Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 123 (2005) 165–177
www.elsevier.com/locate/entcs

Probabilistically Checkable Proofs Over the Reals

Klaus Meer[1,2]

Department of Mathematics and Computer Science
Syddansk Universitet, Campusvej 55, 5230 Odense M, Denmark

Abstract

Probabilistically checkable proofs (PCPs) have turned out to be of great importance in complexity theory. On the one hand side they provide a new characterization of the complexity class NP, on the other hand they show a deep connection to approximation results for combinatorial optimization problems. In this paper we study the notion of PCPs in the real number model of Blum, Shub, and Smale. The existence of transparent long proofs for the real number analogue $NP_{\mathbb{R}}$ of NP is discussed.

Keywords: PCP, real number model, self-testing over the reals.

1 Introduction

One of the most important and influential results in theoretical computer science within the last decade is the PCP theorem proven by Arora et al. in 1992, [1,2]. Here, PCP stands for probabilistically checkable proofs, a notion that was further developed out of interactive proofs around the end of the 1980's. The PCP theorem characterizes the class NP in terms of languages accepted by certain so-called verifiers, a particular version of probabilistic Turing machines. It allows to stabilize verification procedures for problems in NP in the following sense. Suppose for a problem $L \in$ NP and a problem

[1] partially supported by the EU Network of Excellence PASCAL Pattern Analysis, Statistical Modelling and Computational Learning and by the Danish Natural Science Research Council SNF.

[2] Email: meer@imada.sdu.dk

doi:10.1016/j.entcs.2004.04.047

instance $x \in \{0,1\}^*$ we want to check whether a potential proof $y \in \{0,1\}^*$ of polynomial size verifies $x \in L$. Then it is quite likely (unless P = NP) that we have to read all components of y before we can make up a decision whether y proves $x \in L$. The stabilization idea behind the PCP theorem asks for using different proofs which have the following property: if $x \notin L$, then all proofs will be rejected with high probability by inspecting only a certain amount of components in the proof. The surprising result is that the above term 'certain amount' can be taken as 'constantly many'. Being of immense interest already by itself, the PCP theorem has been shown to be at the bottom of approximability properties of NP-hard combinatorial optimization problems. It was used to show several (non-) approximability results not known before. For an introduction and a proof of the PCP theorem see [3,12].

Parallelizing structural complexity theory over finite alphabets and unifying existing approaches in algebraic complexity theory, Blum, Shub, and Smale in 1989 introduced a uniform real number model of computation, now called the BSS machine [6]. In the meanwhile, a lot of work has been done in the BSS model. The textbooks [7,5,4] shed an excellent light on problems considered in this framework.

However, so far neither PCP results nor a notion of approximation over the reals that seems appropriate to the classical one [3] have been studied for the BSS model (interactive proofs, however, have been considered in [13]). Such a study seems both tempting and important due to the significance PCP results have shown in the Turing model.

Here, such an attempt will be described. Note that the original question posed in the PCP theorem makes perfect sense in the real framework as well: Can we have a verification procedure for an $\mathrm{NP}_{\mathbb{R}}$ problem $L \subseteq \mathbb{R}^* := \bigcup_{n=1}^{\infty} \mathbb{R}^n$ such that in case $x \notin L$ for any potential proof y only constantly many real components of y have to be checked in order to reject with high probability? We introduce real verifiers as well as real $\mathrm{PCP}_{\mathbb{R}}$-classes. It is then shown how stable verification proofs for $\mathrm{NP}_{\mathbb{R}}$-complete problems can be constructed. More precisely, we design transparent (i.e. exponentially) long proofs that use a constant number of components, only. This will show $\mathrm{NP}_{\mathbb{R}} \subseteq \mathrm{PCP}_{\mathbb{R}}(poly, O(1))$. In the final section we briefly outline the relation to approximation problems over the reals. Most details can be found in [11].

[3] Here, we do not have in mind approximation issues as treated in numerical analysis. Such a notion likely will lead to problems in the original BSS model unless analytic functions are introduced, see [8] and [10]. Therefore, a more combinatorial notion seems appropriate.

