of variables remains constant, their range becomes larger, thereby achieving smaller, hopefully sub-constant, error-probability. The error-probability is defined to be the fraction of functions that can be satisfied by any single assignment in case the input-instance is not in the original language.

Adopted to the function-system terminology, [BGLR93]'s conjecture is as follows:

Conjecture 3 ([BGLR93]) *For every $L \in \mathsf{NP}$, let $M(n)$ be at most logarithmic in n, there exists a polynomial time procedure that, on input $I \in \{0,1\}^n$, constructs a set of Boolean-functions*

$$\Phi_{L,I} = \{\varphi_1, ..., \varphi_l\}$$

over variables $\mathcal{Y} = \{y_1, ..., y_l\}$ each ranging over $[1, ..., 2^{M(n)}]$ such that

1. *Each $\varphi_i \in \Phi_{L,I}$ depends on a constant number of variables in $\mathcal{Y}$*
2. *For every input instance I:*
   - *if $I \in L$ then there exists an assignment A to the variables $\mathcal{Y}$ such that all $\varphi_i \in \Phi_{L,I}$ evaluate to* `true` *on A.*
   - *if $I \notin L$ then for any assignment A to the variables $\mathcal{Y}$, at most $\frac{l}{2^{M(n)}}$ of $\varphi_i \in \Phi_{L,I}$ evaluate to* `true` *on A.*

The conjecture takes an extreme view, expecting every bit the functions depend on to decrease the probability of error by a factor of 2.

Our Results

Our results attend to the [BGLR93]'s conjecture, and prove it true for a sizable fraction of the appropriate range:

Theorem 4 *Let $M(n)$ be any function such that $M(n) \leq (\log n)^\beta$ for $\beta < 1/4$, and no smaller than $\log^{(i)} n$ for a constant i (where $\log^{(i)}$ denotes i applications of the $\log$ function). Then for every $L \in \mathsf{NP}$, there exists a polynomial time procedure that, on input $I \in \{0,1\}^n$, constructs a set of Boolean-functions*

$$\Phi_{L,I} = \{\varphi_1, ..., \varphi_l\}$$

over variables $\mathcal{Y} = \{y_1, ..., y_l\}$ which range over $[1, ..., 2^{O(M(n))}]$, such that

1. *Each $\varphi_i \in \Phi_{L,I}$ depends on a constant C_M variables in $\mathcal{Y}$.*

2. *For every input instance I:*
   - *if $I \in L$ then there exists an assignment A to the variables $\mathcal{Y}$ such that all $\varphi_i \in \Phi_{L,I}$ evaluate to* `true` *on A.*
   - *if $I \notin L$ then for any assignment A to the variables $\mathcal{Y}$, at most $\frac{l}{2^{M(n)}}$ of $\varphi_i \in \Phi_{L,I}$ evaluate to* `true` *on A.*

In addition, if one is interested in $M(n)$ closer to $\log n$, the following holds as well:

Theorem 5 *Let $M(n)$ be any function which is at most $(\log n)^{1-\beta}$ for any constant $\beta > 0$, and no smaller than $\log^{(i)} n$ for a constant i (where $\log^{(i)}$ denotes i applications of the $\log$ function). Then for every $L \in \mathsf{NP}$, there exists a polynomial time procedure that, on input $I \in \{0,1\}^n$, constructs a set of Boolean-functions*

$$\Phi_{L,I} = \{\varphi_1, ..., \varphi_l\}$$

over variables $\mathcal{Y} = \{y_1, ..., y_l\}$ which range over $[1, ..., 2^{M(n)\cdot(\log M(n))^c}]$, for some constant c, such that

1. *Each $\varphi_i \in \Phi_{L,I}$ depends on a constant C_M variables in $\mathcal{Y}$.*
2. *For every input instance I:*
   - *if $I \in L$ then there exists an assignment A to the variables $\mathcal{Y}$ such that all $\varphi_i \in \Phi_{L,I}$ evaluate to* `true` *on A.*
   - *if $I \notin L$ then for any assignment A to the variables $\mathcal{Y}$, at most $\frac{l}{2^{M(n)}}$ of $\varphi_i \in \Phi_{L,I}$ evaluate to* `true` *on A.*

Background and Related Work

As mentioned above, the characterizations of NP discussed here seem today as a natural extension of Cook's characterization of NP, adjusted so as to become suitable for translation to approximation problems. Arriving at this model, however, required many ideas and methods, some developed independently, and at the time seemed completely unrelated.

The paradigm presented here, finds its origins in the work on interactive proofs of [GMR89, Bab85] as extended by [BGKW88] and interpreted by [FRS88]. The models were characterized in terms of ordinary computational-complexity classes in [LFKN92, Sha92, BFL91], following ideas of [BK89, BLR90, Lip89]. The work in [BFL91], in particular, served as a trigger for [FGL$^+$91], which focused the attention to characterizations of NP with applications to the hardness of approximation problems, thereby allowing ideas and techniques, introduced in many of the works related to the above topics, to be utilized for that purpose. Scaling-down of the result of [BFL91] to the NP level, was obtained independently to [FGL$^+$91] by [BFLS91] according to a variant model which assumes the input is encoded according to some error-correcting code. This variant model, nevertheless, turned out to be later useful for the technique of [AS92].

Two avenues of research were proposed in [FGL$^+$91]: The first being to try establish an exact characterization of NP, following those guidelines. The other called for new translations, taking as a starting point the function-system described there, in order to show additional approximation problems hard. ([FGL$^+$91] could only establish hardness of the approximated version of one problem, namely, finding the largest clique of a given graph).

A characterization of NP was achieved in [AS92]. The proof method of [AS92] involves a subtle technique, which was utilized in many subsequent works. Put simplistically, some characterization of NP in terms of PCP is composed onto another in order to improve the complexity of the latter. This technique is referred to here as "Composition", and an improved, adopted-to-our-terms version of it appears below (lemma 2).

Resolving the first open problem proposed by [FGL$^+$91] — that of obtaining a characterization of NP in terms of PCP — has led to a flourish of results regarding the hardness of approximation problems, some of which could have had been applied equally to what was established by [FGL$^+$91] (so as to obtain almost-NP-hardness results).

Even before this avenue of research was initiated, [PY91] introduced a class of problems — referred to there as MAX-SNP-complete — each translatable to each other by an approximation-preserving translation. Therefore, proving the approximated version of one problem in this class NP-hard implies all others to be so as well. Approximating the number of functions satisfied, given a function-system, is one problem of that class, hence theorem 2 implies approximating all other problems, to within some constant factor, to be NP-hard.

Approximating the chromatic number of a graph, to within pretty large factors, was shown to be NP-hard in [LY94]. In addition, [LY94] showed approximating set-cover, to within logarithmic factors, to be almost NP-hard — if set-cover can be approximated efficiently, then every NP problem were to have deterministic algorithm that run in time $n^{O(\log n)}$.

[LY94] triggered [BGLR93] to improved on the hardness results with regard to set-cover, and, in addition, redefined the PCP hierarchy, and states the above conjecture. [BGLR93] formulated the conjecture for a somewhat different model, in order to be applied so as to obtain NP-hardness result for approximating set-cover to within logarithmic factors. Indeed, combined with some observations, regarding the relation between the two models [**?**], and other properties the characterization should comply with (see [BGLR93]), NP-hardness for set-cover can be deduced. This is to be elaborated in the full version of the paper.

A construction of a function-system of super-polynomial ($n^{O(\log n)}$ to be precise) size, which, however, otherwise complies with the conditions of the [BGLR93] conjecture, is implied by [Raz94]. An exact characterization of NP, however, cannot be deduce from the analysis there.

Many other results, in that area, followed the above mentioned papers, improving on those results, taking them closer to optimum [BGS95, Hås96b, Hås96a, Fei95] — a more appropriate survey of which is to appear in the full version of this paper.

The Proof

Our proof takes the framework suggested in [AS92] as a starting point. This framework, however, has to be improved, as suggested by [ALM$^+$92] (following the technique of [LS91], improved in [FL92]) in order to maintain the number of variables each Boolean-function depends on constant. This scheme, however, is insufficient when sub-constant error-probability is sought after. Further alterations are therefore required, in order to allow undisturbed flow of such small error-probability between various parts of the proof.

The strategy we have chosen, therefore, in presenting the results reported herein, is to not rely on any previous work, rather rework the proof, extending when necessary, making unclear issues precise, and simplifying where we could. The exposition is therefore complete[1], presenting the entire, face-lifted, proof.

The specific paradigm used in our analysis is syntactically different than previous presentation. It should be viewed however simply as an alternative structure, which in essence is not much different to previous ones, however allows us to define precisely the various necessary parameters. Let us emphasize that our exposition by no means turn the proof into a simple one, nevertheless, makes it precise and certainly, despite proving a stronger claim, not more complex than previous presentation, perhaps simpler.

1 The Basic Paradigm

The current section's purpose is to lay the foundations necessary in order to present our proof for the new characterization of NP in terms of PCP. The framework of the proof, inspired by [AS92] however extended in order to fit our purposes, is rather subtle. The proof constructs first some verification systems of complexities far from optimal, and proceeds gradually to improve the complexity achieved.

The process of improving on the complexity of a given system makes a very particular use of other systems already obtained. This is made precise in the next section. In order to allow that process, however, the notion of a verification system has to be generalized. Consequently, this section involves, almost solely, with introducing the notion of a system of Boolean-functions that verifies, in a particular strong sense, membership of a given input in a specified language.

Overview

Let us begin with an informal overview of the precise definition.

Verification-Systems

A Boolean-function-system is simply a collection of Boolean-functions over a common set of variables (not necessarily Boolean), where each of the functions depends on some of the variables.

A verifying-Boolean-function-system Φ is a Boolean-function-system, where some of the variables are designated to represent some input I whose membership in some language L is in question. The other variables serve as auxiliary data for functions of Φ. Φ is said to be membership-verifying for (I in) L if it is the case that: there exists an assignment to the auxiliary variables satisfying all functions exactly if $I \in L$.

[1]provided last sections are filled in

In other words, some of the variables represent some specific input I, and in order for Φ to be a verifying-function-system for L there must exist a satisfying assignment (to the auxiliary variables) if and only if $I \in L$.

The assignment to the auxiliary variables is sometimes referred to as the witness for membership of I in L.

Strong Sense of Verification

The strong sense of verification, mentioned above, relates to the maximal fraction of functions satisfied in case $I \notin L$ (where the maximum is taken over all assignments to the auxiliary variables). The stronger the system, the smaller that fraction is.

This fraction is referred to as the "error-probability" of Φ, a term inspired by the following scenario. Consider a game of two players, induced by any given verifying-function-system Φ, as follows: Fix an input I; the first player — the "Prover" — commits to a value for each of the auxiliary variables. These values supposedly satisfy all functions of Φ. The other player — the "Verifier" — then proceeds to choose randomly a single function φ of Φ and check whether φ is satisfied by the values the Prover has committed to. The Verifier accepts or rejects I accordingly.

If the input I is "good" (i.e. belongs to L), it must be the case that there is an assignment, which the Prover can commit to, that causes the Verifier to accept for whichever φ of Φ chosen. In the other case where the input I is "bad", the Prover may be able to cause the verifier to accept some of the times, however only with a small probability. The maximum fraction of functions satisfied hence corresponds to the probability the verifier may erroneously accept I.

