An offprint from the

DUKE MATHEMATICAL JOURNAL

A Celebration of John F. Nash Jr.

Volume 81, Number 1

Editorial Committee for this special volume

Harold W. Kuhn

Louis Nirenberg

Peter Sarnak

Morris Weisfeld
Managing Editor

Duke University Press

Durham and London 1995

ON THE INTRACTABILITY OF HILBERT'S NULLSTELLENSATZ AND AN ALGEBRAIC VERSION OF "$NP \neq P$?"

MICHAEL SHUB AND STEVE SMALE

Section 1. Introduction. In this paper, we relate an elementary problem in number theory to the intractability of deciding whether an algebraic set defined over the complex numbers (or any algebraically closed field of characteristic zero) is empty.

More precisely, we first conjecture: The Hilbert nullstellensatz is intractable. The Hilbert nullstellensatz is formulated as a decision problem as follows.

Given $f_1, \ldots, f_\ell: \mathbf{C}^n \to \mathbf{C}$, and complex polynomials of degree d_i, $i = 1, \ldots, \ell$, decide if there is a $z \in \mathbf{C}^m$ such that $f_i(z) = 0$ for all i.

There is an algorithm for accomplishing this task. From Hilbert, the answer is no if and only if there are polynomials $g_i: \mathbf{C}^m \to \mathbf{C}$, $i = 1, \ldots, \ell$ with the property

$$(*) \qquad \sum_{i=1}^{\ell} g_i f_i = 1 .$$

Brownawell [2] has made the most decisive next step by finding a good bound on the degrees of these g_i. With that, one may decide if (*) has a solution by linear algebra, since (*) is a finite-dimensional linear system with the g_i's as unknowns. This procedure is called the "effective nullstellensatz."

To say what "intractable" means in our conjecture, it is necessary to have a formal definition of algorithm in this context. That is done in Blum-Shub-Smale [1]. In that paper, algebraic algorithms (called algorithms over $\mathbf{C}$) are described in terms of "machines," which make arithmetical computations and branch according to whether a variable (the first state variable) is zero or not.

In [1], furthermore, one has the concept of a polynomial time algorithm, in particular, a polynomial time decision algorithm over $\mathbf{C}$. This may be expressed in the present setting as

$$T(f) \leqslant s(f)^c \qquad \text{(the } c \text{ power of } s(f) \text{ all } f).$$

Here $f = (f_1, \ldots, f_\ell)$ is the input, $T(f)$ is the number of operations (arithmetic, branching) used to accomplish the decision, and $s(f)$ is the total number of coeffi-

Received 26 May 1994.

The authors were partially supported by National Science Foundation grants. The work was carried out at Instituto de Matemática Pura e Aplicada, Rio de Janeiro.

cients of the f_i (input size). Also c is a universal constant. The input size is

$$s(f) = \sum_{i=1}^{\ell} \binom{m + d_i}{d_i}.$$

Our conjecture is now formally the mathematical statement: There is no polynomial time algorithm over **C** which decides the Hilbert nullstellensatz.

An algebraic version of the computer science problem "$NP \neq P$?" is also introduced in [1]. From that paper, it follows that the nullstellensatz is a universal decision problem in a certain sense. It is "NP complete over **C**." It follows that "$NP \neq P$ over **C**" if and only if our main conjecture is true.

In other words, we may assert that the algebraic version of $NP \neq P$ is true if and only if the Hilbert nullstellensatz is intractable.

Valiant [9] also has an algebraic theory of NP completeness that differs from ours in his focus on "formula size," which is not equivalent to a computational notion. Moreover, his model is not uniform and does not permit branching on a variable $x \neq 0$.

A *computation of length* ℓ of the integer m is a sequence of integers $x_o, x_1, \ldots, x_\ell$ where $x_o = 1$, $x_\ell = m$ and given k, $1 \leqslant k \leqslant \ell$, there are i, j, $0 \leqslant i, j < k$ such that $x_k = x_i \circ x_j$ where $\circ$ is addition, subtraction, or multiplication. We define $\tau\colon Z \to \mathbf{N}$ (the natural numbers) by saying $\tau(m)$ is the minimum length of a computation of m.

The following is easy to check.