2 The problem setting

2.1 The BSS model; Quadratic Polynomial Systems

In the BSS model over $\mathbb{R}$ real numbers are considered as entities. The basic arithmetic operations $+, -, *, :$ can be performed at unit costs, and there is a test-operation "is $x \geq 0$?" reflecting the underlying ordering of the reals. Decision problems now are subsets $L \subseteq \mathbb{R}^* := \bigcup_{n=1}^{\infty} \mathbb{R}^n$. The (algebraic) size of a point $x \in \mathbb{R}^n$ is n. Having fixed these notions it is easy to define real analogues $\mathrm{P}_{\mathbb{R}}$ and $\mathrm{NP}_{\mathbb{R}}$ of the classes P and NP as well as the notion of $\mathrm{NP}_{\mathbb{R}}$-completeness. For more details on the BSS model we refer to [5].

Example 2.1 (Quadratic Polynomial Systems) Let us consider a typical decision problem over the reals. The QPS (Quadratic Polynomial Systems) decision problem is given as:

INPUT: Integers $n, m \in \mathbb{N}$, a system $p := (p_1, \ldots, p_m)$ of m real polynomials in n variables, each of degree at most 2; moreover, each p_i depends on at most 3 variables.

QUESTION: Is there a common real solution $a \in \mathbb{R}^n$ such that $p(a) = 0 \in \mathbb{R}^m$?

Without loss of generality we can assume $m = O(n)$ (by adding dummy variables). The QPS problem is $\mathrm{NP}_{\mathbb{R}}$-complete [5]. Let us here just indicate why it is in $\mathrm{NP}_{\mathbb{R}}$: For an input $n, m, p_1, \ldots, p_m$ we guess a $y \in \mathbb{R}^n$, evaluate $p_i(y)$ for all i and accept iff all results vanish. It is easy to see that this verification procedure needs polynomial (algebraic) running time in the (algebraic) input size, only.

It is the QPS problem for which we are going to construct a real number verifier later on.

2.2 Verifiers and the classical PCP theorem

In this section first we briefly recall the basic notions used to state the classical PCP theorem. Considering an input $\Phi(x_1, \ldots, x_n)$ for the NP-complete 3-SAT decision problem, the 'natural' NP-verification is given by guessing an assignment $y \in \{0, 1\}^n$, plugging it into Φ and checking $\Phi(y) = 1$. Obviously, unless P = NP this procedure in general requires to inspect all components of y in order to get the right answer. The same holds for the above 'natural' verification procedure showing that QPS belongs to $\mathrm{NP}_{\mathbb{R}}$.
The idea behind PCP theorems now is to show the existence of other verification procedures that are more stable in that only a constant number of proof-components have to be checked. The price to pay for it is that an input not belonging to the language under consideration might be accepted with a

certain, though small, probability (recall that in the definition of NP a false verification is always rejected). Formally, this idea is captured by introducing the notion of a verifier.

Definition 2.2 a) Let $r, q : \mathbb{N} \mapsto \mathbb{N}$ be two functions. A $(r(n), q(n))$-restricted verifier V in the Turing model is a particular randomized Turing machine working as follows. For an input $x \in \{0,1\}^*$ of size n and another vector $y \in \{0,1\}^*$ representing a potential membership proof of x in a certain language, the verifier first produces a sequence of $r(n)$ many random bits (under the uniform distribution on $\{0,1\}^{r(n)}$). Given x and these $r(n)$ many random bits V computes in deterministic polynomial time in n the indices of $q(n)$ many components of y. Finally, V uses the input x together with the values of the chosen components of y in order to perform a deterministic polynomial time algorithm. At the end of this algorithm V either accepts or rejects (x, y). We denote by $V(x, y, \rho)$ the result of V supposed the random sequence generated for input (x, y) was ρ.
b) Part a) can be adapted almost word by word in order to define verifiers for the BSS model. The randomized part will be a real number algorithm that tosses coins. The input x and the verification y now belong to $\mathbb{R}^*$. The bit-length of x is replaced by its algebraic size.

Remark 2.3 i) It is important to note that the probability notions used in the definition of real number verifiers still refer to *discrete* sample spaces and their uniform distribution.
ii) Probabilistic BSS machines whose randomization relies on coin toss are studied in [9]. There $\mathrm{BPP}_{\mathbb{R}} = \mathrm{P}_{\mathbb{R}}$ is shown. Our results actually give certain lower bound information on deterministic algorithms simulating $\mathrm{BPP}_{\mathbb{R}}$ computations (if $\mathrm{P}_{\mathbb{R}} \neq \mathrm{NP}_{\mathbb{R}}$).

