Probabilistic Algorithms for Deciding Equivalence of Straight-Line Programs

OSCAR H. IBARRA AND SHLOMO MORAN

University of Minnesota, Minneapolis, Minnesota

Abstract. Let Q be any algebraic structure and $\mathscr{P}$ the set of all total programs over Q using the instruction set $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y, z \leftarrow x/y\}$. (A program is total if no division by zero occurs during any computation.) Let the equivalence problem for $\mathscr{P}$ be the problem of deciding for two given programs in $\mathscr{P}$ whether or not they compute the same function. The following results are proved:

(1) If Q is an infinite field (e.g., the rational numbers or the complex numbers), then the equivalence problem for $\mathscr{P}$ is probabilistically decidable in polynomial time. The result also holds for programs with no division instructions and Q an infinite integral domain (e.g., the integers).
(2) If Q is a finite field, or if Q is a finite set of integers of cardinality ≥ 2, then the equivalence problem is NP-hard.

The case when the field Q is finite but its cardinality is a function of the size of the instance to the equivalence problem is also considered. An example is shown for which a sharp boundary between the classes NP-hard and probabilistically decidable exists (provided they are not identical classes).

Categories and Subject Descriptors: F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*probabilistic computation*; F.1.3 [**Computation by Abstract Devices**]: Complexity Classes—*reducibility and completeness*; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computations in finite fields, computations on polynomials*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; G.3 [**Probability and Statistics**]—*probabilistic algorithms*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Straight-line program, equivalence problem, NP-hard, infinite field, characteristic of a field

1. *Introduction*

Consider the following seemingly simple problem: Given two straight-line programs F_1 and F_2 using only constructs $z \leftarrow 1$, $z \leftarrow x + y$, $z \leftarrow x - y$, $z \leftarrow x * y$, devise an algorithm to determine whether or not F_1 and F_2 are equivalent. An algorithm clearly exists: For each program, derive polynomial expressions for the output variables in terms of the input variables. Then F_1 and F_2 are equivalent if and only if the corresponding expressions (in standard form, i.e., sums of products) are identical. However, in the worst case the process is exponential in the sum of the sizes of the

This research was supported in part by the National Science Foundation under Grant MCS 78-01736.

Authors' present addresses. O. H. Ibarra, Department of Computer Science, University of Minnesota, 136 Lind Hall, 207 Church Street, SE, Minneapolis, MN 55455; S. Moran, Department of Computer Science, Technion—Israel Institute of Technology, Technion City, Haifa 32000, Israel

programs. At present we know of no polynomial-time algorithm for solving this problem.

The equivalence problem for straight-line programs is important because of its relation to a number of decision problems that have recently been studied in the literature. In [4], for example, the equivalence problem for free Boolean graphs was shown to be probabilistically decidable in polynomial time by essentially reducing the problem to the equivalence problem for a restricted class of straight-line programs. In [16], probabilistic algorithms for verifying some polynomial identities were presented. One can easily check that many of the problems discussed in [16] can be reduced to the zero-equivalence problem (i.e., does a program output zero for all inputs?) for straight-line programs. Some important problems involving matrices can also be reduced in polynomial time to the zero-equivalence problem for straight-line programs. For example, we showed in [11] that the problem of deciding for a given positive integer r and a matrix A with polynomial entries (where the polynomials are represented by arithmetic expressions with arbitrary parenthesization using addition, subtraction, multiplication, and exponentiation to a positive integer constant) whether A has rank $\geq r$ is polynomial-time reducible to the zero-equivalence problem for straight-line programs. In [19] (see also [18]) a restricted version of the rank problem for matrices with polynomial entries was shown to be probabilistically decidable.

The main results of this paper are the following:

(1) If Q is an infinite field (e.g., the field of rational numbers or the field of complex numbers), then the zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over Q is probabilistically decidable in polynomial time[1] (Section 2). The result also holds when Q is an infinite integral domain (e.g., the integers). Thus the problems mentioned above (free Boolean graphs, polynomial identities, rank of matrices with polynomial entries) are probabilistically decidable in polynomial time.