PROPOSITION. $\tau(m) \leqslant 2 \log m$.

If m is of the form 2^{2^k}, then $\tau(m) = \log\log m + 2$. The same is essentially true even if m is any power of 2. (All logs are to the base 2.)

We raised the question as to whether $\tau(m) \leqslant (\log\log m)^c$, where c is independent of m. Welington de Melo and Benar F. Svaiter [6] showed by a counting argument that the answer is no. H. Lenstra also tells us that Jeff Shallit answered our question as well. Carlos Gustavo Moreira [7] subsequently gave quite sharp estimates on this problem.

Yet our second question remains unanswered.

Problem. Is there a constant c such that

$$\tau(k!) \leqslant (\log k)^c \quad \text{all } k?$$

Given a sequence of integers a_k, we say that a_k is *easy to compute* if there is a constant c such that $\tau(a_k) \leqslant (\log k)^c$, all $k > 2$, and is hard to compute otherwise. We say that the sequence a_k is *ultimately easy to compute* if there are nonzero integers m_k such that $m_k a_k$ is easy to compute and ultimately hard to compute otherwise.

MAIN THEOREM. *If the sequence of integers $k!$ is ultimately hard to compute, then the Hilbert nullstellensatz is intractable, and consequently the algebraic version of "$NP \neq P$" is true.*

To prove the Main Theorem, we consider an intermediate decision problem which we call *twenty questions*:

$$\text{Input} \quad (k, ht(k), z) \in \mathbf{N} \times \mathbf{N} \times \mathbf{C}.$$

$$\text{Decide if} \quad z \in \{1, 2, \ldots, k\}.$$

Here $ht(k)$ is defined to be the largest natural number less than or equal to $\log k$.

THEOREM 1. *If the Hilbert nullstellensatz is tractable, i.e., if $NP = P$ over $\mathbf{C}$, then there is a machine $\mathcal{M}$ over $\mathbf{C}$ (in the sense of [1]) and a constant c such that $\mathcal{M}$ decides twenty questions in time bounded by* $(\log k)^c$.

THEOREM 2. *If a machine over $\mathbf{C}$ (in the sense of [1]) decides twenty questions in time bounded by* $(\log k)^c$ *for some constant c, then the sequence $k!$ is ultimately easy to compute.*

The Main Theorem follows immediately from Theorems 1 and 2. Theorem 1 is fairly simple in our computational setting; its proof is carried out in Section 2. Most of the substance of our paper is in the proof of Theorem 2. For this τ must be extended to polynomial rings. The algebraic and transcendental constants used by the machine must be circumvented. These arguments are carried out in Sections 3 and 4.

The complexity of deciding twenty questions was considered in a slightly different context in Shub [8]. The paper by Heintz-Morgenstern [5] is related to our work here.

Section 2. Proof of Theorem 1 (of Section 1). We prove Theorem 1 by embedding "twenty questions" in a decision problem (Y, Y_{yes}) which is in NP over $\mathbf{C}$.

Then if $NP = P$ over $\mathbf{C}$, (Y, Y_{yes}) is in P, and there is a machine $\mathcal{M}$ that decides twenty questions in time bounded by $(\log k)^c$, c a constant. Here $\mathcal{M}$ is the restriction of the machine which decides (Y, Y_{yes}) in polynomial time.

The decision problem (Y, Y_{yes}) is described as follows. $Y = \mathbf{C}^\infty$ and $Y_{\text{yes}} = \bigcup_{k \in \mathbf{N}} Y_{\text{yes},k}$ where

$$Y_{\text{yes},k} = \{(k, ht(k), z_1, \ldots, z_{ht(k)}, 0, \ldots) | z_1 \in \{1, \ldots, k\}\}.$$

The embedding of twenty questions in (Y, Y_{yes}) is simply

$$(k, ht(k), z) \to (z, ht(k), z, 1, \ldots, 1, 0, 0, \ldots),$$

where the number of ones is $ht(k) - 1$.

The proof of Theorem 1 is finished by the next proposition.

PROPOSITION. *(Y, Y_{yes}) is in NP over* **C**.