Verifiers now are used to define PCP- and $\mathrm{PCP}_{\mathbb{R}}$-complexity classes. Since the latter are the newly introduced ones, here we only give the definition for the real classes. The PCP-classes in the Turing framework are defined similarly by replacing once again the obvious terms in the BSS setting through those in the Turing model.

Definition 2.4 Let $r, q : \mathbb{N} \mapsto \mathbb{N}$; a real number decision problem $L \subseteq \mathbb{R}^*$ is in class $\mathrm{PCP}_{\mathbb{R}}(r(n), q(n))$ iff there exists a $(r(n), q(n))$-restricted verifier V such that conditions i) and ii) below hold:

i) For all $x \in L$ there is a $y \in \mathbb{R}^*$ such that for all randomly generated strings $\rho \in \{0,1\}^{r(size_{\mathbb{R}}(x))}$ the verifier accepts: $\Pr_{\rho}\{V(x, y, \rho) = \ 'accept'\} = 1$.

ii) For any $x \notin L$ and for each $y \in \mathbb{R}^*$ it is $\Pr_{\rho}\{V(x, y, \rho) = \ 'reject'\} \geq \frac{1}{2}$.

The probability is chosen uniformly over all strings $\rho \in \{0,1\}^{r(size_{\mathbb{R}}(x))}$.

Example 2.5 It is easy to see that $\mathrm{NP} = \mathrm{PCP}(0, poly), \mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(0, poly)$ as well as $\mathrm{PCP}(O(\log n), O(1)) \subseteq \mathrm{NP}, \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1)) \subseteq \mathrm{NP}_{\mathbb{R}}$. Try yourself proving $\mathrm{P} = \mathrm{PCP}(O(\log n), 2)$ in the Turing setting. What's about the analog statement in the BSS model?

The PCP theorem gives the following surprising characterization of NP :

Theorem 2.6 *(*PCP *theorem, [1,2])* $\mathrm{NP} = \mathrm{PCP}(O(\log n), O(1))$.

A central first step in the proof of the above theorem is to show that NP has transparent long proofs, i.e. $\mathrm{NP} \subset \mathrm{PCP}(poly, O(1))$. Some of the important techniques used to prove the full PCP theorem come into play already here. The latter are so-called self-testing and self-correction of linear functions over $\mathbb{Z}_2^n$. In the real framework these techniques have to be generalized to very different domains.

3 Transparent long proofs for $\mathrm{NP}_{\mathbb{R}}$

Our main result establishes the existence of transparent long proofs for $\mathrm{NP}_{\mathbb{R}}$. Formally, it states

Theorem 3.1 $\mathrm{NP}_{\mathbb{R}} \subset \mathrm{PCP}_{\mathbb{R}}(poly, O(1))$.

'Transparent proofs' refers to the fact that only constantly many components of a verification proof have to be inspected. 'Long proofs' reflects that when producing a polynomial number of random bits there are exponentially many different random strings and thus exponentially many proof-components that in principle might be inspected. The corresponding statement $\mathrm{NP} \subset \mathrm{PCP}(poly, O(1))$ is the first major ingredient for proving the PCP theorem over $\mathbb{Z}_2^n$, see Theorem 5 in [1]. We thus may hope for establishing as well the following

Conjecture: $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$.

3.1 Where new difficulties come from

Since polynomial time reductions can be included in the computation of a verifier, for showing Theorem 3.1 it is sufficient to prove that the $\mathrm{NP}_{\mathbb{R}}$-complete QPS problem admits a $(poly, O(1))$-verifier. The main question to solve is to find out what a stable verification proof for QPS should look like. Since arithmetization of the 3-SAT problem is a major ingredient of the classical PCP theorem, it is natural to follow a similar approach. This approach replaces

a satisfying assignment by certain linear (and later: polynomial) functions it generates. Then, tables of the function values replace the assignment itself. Stability of this verification is implied by randomly checking internal consistency (i.e. that the tables do represent linear functions which all arise from a single assignment) and solvability (i.e. that the assignment is satisfying).

However, two severe difficulties arise. First, almost all probability statements needed in the classical proof are heavily relying on closeness of $\mathbb{Z}_2^n$ under addition and an additive shift invariance of the uniform distribution over $\mathbb{Z}_2^n$. This means that for fixed $a, b \in \mathbb{Z}_2^n$ we have

$$\Pr_{x \in \mathbb{Z}_2^n} \{x = b\} = \Pr_{x \in \mathbb{Z}_2^n} \{a + x = b\} \ .$$

Secondly, over $\mathbb{Z}_2^n$ there are no other constants present beside $\{0, 1\}$. Thus, linearity over $\mathbb{Z}_2^n$ is equivalent to the condition

$$\forall x, y \in \mathbb{Z}_2^n : f(x + y) = f(x) + f(y).$$

As soon as real numbers come into play both conditions are violated. Since we cannot enlarge our function tables to $\mathbb{R}^n$, the first major problem to solve is: What is the right domain $\mathcal{X} \subset \mathbb{R}^n$ on which we should guess function values for the verification procedure.

