Reprinted from JOURNAL OF COMPUTER AND SYSTEM SCIENCES
All Rights Reserved by Academic Press, New York and London

Vol. 32, No. 2, April 1986
Printed in Belgium

The Complexity of Elementary Algebra and Geometry

MICHAEL BEN-OR

The Hebrew University, Jerusalem, Israel

DEXTER KOZEN

IBM Research, Yorktown Heights, New York

AND

JOHN REIF

Harvard University, Cambridge, Massachusetts

Received February 7, 1985, revised June 11, 1985

IN MEMORY OF ALFRED TARSKI, 1902–1983

The theory of real closed fields can be decided in exponential space or parallel exponential time. In fixed dimension, the theory can be decided in *NC*. © 1986 Academic Press, Inc.

1. INTRODUCTION

In his 1948 paper [14], Tarski gave a decision procedure for the first-order theory of the real numbers with $+$, $\cdot$, and $=$, commonly known as the theory of real closed fields. His decision procedure was nonelementary (in the complexity-theoretic sense). An elementary decision procedure was given by L. Monk [10], and in 1974, double-exponential-time decision procedures were given by Collins [5] and independently by Monk and Solovay [11]. Various improvements and heuristics notwithstanding, that worst-case bound has stood since that time.

In this paper we give a new algorithm that can be implemented in exponential space or in parallel exponential time. We also conjecture that the problem is complete in exponential space.

The main lemma is of independent interest: an *NC* (i.e., $(\log n)^{O(1)}$ depth and polynomial size) circuit to determine whether a given set of rational, univariate polynomial equations and inequalities has a real solution. It was known how to solve this problem sequentially in polynomial time [5]. The multivariate problem is also in *NC*, provided the number of variables is fixed; but the depth of the circuit grows exponentially with the number of variables.

On the practical side, we do not expect to be able to implement our algorithms

on parallel machines or in VLSI, given the current state of technology. However, some of the techniques we develop may help to simplify the sequential algorithms in current use, notably the cylindric algebraic decomposition method of Collins [5]. Collins' method uses numerical approximation techniques that appear inherently sequential; by contrast, our method is completely algebraic. We make use of recent advances in parallel algorithms for matrix and polynomial algebra [6, 2, 15], and develop and improved polynomial decomposition algorithm [15].

Our parallel algorithm for testing consistency of polynomial equations and inequalities employs a generalized Sturm sequence method, the basic idea of which appears in Tarski's paper [14]. Considerations of efficiency require that the idea be developed considerably further; in its final abstract form, it becomes an elegant tensor identity (Sect. 2.3).

The remainder of the paper is organized as follows. In Section 2 we describe how to test the consistency of univariate polynomial constraints: given a system Σ of constraints of the form $p(x) \leq 0$, $p(x) = 0$, or $p(x) > 0$, does the system have a real solution x? This algorithm is the main building block which will allow us to eliminate one variable in the quantifier elimination procedure. This section is divided into three subsections, as follows. Section 2.1 describes a simplifying precomputation which will map a given system of polynomial equations and inequalities into an equivalent system in which the polynomials are square-free and pairwise relatively prime. Section 2.2 describes Sturm's theorem and a generalization due to Tarski. These results will allow us to derive a linear relationship between the consistent sign assignments to two polynomials $p, q \in \Sigma$ and certain Sturm computations involving polynomials in the Euclidean remainder sequences of p and q. Section 2.3 extends the results of Section 2.2 to a tensor identity, which will allow solutions for two separate systems Σ_1 and Σ_2 to be combined into a solution for $\Sigma_1 \cup \Sigma_2$.

In Section 3, the algorithm of Section 2 is extended inductively to handle systems of multivariate polynomial constraints. The algorithm can be implemented in *NC* for fixed number of variables, and exponential *NC* or sequential exponential space if the number of variables is allowed to grow as the size of the input. In Section 4, we describe how to eliminate quantifiers in the theory of real closed fields using the circuit of Section 3.

2. Testing Consistency of Univariate Polynomial Constraints

Suppose we are given a finite set Σ of rational univariate polynomials $p(x)$, and a *sign assignment* $\sigma: \Sigma \to \{-1, 0, 1\}$ representing the system of equations and strict inequalities

$$\begin{aligned} p(x) &< 0 && \text{if } \sigma(p) = -1, \\ &= 0 && \text{if } \sigma(p) = 0, \\ &> 0, && \text{if } \sigma(p) = 1. \end{aligned}$$

The system is said to be *consistent* if it has a real solution. We write $\|\Sigma\| \leqslant n$ if there are at most n poynomials in Σ, each $p \in \Sigma$ is of degree at most n, and all coefficients of $p \in \Sigma$ can be represented with at most n bits. We will give an algorithm in *NC* (i.e., a uniform family of circuits of depth $(\log n)^{O(1)}$ and size $n^{O(1)}$) to determine whether a given system (Σ, σ) with $\|\Sigma\| \leqslant n$ is consistent. In general, we know of no *NC* circuit that *finds* a real solution of (Σ, σ), even if it is known that one exists; however, we can test consistency and even determine the cardinality of the set of solutions.