Proof. The NP machine operates on variables

$$(u_1, u_2, z_1, \ldots, z_n, w_o, \ldots, w_n, v_{jo}, \ldots, v_{jn} \text{ for } j = 1, 2, 3, 4).$$

It checks if u_2 is an integer by addition of ones. It checks if the input size (given with the input by definition) is $6u_2 + 5$. If so, $n = u_2$. It checks if $w_n = 1$, $w_i(w_i - 1) = 0$, and $v_{ji}(v_{ji} - 1) = 0$ for $i = 0, \ldots, n$ and $j = 1, 2, 3, 4$. It checks if $u_1 = \sum_{i=0}^{n} 2^i w_i$. It sets $x_j = \sum_{i=0}^{n} 2^i v_{ji}$ for $j = 1, 2, 3, 4$. Finally, it checks if $u_1 = z_1 + \sum_{j=1}^{4} x_j^2$. If so, it outputs yes. Note that if the tests are verified, the w's and v's are 0 or 1; u_1, the x_j, and hence z_1 are nonnegative integers, and $u_2 = ht(u_1)$. The time required is a constant times u_2.

Finally, we show that every element of $Y_{yes,k}$ has a positive test. Let

$$(k, ht(k), z_1, \ldots, z_{ht(k)}, 0, \ldots) \in Y_{yes,k}.$$

Then z_1 is a nonnegative integer so that $k - z$ is sum of four integers squared:

$$k - z_1 = x_1^2 + x_2^2 + x_3^2 + x_4^2. \qquad \square$$

Section 3. Easy to compute sequences in rings. In this section, we prove the facts about easy to compute sequences that are needed for the proof of Theorem 2 of the introduction. These concepts are close to those of algebraic complexity theory (see, for example, [4], [5]).

Given a ring (or field) R and generators $g_o, \ldots, g_n$ of R, a *computation of length* ℓ of the element $r \in R$ is a sequence of elements $r_{-n}, \ldots, r_o, r_1, \ldots, r_\ell$, where $r_{-i} = g_i$ for $0 \leqslant i \leqslant n$, $r_\ell = r$. Moreover, given k between 1 and ℓ (inclusive), there are p, q with $-n \leqslant p, q < k$, such that $r_k = r_p \circ r_q$, where $\circ$ is the operation of addition, subtraction, or multiplication (or division by a nonzero element if R is a field).

Define $\tau = \tau_{g_o, \ldots, g_m}: R \to \mathrm{N}$ by $\tau(r)$ is the minimum length of a computation of r. Note that the $\tau: Z \to \mathrm{N}$ of the introduction is a special case.

PROPOSITION 1. *Let $(g_o, \ldots, g_n)$ and $(h_o, \ldots, h_m)$ be two sets of generators of a ring R. Then there is a constant $c > 0$ such that*

$$\tau_{g_o, \ldots, g_n}(r) \leqslant \tau_{h_o, \ldots, h_m}(r) + c, \qquad \textit{all } r \in R.$$

The proof is straightforward.

Proposition 1 allows one to define hard and easy sequences of elements of R, independently of the choice of generators, exactly as in the introduction for Z.

PROPOSITION 2. *Let G and H be finitely generated rings (or fields). Let $\phi: G \to H$ be a ring homomorphism of G onto H. If $g_k \in G$ is an easy to compute sequence, then so is $\phi(g_k) \in H$.*

Proof. Let $e_1, \ldots, e_n$ be a set of generators of G. Then $\phi(e_1), \ldots, \phi(e_n)$ is a set of generators of H. Thus,

$$\tau_{\phi(e_1),\ldots,\phi(e_n)}(\phi(g_k)) \leqslant \tau_{e_1,\ldots,e_n}(g_k), \qquad \text{for all } k. \qquad \square$$

PROPOSITION 3. *Let R be a finitely generated integral domain and K its quotient field.*

(i) If $f_k \in K$ is an easy to compute sequence in K, then there are easy to compute sequences p_k, q_k in R such that $f_k = p_k/q_k$ for all k.

(ii) Let $f_k \in K[t_1, \ldots, t_n, \lambda_1, \ldots, \lambda_m]$ be an easy to compute sequence where $t_1, \ldots, t_n$ are variables and $\lambda_1, \ldots, \lambda_m$ are elements of an extension of K. Then there are easy to compute sequences $p_k \in R[t_1, \ldots, t_n, \lambda_1, \ldots, \lambda_n]$ $q_k \in R$ such that $f_k = p_k/q_k$ for all k.