One may contemplate a similar game in which the verifier is never wrong in its decision, i.e., an error-less verification procedure. A function-system whose game exhibits such a behavior can be trivially constructed, one that consists of a single function which is satisfied exactly when $I \in L$. Such a system, however, seriously exceeds the complexity restrictions applied to function systems constructed below.

Complexity Measures

Our interest herein is with verifying-function-systems featuring small probability of error — significantly smaller than constant — that are, at the same time, quite restricted in complexity according to various parameters. In particular, each function of function-systems constructed below depends on a very small number of variables, and the variables' range is relatively small as well.

The proof for the main theorem — stating that low error-probability verifying-function-systems can be efficiently constructed for all NP languages — starts by constructing function-systems that are not quite as restricted. It then gradually constructs systems of improved complexities, arriving finally at the complexity claimed by the theorem.

Input Scheme

In order to improve the complexity of a function-system Φ, the proof takes advantage of function-systems already constructed. Put somewhat simplistically, each function φ of Φ is replaced by an entire system Ψ_φ, which verifies the assignments to φ's variables to indeed satisfy φ. Part of Ψ_φ's input, therefore, may be auxiliary data. The complexity restrictions imposed on Ψ_φ, however, prohibit Ψ_φ's functions from depending on even a small fraction of those bits. How can then Ψ_φ verify any nontrivial property regarding the assignments to φ's variables?

The definition of a verifying–function-system, consequently, allows the input to be represented using a predefined representation scheme. Some representation schemes, specifically (error-correcting) codes, enable detection of (serious) inconsistencies with only a small number of accesses. It therefore may be possible to construct a function-system whose Boolean-functions depend only on a small (say constant) number of the input-representing variables, still trust the system to verify membership of the encoded (possibly with errors) input in L. Values assigned to the variables designated to represent the input, in that case, are not necessarily a proper representation of a certain input. The system must reject (i.e., allow only a small fraction of its functions satisfied) if major inconsistencies occur.

Another obstacle posed by the application of function-systems within the proof itself, has to do with the fact that the input is broken into several independent segments. The system must be able both to detect inconsistency in any of the segments that constitute its input, as well as verify some property regarding the combined data.

Let us now proceed carefully to the formal definition.

Function-Systems

Let $z_1, ..., z_m$ be variables taking values from some finite set Σ. A *Boolean-Function-System (or simply FS) over* $(z_1, \ldots, z_m)$ is a collection Φ of functions, where each $\varphi \in \Phi$ depends on a subset of $(z_1, ..., z_m)$.

Our interest is with function-systems that are seriously restricted in various parameters. Let us hence introduce the following complexity measures for a function-system Φ:

- $\mathtt{SIZE}(\Phi) = |\Phi|$, i.e., the number of functions in the collection.
- $\mathtt{DEPEND}(\Phi) =$ the maximal number of variables a function φ of Φ depends on.
- $\mathtt{VAR\text{-}SIZE}(\Phi) = \log_2 |\Sigma|$, i.e., the number of bits required in order to represent the value of each variable z_i.
- $\mathtt{COMP}(\Phi) =$ the maximum, over $\varphi \in \Phi$, of the (circuit) complexity of φ.[2]

To try clear the fog, let us present here a vague picture as to the values each of the above measures may take in function-systems constructed below. Assume a function-system Φ, intended to verify membership for input of size n (a notion yet to be formally defined); systems defined below would comply with the following complexity bounds:

[2]Any reasonable notion of deterministic (and in fact even nondeterministic) complexity may replace "circuit" leaving the analysis intact.

- $\mathtt{SIZE}(\Phi)$ would be eventually polynomial, nevertheless, for some intermediate stages it might be larger, though never exceeding size polynomial in the *original* input-size.

- $\mathtt{DEPEND}(\Phi)$ is almost always constant and at most poly-logarithmic in n.

- $\mathtt{COMP}(\Phi)$ is at most polynomial in $\mathtt{DEPEND}(\Phi) \cdot \mathtt{VAR\text{-}SIZE}(\Phi)$, that is, each function φ of Φ is computable in time polynomial in its input size.

Table Manners

Let us now introduce a few manners by which some objects are referred to. Though formally making only a little of a difference, or none at all, those manners should hopefully simplify presentation of some of the technicalities involved.

Many of the objects discussed below are formally defined as functions $f\colon S \to R$ (where S and R are finite sets). Some times, however, it may be simpler to think of them as vectors. Each entry then corresponds to one element in S, the domain of f, and contains an element of its range R. One can thus consider the Hamming distance between two vectors v_1 and v_2, being the fraction of indexes the two vectors differ on:

$$\Delta(v_1, v_2) = \Pr_{i \in_R S}[v_1(i) \neq v_2(i)]$$

where $\in_R$ denotes the uniform choice of an element.

Let us hence define *the γ-ball of a vector v*, $B_\gamma[v]$, to be the set of all vectors v' whose Hamming distance from v is $\leq \gamma$; that is,

$$B_\gamma[v] = \{v' | \Delta(v', v) \leq \gamma\}\,.$$

One specific object, to which this manner of discussion is extensively applied, is an assignment to a FS's variables, which is simply a function from the set of variables to the variables' range (Σ in the above). This function might as well be thought of as a vector, or according to our particular choice of terms, a *table* T. Each variable ($z_1, \ldots, z_m$ in the above) has its own entry in T, and an assignment is simply assigning each entry with a value from the variables' range Σ. The value given by T to z_i is denoted $T[i]$.

Referring to variables by a table, seems to simplify the presentation. It enables considering a table constructed of several smaller tables, each referred to by a distinct index, hence $T = (T_1, \ldots, T_k)$. The number of entries in T_j is denoted $|T_j|$.

A FS Φ can thus be applied to a given table T; the value of each function $\varphi \in \Phi$ *over* T is simply φ's value when its variables are assigned the value given to them by T. Φ is said to be *satisfied over* T accordingly (i.e., if all functions φ in Φ are satisfied by the value given to their variables by T). The variables $z_1, \ldots, z_m$ may therefore be sometimes altogether omitted from our analysis, and instead be referred to as entries in T.

Representation Schemes

Let us now attend to the manner in which the input (whose membership in a language is in question) is expected to be represented.

A function $\rho\colon \{0,1\}^n \to \Sigma^l$, for a finite alphabet Σ, is said to be an *input-representation* (or simply representation) if for every $I \neq I' \in \{0,1\}^n$,

$$\rho(I) \neq \rho(I').$$

In other words, ρ maps each input to a distinct element of Σ^l. Σ^l in turn, may be thought of as a function from $[1, ..., l]$ to Σ, or alternatively as a vector of l entries.

Hence, a representation ρ induces a set of vectors in Σ^l

$$\{\rho(I)\}_{I \in \{0,1\}^n}$$

being the images of all inputs $I \in \{0,1\}^n$. For technical reasons, however, as part of the definition of ρ we would fix a set $\mathbf{e}_\rho \subseteq \Sigma^l$, which is a superset of the images of inputs according to ρ, that is

$$\{\rho(I)\}_{I \in \{0,1\}^n} \subseteq \mathbf{e}_\rho \subseteq \Sigma^l.$$

It is therefore the case that, a vector in Σ^l may be illegitimate, if it is outside $\mathbf{e}_\rho$. Otherwise, it is considered legitimate, however may or may not represent an input $I \in \{0,1\}^n$. Legitimate vectors that do not represent any input, for all practical purposes would be considered as "bad" inputs, i.e., outside the language L.

The representations used herein, in most cases, are in fact codes, i.e., the elements in $\mathbf{e}_\rho$, as vectors, differ on a large fraction of the indexes. Formally, a representation ρ is said to be an *α-code* if for every $v_1 \neq v_2 \in \mathbf{e}_\rho$, v_1 and v_2 agree on at most an α fraction of the coordinates $[1, ..., l]$.

An input-representation ρ, as defined above, is applied on a single input-size n, while our interest is with languages of various input-sizes. Accordingly, ρ is defined to be a sequence of input-representations, one for each n.

It now makes sense to consider the complexity of computing ρ on a given input; we say that ρ is *efficiently-computable* if it is computable in time polynomial in its output-size (which is some intermediate stages may be larger than the input-size).

Verifying–Function-Systems

Consider a language $L \subseteq \{0,1\}^n$,[3] and let $\rho\colon \{0,1\}^n \to \Sigma^l$ be an input-representation according to which the input (possibly belonging to L) is to be represented.

In order to distinguish between the input-representing variables and the auxiliary variables, assume the function-system Φ's table, T, to be constructed of two parts $T = (T_1, T_2)$. T_1 consists of the set of input-representing variables, while T_2 includes the auxiliary variables. It is of course the case then that $|T_1| = l$.

A-priori, the input-representation and auxiliary variables may have different ranges, in which case let Σ be those ranges' union.

Definition 1 *A FS Φ over a table T is said to be an* (ϵ, δ)-Verifying–Function-System for L according to representation ρ (or simply (ϵ, δ)-VFS_ρ) *if:*

[3]This should be interpreted as "consider the restriction of a language $L \subseteq \{0,1\}^*$ to input-size n"

1. *For every $I \in L$ there exists T_2 such that all $\varphi \in \Phi$ are satisfied over $T = (\rho(I), T_2)$.*

   That is, if the values for the input-representing variables are a proper representation of a "good" input, there exists a witness T_2 satisfying all φ in Φ. On the other hand,

2. *For any T_1, if there exists T_2 such that*

$$\Pr_{\varphi \in_R \Phi}[\varphi \text{ is satisfied over } (T_1, T_2)] \geq \epsilon$$

   then there exists $I \in L$ such that T_1 agrees with $\rho(I)$ on at least a δ fraction of the indexes $[1, ..., l]$, that is, $T_1 \in B_{1-\delta}[\rho(I)]$.

   Conversely, if for any $I \in L$, T_1 is outside $B_{1-\delta}[\rho(I)]$ (i.e., any representation of a "good" input agrees with T_1 on less than δ fraction of the entries) then, for any T_2

$$\Pr_{\varphi \in_R \Phi}[\varphi \text{ is satisfied over } (T_1, T_2)] < \epsilon.$$

Condition 1 of the above definition requires Φ to have an assignment to the auxiliary variables, satisfying all its Boolean-functions, in case a good input is properly represented by the values assigned to the input-representing variables. Condition 2 is the subtle one. The parameter ϵ relates to the probability of error — smaller ϵ corresponds to a stronger notion of verification — hence Φ is allowed to have that fraction of its functions satisfied even for the wildest values for the input-representing variables. It is required however, that no larger fraction is satisfied unless a sizable (δ) fraction of the entries agree with the representation of some good input. This requirement is rather weak, however turns out to be sufficient for our needs.

The parameter δ, in VFSs constructed below, is usually fixed to be smaller than ϵ but larger than ϵ^c for some constant c. Note that condition 2 allows T_1's entries to correspond to representations of several "good" inputs. Nonetheless, as long as ρ is an α-code for small enough α — hence representation of distinct inputs disagree almost everywhere — the number of such inputs cannot exceed δ^{-1}. All VFSs constructed below feature input-representations which are α-codes for δ large enough with respect to α, so as to allow only $O(\delta^{-1})$ *permissible* inputs, namely, inputs whose representation agrees with T_1 on a δ fraction of the entries.