For the real closed field algorithm, we will need more: for a given Σ, $\|\Sigma\| \leqslant n$, we will need to produce list of all consistent sign assignments. Although there are exponentially many possible sign assignments, at most $2n^2+1$ of them are consistent. This is because there are at most n^2 roots in all, and the signs of the polynomials are constant in intervals between roots. We show how to produce such a list in *NC* or in space $(\log n)^{O(1)}$.

(why?)

2.1. *Simple Refinement*

A polynomial is *simple* if it is square-free, i.e., if it has only simple roots. A set of polynomials Σ is called *simple* if the elements of Σ are simple and pairwise relatively prime. Given Σ with $\|\Sigma\| \leqslant n$, we show how to produce in *NC* a *simple refinement*, that is, a new set of polynomials Γ such that Γ is simple, $\|\Gamma\| \leqslant O(n^2 \log n)$, and each $p \in \Sigma$ is a product of powers of elements of Γ. A sign assignment for Γ will uniquely determine a sign assignment for Σ. This precomputation step is not essential, but is taken only to simplify the presentation later on.

The straightforward divide-and-conquer approach using iterated gcd computations yields nonpolynomial growth in degree and coefficient size, so care must be taken.

A single polynomial can be refined using the squarefree decomposition algorithm of [15]. Thus we can assume without loss of generality that all $p \in \Sigma$ are simple. Let $\theta(p)$ denote the set of roots of p. Note

$$\begin{aligned} \bigcap_{p \in I} \theta(p) &= \theta(\gcd(I)) \\ \bigcup_{p \in I} \theta(p) &= \theta(\operatorname{lcm}(I)) \\ \neg\theta(p) &= \theta(\operatorname{lcm}(\Sigma)/p) \end{aligned} \tag{1}$$

where $I \subseteq \Sigma$ and $\neg$ denotes complementation in $\theta(\operatorname{lcm}(\Sigma))$, the set of all roots of all $p \in \Sigma$. Since $\|\Sigma\| \leqslant n$,

$$|\theta(\operatorname{lcm}(\Sigma))| \leqslant n^2.$$

The $\theta(p)$, $p \in \Sigma$, are generators of a Boolean algebra on $\theta(\operatorname{lcm}(\Sigma))$ with Boolean operations given by (1). The simple refinement we seek consists of the atoms of this Boolean algebra, i.e., all nonempty sets of the form

$$\bigcap_{p \in I} \theta(p) \cap \bigcap_{p \in I'} \neg\theta(p)$$

where $I \cup I' = \Sigma$ and $I \cap I' = \varnothing$. By (1) these atoms are represented by the polynomials

$$p(I, I') = \gcd(I)/\gcd(\gcd(I), \operatorname{lcm}(I'))$$

of nonzero degree. These will be determined in $\log n$ stages. At some intermediate stage, suppose we have all the atoms of the Boolean algebra generated by $\Sigma_1 \subseteq \Sigma$, given by a list of pairs (I, I') where $I \cup I' = \Sigma_1$ and $I \cap I' = \varnothing$, and all atoms of $\Sigma_2 \subseteq \Sigma$ presented in a similar way. Each atom of $\Sigma_1 \cup \Sigma_2$ is an intersection of an atom (I, I') of Σ_1 and an atom (J, J') of Σ_2. This intersection is represented by $p(I \cup J, I' \cup J')$. Computing all such polynomials and discarding those of degree 0, we are left with a list of pairs (K, K') representing all atoms of $\Sigma_1 \cup \Sigma_2$. After $\log n$ stages we have built a list of all atoms for Σ. The corresponding polynomials provide a simple refinement of Σ. The entire computation can be done in *NC* using the multiple-polynomial gcd algorithm of [15].

If Γ is a simple refinement of Σ, then any consistent sign assignment for Γ gives a consistent sign assignment for Σ. Any consistent sign assignment for Γ assigns at most one 0, since the elements of Γ are relatively prime. We wish to refine Γ further to get Δ such that any consistent sign assignment for Σ is obtained from a consistent sign assignment for Δ in which *exactly* one 0 is assigned. This is done by appending the polynomial

$$r = \operatorname{lcm}(\Sigma)'(x - b)(x + b)$$

to Σ before refining, where

$$b = 1 + \max_{0 \leqslant i \leqslant k-1} |a_i/a_k|,$$

the a_i are the coefficients of $\operatorname{lcm}(\Sigma)$, and a_k is the leading coefficient. All roots of $\operatorname{lcm}(\Sigma)$ lie in the interval $(-b, b)$ [8], and r has a root in every interval between any two roots of $\operatorname{lcm}(\Sigma)$. Thus without loss of generality we can limit our attention to sign assignments that assign exactly one 0.