(2) If Q is a finite field, or if Q is a finite set of integers (of cardinality ≥ 2), then the zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over Q is NP-hard (Section 3). The proofs of these results provide answers to some open problems in the literature [13, 18].

(3) If Q is finite but its cardinality is a function of the size of the instance (i.e., input program) to the zero-equivalence problem, then we can prove a "gap" theorem (Section 4): If the function (which maps the length of the program to the cardinality of the field) grows fast, then the zero-equivalence problem is probabilistically decidable in polynomial time. If the function grows slowly, then the problem is NP-hard.

One can easily extend the results above to hold for the equivalence problem (i.e., deciding if two programs are equivalent) even when division $z \leftarrow x/y$ is allowed, provided only total programs are considered. (A program is total if no division by zero occurs on any input.) When the programs are not guaranteed to be total and the inputs are integers, the zero-equivalence problem is undecidable [10].

Remark. We could add the constructs $z \leftarrow k * x$ and $z \leftarrow x \uparrow k$ (i.e., multiplication and exponentiation by a positive integer constant) to the instruction set of straight-line programs. However, such instructions will not change the computing power

[1] This means that there is a polynomial-time algorithm which uses a random number generator to decide if a program F computes the zero-function. If the algorithm outputs "yes," then F probably computes the zero-function with probability of error $\leq 1/2$. If the algorithm outputs "no," then F does not compute the zero-function for sure Clearly, a probability of error $\leq 1/2^k$ may be obtained by running the algorithm k times.

of straight-line programs, since it can easily be shown that $z \leftarrow k * x$ and $z \leftarrow x \uparrow k$ can be computed by straight-line programs over $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$ of length $O(\log k)$.

We conclude this section by showing the connection between straight-line programs with one output variable and polynomial expressions. For notational convenience we only consider expressions over the integers.

Definition. A *polynomial expression* (*p.e.*) is any expression which can be derived using rules (1)–(4) below:

(1) 1 is a p.e.
(2) Any variable is a p.e.
(3) If α is a p.e. and k is a positive integer, then $k * \alpha$ and $\alpha \uparrow k$ are p.e.'s
(4) If α and β are p.e.'s, then so are $\alpha + \beta$, $\alpha - \beta$, and $\alpha * \beta$. (Parentheses may be used to avoid ambiguity.)

By straightforward coding (see the remark above) we can easily prove the following proposition.

PROPOSITION 1.1. *We can construct for every p.e. E an equivalent straight-line program F over* $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$ *in time polynomial in the size* (= *length of representation*) *of E.* (*It follows that the size of F is polynomial in the size of E.*)

We can generate p.e.'s from straight-line programs of size n by direct substitutions resulting in expressions of size at most $O(((\sqrt{5} + 1)/2)^n)$. A polynomial bound seems unlikely. Consider, for example, the following program, F (x and y are the input variables, z is the output variable, and $n \geq 2$):

```
$w \leftarrow (x + y) \uparrow 2$
$z \leftarrow (w + y) \uparrow 3$
$w \leftarrow (w + z) \uparrow 4$
$z \leftarrow (w + z) \uparrow 5$
.
.
.
$w \leftarrow (w + z) \uparrow (2n)$
$z \leftarrow (w + z) \uparrow (2n + 1)$
```

Clearly, F can be converted to a straight-line program F' over $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$ whose size is polynomial in the size of F. Now, by direct substitutions, a p.e. E denoting the value of z at the end of F can easily be obtained. However, the length of E is exponential in the size of F. (See [9, 17] for related topics.)

The brief discussion above shows that results (e.g., probabilistic algorithms) for straight-line programs are applicable to polynomial expressions, but the converse may not be true.