Multi-Input VFSs

As mentioned above, for sake of the application of VFS within the proof itself, one must also consider the case where the input I, whose membership is in question, is broken into several segments.

Consider a language L whose input consists of k strings $I_1, ..., I_k \in \{0,1\}^n$, that is, $L \subseteq \{0,1\}^{kn}$. Let $\rho: \{0,1\}^n \to \Sigma^l$ be a representation, according to which all segments are to be represented. Let us now define a VFS that verifies membership, of the combined data of k independent segments, in L.

Consider a FS Φ over a table T which is constructed of $k+1$ parts: $T = (T_1, T_2, \ldots, T_{k+1})$, each entry, in each part, taking value from Σ.

The first k parts of T are supposedly representing k segments of input, hence $|T_1| = |T_2| = \cdots = |T_k| = l$. The $(k+1)^{\underline{\text{th}}}$ part supposedly contains the witness for membership of $(I_1, \ldots, I_k)$ in L.

Definition 2 *Φ is said to be a k-*inputs (ϵ, δ)-VFS for L according to representation ρ (or simply (ϵ, δ)-$\mathsf{VFS}_{k,\rho}$) *if:*

1. *For every $I = (I_1, \ldots, I_k) \in L$, there exists T_{k+1} such that all $\varphi \in \Phi$ are satisfied over the table $T = (\rho(I_1), ..., \rho(I_k), T_{k+1})$.*

   That is, as in the above, a "good" input has a witness that causes universal satisfaction. On the other hand,

2. *For any T_1, ..., T_k, if there exists T_{k+1} such that*

   $$\Pr_{\varphi \in_R \Phi}[\varphi \textit{ is satisfied over } (T_1, ..., T_{k+1})] \geq \epsilon$$

   then there exists $I = (I_1, ..., I_k) \in L$ such that for every $j \in [1, ..., k]$, $T_j \in B_{1-\delta}[\rho(I_j)]$.

   Conversely, if for any $I = (I_1, \ldots, I_k) \in L$, there exists an index $j \in [1, ..., k]$ such that T_j agrees with $\rho(I_j)$ on less than δ fraction of the entries, then, for any T_{k+1}

   $$\Pr_{\varphi \in_R \Phi}[\varphi \textit{ is satisfied over } (T_1, ..., T_{k+1})] < \epsilon.$$

   That is, Φ cannot even have an ϵ fraction of its functions satisfied over T_1, ..., T_k, no matter what T_{k+1} is, unless there exist "good" k segments such that each T_j agrees with the representation of the $j^{\underline{\text{th}}}$ segment on a sizable (δ) fraction.

This definition assumes all k input segments $I_1, ..., I_k$ to be of same length and represented according to the same representation ρ. These definitions can be generalized naturally to consider input segments of different length, or where different representations are used for different segments.

The Game

Let us return, for a moment, to the setting mentioned above, regarding the two-players game induced by a functions-system. For a VFS Φ the game induced is as follows: The first player — the Prover — commits to values for the input-representing variables and the auxiliary variables, claiming that a certain good input $I \in L$ is properly represented in the first set, and that the values for the auxiliary variables are a witness to the membership of I in L. The second player — the Verifier — picks a single random Boolean-function φ of Φ and checks whether it is satisfied by the values committed to by the Prover.

Let us consider the complexity measures defined above for VFS and figure out their implications to the induced game. First, the restrictions implied are all concerned with the Verifier, as the Prover remains the all-powerful nondeterministic guesser. The parameter $\mathtt{SIZE}(\Phi)$ is the number of options the Verifier may choose from, hence $\log(\mathtt{SIZE}(\Phi))$ is the number of random bits consumed by the Verifier. Assume the Prover writes down the values it commits to, and that they are arranged in blocks; VAR-SIZE(Φ) determines the size of each block, while DEPEND(Φ) fixes the maximum number of blocks the Verifier reads for any specific random choice. COMP(Φ) is an upper bound on the complexity of verifying the values read to indeed satisfy the chosen Boolean-function.

ϵ, as mentioned above, is the error-probability of the induced game. If less than δ fraction of the input-representing variables agree with any "good" input, the probability of the Verifier to accept, must be smaller than ϵ.

Efficient Constructibility

The proof below involves constructing a VFS (of some significantly restricted complexity) for every NP language L and input-size n. One critical complexity aspect, neglected so far in our discussion, has to do with how hard it is to construct a VFS. A VFS Φ, complying with whichever complexity bounds, however such that to construct Φ takes time exponential in n, might not be very useful. E.g., an efficient procedure that decides, given Φ and an input I, whether I is in or outside L, would not at all imply an efficient decision-procedure for L, in that case.

Hence, in order for a characterization of NP in terms of PCP to be of use, e.g. in order to facilitate NP-hardness results, one has to insist on the construction of the VFS Φ to be in P, i.e., that there is an algorithm which, given a Turing-machine for L and input size n, constructs Φ in time polynomial in n.

Accordingly, all constructions of function-systems the proof involves with are efficient, in the sense that they run in time polynomial in the size of the outputted system[4]. We would write "there exists (constructible) X" to mean not only that X exists but that it can actually be computed efficiently (i.e., in time polynomial in its size).

The PCP Hierarchy

The PCP hierarchy [AS92, BGLR93] classifies languages L according to complexity of a VFS that exists (constructible) for L. Each of the parameters and complexity measures of the constructed VFS plays some role — either crucial or only of little importance — in that classification.

In order to construct a VFS for L, one first needs to fix the number of independent segments that constitute its input, and the input-representation according to which the input is expected. Note however, that those parameters are considered only for the sake of intermediate stages within the proof. For the purpose of classification of computational-complexity classes in terms of the PCP hierarchy, one should be contented with considering VFSs in which those parameters become trivial. Namely, VFSs whose input comes in one segment, and where the input-representation is any efficiently-computable α-code, for $\alpha < \delta$. (If α were larger, the VFS may not be useful, as an assignment to the input-representing variables may agree with the representation of many inputs, on a δ fraction).

[4]which may be quite larger than the input-size in some intermediate stages, however never exceeding size polynomial in the original input-size.

As it turns out, the functions of VFSs constructed below, usually all depend on only a constant number of variables, and are computable in time polynomial in the total size of their input (which is `DEPEND` times `VAR-SIZE`), hence the `DEPEND` and `COMP` parameters play a little role in what is to follow. The fraction of input-representing variables required for an input to be considered permissible (i.e., δ) would always be smaller than ϵ and larger than a constant power of ϵ.

The major parameters, therefore, whose interplay is crucial to the proof, are the error-probability (ϵ), the number of bits required to represent each variable (`VAR-SIZE`), and the number of functions of a system (`SIZE`).

Definition 3 (PCP-Hierarchy) *Let $k, \epsilon, \delta, D, S, V$ be functions of n,*[5] *and ρ be an input-representation (being itself a function of n)*[6]*.*

The class

$$\mathsf{PCP}_{k,\rho}[\, D \ , \ S \ , \ V \ , \ \epsilon \ , \ \delta \,]$$

is defined to consist of all languages L for which there exists (constructible) an (ϵ, δ)-$\mathsf{VFS}_{k,\rho}$ Φ for L, which complies with the following complexity bounds:

- $\mathtt{DEPEND}(\Phi) \leq D(n)$
- $\mathtt{SIZE}(\Phi) \leq S(n)$
- $\mathtt{VAR\text{-}SIZE}(\Phi) \leq V(n)$
- $\mathtt{COMP}(\Phi) \leq (D(n) \cdot V(n))^C$*, for some constant C.*

It now becomes natural to consider relationships between various classes of languages, either within the PCP hierarchy, or with respect to other computational-complexity classes.

Our interest herein is, in particular, with NP and how low can it be placed within the PCP hierarchy. Note however that it is unlikely for NP to be characterized in terms of the PCP hierarchy by a class whose `SIZE` is bounded from above by n^{const}, for a fixed, global constant *const*. If NP were thus characterized, it would imply that all NP languages have witnesses of size limited to a fixed polynomial, which in turn has revolutionary consequences.

Accordingly, when defining a PCP class, some of the parameters may include the $O(x)$ notation (e.g., as in $n^{O(1)}$ below). Such a class should be interpreted as the union over all constants c of the PCP class induced by replacing $O(x)$ by $c \cdot x$. Consequently, for any language L there may be a different constant for which the containment holds.

The Target

It is thus possible now to phrase our results in terms of the PCP hierarchy. To start with, we present the following characterization of NP — one which already features sub-constant error-probability, moreover, comes close to the optimal relation between `VAR-SIZE` and error-probability:

[5] some of which may possibly be constant functions

[6] that is, ρ is in fact a sequence of input-representations, one for each input-size n

Theorem 6 *There exists a constant D and an efficiently-computable α-code ρ such that*

$$\mathsf{NP} \subseteq \mathsf{PCP}_{1,\rho}[\, D \,,\, S \,,\, V \,,\, \epsilon \,,\, \delta \,]$$

where

- $\epsilon(n) = 2^{-\sqrt{\log n}}$
- $S(n) = n^{O(1)}$
- $V(n) = O(\log n)$
- $\delta(n) > \alpha(n)$

In other words, theorem 6 asserts that, any NP language has (constructible) polynomial-size VFS that exhibits error-probability as small as $2^{-\sqrt{\log n}}$ and where each function depends on a constant number of variables. Each of those variables requires at most a logarithmic number of bits for its representation.

The characterization of NP of theorem 6 is quite close to optimal, as the error-probability cannot be smaller than exponentially small in $V(n)$. Furthermore, the error-probability, for a polynomial-`SIZE` VFS, unless being 0, is, at best, polynomially small. Our results in fact achieve the optimal relation between `VAR-SIZE` and the error-probability (ϵ) (up to a constant factor in the exponent) for a sizeable fraction of the sensible range:

Theorem 7 *Let $M(n)$ be any function not exceeding $(\log n)^\beta$ for some $\beta < 1/4$, and which is at least $\log^{(i)} n$ for a constant i (where $\log^{(i)}$ denotes i applications of the $\log$ function). Then, there exists a constant D (which varies according to the specific M considered), and an efficiently-computable α-code ρ, such that:*

$$\mathsf{NP} \subseteq \mathsf{PCP}_{1,\rho}[\, D \,,\, S \,,\, V \,,\, \epsilon \,,\, \delta \,]$$

for

- $\epsilon(n) = 2^{-M(n)}$
- $S(n) = n^{O(1)}$
- $V(n) = O\left(M(n)\right)$
- $\delta(n) > \alpha(n)$

If one is interested in the relation between the error-probability and `VAR-SIZE` for $M(n)$ beyond $\log^{1/4} n$, the following holds as well:

Theorem 8 *Let $M(n)$ be a function such that $\log^{(i)} n \leq M(n) \leq (\log n)^{1-\beta}$ for some constant i and $\beta > 0$. Then, there exists a constant D (which varies according to the specific M considered), and an efficiently-computable α-code ρ, such that:*

$$\mathsf{NP} \subseteq \mathsf{PCP}_{1,\rho}[\, D \,,\, S \,,\, V \,,\, \epsilon \,,\, \delta \,]$$

for

- $\epsilon(n) = 2^{-M(n)}$
- $S(n) = n^{O(1)}$
- $V(n) = O\left(M(n) \cdot \log^c M(n)\right)$, *for some fixed constant* c
- $\delta(n) > \alpha(n)$

The next three sections contain an exposition of the principal elements of the proof for theorem 6. The intention is to present the framework of the proof, and defer as many as details for as long as possible. It should hopefully be the case that, by the end of those sections, the reader may trust the proof, assuming a few gaps — taking the form of several lemmas whose proof is left for later sections — are filled.