2.2. *A Generalization of Sturm's Theorem*

We have reduced the problem to the following: given p and Σ, $\{p\} \cup \Sigma$ simple, $\|\{p\} \cup \Sigma\| \leqslant n$, list all consistent sign assignments assigning 0 to p and -1 or 1 to the elements of Σ.

If $\Sigma = \varnothing$, the problem is merely to determine whether p has any real roots. This can be solved using Sturm sequences (see [13]). This technique uses the coefficients of the polynomial remainder sequence for p and p', which can be obtained as subresultants, or determinants of submatrices of the Sylvester matrix of coefficients of p and p', in *NC* [4].

Sturm sequences work as follows. Let p_0, p_1 be square-free, relatively prime polynomials in one variable, and let $a, b \in \mathbb{R}$, $a < b$, such that neither a nor b is a root of p_0 or p_1. Consider the Euclidean remainder sequence

$$\begin{array}{l} p_0 \\ p_1 \\ \dots \\ p_{k+1} = q_k p_k - p_{k-1} \\ \dots \\ p_n \end{array}$$

where p_{k+1} is the negative of the remainder obtained by dividing p_{k-1} by p_k. p_n is a constant nonzero polynomial, since by assumption p_0, p_1 are relativly prime. Count the number of sign changes in $p_0(a),..., p_n(a)$, count the number of sign changes in $p_0(b),..., p_n(b)$, and subtract. Denote the result of this computation by $S(p_0, p_1, a, b)$. Sturm's theorem states that $S(p, p', a, b)$ is the number of real roots of p in the interval (a, b).

For sufficiently small a and sufficiently large b, the value of $S(p_0, p_1, a, b)$ is independent of the choice of a and b. We denote this value by $S(p_0, p_1)$. By Sturm's theorem, $S(p, p')$ is the number of real roots of p. $S(p_0, p_1)$ can be computed efficiently as follows: if $l(q)$ is the leading term of q, then evaluating $l(q)$ at -1 gives the same sign as evaluating q at any number smaller than all the real roots of q; similarly, evaluating $l(q)$ at 1 gives the same sign as evaluating q at any number larger than all the real roots of q. Thus if we substract the number of sign changes in the sequence $l(p_0)(1),..., l(p_n)(1)$ from the number of sign changes in $l(p_0)(-1),..., l(p_n)(-1)$, the result is $S(p_0, p_1)$. Define

$$c_q = \{x \mid p(x) = 0 \text{ and } q(x) > 0\}$$
$$\bar{c}_q = \{x \mid p(x) = 0 \text{ and } q(x) < 0\}$$

where q is simple and relatively prime to p. Sturm's theorem then says that

$$S(p, p') = |c_q| + |\bar{c}_q|.$$

The following generalization is implicit in Tarski's paper [14].

Lemma (Tarski). $S(p, p'q) = |c_q| - |\bar{c}_q|$.

Proof. Let $p_0,..., p_n$ be the Euclidean remainder sequence as defined above with $p_0 = p$ and $p_1 = p'q$. Consider a point t moving from $-\infty$ to $+\infty$. Let $S(t)$ denote the number of changes of sign in the sequence $p_0(t),..., p_n(t)$. $S(t)$ is constant between roots of the p_i, so $S(t)$ can change only when t skips over a root of some

p_i. The assumption that p is simple implies that p_n is constant and nonzero. At a root of p_i, $0<i<n$, $p_{i+1}(t)=-p_{i-1}(t)\neq 0$, so the net change in $S(t)$ as t skips over that root is 0. Thus the only roots that can change the value of $S(t)$ are roots of p. There are four cases, depending on the signs of p' and q at a root of p. If $p'>0$ and $q>0$, then $p'q>0$ and $S(t)$ decreases by 1 as we jump over that root. If $p'>0$ and $q<0$, then $p'q<0$ and $S(t)$ increases by 1. If $p'<0$ and $q>0$, then $p'q<0$ and $S(t)$ decreases by 1. If $p'<0$ and $q<0$, then $p'q>0$ and $S(t)$ increases by 1. Thus $S(t)$ decreases by 1 whenever $q>0$, and increases by 1 whenever $q<0$, so the net gain or loss going from $-\infty$ to $+\infty$ is as stated in the lemma. ▌

2.3. *A Tensor Identity*

If $\Sigma=\{q\}$, the values of $S(p,p')$ and $S(p,p'q)$ determine the values of $|c_q|$ and $|\bar{c}_q|$ by solving a simple linear system of order 2:

$$\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}|c_q|\\ |\bar{c}_q|\end{bmatrix}=\begin{bmatrix}S(p,p')\\ S(p,p'q)\end{bmatrix}.$$