2. *The Zero-Equivalence Problem for Programs over Infinite Fields*

In this section we show that the zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over infinite fields is probabilistically decidable in polynomial time.

When we are dealing with zero-equivalence, we assume that the programs have exactly *one* output variable. $F(x_1, \ldots, x_t)$ (or simply F) will denote a program with input variables $x_1, \ldots, x_t$. We also use $F(x_1, \ldots, x_t)$ to denote the input–output function computed by F. Length(F) is the number of instructions in F, while size(F) is the length of the binary representation of F.

We begin with the following lemma.

LEMMA 2.1. *For each program $F(x_1, \ldots, x_t)$ over a field Q there exists a polynomial $P_F(x_1, \ldots, x_t)$ such that $P_F(x_1, \ldots, x_t) = F(x_1, \ldots, x_t)$. If length($F$) $= r$, then deg(P_F) $\leq 2^r$.*[2] *If Q has characteristic $p < \infty$, then the coefficients of the polynomial are integers between* 0 *and $p - 1$; otherwise, the coefficients are integers between $-\infty$ and $+\infty$.*

PROOF. By induction on r. □

LEMMA 2.2. *Let Q be an infinite field, $P(x_1, \ldots, x_t)$ a nonzero polynomial of degree $\leq d$ over Q, and A a subset of Q, $|A| = k \geq d$. Then there are at least $(k - d)^t$ elements $\bar{a} = (a_1, \ldots, a_t)$ in A^t such that $P(\bar{a}) \neq 0$.*

PROOF. The proof is by induction on t. For $t = 1$ the lemma follows from the fact that a one-variable polynomial of degree d has at most d roots in Q.

Assume the lemma holds for $t \geq 1$, and consider a nonzero polynomial $P(x_1, \ldots, x_t, x_{t+1})$ of degree $\leq d$. Then there exists a $t + 1$ tuple $(a_1, \ldots, a_t, c)$ such that $P(a_1, \ldots, a_t, c) \neq 0$. Hence the polynomial $P(x_1, \ldots, x_t, c) = \hat{P}(x_1, \ldots, x_t)$ is not a zero polynomial. Clearly, deg($\hat{P}$) $\leq d$. Hence, by the induction hypothesis, there are at least $(k - d)^t$ t-tuples $(a_1, \ldots, a_t)$ in A^t such that $\hat{P}(a_1, \ldots, a_t) = P(a_1, \ldots, a_t, c) \neq 0$. For each such tuple, the one-variable polynomial $\tilde{P}(x_{t+1}) = P(a_1, \ldots, a_t, x_{t+1})$ is not a zero polynomial, and therefore there are at least $k - d$ elements a_{t+1} in A such that $\tilde{P}(a_{t+1}) = P(a_1, \ldots, a_t, a_{t+1}) \neq 0$. The lemma follows. □

COROLLARY 2.1. *Let $P(x_1, \ldots, x_t)$, d, k, and A be as in Lemma* 2.1. *Let $k \geq 2td$, and let $\bar{a} = (a_1, \ldots, a_t)$ be a random element of A^t. Then $P(\bar{a}) = P(a_1, \ldots, a_t) \neq 0$ with probability $\geq \frac{1}{2}$.*

PROOF

$$\begin{aligned}\text{Prob}(P(\bar{a}) \neq 0) &= \frac{|\{\bar{a} \mid \bar{a} \text{ in } A^t, P(\bar{a}) \neq 0\}|}{|A^t|} \\ &\geq \left(\frac{k-d}{k}\right)^t \geq \left(1 - \frac{1}{2t}\right)^t \\ &\geq \frac{1}{2}.\end{aligned}$$ □

In the proof of the main result of this section we have to distinguish between two cases. First, we deal with the case when the field F over which the programs are defined is the rational numbers (this represents the case when F is any field of infinite characteristic). Later we deal with the case when F is an infinite field with finite characteristic. .