The structure of the proof is still not immediately unveiled. At first, the initial construction of a VFS for every L in NP, one which achieves reasonable complexity, is assumed. It is shown then how the `VAR-SIZE` parameter of that construction can be improved so as to obtain a VFS of desired complexity. Our attention then shifts back to the initial construction, whose basic ingredients are presented in a following section. Finally, some implementation issues are discussed, setting the scene for the proofs of the missing lemmas, allowing them to be completely independent of each other.

Theorem 8 follows by similar techniques. Theorem 7 needs some additional ideas. Both are covered in section 7.

2 Proof of Theorem 6

The proof for theorem 6 breaks into two lemmas: The first — the "Initial Construction" — places NP low in the PCP hierarchy, however much higher than the claim. The second lemma — the "Composition" lemma — shows how, in some cases, the complexity of a PCP class can be altered so as to endure a slight increase in `SIZE`, in return for a reduction in `VAR-SIZE`, still maintaining a similar error-probability.

The Initial-Construction

The first lemma proceeds as follows:

Lemma 1 (Initial-Construction) *For any* $\epsilon(n)$, $s(n) \geq n$, *and a constant* $c_1 \geq 1$, *satisfying* $s(n) \cdot \epsilon(n)^{c_1} \geq 1$, *there exists an efficiently-computable* $(\epsilon(n)^{c_1})$*-code* ρ,[7] *such that for any constant* k

$$\mathsf{NP} \subseteq \mathsf{PCP}_{k,\rho}\left[\, c_2 \cdot k \ , \ S \ , \ V \ , \ \epsilon \ , \ \delta \,\right]$$

for some fixed[8] *constant* c_2 *and where*

- $S(n) \leq s(n)^{O(1)}$

[7] being a function, not only of n, but of ϵ, s and c_1 as well

[8] i.e., one can replace c_2 with a specific number

- $\delta(n) = \epsilon(n)^2/10k$, *and*
- $V(n) = 2^{O\left(\log\log n + \log\frac{1}{\epsilon(n)}\right)\cdot \log n/\log s(n)} \cdot (\log s(n) \cdot \log n)^{c_3}$, *for some fixed constant* c_3.

In other words, lemma 1 asserts the following: Assume an NP language L and an input-size and number of segments. Fix now a desired error-probability. For any size of VFS $S(n)$ (polynomial in input-size or larger), there exists (constructible) a VFS complying with the specified parameters, and whose VAR-SIZE is the monstrous function described by $V(n)$.

It is worthwhile noting that the constant factors in the expressions for S and V (to be plugged in for the O notation) are not so arbitrary. It is in fact the case that for any NP language L, the constructed VFS's SIZE is a fixed power of the nondeterministic time-complexity of L, while V is some fixed function of it.

The VAR-SIZE parameter $V(n)$ of the VFS claimed in lemma 1 varies according to the desired error-probability, $\epsilon(n)$, and SIZE, $S(n)$. Let us consider a few concrete instances of that general tradeoff. For simplicity, let us fix $\delta(n)$ to be $\epsilon(n)^2/10k$, therefore it is omitted. Assume, in addition, c_1 to be a large enough constant, and ρ to be the $(\epsilon(n)^{c_1})$-code assumed to exist in lemma 1. Note that ρ is a function of the other parameters in the lemma.

First, say one is interested in maintaining polynomial SIZE while obtaining good ratio between VAR-SIZE and ϵ:

Corollary 1 *For any constant* c_ϵ

$$\mathsf{NP} \subseteq \mathsf{PCP}_{k,\rho}[\, c_D \cdot k \,,\ S \,,\ V \,,\ \epsilon \,, \delta]$$

for some fixed constant c_D*, and where* $S(n) = n^{O(1)}$, $\epsilon(n) = 2^{-c_\epsilon\sqrt{\log n}}$*, and* $V(n) = 2^{O(\sqrt{\log n})}$.

Lemma 1 can also be applied so as to obtain polynomially small error-probability; in which case, however, the SIZE parameter becomes super-polynomial:

Corollary 2 *For any constant* c_ϵ

$$\mathsf{NP} \subseteq \mathsf{PCP}_{k,\rho}[\, c_D \cdot k \,,\ S \,,\ V \,,\ \epsilon \,, \delta]$$

for some fixed constant c_D*, and where* $S(n) = n^{O(\log n)}$, $\epsilon(n) = 1/n^{c_\epsilon}$*, and* $V(n) = O((\log n)^{c_4})$*, for some fixed constant* c_4.

Lemma 1 can also be applied to obtain VFSs of SIZE much larger than the input-size, and in return have the error probability much smaller:

Corollary 3 *For any constant* $c_\epsilon > 0$

$$\mathsf{NP} \subseteq \mathsf{PCP}_{k,\rho}[\, c_D \cdot k \,,\ S \,,\ V \,,\ \epsilon \,, \delta]$$

for some fixed constant c_D*, and where* $S(n) = n^{O(n^{1/c_\epsilon})}$, $\epsilon(n) = 2^{-n^{1/c_\epsilon}}$*, and* $V(n) = O(n^{1/c_\epsilon} \cdot (\log n)^{c_4})$*, for some fixed constant* c_4.

The proof for lemma 1 is to be found in section 4. Let us now attend to the Composition lemma.

Composition Lemma

The Composition lemma, as mentioned above, aims at showing containment of one PCP class (e.g., the one of corollary 1) in another, so that the parameter `SIZE` increases polynomially, while `VAR-SIZE` reduces drastically (leaving the error-probability pretty much the same).

Lemma 2 (Composition) *Assume a* PCP *class*

$$\mathcal{C} = \mathsf{PCP}_{k,\rho_1}[\, D_1 \,,\, S_1 \,,\, V_1 \,,\, \epsilon_1 \,,\, \delta_1 \,]$$

where ρ_1 is an efficiently-computable α_1-code, and assume (for simplicity) D_1 is constant; hence let us write D_1 instead of $D_1(n)$.

Assume, in addition, a characterization of P *in terms of* PCP*, for input of D_1 segments:*

$$\mathsf{P} \subseteq \mathsf{PCP}_{D_1,\rho_2}[\, D_2 \,,\, S_2 \,,\, V_2 \,,\, \epsilon_2 \,,\, \delta_2 \,]$$

where ρ_2 is an efficiently-computable α_2-code for some $\alpha_2(n) < \delta_2(n)/2$. [9]

The class $\mathcal{C}$ is then contained in another PCP *class; namely, there exists an efficiently-computable $(\alpha_1 + \alpha_2)$-code ρ such that*

$$\mathcal{C} \subseteq \mathsf{PCP}_{k,\rho}[\, D \,,\, S \,,\, V \,,\, \epsilon \,,\, \delta]$$

for

- *$D(n) = D_2(V_1(n))$; (Hence, if D_2 is a constant function then $D = D_2$)*
- *$S(n) = S_1(n) \cdot S_2(V_1(n))$; (Hence, as long as $S_1(n)$ and $S_2(V_1(n))$ are polynomial in n, so is $S(n)$)*
- *$V(n) = V_2(V_1(n))$; (Hence, if $V_2(x)$ is significantly smaller than x, $V(n)$ is much smaller than $V_1(n)$)*
- $\epsilon(n) = \epsilon_2(V_1(n)) + \epsilon_1(n) \cdot \left(\frac{\delta_2(V_1(n))}{2}\right)^{-D_1}$
- $\delta(n) = \delta_1(n) \cdot \delta_2(V_1(n))$

Informally put, the Composition lemma takes advantage of the containment of P low in the PCP hierarchy as a catalyst for the transformation of a given VFS Φ into one which is slightly larger, however much more restricted in other parameters. The construction that implements this transformation involves replacing each function φ of Φ with an entire VFS Ψ_φ that verifies the assignment to the variables which φ depends on, to indeed satisfy φ. (For that purpose, however, those assignments must be encoded according to what is expected by Ψ_φ).

The proof of lemma 2 appears shortly below; let us first show how it implies theorem 6.

[9] Note that, as mentioned above, a characterization of P for fixed S_2 and V_2 (without the O notation) is not likely to be established. It is only necessary here, for the proof to follow through, that all Boolean-functions of the VFS assumed to exist for any language in $\mathcal{C}$, are contained in the PCP class of the right-hand term (adjusted to their input-size which is $D_1 \cdot V_1(n)$). Alternatively, one may assume that S_2 and V_2 are expressed using the O notation, however, it is required then that, for any language L, S_2 and V_2 are fixed functions of the nondeterministic time-complexity of L. Another alternative is...

Combining the Two Lemmas

Since P is clearly contained in NP, the initial construction (of lemma 1) may be applied both as the PCP class to transform, as well as the catalyst for that transformation.

Let the class $\mathcal{C}$ be the one known to contain NP by corollary 1 (of polynomial SIZE and where the error-probability is $2^{-c_1\sqrt{\log n}}$ and VAR-SIZE of $2^{O(\sqrt{\log n})}$).

Let the characterization of P — the catalyst — be that of corollary 2 (of $n^{O(\log n)}$ SIZE, error-probability $1/n^{c_2}$, and poly-logarithmic VAR-SIZE) set so that k is the DEPEND parameter of $\mathcal{C}$.

The Composition lemma then implies that:

$$\mathsf{NP} \subseteq \mathsf{PCP}_{1,\rho}[\, D\ ,\ S\ ,\ V\ ,\ \epsilon\ ,\ \delta\,]$$

for

- constant D
- $S(n) = n^{O(1)}$
- $V(n) = O(\log^{c'} n)$, for some fixed constant c'
- $\epsilon(n) = 2^{-c\sqrt{\log n}}$ (for any desired constant c)

In order to further reduce VAR-SIZE, this characterization is plugged-in again as the class $\mathcal{C}$ in the Composition lemma. Let the characterization of P, in this case, be that of corollary 3, set so that k is the DEPEND parameter of $\mathcal{C}$, and $c_\epsilon = 2c'$. The Composition lemma then implies that:

$$\mathsf{NP} \subseteq \mathsf{PCP}_{1,\rho}[\, D\ ,\ S\ ,\ V\ ,\ \epsilon\ ,\ \delta\,]$$

for some fixed constant D, $S(n) = n^{O(1)}$, $\epsilon(n) = 2^{-c\sqrt{\log n}}$ (for some constant c), and

$$V(n) = O\left(\sqrt{\log n} \cdot (\log\log n)^{c'}\right)$$

for some fixed constant c'.

For simplicity, the parameters δ, c_1, and α were ignored throughout this analysis; the exact choice of the constants mentioned can ensure their values to be appropriate.