This domain $\mathcal{X}$ has to be finite and has to involve real numbers present in the concrete problem instance.The uniform distribution on such a $\mathcal{X}$ usually will neither be shift invariant nor will $\mathcal{X}$ even be closed under additive shifts. This problem gets worse by the fact that linearity on $\mathcal{X}$ now also requires the consideration of real scalar factors. Again, it is not obvious from which finite sets of reals those scalar factors should be taken. Rubinfeld and Sudan [14] have treated related problems with respect to so-called rational domains which are certain well-structured subsets of $\mathbb{Q}$. We shall show how such ideas can be generalized to those subsets of reals that come up in relation with $\mathrm{NP}_{\mathbb{R}}$-complete problems.

3.2 Linear functions related to QPS

Consider an instance of QPS. For real polynomials $p_1, \ldots, p_m$ over n variables define

$$P(y, \tilde{r}) := \sum_{i=1}^{m} p_i^2(y) \cdot \tilde{r}_i \ , \ \tilde{r} = (\tilde{r}_1, \ldots, \tilde{r}_m) \in \mathbb{Z}_2^m. \tag{1}$$

Then it is $P(y, \tilde{r}) \geq 0$ for all $y \in \mathbb{R}^n, \tilde{r} \in \mathbb{Z}_2^m$; moreover, $P(a, \tilde{r}) = 0$ for all

$\tilde{r} \in \mathbb{Z}_2^m$ only if $a \in \mathbb{R}^n$ is a common zero and otherwise

$$\Pr_{\tilde{r} \in \mathbb{Z}_2^m} \{P(a, \tilde{r}) > 0\} \geq \frac{1}{2}. \tag{2}$$

The structure of P is most important for what follows. It is easily seen that this structure can be splitted into two different parts, one only depending on the real coefficients of the input polynomials p_i, the other depending on an assignment $a \in \mathbb{R}^n$ for the variables y. We have

Lemma 3.2 *There are linear functions A, B, C, D depending on $a \in \mathbb{R}^n$, only, as well as linear functions $L_A, \ldots, L_D, E : \mathbb{Z}_2^m \mapsto \mathbb{R}^n, \mathbb{R}^{n^2}, \mathbb{R}^{n^3}, \mathbb{R}^{n^4}, \mathbb{R}$, respectively, such that*

$$\forall\ \tilde{r} \in \mathbb{Z}_2^m \quad P(a, \tilde{r}) = E(\tilde{r}) + A \circ L_A(\tilde{r}) + \ldots + D \circ L_D(\tilde{r}).$$

More precisely,

$$A : \mathbb{R}^n \mapsto \mathbb{R}\ ,\ A(x_1, \ldots, x_n) = \sum_{i=1}^{n} a_i \cdot x_i\ \forall\ x \in \mathbb{R}^n;$$

$$B : \mathbb{R}^{n^2} \mapsto \mathbb{R}\ ,\ B(y_1, \ldots, y_{n^2}) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j \cdot y_{ij}\ \forall\ y \in \mathbb{R}^{n^2}$$

(where we denote by y_{ij} the argument $(i-1)n + j$);

$$C : \mathbb{R}^{n^3} \mapsto \mathbb{R}\ ,\ C(z_1, \ldots, z_{n^3}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} a_i a_j a_k \cdot z_{ijk}\ \forall\ z \in \mathbb{R}^{n^3}$$

(where z_{ijk} denotes the argument $(i-1)n^2 + (j-1)n + k$);

$$D : \mathbb{R}^{n^4} \mapsto \mathbb{R}\ ,\ D(w_1, \ldots, w_{n^4}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{\ell=1}^{n} a_i a_j a_k a_\ell \cdot w_{ijk\ell}\ \forall\ w \in \mathbb{R}^{n^4}$$

(where $w_{ijk\ell}$ denotes the argument $(i-1)n^3 + (j-1)n^2 + (k-1)n + \ell$).