2.1 THE ZERO-EQUIVALENCE PROBLEM OVER THE RATIONALS

PROPOSITION 2.1. *A program $F \equiv 0$ (i.e., computes the zero-function) over the rationals if and only if $F \equiv 0$ over the integers.*

PROOF. This follows from Lemma 2.1 □

Proposition 2.1 shows that deciding zero-equivalence for programs over the rationals is equivalent to deciding zero-equivalence for programs over the integers.

[2] deg(P_F) denotes the degree of $P_F = \max\{$sum of powers of the variables in any term$\}$.

Hence we assume in the remainder of this subsection that the inputs to the programs are integers.

Definition. Let F be a program and m a positive integer. Then $F_{(m)}$ is the program obtained from F by replacing each instruction by the equivalent instruction modulo m. Thus $z \leftarrow 1$, $z \leftarrow x + y$, $z \leftarrow x - y$, and $z \leftarrow x * y$ are replaced by the instructions $z \leftarrow 1 \pmod{m}$, $z \leftarrow x + y \pmod{m}$, $z \leftarrow x - y \pmod{m}$, and $z \leftarrow x * y \pmod{m}$, respectively.

The relationship between F and $F_{(m)}$ is given by the following lemma.

LEMMA 2.3. *$F(x_1, \ldots, x_t) \pmod{m} = F_{(m)}(x_1, \ldots, x_t)$.*

PROOF. By induction on length(F) and the fact that $[x \pmod{m}\ \theta\ y \pmod{m}] \pmod{m} = x\ \theta\ y \pmod{m}$ for $\theta = +, -, *$. □

The next lemma gives an upper bound on the size of $F(x_1, \ldots, x_t)$.

LEMMA 2.4. *Let $a_1, \ldots, a_t$ be input integers such that $|a_i| \leq a$, where $a \geq 2$. Then $|F(a_1, \ldots, a_t)| \leq a^{2^r}$, where r = length (F).*

PROOF. By induction on r. □

LEMMA 2.5. *Let k be a given integer, $1 \leq |k| \leq 2^{2n2^n}$, and let m be an integer chosen at random from the set $M = \{1, 2, \ldots, 2^{2n}\}$. Then for all sufficiently large n, $k \not\equiv 0 \pmod{m}$ with probability $\geq 1/4n$.*

PROOF. By the prime number theorem [7], the number of primes smaller than 2^{2n} tends to $2^{2n}/\ln 2^{2n}$ and hence, for large n, is greater than $2^{2n}/2n$. Let $p_1, \ldots, p_s$ be the distinct prime divisors of k. Clearly, $s \leq \log_2 k \leq 2n2^n$. (If $k = 1$, then $s = 0$.) Hence, for large enough n there are at least $2^{2n}/2n - 2n2^n$ primes smaller than 2^{2n} which do not divide k. Since for $n \geq 10$, $2^{2n}/2n - 2n2^n > 2^{2n}/4n$, the lemma follows. □

We are now ready to prove the main result of this subsection.

LEMMA 2.6. *The zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over the rationals is probabilistically decidable in polynomial time.*

PROOF. Let $F(x_1, \ldots, x_t)$ be a program of length r. Let $A = \{1, 2, \ldots, 2t2^r\}$ and $M = \{1, 2, \ldots, 2^{2n}\}$, where $n = r + t$. The following algorithm probabilistically decides if $F(x_1, \ldots, x_t)$ computes the zero-function.

```
begin
  for $i = 1$ to $8n$ do
    begin
      choose a random element $(\bar{a}, m)$ in $A^t \times M$
      compute $F_{(m)}(\bar{a})$
      if $F_{(m)}(\bar{a}) \neq 0$ then [output ('no'); halt]
    end
  output ('yes')
  halt
end
```