Assuming lemma 1 and lemma 2, the proof for theorem 6 is hereby complete. $\square$

3 Proof of the Composition Lemma

Composition of Representations

Consider an input-representation

$$\rho_1 \colon \{0,1\}^n \to \Sigma^{l_1}$$

and another representation whose domain is Σ:

$$\rho_2 \colon \{0,1\}^{\log_2|\Sigma|} \to \Pi^{l_2}$$

where ρ_2's input is described as $\{0,1\}^{\log_2|\Sigma|}$ in order to emphasize it is treated as a string of bits[10]. Let us define the composition of ρ_2 onto ρ_1

$$\rho_2 \circ \rho_1 \colon \{0,1\}^n \to \Pi^{l_1 \cdot l_2}$$

to be the application of ρ_2 independently to each entry of ρ_1, i.e., given any input $I \in \{0,1\}^n$, for any $i_1 \in [1,...,l_1]$, and $i_2 \in [1,...,l_2]$,

$$\rho_2 \circ \rho_1(I)[i_1,i_2] = \rho_2\left(\rho_1(I)[i_1]\right)[i_2]$$

where $v[i]$ denotes the $i^{\underline{\text{th}}}$ entry of v, and $\rho_2 \circ \rho_1(I)$ is treated as an l_1 by l_2 matrix (instead of an $(l_1 \cdot l_2)$-size vector).

Note that if ρ_1 is an α_1-code and ρ_2 is an α_2-code, then $\rho_2 \circ \rho_1$ is an $(\alpha_1 + \alpha_2)$-code.

Composition of VFSs

Let $L \subseteq \{0,1\}^{k \cdot n}$ denote a language of the class $\mathcal{C}$ assumed in the lemma. Let Φ be an (ϵ_1, δ_1)-VFS_{k,ρ_1} which is implied by the assumption of the lemma regarding $\mathcal{C}$ (in particular such that $\mathtt{DEPEND}(\Phi) \le D_1$, $\mathtt{SIZE}(\Phi) \le S_1(n)$ and $\mathtt{VAR\text{-}SIZE}(\Phi) \le V_1(n)$).

Every φ of Φ, hence, is a Boolean-function depending on at most D_1 variables, each of range Σ; let us denote by

$$[[\varphi]] \subseteq \{0,1\}^{D_1 \cdot \log_2|\Sigma|}$$

the set of assignments, to φ's variables (ordered arbitrarily), that satisfy φ.

Let $m = V_1(n)$; it follows, by the containment of P within the PCP-hierarchy assumed by the lemma (the catalyst), that there exists, for every φ, Ψ_φ such that

$$\Psi_\varphi \text{ is an } \quad (\epsilon_2, \delta_2)\text{-}\mathsf{VFS}_{D_1,\rho_2} \quad \text{for } [[\varphi]]$$

that complies with the following bounds:

- $\mathtt{SIZE}(\Psi_\varphi) \le S_2(m)$.
- $\mathtt{DEPEND}(\Psi_\varphi) \le D_2(m)$.
- $\mathtt{VAR\text{-}SIZE}(\Psi_\varphi) \le V_2(m)$.
- $\mathtt{COMP}(\Psi_\varphi) \le (D_2(m) \cdot V_2(m))^{C_1}$, for some constant C_1.

Ψ_φ is applied over a table consisting of $D_1 + 1$ parts (one part for each of the variables φ depends on, and an additional part for the witness that the assignment to those variables satisfies φ).

[10] Σ's elements are to be 1-to-1 mapped to all strings of bits of that length

Consider now Φ's table, $T = (T_1, ..., T_{k+1})$, and let $\hat{T}_j$, for every $j \in [1, .., k+1]$, be the table T_j except a each variable is replaced by an entire (sub-)table. Each smaller table supposedly represents the assignment to its respective variable according to ρ_2 (hence having l_2 entries of range Π). Let, in addition, $\hat{T}_w$ be a table consisting of $\texttt{SIZE}(\Phi)$ smaller tables, one part T_φ for each φ of Φ, which contains the auxiliary variables for Ψ_φ.

Each Ψ_φ may now be considered over the following, combined table:

$$\hat{T} = (\hat{T}_1, ..., \hat{T}_k, (\hat{T}_{k+1}, \hat{T}_w))$$

which is a $(k+1)$-parts table, whose first k parts correspond to Φ's input, which however is expected according to $\rho_2 \circ \rho_1$. The last part of $\hat{T}$, the witness part, is divided into two tables; the first, $\hat{T}_{k+1}$, contains one small table for each auxiliary variable of Φ (i.e., of T_{k+1}), which supposedly represents, according to ρ_2, the assignment to the respective variable. The second part consists of one table for each φ of Φ, containing the auxiliary data for its respective Ψ_φ.

Note that each Boolean-function of any Ψ_φ may depend on variables of at most $D_1 + 1$ of the smaller tables.

Assume w.l.o.g. that all Ψ_φ are of same $\texttt{SIZE}$, and let

$$\hat{\Phi} = \bigcup_{\varphi \in \Phi} \Psi_\varphi$$

be a FS over $\hat{T}$. It is clearly the case that $\hat{\Phi}$ complies with the following complexity bounds:

- $\texttt{SIZE}(\hat{\Phi}) \leq \texttt{SIZE}(\Phi) \cdot S_2(m)$.
- $\texttt{DEPEND}(\hat{\Phi}) = \max_{\varphi \in \Phi} \texttt{DEPEND}(\Psi_\varphi)$.
- $\texttt{VAR-SIZE}(\hat{\Phi}) = \max_{\varphi \in \Phi} \texttt{VAR-SIZE}(\Psi_\varphi)$.
- $\texttt{COMP}(\hat{\Phi}) = \max_{\varphi \in \Phi} \texttt{COMP}(\Psi_\varphi)$.

Furthermore,

Claim 3 *As long as ρ_2 is a $\frac{\delta_2}{2}$-code,*

$$\hat{\Phi} \textit{ is an } \quad (\epsilon, \delta)\text{-}\mathsf{VFS}_{k, \rho_2 \circ \rho_1} \quad \textit{for } L$$

where $\delta = \delta_1 \cdot \delta_2$ and

$$\epsilon = \epsilon_2 + \epsilon_1 \cdot (2 \cdot {\delta_2}^{-1})^{D_1}.$$

Proof for Claim 3

One needs to prove that (1) any input in L (represented according to $\rho_2 \circ \rho_1$) has a satisfying witness for $\hat{\Phi}$; and (2) that an ϵ fraction of satisfaction in $\hat{\Phi}$ implies a permissible good input.

(1) Given $(I_1, ..., I_k) \in L$, let T_{k+1} be the witness of membership in L, i.e. such that Φ is satisfied over $(\rho_1(I_1), ..., \rho_1(I_k), T_{k+1})$. To obtain $\hat{T}_{k+1}$, apply ρ_2 to each variable of T_{k+1}. Since all φ of Φ are satisfied over the table $(\rho_1(I_1), ..., \rho_1(I_k), T_{k+1})$, it must be the case that there are satisfying assignments to the auxiliary variables of each Ψ_φ; let $\hat{T}_w$ be the collection of all those assignments. It must then be the case that every $\hat{\varphi}$ of $\hat{\Phi}$ is satisfied over the table

$$(\rho_2 \circ \rho_1(I_1), \quad \ldots, \quad \rho_2 \circ \rho_1(I_k), \quad (\hat{T}_{k+1}, \quad \hat{T}_w)).$$

(2) Given $\hat{T} = (\hat{T}_1, \ldots, \hat{T}_k, (\hat{T}_{k+1}, \hat{T}_w))$ such that

$$\Pr_{\hat{\varphi} \in_R \hat{\Phi}} \left[\hat{\varphi} \text{ is satisfied over } \hat{T}\right] \geq \epsilon$$

let us exhibit an $(I_1, \ldots, I_k) \in L$ such that, for every $j \in [1, \ldots, k]$, $\hat{T}_j \in B_{1-\delta}[\rho_2 \circ \rho_1(I_j)]$.

Let us rephrase that statement, regarding the error probability of $\hat{\Phi}$ over $\hat{T}$, as follows:

$$\Pr_{\varphi \in_R \Phi \ , \ \psi \in_R \Psi_\varphi} \left[\psi \text{ is satisfied over } \hat{T}\right] \quad \geq \quad \epsilon_2 + \epsilon_1 \cdot (2 \cdot {\delta_2}^{-1})^{D_1}$$

which implies that, let Ω denote the set of φ of Φ for which

$$\Pr_{\psi \in_R \Psi_\varphi} \left[\psi \text{ is satisfied over } \hat{T}\right] \geq \epsilon_2$$

then

$$\Pr_{\varphi \in_R \Phi} [\varphi \in \Omega] \quad \geq \quad \epsilon_1 \cdot (2 \cdot {\delta_2}^{-1})^{D_1}.$$

Consider now a labeling of the variables of Φ, each with the set of all the values that are permissible according to its respective table — each variable is hence labeled by at most $2{\delta_2}^{-1}$ values.

It then follows that for all $\varphi \in \Omega$, there exists a choice of one value for every variable φ depends on, out of the set that labels that variable, such that the chosen values satisfy φ.

It is now easy to show, applying the probabilistic method, that there exists an assignment, of a single value to each variable of Φ, that satisfies a $(\frac{\delta_2}{2})^{D_1}$ fraction of Boolean-functions in Ω. The proof simply considers a random choice of one element out of each variable's label, and concludes there is a positive probability that such an assignment satisfies that fraction of functions in Ω, which implies its existence.

This completes the proof of claim 3. □

The proof of the Composition lemma (lemma 2) is hereby complete as well. □

4 The Initial Construction

The proof of the initial construction (lemma 1) is carried out via a two-phase process, simplistically described as follows: In order to construct a VFS for a language L, one first constructs a VFS that verifies membership in L provided the input-representing variables can be trusted to all correspond to a single representation (i.e., their combined values belong to $\mathbf{e}_\rho$). Independently, a function system of some special characteristics is constructed, whose sole purpose is to filter out values that do not conform to some notion of global consistency (according to the input-representation ρ). Combined, the systems of those two phases form the desired VFS for L.

This method can be applied, independently, to all input tables, so that the combined systems form the desired VFS. Furthermore, such a filter can be applied also to the auxiliary table (which however is expected not encoded according to any representation). Such a transformation may possibly reduce the DEPEND parameter of the constructed VFS, if each Boolean-function of the filter depends on a smaller number of variables.

Presuming-VFSs

Let us first define a VFS that verifies membership in a language L, provided $T_1, \ldots, T_k$ each belongs to $\mathbf{e}_\rho$.