The 2×2 matrix is denoted A_1. The value of $|c_q|$ (resp. $|\bar{c}_q|$) determines whether the sign assignment $p=0$, $q>0$ (resp. $p=0$, $q<0$) is consistent. For $\Sigma=\{q_1,q_2\}$, there are four linear equations in four unknowns:

$$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}\begin{bmatrix}|c_1\cap c_2|\\ |\bar{c}_1\cap c_2|\\ |c_1\cap \bar{c}_2|\\ |\bar{c}_1\cap\bar{c}_2|\end{bmatrix}=\begin{bmatrix}S(p,p')\\ S(p,p'q_1)\\ S(p,p'q_2)\\ S(p,p'q_1q_2)\end{bmatrix}.$$

Here c_i and $\bar{c}_i$ abbreviate c_{q_i} and $\bar{c}_{q_i}$, respectively. The 4×4 matrix A_2 is the Kronecker (tensor) product of A_1 with itself:

$$\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{bmatrix}=\begin{bmatrix}A_1 & A_1\\ A_1 & -A_1\end{bmatrix}=A_1\otimes A_1.$$

A_2 is nonsingular, since Kronecker products of nonsingular matrices are again nonsingular. After making the four Sturm queries on the right, the system can be solved for the four unknowns. Each unknown corresponds to a sign assignment to q_1 and q_2; the unknown is nonzero exactly when the corresponding sign assignment is consistent.

In general, for $\Sigma=\{q_1,\dots,q_n\}$, we get a $2^n\times 2^n$ linear system $A_nc=s$, where c is a vector consisting of all elements of the form

$$|d_1\cap\cdots\cap d_n|,\qquad d_i\in\{c_i,\bar{c}_i\},\qquad 1\leqslant i\leqslant n,$$

and s is a vector of all elements of the form

$$S\left(p, p' \prod I\right)$$

where $\prod I = \prod_{q \in I} q$, $I \subseteq \Sigma$. A_n is a $2^n \times 2^n$ Hadamard matrix, obtained by taking the Kronecker product of n copies of A_1. A_n is nonsingular (in fact, $A_n^{-1} = 2^{-n} A_n$), so we could in principle make all the Sturm queries s and solve the system for c. The nonzero elements of c correspond to the consistent sign assingments. Unfortunately, the system is far too big for this computation to be done in NC.

We now make the key observation that since $\|\{p, q_1, ..., q_n\}\| \leqslant n$, p is of degree at most n, therefore all but at most n of the elements of c are 0. Thus $A_n c = s$ is equivalent to a much smaller system, obtained as follows: drop out the zero elements of c and the corresponding columns of A_n to obtain a rectangular system of order at most $2^n \times n$. The resulting matrix is of full rank, therefore a basis can be found among its rows. Drop out all rows of the matrix not contained in this basis, and the corresponding elements of s. We are left with a nonsingular square system of order at most n, in which the vector c contains all nonzero elements of the form

$$|d_1 \cap \cdots \cap d_n|, \qquad d_i \in \{c_i, \bar{c}_i\}, \qquad 1 \leqslant i \leqslant n.$$

We described the smaller system by constructing the $2^n \times 2^n$ system first and then reducing it. This cannot be done in NC. However, we show below that the smaller system can be constructed without constructing all of A_n first. The computation will proceed in stages; at each stage, two $n \times n$ solutions for subsets $\Gamma, \Gamma' \subseteq \Sigma$ will be combined via a tensor product construction, given in detail below, to yield an $n^2 \times n^2$ solution for $\Gamma \cup \Gamma'$. This $n^2 \times n^2$ solution will then be reduced to an $n \times n$ solution as above (delete the zero elements of c and the corresponding columns of A; find a basis among the rows of the resulting matrix, and delete the other rows and the corresponding elements of s).

The tensor product construction proceeds as follows. Suppose $\Gamma, \Gamma' \subseteq \Sigma$, c and c' are vectors of length m and n, respectively, such that c contains all nonzero elements of the form

$$\left|\bigcap_{q \in I} c_q \cap \bigcap_{q \in \Gamma - I} \bar{c}_q\right|, \qquad I \subseteq \Gamma$$

and c' contains all nonzero elements of the form

$$\left|\bigcap_{q \in J} c_q \cap \bigcap_{q \in \Gamma' - J} \bar{c}_q\right|, \qquad J \subseteq \Gamma',$$

s and s' are vectors of length m and n, respectively, containing elements of the form

$$S\left(p, p' \prod I\right), \qquad I \subseteq \Gamma,$$

$$S\left(p, p' \prod J\right), \qquad J \subseteq \Gamma',$$

respectively, and A and A' are nonsingular square matrices of order m and n, respectively, such that

$$Ac = s, \qquad A'c' = s'.$$

Moreover, assume that these equations hold independently of the sizes of the nonempty sets of the form

$$\bigcap_{q \in I} c_q \cap \bigcap_{q \in \Sigma - I} \bar{c}_q, \qquad I \subseteq \Sigma$$

(this assumption will be explained more fully below). Let $A \otimes A'$ denote the Kronecker product of A and A', obtained by replacing each entry a_{ij} of A by the matrix $a_{ij}A'$. $A \otimes A'$ is nonsingular since A and A' are. We index the entries of $A \otimes A'$ by four indices i, j, k, l, where i, j give the position of the block $a_{ij}A'$ and k, l give the position of the entry $a_{ij}a'_{kl}$ within the block.