Let G be the event $F_{(m)}(\bar{x}) \not\equiv 0$ and H the event $F(\bar{x}) \not\equiv 0$. (By Lemma 2.3, $F_{(m)}(\bar{x}) = F(\bar{x}) \pmod{m}$, and hence $G \subseteq H$.) If $F \equiv 0$ (i.e., F computes the zero-function), then the algorithm will output 'yes' no matter what random elements $(\bar{a}, m)$ are chosen inside the for-loop. Now suppose that $F \not\equiv 0$. In this case, since $G \cap H = G$, we have that $\text{Prob}(G) = \text{Prob}(H)\text{Prob}(G|H)$. By Corollary 2.1 and the fact that

$\deg(P_F) \le 2^r$ (Lemma 2.1), $\text{Prob}(H) \ge \frac{1}{2}$. Let $\bar{x} = (a_1, \ldots, a_t)$ be any element from A^t. Then $|a_i| \le 2n2^n \le 2^{2n}$, and hence, by Lemma 2.4, $|F(\bar{x})| \le 2^{2n2^n}$. Hence, by Lemma 2.5, if $F(\bar{x}) \ne 0$, then $\text{Prob}(F(\bar{x}) \ne 0 \pmod m) \ge 1/4n$, where m is a random element from M. Since $F_{(m)}(\bar{x}) = F(\bar{x}) \pmod m$, this means that $\text{Prob}(G \mid H) \ge 1/4n$, and therefore $\text{Prob}(G) \ge 1/8n$. Now the algorithm will output 'yes' if and only if $F_{(m)}(\bar{x}) = 0$ for all $8n$ random samplings of $\bar{x}$ and m. The probability of this event happening is $\le (1 - 1/8n)^{8n} < \frac{1}{2}$. Hence, if $F \not\equiv 0$, the algorithm will output 'no' with probability $> \frac{1}{2}$.

Since the algorithm uses modulo m arithmetic for $m \le 2^{2n}$, the time complexity of the algorithm is polynomial in size(F). This completes the proof. □

2.2. The Zero-Equivalence Problem for Infinite Fields with Finite Characteristic. A field Q has finite characteristic if there exists a positive integer n such that $n\alpha = 0$[3] for every element α in Q. The characteristic of Q is then defined to be the minimal positive integer n which has this property. Throughout the rest of this subsection Q denotes an infinite field of finite characteristic, and p denotes the characteristic of Q (p must be a prime number).

Definition. For α in Q, $\alpha \ne 0$, order(α) $= n$ if $\alpha^n = 1$ and $\alpha^m \ne 1$ for $0 < m < n$. If $\alpha^n \ne 1$ for all positive integer n, then order(α) $= \infty$. It is known that if order(α) $\ge p^d - 1$ for some d, then $P(\alpha) \ne 0$ for any polynomial $P(x)$ with coefficients in GF(p) of degree $< d$ [3].

Lemma 2.7. *For each integer n there is an α in Q such that order(α) $> n$.*

Proof. Suppose the lemma is false. Then for some integer n_0, $n_0 = \max\{\text{order}(\alpha) \mid \alpha \text{ in } Q\}$. Then it can easily be shown that order(α) divides n_0 for each α in Q [3]. Hence, all the elements in Q are roots of the polynomial $x^{n_0+1} - x$, which means that $|Q| \le n_0 + 1$. This contradicts the infiniteness of Q. □

We need the following definition for our next lemma.

Definition. Let GF(p) be the Galois field of integers modulo p. Then GF(p)[x] is the ring of one-variable polynomials over GF(p), where addition and multiplication are defined in the standard way. (The zero element of GF(p)[x] is the zero polynomial.)