Definition 4 *A FS Φ over a table $(T_1, \ldots, T_{k+1})$ is said to be a k-*inputs ϵ–Presuming–VFS for L according to representation ρ (or simply ϵ-Presuming-$\mathsf{VFS}_{k,\rho}$) *if:*

1. *For every $(I_1, \ldots, I_k) \in L$ there exists T_{k+1} such that all $\varphi \in \Phi$ are satisfied over the table $(\rho(I_1), \ldots, \rho(I_k), T_{k+1})$.*

2. *For any other values $T_1, \ldots, T_k \in \mathbf{e}_\rho$ (i.e., such that $(T_1, \ldots, T_k)$ is not a representation of an input in L), there exists no T_{k+1} such that*

$$\Pr_{\varphi \in_R \Phi}[\varphi \textit{ is satisfied over } (T_1, \ldots, T_k, T_{k+1})] \geq \epsilon.$$

Consistent-Readers

Assume now an ϵ-presuming-$\mathsf{VFS}_{k,\rho}$ Φ for L. That is, Φ verifies, with error-probability ϵ, membership in L, as long as the input-representing variables can be trusted to be all consistent with some vector in $\mathbf{e}_\rho$. Let us now move on to the second phase in which Φ is transformed into a non-presuming VFS $\hat{\Phi}$ (however one that expects its input according to a different input-representation $\hat{\rho}$). Clearly, if there were some function-system that ensures the values of the input-representing variables to be all consistent with a single vector of $\mathbf{e}_\rho$, Φ could be combined with that function-system so as to form a non-presuming VFS. Such a function-system, however, can be constructed only with complexities far exceeding our requirements.

The definition of an (ϵ, δ)-VFS (definition 2), however, is carefully stated so that a non-negligible (ϵ) fraction of functions satisfied implies only that a non-negligible (δ) fraction of the input-representing variables agree with the representation of I for some good $I \in L$.

Weak Notion of Consistency

Given a table supposedly representing some input, let us thus distinguish between "permissible" vectors — namely those agreeing with that table on at least a δ fraction of the entries — and the other, ordinary ones:

Assume a table $\tilde{T}: [1, \ldots, \tilde{l}] \to \tilde{\Sigma}$ supposedly representing some input according to representation $\tilde{\rho}: \{0,1\}^{\tilde{n}} \to \tilde{\Sigma}^{\tilde{l}}$. A vector $v \in \tilde{\Sigma}^{\tilde{l}}$ is said to be *$\tilde{T}$-permissible (with respect to δ and $\tilde{\rho}$)* if $v \in \mathbf{e}_{\tilde{\rho}}$, and $\tilde{T} \in B_{1-\delta}[v]$ (i.e., v agrees with $\tilde{T}$ on at least a δ fraction of the entries).[11]

[11]Note that if $\tilde{\rho}$ is an α-code, for small enough α, there can be at most δ^{-1} distinct $\tilde{T}$-permissible vectors.

Returning now to the presuming-VFS Φ, it suffices if the values for the input-representing variables, each φ of Φ depends on, can be trusted (with very high probability) to be consistent with one of the permissible vectors. A FS that verifies global-consistency according to this weaker notion — i.e., one that with probability at most ϵ returns values that disagree with one of the permissible vectors — may still be composed with Φ to form a non-presuming VFS.

The Setting

Let us first consider the case where the input is constructed of a single segment.

The presuming-VFS Φ induces a collection of tuples of entries, one tuple τ_φ for each φ of Φ, each of which consisting of the set of indexes to the input-representing variables that φ depends on.

Let $\mathcal{T}$ denote, from now on, such a collection of tuples (of variables to be consistently-read), i.e., $\mathcal{T}$ is a set of tuples of indexes to entries of ρ.

Let $\hat{\rho}: \{0,1\}^n \rightarrow \hat{\Sigma}^{\hat{l}}$ be the input-representation according to which $\hat{\Phi}$'s input is expected. Assume a table T_1 supposedly representing some input I according to $\hat{\rho}$. Assume, in addition, a table T_2 that would serve as auxiliary data for the reader.

Reader for a Single Tuple

Consider one of the tuples $\tau \in \mathcal{T}$. A $[\rho, \tau, \hat{\rho}]$-*reader* is a non-Boolean function-system[12] Ψ_τ over a table (T_1, T_2), in which every function may evaluate in one of the following ways:

1. ψ of Ψ returns a vector assigning each element of τ with a value in ρ's range Σ; that is, ψ evaluates to an element in the set of functions $\{g \mid g: \tau \rightarrow \Sigma\}$.

   Informally put, ψ, with a few accesses to the input-representing variables of T_1 and the auxiliary variables of T_2, obtains some values that correspond to τ's entries in ρ (values which are hopefully consistent).

   Alternatively, ψ may announce an inconsistency found:

2. ψ of Ψ returns a special symbol `fault`.

T_1 induces a set of T_1-permissible vectors of $\mathbf{e}_{\hat{\rho}}$ (with respect to δ). A function ψ of Ψ_τ is said to be *erroneous* over (T_1, T_2) if it is non-faulty (i.e., does not evaluate to the special symbol `fault`) however the values assigned to τ's variables disagree with $v \in \mathbf{e}_\rho$ for any T_1-permissible vector v.[13]

[12] A non-Boolean function-system is just the same as a Boolean-function-system except the range of its functions may be non-Boolean.

[13] A transformation of vectors in $\mathbf{e}_{\hat{\rho}}$ to vectors in $\mathbf{e}_\rho$ — one which extends the natural $\hat{\rho}(I)$ to $\rho(I)$ — is implicitly assumed here. Hence the set of T_1-permissible vectors according to $\hat{\rho}$ induces a set of T_1-permissible vectors according to ρ. All pairs of $\hat{\rho}$ and ρ used herein, naturally induce such a transformation. Otherwise, any arbitrary extension of $\hat{\rho}(I)$ to $\rho(I)$ would do.

The Full Version

Assume now a set of τ-readers $\{\Psi_\tau\}$, one for each $\tau \in \mathcal{T}$ (all of which over the same two-parts table (T_1, T_2)), and so that the sizes of those readers, $|\Psi_\tau|$, are all the same. Their union

$$\Psi = \bigcup_{\tau \in \mathcal{T}} \Psi_\tau$$

is said to be a $[\rho, \mathcal{T}, \hat{\rho}]$-$(\epsilon, \delta)$-*consistent-reader* if the following holds:

1. For every $I \in \{0,1\}^n$, there exists T_2 such that the values of all ψ of Ψ, over $(\hat{\rho}(I), T_2)$, are (non-faulty and) consistent with $\rho(I)$.

   That is, for any proper representation of an input string I, each function $\psi \in \Psi_\tau$ returns the restriction of $\rho(I)$ to τ.

2. For any (T_1, T_2),
$$\Pr_{\psi \in_R \Psi}[\psi \text{ is erroneous over } (T_1, T_2)] \le \epsilon.$$

In other words, on one hand there is always a satisfactory witness in case T_1 is a proper $\hat{\rho}$-representation of some input. On the other hand, the total fraction of functions ψ which (are non-faulty however) return values that disagree with all T_1-permissible vectors of $\mathbf{e}_\rho$ is at most ϵ.

Combination

The non-presuming VFS $\hat{\Phi}$ is defined to be the outcome of combining the presuming-VFS Φ and its consistent-reader Ψ as follows: Let $\hat{\Phi} = \Phi_{[\Psi]}$ be a $\mathsf{VFS}_{\hat{\rho}}$ for L, consisting of one Boolean-function $\hat{\varphi}_{\varphi,\psi}$ for every φ of Φ and ψ of its reader Ψ_{τ_φ} (recall that τ_φ is the tuple of $\mathcal{T}$ induced by φ). $\hat{\varphi}_{\varphi,\psi}$ evaluates to `false` in case ψ evaluates to `fault`, and otherwise returns the value of φ, where the values for the input-representing variables are taken to be those returned by ψ.

Each φ of Φ is hence replaced (in order to obtain $\hat{\Phi}$) by a set of functions, one for each ψ of its reader Ψ_{τ_φ}, which serves as a filter for φ. Any rejection by ψ causes a rejection by $\hat{\varphi}_{\varphi,\psi}$, while proper values are forwarded to the appropriate φ.

Obviously, $\hat{\Phi} = \Phi_{[\Psi]}$ complies with the following complexity bounds:

- $\mathtt{SIZE}(\hat{\Phi}) \le \mathtt{SIZE}(\Phi) \cdot \mathtt{SIZE}(\Psi)$.
- $\mathtt{COMP}(\hat{\Phi}) \le \mathtt{COMP}(\Phi) + \mathtt{COMP}(\Psi)$.

Note that $\mathtt{DEPEND}(\hat{\Phi})$ may be smaller than $\mathtt{DEPEND}(\Phi)$. This transformation therefore, if applied to the auxiliary table as well, may be — and in fact is — utilized in order to significantly reduce the `DEPEND` parameter of a system.

Consistent-Readers for all Tables

Let us generalize the above definitions to the multi-input case, while, in addition, allow a consistent-reader not only for the input-representing tables but for the auxiliary table as well. The consistent-readers for the input-representing tables would be utilized in order to transform a presuming VFS into a non-presuming one. The reader for the auxiliary table may help reducing the DEPEND parameter.

Let Φ now be an ϵ-presuming-$\mathsf{VFS}_{k,\rho}$ for a language L. Φ's variables are hence arranged in $k+1$ tables $T_1, ..., T_k, T_{k+1}$. For each $j \in [1, ..., k+1]$, let

$$\mathcal{T}_j = \{\tau_{\varphi,j}\}_{\varphi \in \Phi}$$

where each $\tau_{\varphi,j}$ is the tuple of all indexes to variables in T_j that φ depends on.

Assume now, for every $j \in [1, ..., k]$, a $[\rho, \mathcal{T}_j, \hat{\rho}]$-$(\epsilon, \delta)$-consistent-reader Ψ_j over two-parts table $(T_{j,1}, T_{j,2})$; the consistent-reader for each table would serve as a filter for that table. As to the auxiliary table T_{k+1}, since it is not expected encoded according to any representation, let $\mathcal{I}$ denote the identity representation, and assume Ψ_{k+1} to be a $[\mathcal{I}, \mathcal{T}_{k+1}, \rho']$-$(\epsilon', \delta')$-consistent-reader over a two-parts table (T'_1, T'_2).

$\hat{\Phi} = \Phi_{[\Psi_1, ..., \Psi_k, \Psi_{k+1}]}$ is defined, in a manner similar to the single reader case, to have one function

$$\hat{\varphi}_{\varphi, \psi_1, ..., \psi_{k+1}}$$

for every φ of Φ and ψ_j of its $\tau_{\varphi,j}$-reader $\Psi_{j,\tau_{\varphi,j}}$, within Ψ_j. The Boolean-function $\hat{\varphi}_{\varphi,\psi_1,...,\psi_{k+1}}$ evaluates to `false` in case any of ψ_j returns `fault`. It otherwise evaluates to the result of applying φ on the values returned by ψ_js for φ's variables, each according to the table it belongs to.

Ψ_1, ..., Ψ_{k+1} hence serve as filters for φ of Φ in accessing values of each of the tables. The first k consistent-readers transform Φ into a non-presuming $\hat{\Phi}$, while the last one is only utilized in order to reduce the DEPEND parameter.