The vectors c and c' are combined into a new column vector cc' consisting of m column vectors placed end-to-end, each one of length n. The entries of cc' are indexed by two indices i, j, where i gives the position of the block and j gives the position of the entry within the block. The i, jth entry of cc' is $|e \cap e'|$, where $|e|$ is the ith entry of c and $|e'|$ is the jth entry of c'.

The vectors s and s' are combined into a new column vector ss' consisting of m column vectors placed end-to-end, each one of length n. The entries of ss' are indexed by i, j as above. The i, jth entry of ss' is

$$S\left(p, p' \prod (I \Delta J)\right)$$

where $S(p, p' \prod I)$ is the ith entry of s, $S(p, p' \prod J)$ is the jth entry of s', and Δ denotes exclusive-or of sets.

The following equation gives the relationship between these constructs.

LEMMA. $(A \otimes A')(cc') = ss'$. *Moreover, this equation holds independently of the sizes of the nonempty sets of the form*

$$\bigcap_{q \in I} c_q \cap \bigcap_{q \in \Sigma - I} \bar{c}_q, \qquad I \subseteq \Sigma.$$

Proof. In order to explain the use of the tensor product, we must first reformulate the lemma in terms of linear algebra. Let B be the Boolean algebra of subsets of roots of p generated by the sets $c_q, q \in \Sigma$, with the usual set-theoretic

Boolean operations. B extends to a real vector space $\hat{B}$, namely the free vector space over $\mathbb{R}$ generated by the atoms (minimal nonzero elements) of B. If B has k atoms, then B is isomorphic to $\{0, 1\}^k$ with componentwise Boolean operations, and $\hat{B}$ is isomorphic to $\mathbb{R}^k$. We may then regard B as a subset of $\hat{B}$ via the inclusion $\{0, 1\} \subseteq \mathbb{R}$. The atoms of B correspond to bit strings $(0, 0,..., 0, 1, 0,..., 0)$, and these also provide a basis for $\hat{B}$. Define a multiplication on $\hat{B}$ componentwise:

$$(a_1,..., a_k) \cdot (b_1,..., b_k) = (a_1 b_1,..., a_k b_k).$$

Under this operation, $\hat{B}$ becomes a commutative algebra with identity $(1, 1,..., 1)$, which we denote by 1. Restricted to B, multiplication is just intersection. In fact, all the Boolean operations of B can be expressed using the arithmetic operations of $\hat{B}$:

$$a \cap b = a \cdot b$$
$$a \cup b = a + b - a \cdot b$$
$$\bar{a} = 1 - a.$$

The identity element 1 of $\hat{B}$ is the top element of B, and the zero element 0 of $\hat{B}$ is the bottom element of B.

The size function $|\cdot|$ is a finitely additive, real-valued function on B, and as such extends uniquely to a linear functional

$$\mu\colon \hat{B} \to \mathbb{R}.$$

Thus μ is an element of the dual space $\hat{B}^*$ of $\hat{B}$.

We now describe the elements of c and s in terms of $\hat{B}$ and μ. For $I, J \subseteq \Sigma$, define $\prod I = \prod_{q \in I} q$ and define

$$\alpha_{I,J} = \prod_{q \in I} c_q \cdot \prod_{q \in J} (1 - c_q),$$
$$\sigma_I = 2c_{\prod I} - 1.$$

The elements of c are all of the form

$$\left| \bigcap_{q \in I} c_q \cap \bigcap_{q \in \Gamma - I} \bar{c}_q \right| = \mu\left(\prod_{q \in I} c_q \cdot \prod_{q \in \Gamma - I} (1 - c_q) \right)$$
$$= \mu(\alpha_{I, \Gamma - I}), \qquad I \subseteq \Gamma$$

and, by Tarski's lemma, those of s are all of the form

$$S\left(p, p' \prod I \right) = |c_{\prod I}| - |\bar{c}_{\prod I}|$$
$$= \mu(c_{\prod I}) - \mu(1 - c_{\prod I})$$
$$= \mu(\sigma_I), \qquad I \subseteq \Gamma.$$

Similarly, the elements of c' and s' are of the form $\mu(\alpha_{J,\Gamma'-J})$, $J \subseteq \Gamma'$, and $\mu(\sigma_J)$, $J \subseteq \Gamma'$, respectively.