Lemma 2.8. *$F \equiv 0$ over Q if and only if $F \equiv 0$ over GF(p)[x].*

Proof. Let $F(x_1, \ldots, x_t)$ be a program and r = length(F). Let d be the smallest integer satisfying $p^d > 2t2^r$. Define a set $A \subseteq \text{GF}(p)[x]$ by $A = \{a(x) \mid a(x) \text{ in } \text{GF}(p)[x], \deg(a(x)) < d\}$. Note that $|A| = p^d$. Let α in Q be such that order(α) $> p^{2^r d}$. (Such an α exists by Lemma 2.7.) Let $A_\alpha \subseteq Q$ be the set $A_\alpha = \{a(\alpha) \mid a(x) \text{ in } A\}$. We claim that $|A| = |A_\alpha|$. (Otherwise, $a_1(\alpha) = a_2(\alpha)$ for some $a_1(x)$, $a_2(x)$ such that $\deg(a_1(x)) < d$, $\deg(a_2(x)) < d$. Then α is a root of the polynomial $a_1(x) - a_2(x)$ whose degree $< d$. This is impossible, since order(α) $> p^d$.) It follows, by Lemma 2.2, that $F \equiv 0$ over Q if and only if $F \equiv 0$ over A_α. By a similar argument, $F \equiv 0$ over GF(p)[x] if and only if $F \equiv 0$ over A.

Suppose now that $F \equiv 0$ over GF(p)[x]. Then $F(a_1(x), \ldots, a_t(x)) = 0$ for all $a_1(x), \ldots, a_t(x)$ in A. In particular, $F(a_1(\alpha), \ldots, a_t(\alpha)) = 0$ for all $a_1(\alpha), \ldots, a_t(\alpha)$ in A_α, which implies that $F \equiv 0$ over Q. On the other hand, if $F \not\equiv 0$ over GF(p)[x],

[3] $n\alpha = \alpha + \cdots + \alpha$ (n times)

then $F \not\equiv 0$ over A. Hence, for some $a_1(x), \ldots, a_t(x)$ in A, the polynomial $P_F(a_1(x), \ldots, a_t(x)) = F(a_1(x), \ldots, a_t(x))$ is a nonzero polynomial. Call this polynomial $\hat{a}(x)$. Now $\deg(\hat{a}(x)) < 2^r d$, and since $\text{order}(\alpha) > p^{2^r d}$, $\hat{a}(\alpha) = F(a_1(\alpha), \ldots, a_t(\alpha)) \neq 0$. It follows that $F \not\equiv 0$ over Q. □

We have shown that deciding zero-equivalence for programs over Q is equivalent to deciding zero-equivalence for programs over GF(p)[x]. We shall assume, therefore, that the inputs to the program F are elements of GF(p)[x]. (Note that a polynomial of degree d over GF(p) can be represented by its coefficients in $O(d)$ space.) One can easily verify that Corollary 2.1 still holds if Q is replaced by the ring GF(p)[x]:

COROLLARY 2.1′. *Let $P(x_1, \ldots, x_t)$ be a nonzero polynomial of degree $\leq d$ over GF(p)[x]. Let A be a subset of GF(p)[x] such that $|A| \geq 2td$. Let $\bar{a} = (a_1, \ldots, a_t)$ be a random element of A^t. Then $P(\bar{a}) \neq 0$ with probability $\geq \frac{1}{2}$.*

Definiton. For a polynomial $q(x)$ in GP(p)[x], F_q is the program obtained by replacing all instructions in F by the equivalent instructions modulo $q(x)$.

The proof of the following lemma is similar to that of Lemma 2.3 and is omitted.

LEMMA 2.9. *Let F be a program with t input variables, and let $q(x)$ be in GF(p)[x]. Then $F_q(a_1(x), \ldots, a_t(x)) = F(a_1(x), \ldots, a_t(x))$ (mod $q(x)$) for all $a_1(x), \ldots, a_t(x)$ in GF(p)[x].*

LEMMA 2.10. *Let $\hat{a}(x)$ in GF(p)[x] be given such that $\deg(\hat{a}(x)) < 2^d$, where $d \geq 2$. Then there are at least $p^{2d}/8d$ monic polynomials of degree $2d$ over GF(p) which do not divide $\hat{a}(x)$. (Note that there are exactly p^{2d} monic polynomials of degree $2d$.)*

PROOF. Clearly, there are at most $2^d/2d$ distinct monic irreducible polynomials of degree $2d$ which divide $\hat{a}(x)$. By [3, Eq. 3.37] there are more than $p^{2d}/4d$ monic irreducible polynomials of degree $2d$ ($d \geq 2$). The result follows from the fact that for $d \geq 2$, $p^{2d}/8d \geq 2^d/2d$ for all primes p. □

We can now prove the following.