$\hat{\Phi} = \Phi_{[\Psi_1, ..., \Psi_k, \Psi_{k+1}]}$ complies with the following complexity bounds:

- $\mathtt{SIZE}(\hat{\Phi}) \leq \mathtt{SIZE}(\Phi) \cdot \prod_{j=1}^{k+1} \mathtt{SIZE}(\Psi_j)$.
- $\mathtt{DEPEND}(\hat{\Phi}) \leq \sum_{j=1}^{k+1} \mathtt{DEPEND}(\Psi_j)$.
- $\mathtt{VAR\text{-}SIZE}(\hat{\Phi}) = \max_{j=1}^{k+1} \mathtt{VAR\text{-}SIZE}(\Psi_j)$.
- $\mathtt{COMP}(\hat{\Phi}) \leq \mathtt{COMP}(\Phi) + \sum_{j=1}^{k+1} \mathtt{COMP}(\Psi_j)$.

The following lemma shows $\hat{\Phi}$ to be the desired VFS for L:

Lemma 4 *If ρ' is a $\frac{\delta'}{2}$-code, then $\hat{\Phi} = \Phi_{[\Psi_1, ..., \Psi_k, \Psi_{k+1}]}$ is a (non-presuming) $(\hat{\epsilon}, \hat{\delta})$-$\mathsf{VFS}_{k,\hat{\rho}}$ for L with*

$$\hat{\epsilon} = \frac{2}{\delta'} \cdot (k+1) \cdot \epsilon + \epsilon' \quad , \quad \hat{\delta} = \delta.$$

Proof: It is only left to prove that the error-probability of $\hat{\Phi}$ is indeed $\hat{\epsilon}$.

First consider the VFS $\tilde{\Phi} = \Phi_{[\Psi_1,\ldots,\Psi_k]}$, defined in a manner similar to $\hat{\Phi}$ using consistent-readers for the first k tables only. As explained above, the first k consistent-readers transform the VFS Φ into a non-presuming VFS ($\tilde{\Phi}$). Obviously, the error-probability of $\tilde{\Phi}$ is the sum of the error-probabilities of $\Phi, \Psi_1, \ldots, \Psi_k$. Therefore, $\tilde{\Phi} = \Phi_{[\Psi_1,\ldots,\Psi_k]}$ is a (non-presuming) $(\tilde{\epsilon}, \tilde{\delta})$-$\mathsf{VFS}_{k,\hat{\rho}}$ for L with

$$\tilde{\epsilon} = (k+1)\cdot\epsilon \quad , \quad \tilde{\delta} = \delta.$$

Now consider the last consistent-reader, Ψ_{k+1}. Since ρ' is a $\frac{\delta'}{2}$-code, the table T'_1 has at most $2/\delta'$ permissible values (according to ρ' and δ'), referred to as *permissible witnesses.*

We would like to prove that an $\hat{\epsilon}$ fraction of satisfaction in $\hat{\Phi}$ implies a permissible good input. Assume therefore that an $\hat{\epsilon}$ fraction of functions $\hat{\varphi}_{\varphi,\psi_1,\ldots,\psi_{k+1}}$ in $\hat{\Phi}$ are satisfied over a table. By the definition, for any satisfied $\hat{\varphi}_{\varphi,\psi_1,\ldots,\psi_{k+1}}$, ψ_{k+1} is non-faulty, and it may be erroneous for at most ϵ' fraction of functions. If ψ_{k+1} is not erroneous, however, it must agree with one of the $2/\delta'$ permissible witnesses.

Therefore an

$$\frac{\hat{\epsilon}-\epsilon'}{2/\delta'}$$

fraction of functions $\hat{\varphi}_{\varphi,\psi_1,\ldots,\psi_{k+1}}$ in $\hat{\Phi}$ are both: 1) satisfied, and 2) the corresponding ψ_{k+1} agrees with one specific permissible witnesses. Since

$$\frac{\hat{\epsilon}-\epsilon'}{2/\delta'} = \tilde{\epsilon},$$

the existence of permissible good input follows from the fact that $\tilde{\Phi}$ is a $(\tilde{\epsilon}, \delta)$-$\mathsf{VFS}_{k,\hat{\rho}}$ for L. □

Proof of Lemma 1

Let us now attend to the proof of lemma 1, the initial construction. The construction is carried out following the process described above: first constructing, for any $L \in \mathsf{NP}$, a presuming-VFS; one, moreover, whose `DEPEND` parameter is much larger than desired, namely poly-logarithmic in n. That construction is then combined with a consistent-reader for the representation used, so as to form a (non-presuming) VFS for L whose `DEPEND` parameter is constant.

Let us still defer a little the specifics of the representations used by our analysis. Let us instead introduce two classes of efficiently-computable input-representations, LDE and PLDE — whose precise definitions appear below in section 4.1 — both classes consisting of one representation for input-size n, ϵ, s and c_1 (where $s\cdot\epsilon^{c_1} \geq 1$, $s \geq n$, and $c_1 \geq 1$). For any specific setting to these parameters, $\mathsf{LDE}[n, s, \epsilon, c_1]$ is a (ϵ^{c_1})-code, encoding each string of length n, by a vector whose size is bounded from above by a small, fixed polynomial in s.

LDE is defined so as to allow an ϵ-presuming-VFS for any $L \in \mathsf{NP}$ according to $\rho = \mathsf{LDE}[n, s, \epsilon, c_1]$ (lemma 5 below). $\mathbf{e}_\rho$ is in particular a geometric-code[RS96], enabling the application of the consistency-test there. Accordingly, $\mathsf{PLDE}[n, s, \epsilon, c_1]$ is defined to be the plane-representation of $\mathsf{LDE}[n, s, \epsilon, c_1]$. Consequently, it is possible to construct a consistent-reader, that expects its input according to $\mathsf{PLDE}[n, s, \epsilon, c_1]$ and returns values according to $\mathsf{LDE}[n, s, \epsilon, c_1]$, whose error-probability is ϵ (lemma 6 below).

Note that each setting for the parameters of LDE and PLDE results in a representation of a different alphabet, hence implying a VFS of different VAR-SIZE. The reader may choose to proceed here, or alternatively to read section 4.1 first.

The first step — the construction of a non-presuming VFS for any $L \in \mathsf{NP}$ — is as follows:

Lemma 5 (Presuming-VFS) *Let $s(n) \geq n$, $\epsilon(n)$, and $c_1 \geq 1$, satisfying $s(n) \cdot \epsilon(n)^{c_1} \geq 1$. For any $L \in \mathsf{NP}$, and any constant k there exists (constructible) ϵ-presuming-$\mathsf{VFS}_{k,\rho}$ Φ for L, where $\rho = \mathsf{LDE}[n, s(n), \epsilon(n), c_1]$ such that:*

- $\mathtt{SIZE}(\Phi) \leq s(n)^{O(1)}$
- $\mathtt{DEPEND}(\Phi) \leq (\log n)^{c_2}$*, for some constant c_2*
- $\mathtt{VAR\text{-}SIZE}(\Phi) \leq 2^{O\left(\log\log n + \log\frac{1}{\epsilon(n)}\right)\cdot \log n/\log s(n)} \cdot (\log s(n) \cdot \log n)^{c_3}$*, for some constant c_3*
- $\mathtt{COMP}(\Phi) \leq (\mathtt{DEPEND}(\Phi) \cdot \mathtt{VAR\text{-}SIZE}(\Phi))^{c_4}$*, for some constant c_4*

The proof for lemma 5 can be found below in section 6.

Now, in order to transform Φ into a non-presuming VFS, one needs to construct a consistent-reader for $\rho = \mathsf{LDE}[n, s(n), \epsilon(n), c_1]$:

Lemma 6 (Consistent-Reader) *Let $\rho = \mathsf{LDE}[n, s, \epsilon, c_1]$ and $\hat{\rho} = \mathsf{PLDE}[n, s, \epsilon, c_1]$. Assume a collection of tuples $\mathcal{T}$, each tuple of size at most $(\log n)^c$, for some global constant c.*[14] *Then, there exists (constructible) $[\rho, \mathcal{T}, \hat{\rho}]$-$(\epsilon, \frac{\epsilon}{4})$-consistent-reader Ψ, which complies with the following complexity bounds:*

- $\mathtt{SIZE}(\Psi) \leq |\mathcal{T}| \cdot s^{c_2}$*, for some constant c_2.*
- $\mathtt{DEPEND}(\Psi) = 3$
- $\mathtt{VAR\text{-}SIZE}(\Psi) \leq 2^{O\left(\log\log n + \log\frac{1}{\epsilon}\right)\cdot \log n/\log s} \cdot (\cdot \log n)^{c_3}$*, for some constant c_3.*
- $\mathtt{COMP}(\Psi) \leq (\mathtt{DEPEND}(\Psi) \cdot \mathtt{VAR\text{-}SIZE}(\Psi))^{c_4}$*, for some constant c_4.*

The proof for lemma 6 can be found in section 5.

Lemma 1 is now proved by lemma 4, combined with lemma 5 and lemma 6, where ϵ', δ' (in lemma 4) are taken to be of the order of $\sqrt{\epsilon}$.

Note, that even though Ψ_{k+1} is a consistent-reader for the identity representation $\mathcal{I}$, which is not a member of the class LDE, lemma 6 can still be used to construct Ψ_{k+1}. This is true because every representation $\rho \in \mathsf{LDE}$ will be an extension of $\mathcal{I}$, that is, the restriction of any $\rho(I)$ to some specific entries will give I. Therefore, any consistent-reader for any $\rho \in \mathsf{LDE}$ is in particular a consistent reader for $\mathcal{I}$ as well. $\square$

[14] Being c_2 of the Presuming lemma (lemma 5) above

4.1 Low-Degree-Extension

Let us now attend to the precise definition of the two classes of codes LDE and its plane-representation variant PLDE. Let us stress that even though the Initial-Construction is proven using a particular code, another construction, plausibly using a different code, as long as it satisfies these conditions, may replace the specific one used here.

$\rho = \mathsf{LDE}[n, s, \epsilon, c_1]$ is defined to be a geometric-code representing each n-bit string by a vector of size bounded from above by a fixed polynomial of s (say $2s^2$). ρ induces a set $\mathbf{e}_\rho$, which is a ϵ^{c_1}-geometric-code, that is, a set of functions (vectors) over a geometric-space, $\mathcal{F}^d$, for some finite-field $\mathcal{F}$ and dimension d. Let us assume here, for simplicity, $\mathcal{F}$ to be Z_p, for some integer p $(= |\mathcal{F}|)$.

The size of $\mathcal{F}$ is set to be the smallest (prime) such that

$$|\mathcal{F}| \geq \left(\epsilon^{-c_1} \cdot (\log n)\right)^c$$

for some large enough, fixed constant c. The dimension d is then set to be the smallest integer such that

$$d \geq \frac{2 \log s}{\log |\mathcal{F}|}.$$

In addition, let the set $H = [0, .., h-1]$, be such that h is the smallest integer satisfying $h^d \geq n+1$; hence, by the definition of d, $h < |\mathcal{F}|$. The set $\mathbf{e}_\rho$ is defined to consist of all d-variable polynomials

$$f \colon \mathcal{F}^d \to \mathcal{F}$$

of total degree $d \cdot |H|$.

H^d is a subset of $\mathcal{F}^d$ (which is the set of entries to $f \in \mathbf{e}_\rho$). H also induces a natural mapping of the indexes $[0, .., n+1]$ to H^d, simply by considering the base-$|H|$ representation of the index, which requires d digits. Let us assume for simplicity that $\left|H^d\right|$ is exactly $n+1$.