Let

$$c = \langle \mu(\alpha_i) \mid 1 \leqslant i \leqslant m \rangle, \qquad s = \langle \mu(\sigma_i) \mid 1 \leqslant i \leqslant m \rangle,$$
$$c' = \langle \mu(\alpha'_j) \mid 1 \leqslant j \leqslant n \rangle, \qquad s' = \langle \mu(\sigma'_i) \mid 1 \leqslant j \leqslant n \rangle.$$

We claim that the i, jth element of cc' is $\mu(\alpha_i \cdot \alpha'_j)$ and the i, jth element of ss' is $\mu(\sigma_i \cdot \sigma'_j)$. By construction, if $\alpha_i = \alpha_{I,\Gamma-I}$, $I \subseteq \Gamma$, and $\alpha'_j = \alpha_{J,\Gamma'-J}$, $J \subseteq \Gamma'$, then the i, jth element of cc' is

$$\mu(\alpha_{I \cup J, (\Gamma - I) \cup (\Gamma' - J)}).$$

Also by construction, if $\sigma_i = \sigma_I$, $I \subseteq \Gamma$, and $\sigma'_j = \sigma_J$, $J \subseteq \Gamma'$, then the i, jth element of ss' is

$$\mu(\sigma_{I \Delta J}).$$

Thus in order to prove the claim it suffices to prove the two equations

$$\alpha_{I \cup J, K \cup L} = \alpha_{I,K} \cdot \alpha_{J,L},$$
$$\sigma_{I \Delta J} = \sigma_I \cdot \sigma_J$$

in $\hat{B}$. The first is immediate from the definition of $\alpha_{I,J}$. For the second, we use the fact that

$$\begin{aligned} c_{qr} &= (c_q \cap c_r) \cup (\bar{c}_q \cap \bar{c}_r) \\ &= c_q c_r + (1 - c_q)(1 - c_r) - c_q(1 - c_q)\, c_r(1 - c_r) \\ &= 2c_q c_r - c_q - c_r + 1 \end{aligned}$$

to get

$$2c_{qr} - 1 = (2c_q - 1)(2c_r - 1),$$
$$2c_{q^2} - 1 = (2c_q - 1)^2 = 1.$$

It follows that

$$\begin{aligned} \sigma_I \cdot \sigma_J &= (2c_{\prod I} - 1)(2c_{\prod J} - 1) \\ &= 2c_{\prod I \cdot \prod J} - 1 \\ &= (2c_{\prod (I \Delta J)} - 1)(2c_{\prod (I \cap J)} - 1)^2 \\ &= \sigma_{I \Delta J}. \end{aligned}$$

This establishes the claim.

We are now ready to restate the lemma. Recall the assumption that $Ac = s$ and $A'c' = s'$ hold independently of the sizes of the nonempty atoms of B. This says that the equations

$$A(\langle \mu(\alpha_i)\rangle) = \langle \mu(\sigma_i)\rangle$$
$$A'(\langle \mu(\alpha'_j)\rangle) = \langle \mu(\sigma'_j)\rangle$$

hold independently of the choice of (integral-valued) μ. Among such μ, a basis for $\hat{B}^*$ can certainly be found. This implies that the above equations hold for all $\mu \in \hat{B}^*$. The lemma is then equivalent to:

If $A(\langle \mu(\alpha_i)\rangle) = \langle \mu(\sigma_i)\rangle$ and $A'(\langle \mu(\alpha'_j)\rangle) = \langle \mu(\sigma'_j)\rangle$ for all $\mu \in \hat{B}^*$, then $(A \otimes A')(\langle \mu(\alpha_i \cdot \alpha'_j)\rangle) = \langle \mu(\sigma_i \cdot \sigma'_j)\rangle$ for all $\mu \in \hat{B}^*$. (2)

Writing $A(\langle \alpha_i \rangle) = \langle \sigma_i \rangle$, $1 \leqslant i \leqslant m$, to denote the m equations

$$\sum_{j=1}^{m} A_{ij}\alpha_j = \sigma_i, \qquad 1 \leqslant i \leqslant m$$

in $\hat{B}$, it is clear that $A(\langle \mu(\alpha_i)\rangle) = \langle \mu(\sigma_i)\rangle$ for all $\mu \in \hat{B}^*$ if and only if $A(\langle \alpha_i \rangle) = \langle \sigma_i \rangle$. Thus (2) is equivalent to

If $A(\langle \alpha_i \rangle) = \langle \sigma_i \rangle$ and $A'(\langle \alpha'_j \rangle) = \langle \sigma'_j \rangle$, then $(A \otimes A')(\langle \alpha_i \cdot \alpha'_j \rangle) = \langle \sigma_i \cdot \sigma'_j \rangle$. (3)

A simple calculation verifies (3): the ijth element of $(A \otimes A')(\langle \alpha_i \cdot \alpha'_j \rangle)$ is

$$\sum_{k=1}^{m} \sum_{l=1}^{n} A_{ik} A'_{jl}(\alpha_k \cdot \alpha'_l) = \left(\sum_{k=1}^{m} A_{ik}\alpha_k\right) \cdot \left(\sum_{l=1}^{n} A'_{jk}\alpha'_l\right) = \sigma_i \cdot \sigma'_j.$$