Whereas functions like $A, \ldots, D$ occur as well in the Turing setting, the $L_A, \ldots, L_D$ are new. Their presence creates most of the problems because evaluation on $\tilde{r} \in \mathbb{Z}_2^m$ gives real vectors depending on the actual input (i.e. its real coefficients). Thus, $A, \ldots, D$ have to be evaluated on real number domains. This results in the necessity of developing new techniques for self-testing and -correcting those linear functions.

A natural first attempt for the right real domain on which to test linearity of, say, A is $L_A(\mathbb{Z}_2^m)$. However, in general when choosing $x, y \in L_A(\mathbb{Z}_2^m)$ the element $x + y$ will not any more belong to this set (due to the fact that for $\tilde{r}, \tilde{t} \in \mathbb{Z}_2^m$ we now have to compute $\tilde{r} + \tilde{t}$ over $\mathbb{R}^m$ instead of $\mathbb{Z}_2^m$). Thus, this idea

eventually leads to infinite sets which cannot be used as domains on which to guess function values. Instead, we extend an idea developed by Rubinfeld and Sudan [14] for so-called rational domains. Our extension applies to a much larger amount of finite subsets over $\mathbb{R}$ and covers all those potential domains that might be generated in the above explained way by an instance for QPS. A second problem that has to be taken into account is that linearity is not any longer equivalent to the additivity condition, only. It might be the case that $L_A(\mathbb{Z}_2^m)$ splits into several components that will not result from each other by additive shifts. Then even if A satisfies additivity on the considered domain we have to exclude the situation where A corresponds to different linear functions, one on each component. Therefore, another test checking scalar multiplicativity has to be performed. Here, once more a question is from which domain we have to choose the scalars.

The outline of how the verifier works is as follows:

(i) As guess the verifier uses function tables for A, B, C, D on appropriate domains. These tables will have an exponential size in the input size.

(ii) Check that all functions are linear with high probability.

(iii) Check that all functions arise from the same vector $a \in \mathbb{R}^n$ with high probability.

(iv) For randomly chosen $\tilde{r} \in \mathbb{Z}_2^m$ compute the value of $P(a, \tilde{r})$ by looking into the function tables; check whether the result is 0.

4 Proof details

In this section we collect the mathematical details necessary to prove the main result. Full proofs can be found in [11].

4.1 Testing additivity

Let us first consider the map A. For the linear map $L_A : \mathbb{Z}_2^m \mapsto \mathbb{R}^n$ let $C_0 := \{\lambda_1, \ldots, \lambda_K\} \subset \mathbb{R}$ be the multiset of all entries in the matrix representation of L_A, without loss of generality $\lambda_1 := 1, K = O(n)$. The set $\mathcal{X}_0$ is the domain on which we want to guarantee additivity of A with high probability. It is defined as

$$\mathcal{X}_0 := \left\{ \sum_{i=1}^{K} s_i \cdot \lambda_i \mid s_i \in \{0,1\} \text{ for } 1 \leq i \leq K \right\}^n \subset \mathbb{R}^n \,. \tag{3}$$

Note that the following inclusions hold: $\mathbb{Z}_2^n \subseteq \mathcal{X}_0, L_A(\mathbb{Z}_2^m) \subseteq \mathcal{X}_0$ and all sums of $\leq K$ many terms $\lambda \cdot z$ for $\lambda \in C_0, z \in \mathbb{Z}_2^n$ belong to $\mathcal{X}_0$. As indicated

before, in order to perform self-testing successfully we consider a function table of A's values on a much larger set. This set is defined by means of

$$\mathcal{X}_2 \;:=\; \left\{ \sum_{i=1}^{K} s_i \cdot \lambda_i \mid s_i \in \{-n^3, -n^3+1, \ldots, n^3\} \text{ for } 1 \le i \le K \right\}^n \;\subset\; \mathbb{R}^n \;. \tag{4}$$

The verification proof that is used to show linearity of A on $\mathcal{X}_0$ is given as the table of values of A on elements from $\mathcal{X}_2 \oplus \mathcal{X}_2 \oplus \mathcal{X}_2$.[4] It is important to note that the use of $\mathcal{X}_2$ 'stabilizes' the verification procedure in that it is much larger than $\mathcal{X}_0$. This results in the fact that $\mathcal{X}_2$ is almost invariant under additive shifts with elements from $\mathcal{X}_0$:

Lemma 4.1 $\forall\, x \in \mathcal{X}_0$ *it is* $\frac{|x+\mathcal{X}_2 \cap \mathcal{X}_2|}{|\mathcal{X}_2|} \ge 1 - \frac{c}{n}$ *for a constant* $c > 0$.