It now remains to determine which input does such a function $f \in \mathbf{e}_\rho$ represents (if at all), i.e., determine $\rho(I)$ for every n-bits string I.

Some of the vectors (functions) $f \in \mathbf{e}_\rho$, if restricted to H^d, result in a Boolean-function, i.e., assign values to $[1, ..., n+1]$ that are all either 0 or 1. Any such function f whose $(n+1)^{\underline{\text{th}}}$ bit is 1,[15] represents the input $I \in \{0,1\}^n$ consisting of f's values on all other indexes. Every input $I \in \{0,1\}^n$, on the other hand, viewed as a Boolean-function over H^d (where the $(n+1)^{\underline{\text{th}}}$ bit is set to 1), has a unique extension into a function $\hat{I} \colon \mathcal{F}^d \to \mathcal{F}$ in $\mathbf{e}_\rho$.

By the properties of low-degree polynomials, it follows that ρ is an α-code for

$$\alpha = \frac{d \cdot h}{|\mathcal{F}|}.$$

Plane Representation

$\rho = \mathsf{LDE}[n, s, \epsilon, c_1]$ is, as mentioned above, a geometric-code, hence its plane-representation is defined as follows: Consider all planes of $\mathcal{F}^d$, i.e., all affine subspaces of dimension 2. Assume some linear 1-to-1 mapping between every plane $\wp$ of $\mathcal{F}^d$ and $\mathcal{F}^2$.

[15] For some technical reasons occuring in the Presuming lemma, one of the bits of the string must be assumed to be fixed to 1.

Now consider a function F assigning a polynomial $F(\wp)\colon \mathcal{F}^2 \to \mathcal{F}$ of degree $d \cdot |H|$ to each plane $\wp$ of $\mathcal{F}^d$. $\hat{\rho} = \mathsf{PLDE}[k, n, s, \epsilon]$ is defined to be such that $\mathbf{e}_{\hat{\rho}}$ is the set of all F for which there exists an $f \in \mathbf{e}_\rho$ where each plane $\wp$ is assigned by F the restriction of f to $\wp$. $\hat{\rho}(I)$ is simply the function F that results from $\rho(I)$.

5 Consistent-Reader for LDE (sketch)

In this section we show the construction of a consistent-reader for input representations in the class LDE, and prove lemma 6. The consistent-reader is based on the new low-degree-test of [RS96], combined with some known methods from [LS91, FL92, ALM+92].

Low-Degree-Test

Let $\rho = \mathsf{LDE}[n, s, \epsilon, c_1]$, and let $\hat{\rho} = \mathsf{PLDE}[n, s, \epsilon, c_1]$ be the corresponding plane-representation. Let $\mathcal{F}, d, h$ be the corresponding parameters, as described in section 4.1, and denote

$$\alpha = \frac{d \cdot h}{|\mathcal{F}|}.$$

For some (large enough) fixed constant c, let ϵ, δ be such that

$$\epsilon \geq c \cdot d \cdot \alpha^{1/c} \quad , \quad \delta = \epsilon/4.$$

Let T_1 be a table, supposedly representing some input $I \in \{0,1\}^n$ according to the representation $\hat{\rho}$, and denote by $\mathcal{S}[\delta]$ the set of T_1-permissible values according to $\hat{\rho}$ and δ.

A new notion of low-degree-tests was introduced in [RS96]. A new low-degree-test (according to the new notion) for such a table, T_1, was introduced as well. This test induces the following function-system Φ: For every two planes p_1, p_2 of $\mathcal{F}^d$, intersecting by a line, Φ has one function φ_{p_1,p_2}, defined in the following way: φ_{p_1,p_2} is satisfied iff $T_1(p_1), T_2(p_2)$ agree on their intersection.

A function φ_{p_1,p_2} is *erroneous* over T_1 if it is satisfied, however every T_1-permissible value (i.e. a value in $\mathcal{S}[\delta]$) disagrees with either $T_1(p_1)$ or $T_1(p_2)$. It was proved in [RS96] that the fraction of erroneous functions in Φ is at most $\epsilon/2$, for ϵ, δ as above.

Consistent-Reader for one Tuple

Consider one tuple τ, consisting of l indexes to entries of the representation ρ, that is, a tuple of l points in $\mathcal{F}^d$. For every element $x \in \mathcal{F}^d$, let $\Gamma[\tau, x]$ be a 1-dimensional polynomial-curve in $\mathcal{F}^d$, defined in the following way: $\Gamma[\tau, x]$ is the curve of smallest degree, whose first l points are the l points in τ (in some fixed order), and whose $(l+1)^{\underline{\text{th}}}$ point is x. Note that if x is uniformly distributed on $\mathcal{F}^d$, then every other point of $\Gamma[\tau, x]$, (except the first l points), is also uniformly distributed on $\mathcal{F}^d$.

Obviously, $\Gamma[\tau, x]$ is a curve of degree at most l, (that is, every coordinate of $\Gamma[\tau, x]$ is defined by a one-variable polynomial of degree l). Therefore, the restriction of $\rho(I)$ to any such $\Gamma[\tau, x]$ can be described by a one-variable polynomial of degree at most $l \cdot \alpha\,|\mathcal{F}|$.

Let T_1 be a table as above, and let T_2 be another table, supposedly containing, for every x, the restriction of $\rho(I)$ to $\Gamma[\tau, x]$.

The reader Ψ_τ is defined in the following way: Ψ_τ has one function $\psi_{p_1,p_2,x,y}$ for every pair of planes p_1, p_2, and pair of points x, y, such that p_1, p_2 intersect by a line containing y, and y is a point on the curve $\Gamma[\tau, x]$ – other than the first l points. The function $\psi_{p_1,p_2,x,y}$ is defined in the following way: If $T_1(p_1), T_2(p_2)$ agree on their intersection, and they both agree with $T_2(x)$ on the point y then $\psi_{p_1,p_2,x,y}$ returns for every point in τ the value given to it by $T_2(x)$. Otherwise, $\psi_{p_1,p_2,x,y}$ outputs `fault`.

$\psi_{p_1,p_2,x,y}$ is *erroneous* over (T_1, T_2) if it is non-faulty, and disagrees with all T_1-permissible values (i.e. values in $\mathcal{S}[\delta]$). How can $\psi_{p_1,p_2,x,y}$ be erroneous ? If $T_2(x)$ agrees with one of the permissible values then $\psi_{p_1,p_2,x,y}$ is not erroneous. However, If $T_2(x)$ disagrees with all permissible values then it may agree with each one of them on at most $l \cdot \alpha\,|\mathcal{F}|$ points y (this is because, as mentioned above, $T_2(x)$ can be described by a one-variable polynomial of degree $l \cdot \alpha\,|\mathcal{F}|$). Thus, if $T_2(x)$ disagrees with all the permissible values, the number of points y where an agreement with at least one of the permissible values occures, is at most

$$|\mathcal{S}[\delta]| \cdot l \cdot \alpha\,|\mathcal{F}| \leq \delta^{-1} \cdot l \cdot \alpha\,|\mathcal{F}|$$

(since the number of permissible values is at most δ^{-1}).

Therefore, it is not hard to see that the total error-probability, that is the fraction of erroneous functions $\psi_{p_1,p_2,x,y} \in \Psi_\tau$ is at most

$$\epsilon_\Psi = \epsilon/2 + \delta^{-1} \cdot l \cdot \alpha \cdot \frac{|\mathcal{F}|}{|\mathcal{F}| - l} \leq \epsilon.$$

Consistent-Reader for many Tuples

We can now combine the readers above to get a consistent reader for any set $\mathcal{T}$, with the same ϵ, δ. Lemma 6 is hence proved. □

6 Presuming-VFS for NP (sketch)

In this section we prove lemma 5. The proof for $k = 1$ is the standard *sum-check* procedure of [LFKN92, Sha92, BFL91, FGL+91, BFLS91]. The proof for larger k follows by the same sum-check procedure, combined with a simple "concatenation lemma", similar to the one of [AS92]. Note that since the VFS is *presuming*, no low-degree-test is needed.

A complete proof will appear in the full version of this paper.

7 Proof of Theorem 8, and Theorem 7

Theorem 8 follows by similar techniques as the one of theorem 6, that is, by applying the Initial-Construction-lemma and the Composition-lemma. It may be necessary, however, to apply the composition-lemma more than twice. For example; if one wants to achieve $\epsilon = 2^{-(\log n)^{1-1/r}}$ one has to apply the composition lemma r times.

In order to prove theorem 7 from theorem 8, one should apply the composition lemma once more. This time a new catalyst is used, one that places NP within a PCP class with; constant DEPEND, VAR-SIZE $= M(n)$, and $\epsilon = 2^{-M(n)}$. The SIZE parameter, even though being exponential in the input-size, is nevertheless polynomial in the *original* input-size. The characterization used (for the catalyst) is proven in a lemma generalizing the one of [ALM+92], using larger fields, rather than the 2-elements field used in [ALM+92].

A complete proof will appear in the full version of this paper.

References

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 13–22, 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. In *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 2–13, 1992.

[Bab85] L. Babai. Trading group theory for randomness. In *Proc. 17th ACM Symp. on Theory of Computing*, pages 421–429, 1985.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd ACM Symp. on Theory of Computing*, pages 21–31, 1991.

[BGKW88] M. Ben-or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi prover interactive proofs: How to remove intractability assumptions. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 113–121, 1988.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient multi-prover interactive proofs with applications to approximation problems. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 113–131, 1993.

[BGS95] M. Bellare, O. Goldreich, and M. Sudan. Free bits and nonapproximability. In *Proc. 27th ACM Symp. on Theory of Computing*, 1995. to appear.

[BK89] M. Blum and S. Kannan. Designing programs that check their work. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 86–97, 1989.

[BLR90] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 73–83, 1990.

[Coo71] S. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd ACM Symp. on Theory of Computing*, pages 151–158, 1971.

[Fei95] U. Feige. Set cover has a theshold at $\ln n$. Unpublished Manuscript, 1995.

[FGL+91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991.

[FL92] U. Feige and L. Lovász. Two-prover one-round proof systems: Their power and their problems. In *Proc. 24th ACM Symp. on Theory of Computing*, pages 733–741, 1992.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings of the 3rd Conference on Structure in Complexity Theory*, pages 156–161, 1988.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proofs. *SIAM J. of Computation*, 18:186–208, 1989.

[Hås96a] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proc. 37th IEEE Symp. on Foundations of Computer Science*, 14–16 October 1996.

[Hås96b] Johan Håstad. Testing of the long code and hardness for clique. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 11–19, Philadelphia, Pennsylvania, 22–24 May 1996.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In Miller and Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, October 1992.

[Lip89] R. Lipton. Efficient checking of computations. In *Proceedings of 6th STACS*, 1989.

[LS91] D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXP-TIME. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 13–18, 1991.

[LY94] Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41(5):960–981, 1994.

[PY91] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[Raz94] R. Raz. A parallel repetition theorem. Manuscript, 1994.

[RS96] R. Raz and S. Safra. A sub-constant error-probability PCP characterization of NP; part II: The consistecy-test. Manuscript, 1996.

[Sha92] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, October 1992. Prelim. version in 1990 FOCS, pages 11–15.