Proof. We can assume that the generators are $g_1, \ldots, g_\ell, t_1, \ldots, t_n, \lambda_1, \ldots, \lambda_m$ where the g_i generate R. Now use the instructions for computing f_k to compute (p_k, q_k). For example,

$$(p_k, q_k) + (p_j, q_j) = (p_i q_j + p_j q_i, q_i q_j). \qquad \square$$

THEOREM 1. *Let $f_i \in \mathbf{Q}(t, \lambda_1, \ldots, \lambda_m)$ be an easy to compute sequence of nontrivial rational functions in the variable t and transcendentally independent complex numbers $\lambda_1, \ldots, \lambda_m$. Then there is an easy to compute sequence of integral polynomials $p_i \in Z[t]$ such that $p_i \neq 0$ for all i and for $z \in \mathbf{Q}$, $p_i(z) = 0$ whenever $f_i(z, \lambda_1, \ldots, \lambda_m) = 0$.*

For the proof we use two lemmas.

LEMMA 1. *Let a polynomial $f \in Z[t_o, t_1, \ldots, t_m]$ have degree d. If f is zero on every integer point in the cube in $\mathbf{R}^{m+1}$ centered at $(0, \ldots, 0)$ with side having length $(d+1)$, then f is identically zero.*

The proof is a straightforward induction on m.

LEMMA 2. *Let $f_i \in Z[t, \lambda_1, \ldots, \lambda_m]$ be an easy to compute sequence of nontrivial integral polynomials in the variable t and transcendentally independent complex numbers $\lambda_1, \ldots, \lambda_m$. Then there is an easy to compute sequence of nontrivial integral polynomials $p_i \in Z[t]$ such that for $z \in \mathbf{Q}$, $p_i(z) = 0$ whenever $f_i(z, \lambda_1, \ldots, \lambda_m) = 0$.*

For the proof, we may assume that $1, t, \lambda_1, \ldots, \lambda_m$ are the generators of $Z[t, \lambda_1, \ldots, \lambda_m]$ that we use for defining computational length.

Let n_i be the computational length of f_i. Then the degree of f_i is less than $2^{n_i} + 1$. Using Lemma 1, considering the λ_i as variables, there is an $(m+1)$-tuple of integers $(k_{io}, \ldots, k_{im}) = k$ such that $f_i(k_{io}, \ldots, k_{im}) \neq 0$ and $|k_{i,j}| \leqslant 2^{n_i} + 1$, for all i, j. By the proposition of Section 1, $\tau(k_{i,j}) \leqslant 2(n_i + 1)$. Write $f_i(t, \lambda_1, \ldots, \lambda_m) = \sum_I a_{i,I}(t)\lambda^I$ where I is a multi-index (a finite sum, of course). Since λ_j, $j = 1, \ldots, m$, are transcendentally independent, $f_i(z, \lambda_1, \ldots, \lambda_m) = 0$ for a rational number z if and only if $a_{iI}(z) = 0$ for all I.

Let $k_i' = (k_{i1}, \ldots, k_{im})$. Then if $f_i(z, \lambda_1, \ldots, \lambda_m) = 0$ for some rational z, we see that $p_i(t) = \sum_I a_{i,I}(t)(k_i')^I$ vanishes at $t = z$. Note also that $p_i(k_{i,o}) \neq 0$.

Finally, by computing k_{ij} first and substituting k_{ij}, $j = 1, \ldots, m$ in the instructions for computing f_i, p_i is computed with computational length at most $n_i + 2m(n_i + 1)$, and so p_i is an easy to compute sequence.

Now we return to the proof of Theorem 1. By Proposition 3 (i), one finds easy to compute sequences p_i, q_i' in $\mathbf{Q}[t, \lambda_1, \ldots, \lambda_m]$ such that $p_i'/q_i' = f_i$. By Proposition 3 (ii), we find easy to compute sequences $p_i'' \in Z[t, \lambda_1, \ldots, \lambda_m]$, $q_i'' \in Z$ such that $p_i' = p_i''/q_i''$. Thus p_i'' is not zero. Also $p_i''(z, \lambda_1, \ldots, \lambda_m) = 0$ if z is rational and $f_i(z, \lambda_1, \ldots, \lambda_m) = 0$. Now using p_i'' in Lemma 2 finishes the proof.