The verifier performs a first test

Test 1: Choose $0 < \delta_1 < 1$. For $i = 1$ to $k := \lceil \frac{2}{\delta_1} \rceil$ do

i) pick randomly elements x, y from $\mathcal{X}_2$;

ii) if $A(x+y) \neq A(x) + A(y)$ reject.

If all test pairs satisfy additivity accept A.

Proposition 4.2 *If A passes Test 1, then with high probability A is close (in a certain probabilistic sense defining $g_A^+(a)$ as the majority result among $A(a+x) - A(x), x \in \mathcal{X}_2$) to a function g_A^+ that satisfies additivity on $\mathcal{X}_0$.*

4.2 Scalar multiplicativity

Similarly as above we enlarge C_0 to a set C_1 given as

$$C_1 \;=\; \left\{ \prod_{i=1}^{K} \lambda_i^{t_i} \;,\; t_i \in \{-n, \ldots, n\} \right\} \tag{5}$$

C_1 is almost invariant with respect to scalar multiplication with a $\lambda \in C_0$:

Lemma 4.3 $\forall\, \lambda \in C_0$ *it is* $\frac{|\lambda \cdot C_1 \cap C_1|}{|C_1|} \ge 1 - \frac{1}{n}$.

The next test is

Test 2: Let $\delta_2 > 0$ be fixed. For $i = 1$ to $k := \lceil \frac{2}{\delta_2} \rceil$ do

i) pick random elements $\mu \in C_1, x \in \mathbb{Z}_2^n$;

ii) if $\dfrac{A(\mu \cdot x)}{\mu} \neq A(x)$ reject.

[4] Though there is no $\mathcal{X}_1$ used here it is in the proofs of [11]. We therefore prefer not to change notations.

If all test tuples satisfy equality accept A.

Proposition 4.4 *Let $0 < \delta_2 < \frac{1}{8}$. For n large enough it holds*

a) If A passes Test 2 with respect to δ_2 without rejection, then there exists a set $M \subseteq \mathbb{Z}_2^n$ containing a basis of $\mathbb{R}^n$ and a function g_A^* that satisfies scalar multiplicativity for scalars from C_0 and values $x \in M$ such that with high probability A is close to g_A^* (in a similar, though more complicated sense as above).

b) Suppose Tests 1 and 2 were performed without rejection for A. If A is a linear function on $\mathcal{X}_0$ (with respect to additivity) and on $C_0 \times M$ (with respect to multiplicativity), then A equals g_A^+ and g_A^* and both are the same linear function on $\mathbb{R}^n$. We denote the latter by g_A; similarly for g_B, g_C, and g_D, respectively.

The verifier performs similar tests for B, C, D (on the appropriate domains). At that point of the verification procedure we know that inconsistencies in the four function tables with respect to linearity are realized with high probability (arbitrarily close to 1) by constantly many inspections of certain function values.

4.3 Self-correction, consistency, solvability

It remains to check whether functions $A, \ldots, D$ all result from a single assignment $a \in \mathbb{R}^n$ and whether the latter is a zero. Due to the particular representation for $P(a, \tilde{r})$ we used most of the remaining steps now can be performed with small changes similarly as it is done over $\mathbb{Z}_2$.

In a third block of tests we compare pairs of functions $(A, B), (A, C)$, and (A, D) to figure out whether they result from the same assignment a. This is done by self-correcting these functions in the usual manner. For example, checking whether $g_A : \mathbb{R}^n \mapsto \mathbb{R}$ and $g_B : \mathbb{R}^{n^2} \mapsto \mathbb{R}$ result from the same a we check for randomly chosen points $x, x' \in \mathbb{Z}_2^n$ whether $g_A(x) \cdot g_A(x') = g_B(x \otimes x')$, where $x \otimes x' := (x_1 x'_1, x_1 x'_2, \ldots, x_n x'_n)$ corresponds to the appropriate assignment for the variable vector y used in the definition of B, see Lemma 3.2. Self-correction is used in order to make sure that with high probability the correct values for g_A and g_B are computed.

Definition 4.5 The random function SC-A is defined as follows: For $x \in \mathbb{Z}_2^n$ (note that $\mathbb{Z}_2^n \subset \mathcal{X}_0$) pick a random $y \in \mathcal{X}_2$ and return as result the value $A(x + y) - A(y)$. Similarly for SC-B.