This completes the proof. ▮

Let $\Sigma = \{q_1, ..., q_n\}$ and let $\|\{p\} \cup \Sigma\| \leqslant n$. We use the above lemma to construct, in $\log n$ stages, an $n \times n$ system $Ac = s$ such that c gives all nonzero sets of the form

$$|d_1 \cap \cdots \cap d_n|, \qquad d_i \in \{c_i, \bar{c}_i\}, \qquad 1 \leqslant i \leqslant n,$$

or equivalently, all consistent sign assignments to the $q \in \Sigma$ at roots of p. In the first stage, we solve the n problems $\{p, q_1\}, ..., \{p, q_n\}$ in parallel. Then we combine adjacent solutions using the tensor product construction of the previous lemma in parallel to get solutions for the $n/2$ problems $\{p, q_1, q_2\}, ..., \{p, q_{n-1}, q_n\}$. We continue in this fashion, combining adjacent solutions in parallel; every stage doubles the number of q_i in each solution. If at any time the order of the system exceeds n, we reduce it to an equivalent order n system as described above. After $\log n$ stages we have an $n \times n$ solution for the entire set $\{p, q_1, ..., q_n\}$. At no time does the order of any intermediate system exceed n^2. Each stage requires the solution of a nonsingular system of order at most n^2, as well as the computation of a basis; all these computations can be done in NC [6, 15, 2].

3. Testing Consistency of Multivariate Polynomial Constraints

The construction of Section 2 produced a family of circuits, denoted generically by C, to list all consistent sign assignments of a set Σ of polynomials, $\|\Sigma\| \leqslant n$. It is crucial to observe that C did not need to know the actual values of the coefficients of $p \in \Sigma$, but only the *signs* of certain polynomials in the coefficients of p. These polynomials arose in the Sturm computations, gcd computations, subresultants, etc.

This observation allows us to construct circuits to handle multivariate polynomials. Given a set of polynomials $\Sigma[x_1,..., x_k]$ in $\mathbb{Q}[x_1,..., x_k]$, we write them as polynomials in x_k with coefficients in the polynomial ring $\mathbb{Q}[x_1,..., x_{k-1}]$. In order to list the consistent sign assignments of $\Sigma[x_1,..., x_k]$, we need only know the signs of certain polynomials $\Sigma[x_1,..., x_{k-1}]$ in $\mathbb{Q}[x_1,..., x_{k-1}]$. A rough complexity analysis of the construction of Section 2 reveals that $\longleftarrow$ why?

$$\|\Sigma[x_1,..., x_{k-1}]\| \leqslant O(\|\Sigma[x_1,..., x_k]\|^3).$$

Suppose we have built a circuit C_{k-1} to list all consistent sign assignments of any input set of polynomials Σ over $\mathbb{Q}[x_1,..., x_{k-1}]$, $\|\Sigma\| \leqslant c\, \|\Sigma[x_1,..., x_k]\|^3$. If we knew the set $\Sigma[x_1,..., x_{k-1}]$ in advance, we could apply C_{k-1} to $\Sigma[x_1,..., x_{k-1}]$ to obtain its consistent sign assignments; then each consistent sign assignment σ of $\Sigma[x_1,..., x_{k-1}]$ would provide enough input information to the circuit C to enable it to list all consistent sign assignments of $\Sigma[x_1,..., x_k]$ consistent with σ. By doing this in parallel for all such σ, we would get the circuit C_k listing all consistent sign assignments of $\Sigma[x_1,..., x_k]$.

The situation is a bit more complicated than that just described, because the polynomials $\Sigma[x_1,..., x_{k-1}]$ are not known to C at the time of input but are generated along the way. Thus C must use C_{k-1} as a subroutine, calling it at each level to incorporate new polynomials into $\Sigma[x_1,..., x_{k-1}]$ as they are generated, and producing new consistent sign assignments that extend the old assignments with signs for the new polynomials. Using the rough estimate

$$\|\Sigma[x_1,..., x_{k-1}]\| \lesssim \|\Sigma[x_1,..., x_k]\|^3,$$

we get

$$\|\Sigma[x_1,..., x_i]\| \lesssim \|\Sigma[x_1,..., x_k]\|^{3^{k-i}}, \qquad i \leqslant k.$$

The depth of C_k is the product of the depth of C, roughly $(\log \|\Sigma[x_1,..., x_k]\|)^3$, and the depth of C_{k-1}. Inductively this gives

$$\begin{aligned}
\operatorname{depth}(C_k) &\lesssim \prod_{i=0}^{k} (\log \|\Sigma[x_1,..., x_i]\|)^3 \\
&\lesssim \prod_{i=0}^{k} (\log(\|\Sigma[x_1,..., x_k]\|^{3^i}))^3 \\
&\lesssim 9^{k^2} (\log \|\Sigma[x_1,..., x_k]\|)^{3k},
\end{aligned}$$

which is still in NC for fixed number of variables k. If the number of variables can grow linearly with the size of the input, however, then the depth becomes exponential. This is the source of the exponential upper bound in the quantifier elimination procedure that follows.