Section 4. Proof of Theorem 2 (of Section 1). The proof is preceded by two propositions.

Let a machine $\mathcal{M}$ solve a decision problem (K^∞, Y) where K is a field of characteristic 0 branching on $x = 0$ or $x \neq 0$. Suppose there is a sequence n_k of positive integers with $\mathcal{M}$ halting at time $T(k)$ on inputs of size n_k. Let $K^{n_k} \subset K^\infty$ be the n_k-fold cartesian product of K and $Y_k = Y \cap K^{n_k}$. Under the hypothesis that Y_k is a proper, nonempty subvariety of K^{n_k}, we define the kth *canonical path* as the computation path which at each branch node is taken by a Zariski-dense set of inputs in K^{n_k}.

Thus, a canonical path may be described as a certain sequence $\gamma_1\gamma_2\cdots\gamma_\ell$, $\ell < T(k)$ where each γ_j is a branch node, and γ_{j+1} is the node encountered by almost all inputs subsequent to γ_j. We omit γ_j in the case that all inputs arriving at γ_j take the same branch (see Cucker-Shub-Smale [3]). Branching is determined by a condition $x_1 = 0$ or not. Then x_1 is represented by a rational function G_j defined almost everywhere on K^{n_k}.

It is easy to see that the computational length of G_j is bounded by $c_1 j + c_2$ where c_1, c_2 are constants.

Let $H_k = \Pi G_j$.

PROPOSITION 1. *The rational function H_k defined almost everywhere on K^{n_k} vanishes on Y_k, but is not identically zero. Its computational length is bounded by $c_1 T(k) + c_2$.*

Proof. The machine must answer no on a Zariski-dense set of points of K^{n_k}, so Y_k must be contained in the union of the varieties $V_j = \{x | G_j(x) = 0\}$. This proves the first assertion.

The last assertion is a special case of the remark preceding Proposition 1.

PROPOSITION 2. *Let $\hat{\mathcal{M}}$ be a machine over a field $\hat{K}$ which is a finite algebraic extension of a field K. Then there is a machine $\mathcal{M}$ over K and a constant $c > 0$ such* *that for any decision problem (Y, Y_{yes}), $Y \subset \hat{K}^\infty$, decided by $\hat{\mathcal{M}}$, $(Y \cap K^\infty, Y_{\text{yes}} \cap K^\infty)$ is decided by $\mathcal{M}$; and, moreover, the stopping time satisfies:*

$$T_{\mathcal{M}}(y) \leqslant cT_{\hat{\mathcal{M}}}(y), \qquad y \in Y \cap K^\infty.$$

Proof. We may assume that at any computation node, the computation performed is either addition, multiplication, subtraction, or division of two elements of $\hat{K}$. We may regard $\hat{K}$ as a vector space over K of some fixed dimension q. Thus, $\hat{K}$ can be represented as K^q where the embedding $K \subset \hat{K}$ is the inclusion of K in K^q as the first coordinate. Now addition and multiplication are represented by fixed symmetric bilinear maps

$$B_+ : K^q \times K^q \to K^q$$

$$B_\times : K^q \times K^q \to K^q.$$

Division of b by a is accomplished by solving the linear system $B_\times(a, y) = b$ for y by Gaussian elimination. This requires on the order of q^3 steps. To define $\mathcal{M}$, replace $\hat{K}^\infty$ by $(K^q)^\infty$. The input of K^∞ in $\hat{K}^\infty$ is replaced by the input K^∞ as the first coordinates in $(K^q)^\infty$. Multiplication nodes are replaced by $B_\times$ and addition nodes by B_+. Subtraction nodes are replaced by -1 followed by B_+. Branching is done on the coordinates of K^q.