Test 3 (Consistency): Let $0 < \delta_4 < 1$ be fixed. For $i = 1$ to $k := \lceil \frac{\log \delta_4}{\log \frac{7}{8}} \rceil$ do

i) pick $x, x' \in \mathbb{Z}_2^n$ randomly.

ii) Pick $y, y', y'' \in \mathcal{X}_2$ according to the uniform distribution on $\mathcal{X}_2$.

iii) If $SC\text{-}A(x) \cdot SC\text{-}A(x') \neq SC\text{-}B(x \otimes x')$ reject.

If all test points satisfy equality accept.

Proposition 4.6 *Suppose A and B pass Tests 1 and 2; suppose furthermore that the corresponding linear function $g_A : \mathbb{R}^n \mapsto \mathbb{R}$ originates from a vector $a = (a_1, \ldots, a_n) \in \mathbb{R}^n$ via $g_A(x) = \sum_{i=1}^{n} a_i \cdot x_i$ and the corresponding $g_B : \mathbb{R}^{n^2} \mapsto \mathbb{R}$ originates from a vector $b = (b_{11}, \ldots, b_{nn}) \in \mathbb{R}^{n^2}$ via $g_B(y) = \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij} \cdot y_{ij}$. If $a \otimes a \neq b$, then Test 3 rejects with a probability of at least $1 - \delta_4$.*

Finally, checking the $a \in \mathbb{R}^n$ to be a zero is done according to (2):

Test 4 (Satisfiability): For $i = 1$ to $k := \lceil \frac{\log \delta_5}{\log \frac{1}{2}} \rceil$ do

i) pick $\tilde{r} \in \mathbb{Z}_2^m$ randomly according to the uniform distribution on $\mathbb{Z}_2^m$.

ii) Evaluate $P(a, \tilde{r})$; if the result is different from 0 reject.

If $P(a, \tilde{r})$ vanishes for all test points $\tilde{r}$ accept.

Proof. (of Main Theorem 3.1) All above tests are applied to all functions $A, \ldots, D$ for appropriately chosen values of the probabilities involved. If one of the tests gives a contradiction the verifier rejects. If all tests pass without contradiction it accepts. It is clear that if the input is a solvable instance of QPS and if the guessed function tables result from a common zero, then no test will fail. The verifier accepts with probability one. Contrary, if the given QPS instance has no common root, then one of the conditions checked in one of the tests is violated with high probability. This will happen by inspecting constantly many values of the function tables involved, only. The size of these tables is exponential. □

5 Conclusions

It is clearly the most interesting future question to prove a full version of the classical PCP theorem in the BSS model, i.e. to show (or disprove) the

$$\textbf{Conjecture} : \mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1)).$$

Another interesting topic is the relation to approximation issues. Here, we think of a kind of semi-combinatorial approximation. Note that in the typical setting of combinatorial approximation the set of feasible solutions for

a problem instance (like Hamilton cycles for the TSP problem) is finite. Each of them gives a value for the objective function, and the task is to approximate the optimal solution as best as possible by polynomial time algorithms. The situation over the real is different in that we cannot require the set of feasible solutions to be finite any more. Usually, we then also have to take care about existence problems for optimal solutions. A typical optimization problem we do not consider to be appropriate in such a framework would be to compute the minimal norm solution of a polynomial system. A typical problem we do consider to be meaningful in relation with approximation matters and real PCPs is the following:

Definition 5.1 The MAX-QPS optimization problem is defined as follows: Given an instance $n, m \in \mathbb{N}, p_1, \ldots, p_m$ of the QPS decision problem, find the maximal number of polynomials among the p_i's that simultaneously can be made zero by an assignment $y \in \mathbb{R}^n$.