LEMMA 2.11. *The zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over Q is probabilistically decidable in polynomial time.*

PROOF. Let $F(x_1, \ldots, x_t)$ be a program of length r. By Lemma 2.8, it is sufficient to decide if $F \not\equiv 0$ over GF(p)[x]. Let d be the smallest integer satisfying $p^d > 2t2^r$. (Note that $d = O(r)$, and d can be found in time polynomial in r.) Let $A = \{a(x) \mid a(x) \text{ in GF}(p)[x], \deg(a(x)) < d\}$. Let $M = \{q \mid q$ is a monic polynomial of degree $2(d + r)$ over GF(p)$\}$. The algorithm below probabilistically decides if $F(x_1, \ldots, x_t)$ computes the zero-function. The complexity and probabilistic analysis of this algorithm are similar to that of the algorithm given in the proof of Lemma 2.6, knowing that the following hold for random $(a_1(x), \ldots, a_t(x))$ in A^t and $q(x)$ in M:

(i) $\deg(P_F(a_1(x), \ldots, a_t(x)) \leq d2^r$. (This follows from Lemma 2.1.)
(ii) If $F \not\equiv 0$, then $\text{Prob}(F(a_1(x), \ldots, a_t(x)) \neq 0) \geq \frac{1}{2}$ (by Corollary 2.1′).
(iii) If $F(a_1(x), \ldots, a_t(x)) \neq 0$, then $\text{Prob}(F(a_1(x), \ldots, a_t(x)) \neq 0 \text{ (mod } q(x))) \geq 1/8(d + r)$ (by Lemma 2.10, (i), and the inequality $d2^r < 2^{d+r}$ for $d \geq 1$).

The details are omitted.

```
begin
  for $i = 1$ to $16(d + r)$ do
    begin
      choose a random element $((a_1(x), \ldots, a_t(x)), q(x))$ in $A^t \times M$
      compute $F_q(a_1(x), \ldots, a_t(x))$
      if $F_q(a_1(x), \ldots, a_t(x)) \neq 0$ then [output('no'); halt]
    end
  output('yes')
  halt
end
```
□

Combining Lemma 2.6 and 2.11, we obtain the main result of this section.

THEOREM 2.1. *The zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over any infinite field F is probabilistically decidable in polynomial time.*

PROOF. If F is of inifinite characteristic, then F contains an isomorphic image of the rational numbers, and hence the theorem follows from Lemma 2.6. If F is of finite characteristic, then the theorem holds by Lemma 2.11. □

Remark. The proof of Theorem 2.1 can be easily modified to show that the zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over any infinite integral domain (i.e., infinite commutative ring with no zero divisors) which contains the unit element is probabilistically decidable in polymomial time.

3. *NP-Hard Zero-Equivalence Problems*

In this section we show that the zero-equivalence problem for straight-line programs becomes NP-hard when we restrict the input domain to be a finite set. We shall consider two cases: (a) when the input domain is a finite set of integers, and (b) when the input domain is a finite field.