One can see using the isomorphism between K^q and $\hat{K}$ that on inputs $y \in Y \cap K^\infty$, $\mathcal{M}$ gives the same answers as $\hat{\mathcal{M}}$ with the desired time bound where c is on the order q^3. □

Proof of Theorem 2. Assume that $\hat{\mathcal{M}}$ decides twenty questions in time bounded by $(\log k)^c$. Let $\mu_1, \dots, \mu_\ell$ be the nonrational constants of $\hat{\mathcal{M}}$ so that we may view $\hat{\mathcal{M}}$ as a machine over $\mathbf{Q}(\mu_1, \dots, \mu_\ell)$. Now $\mathbf{Q}(\mu_1, \dots, \mu_\ell)$ is a finite algebraic extension of a finitely generated purely transcendental extension $\mathbf{Q}(\lambda_1, \dots, \lambda_m)$ of $\mathbf{Q}$. Thus, by Proposition 2, there is a machine $\mathcal{M}$ over $\mathbf{Q}(\lambda_1, \dots, \lambda_m)$ which solves twenty questions restricted to $\mathbf{Q}(\lambda_1, \dots, \lambda_m)$. Thus, on the input of $(k, ht(k), z)$, $z \in \mathbf{Q}(\lambda_1, \dots, \lambda_m)$, $\mathcal{M}$ decides if $z \in \{1, \dots, k\}$ in time bounded by $c_1(\log k)^c$.

Then, by Proposition 1, there are nontrivial rational functions $f_k \in \mathbf{Q}(t, \lambda_1, \dots, \lambda_m)$ which vanish on $\{1, \dots, k\}$ and whose computational length in $\mathbf{Q}(t, \lambda_1, \dots, \lambda_m)$ is bounded by $c_2(\log k)^c + c_3$. Here c_3 is a bound for the computational length of rational constants introduced by the machine $\hat{\mathcal{M}}$, and c_3 depends only on $\hat{\mathcal{M}}$. Therefore, f_k is an easy to compute sequence of nontrivial rational functions. By Theorem 1 of Section 3, there is an easy to compute sequence of nontrivial polynomials $p_k \in \mathbf{Z}[t]$ vanishing on $\{1, \dots, k\}$. By Lemma 1 of Section 3, there is an integer m such that $p_k(m) \neq 0$, $|m| \leqslant 2^\ell + 1$ where ℓ is the computational length of p_k. So $\tau(m) \leqslant \ell + 1$. We may assume $|m|$ is minimal with these properties. Then p_k is zero at each integer between zero and m. Evaluating p_k at m gives a computational length of at most $2\ell + 1$ for $p_k(m)$. Since $p_k(m)$ is an integral multiple of $k!$, the sequence $k!$ is ultimately easy to compute. □

References

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

[2] W. D. Brownawell, *Bounds for the degrees in the Nullstellensatz*, Ann. of Math. (2) **126** (1987), 577–591.

[3] F. Cucker, M. Shub, and S. Smale, *Separation of complexity classes in Koiran's weak model*, to appear in Theoret. Comput. Sci.

[4] J. Heintz, "On the computational complexity of polynomials and bilinear mappings: A survey," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, Lecture Notes in Comput. Sci., **356**, Springer-Verlag, Berlin, 1989, 269–300.

[5] J. Heintz and J. Morgenstern, *On the intrinsic complexity of elimination theory*, J. Complexity **9** (1993), 471–498.

[6] W. de Melo and B. F. Svaiter, *The cost of computing integers*, to appear in Proc. Amer. Math. Soc.

[7] C. G. Moreira, *On asymptotic estimates for arithmetic cost functions*, to appear.

[8] M. Shub, "Some remarks on Bezout's Theorem and complexity theory" in *From Topology to Computation: Proceedings of the Smalefast*, ed. by M. W. Hirsch, J. E. Marsden, and M. Shub, Springer-Verlag, New York, 1993, 443–455.

[9] L. G. Valiant, "Completeness classes in algebra," Conference Record of the Eleventh Annual ACM Symposium on Theory of Computing, ACM, New York, 1979, 249–261.

Shub: T. J. Watson Research Center, IBM Research, Yorktown Heights, New York 10598, USA; shub@watson.ibm.com

Smale: Department of Mathematics, City University of Hong Kong, Tat Chee Ave., Kowloon, Hong Kong; masmale@cityu.edu.hk