Here, the set of feasible solutions might be infinite, but the optimal value of the objective function does exist. Taking into account this and some other aspects (like not requiring an approximation algorithm to compute a feasible solution, but only guaranteeing the existence of a solution of a certain quality), we can define real versions of approximation classes such as $\mathrm{APX}_{\mathbb{R}}$, $\mathrm{PTAS}_{\mathbb{R}}$, $\mathrm{FPTAS}_{\mathbb{R}}$. It is then, for example, possible to show

Theorem 5.2 *Suppose the conjecture* $NP_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), O(1))$ *holds. Then there exists no fully polynomial time approximation scheme (*$\mathrm{FPTAS}_{\mathbb{R}}$*) for* MAX-QPS *in the BSS model unless* $\mathrm{P}_{\mathbb{R}} = \mathrm{NP}_{\mathbb{R}}$. □

An even more interesting problem w.r.t. the above conjecture is the Maximum-Circuit-Acceptance-Problem MAX-q-CAP. Here, as input we consider numbers $n, m \in \mathbb{N}$ and m many algebraic circuits $C_1, \ldots, C_m$. Each circuit has a constant number q of input nodes labelled by indices $i_1, \ldots, i_q \in \{1, \ldots, n\}$. There might be additional input nodes labelled by real constants. Each circuit computes as result 'accept' or 'reject'. Then the question is to compute the maximal number of circuits that simultaneously accept an input $y \in \mathbb{R}^n$ (where for $y \in \mathbb{R}^n$ a circuit takes as its q inputs the corresponding components $y_{i_1}, \ldots, y_{i_q}$).

Note that MAX-q-CAP is $\mathrm{NP}_{\mathbb{R}}$-hard for $q \geq 3$.. It then can be shown

Theorem 5.3 *Let* $q \in \mathbb{N}$ *be constant. Then* $\mathrm{NP}_{\mathbb{R}} = \mathrm{PCP}_{\mathbb{R}}(O(\log n), q)$ *iff there exists a polynomial time BSS reduction* Φ *from* QPS *to* MAX-q-CAP *and an* $\epsilon > 0$ *such that*

- *if a polynomial system* $p := (p_1, \ldots, p_m)$ *has a common zero all circuits* $\Phi(p)$ *are simultaneously satisfiable and*

- *if p has no common zero at most $\frac{1}{1+\epsilon}$ many circuits among $\Phi(p)$ can be simultaneously satisfied.*

The theorem is a real number version of a well known similar statement concerning MAX-3-SAT. However, it is unclear whether it holds as well for the 'more natural' analogue MAX-QPS.

Discussions and proofs of these results are postponed to a future paper.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy: Proof verification and hardness of approximation problems. Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, 14–23, 1992.

[2] S. Arora, S. Safra: Probabilistic checking proofs: A new characterization of NP. Journal of the ACM 45, 70–122, 1998. Preliminary version: Proc. of the 33rd Annual IEEE Symposium on the Foundations of Computer Science, 2–13, 1992.

[3] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties. Springer (1999).

[4] S. Basu, R. Pollack, M.F. Roy: Algorithms in Real Algebraic Geometry. Vol. 10 of *Algorithms and Computation in Mathematics*. Springer, 2003.

[5] L. Blum, F. Cucker, M. Shub, S. Smale: Complexity and Real Computation. Springer, 1998.

[6] L. Blum, M. Shub, S. Smale: On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. Bull. Amer. Math. Soc., vol. 21, 1–46, 1989.

[7] P. Bürgisser, M. Clausen, M.A. Shokrollahi: Algebraic Complexity Theory, volume 315 of *Grundlehren*. Springer, 1997.

[8] T. Chadzelek, G. Hotz: Analytic machines. Theoret. Comput. Sci., 219(1– 2), 151–167, 1999.

[9] F. Cucker, M. Karpinski, P. Koiran, T. Lickteig, K. Werther: On real Turing machines that toss coins. Proc. 27th STOC, 335–342, 1995.

[10] J. Makowsky, K. Meer: On the complexity of combinatorial and metafinite generating functions of graph properties in the computational model of Blum, Shub and Smale. In: Proceedings CSL 2000; Peter G. Clote and Helmut Schwichtenberg (Eds.), Springer LNCS 1862, 399–410, 2000.

[11] K. Meer: Transparent long proofs: A first PCP theorem for $\mathrm{NP}_{\mathbb{R}}$. Extended abstract to appear in: 31st International Colloquium on Automata, Languages and Programming ICALP 2004, Turku, Springer LNCS.

[12] D. Hougardy, H.J. Prömel, A. Steger: Probabilistically checkable proofs and their consequences for approximation algorithms. Discrete Mathematics 136, 175–223, 1994.

[13] S. Ivanov, M. de Rougemont: Interactive Protocols on the reals. Computational Complexity 8, 330–345, 1999.

[14] R. Rubinfeld, M. Sudan: Self-testing polynomial functions efficiently and over rational domains. Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms (Orlando, FL, 1992), ACM, 23–32, 1992.