4. Elimination of Quantifiers

Once we have a circuit to produce the $\Sigma[x_1,..., x_k]$ and their consistent sign assignments, the actual quantifier elimination circuit is straightforward. Suppose we want to decide the truth of the sentence

$$Q_1 x_1 \cdots Q_n x_n B(x_1,..., x_n)$$

where $B(x_1,..., x_n)$ is a Boolean combination of polynomial equations and inequalities in $x_1,..., x_n$. Let $\Sigma[x_1,..., x_n]$ be this set of polynomials, and generate the sets $\Sigma[x_1,..., x_i]$, $1 \leqslant i \leqslant n$, as in Section 3. If $Q_1 = \exists$, construct an $\bigvee$-branch with root r and one leaf for each consistent sign assignment of $\Sigma(x_1)$. If $Q_1 = \forall$, the construction is the same, except we use an $\bigwedge$-branch instead of an $\bigvee$-branch. For each leaf, the consistent sign assignment σ associated with that leaf determines a set of consistent sign assignments for $\Sigma(x_1, x_2)$, namely those assignments that are consistent with σ. If $Q_2 = \exists$, we again construct an $\bigvee$-branch from σ, and each new leaf is associated with a consistent sign assignment of $\Sigma(x_1, x_2)$ that is consistent with σ. Continuing in this fashion, at the bottom of the circuit we have consistent sign assignments for $\Sigma[x_1,..., x_n]$, which determine the truth or falsity of $B(x_1,..., x_n)$. Starting from these truth values, the circuit associates a Boolean value with each node, computing upward toward the root r. The final Boolean value associated with r is *true* iff $Q_1 x_1 \cdots Q_n x_n B(x_1,..., x_n)$ is true. The circuit is not really a tree but a directed acyclic graph; the depth and size of the circuit are roughly those of the circuit C_n constructed in Section 3.

The exponential-depth circuits produced above are uniform and can be simulated in exponential space [3]. We have thus shown

THEOREM. *The theory of real closed fields can be decided in deterministic exponential space or parallel exponential time. In fixed dimension, the theory can be decided in NC.*

Since complex numbers can be encoded as pairs of reals, we also have

COROLLARY. *The theory of the complex numbers under $+$, $\cdot$, and $=$ can be decided in exponential space or parallel exponential time. In fixed dimension, the theory can be decided in NC.*

REFERENCES

1. L. BERMAN, The complexity of logical theories, *Theoret. Comput. Sci.* **11** (1980), 71–77.
2. BORODIN, HOPCROFT, AND VON ZUR GATHEN, Fast parallel matrix and gcd computations, *in* "*Proc. 23rd Symp. on Foundations of Computer Science*," 1982, pp. 65–71.
3. A. BORODIN, On relating time and space to size and depth, *SIAM J. Comput.* **6**, No. 4 (1977), 733–744.
4. W. BROWN AND J. F. TRAUB, On Euclid's algorithm and the theory of subresultants, *J. Assoc. Comput. Mach.* **18** (1971), 505–514.
5. G. E. COLLINS, Quantifier elimination for real closed fields by cylindric algebraic decomposition, *in* "*Proc. 2nd* GI *Conference on Automata Theory and Formal Languages*," Lect. Notes in Comput. Sci. Vol. 35, pp. 134–183, Springer-Verlag, Berlin, 1975.
6. L. CSANKY, Fast parallel matrix inversion algorithms, *SIAM J. Comput.* **5** (1976), 618–623.
7. M. J. FISCHER, AND M. O. RABIN, Super-exponential complexity of Presburger arithmetic, *in* "*Proc.* AMS *Symp. on Complexity of Real Computational Processes*," Vol. 7, 1974.
8. M. MARDEN, *Geometry of Polynomials*. Amer. Math. Soc., Providence, 1966.
9. E. W. MAYR AND A. R. MEYER, The complexity of the word problem for commutative semi-groups and polynomial ideals, *Adv. in Math.* **46** (1982), 305–329.
10. L. MONK, An elementary-recursive decision procedure for Th($\mathbb{R}$, $+, \cdot$), manuscript, University of California, Berkeley, 1974.
11. L. MONK, "Elementary Recursive Decision Procedures," Ph. D. thesis, University of California, Berkeley, 1974.
12. R. JENKS (Ed.), "Proc. ACM Symp. on Symbolic and Algebraic Computation," Yorktown Heights, N.Y., 1976.
13. J. T. SCHWARTZ AND M. SHARIR, On the piano mover's problem. II. General techniques for computing topological properties of real algebraic manifolds, *Adv. in Appl. Math* **4** (1983), 298–351.
14. A. TARSKI, "A Decision Method for Elementary Algebra and Geometry," Univ. of Calif. Press, Berkeley, 1948; 2nd ed. 1951.
15. J. VON ZUR GATHEN, Parallel algorithms for algebraic problems, *in* "*Proc. 15*th ACM *Symp. on Theory of Computing*," April 1983, pp. 17–23.

Printed by the St. Catherine Press Ltd., Tempelhof 41, Bruges, Belgium