THEOREM 3.1. *Let D be any finite set of integers with at least two elements. Then the zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over D is NP-hard.*

PROOF. We shall reduce the satisfiability problem for Boolean formulas in conjunctive normal form (CNF) to our problem. So let $L = C_1 \cdots C_m$ be a Boolean formula in CNF over variables $x_1, \ldots, x_n$. Assume that each C_i contains exactly three literals. (A literal is a variable or its negation.) We shall construct a program F such that $F(a_1, \ldots, a_n) = 0$ for all a_i in D if and only if L is not satisfiable. F has input variables $X_1, \ldots, X_n$, output variable z, and auxiliary variables which include $\bar{X}_1, \ldots, \bar{X}_n, z_1, \ldots, z_m$, etc. Assume that $D = \{\alpha_1, \ldots, \alpha_k\}$, where $k \geq 2$ and $\alpha_1 < \cdots < \alpha_k$. The relationship between the logical formula L and the program F constructed below is as follows: For each truth assignment to the variables $x_1, \ldots, x_n$ there corresponds one or more input assignments to the variables $X_1, \ldots, X_n$ such that the truth assignment satisfies L if and only if each of the corresponding input assignments produces a nonzero output. The correspondence between the assignments is the following: If x_i is assigned 'false,' then X_i is assigned α_1. If x_i is assigned 'true,' then X_i is assigned α_j for some $j > 1$. The program F is the following:

Segment 1. For $i = 1, 2 \ldots, n$, write the code to perform the following task:

$X_i \leftarrow X_i - \alpha_1$

(If the input is $(a_1, \ldots, a_n)$ in D^n, then after segment 1, $X_i = 0$ if $a_i = \alpha_1$; otherwise, $X_i > 0$.)

Segment 2. For $i = 1, 2, \ldots, n$, write the code to perform the following task:

$\bar{X}_i \leftarrow ((\alpha_2 - \alpha_1) - X_i)((\alpha_3 - \alpha_1) - X_i) \cdots ((\alpha_n - \alpha_1) - X_i)$

(If $X_i = 0$, then $\bar{X}_i > 0$; if $X_i > 0$, then $\bar{X}_i = 0$.)

Segment 3. For $i = 1, 2, \ldots, m$, write the code to perform the following task:

$z_i \leftarrow$ sum of the variables representing the literals of C_i

(If C_i is satisfied, then $z_i > 0$; if C_i is not satisfied, then $z_i = 0$.)

Segment 4

$z \leftarrow 1$
$z \leftarrow z * z_1$
$\vdots$
$z \leftarrow z * z_m$

Clearly, we can construct the program F in time polynomial in the size of L. (Note that an instruction of the form $z \leftarrow c$, where $c > 1$, can be coded over $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$ using at most $O(\log c)$ instructions.) Moreover, F computes the zero-function if and only if L is not satisfiable. The result now follows, since the satisfiability problem is NP-hard [5]. □

Note. The proof of Theorem 3.1 above implies that the inequivalence problem for polynomial expressions over a finite set of integers [13] is NP-complete, thus settling an open problem in [13].

Remark. One can easily verify that when D has exactly one element, then the zero-equivalence problem is probabilistically decidable in polynomial time. (Simply evaluate the program for the given value in D using modular arithmetic, where the modulus is chosen randomly.)

THEOREM 3.2. *Let Q be any finite field. Then the zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over Q is NP-hard.*

PROOF. Since every finite field is isomorphic to $GF(p^s)$ for some prime p and positive integer s, we may assume that $Q = GF(p^s)$. Let $L = C_1 \cdots C_m$ be a Boolean formula in CNF over variables $x_1, \ldots, x_n$, where each C_i contains exactly three literals. We shall construct a program F such that $F(a_1, \ldots, a_n) = 0$ for all a_i in Q if and only if L is not satisfiable. F has input variables $x_1, \ldots, x_n$, output variable z, and auxiliary variables which include $\bar{x}_1, \ldots, \bar{x}_n, z_1, \ldots, z_m$, etc. The description of F follows:

Segment 1. For $i = 1, 2, \ldots, n$, write the code to perform the following task:

$x_i \leftarrow x_i^{p^s-1}$

(Clearly, the code can be written to have at most $O(\log p^s) = O(s)$ instructions. If the input is $(a_1, \ldots, a_n)$ in Q^n, then after the segment, $x_i = 0$ if $a_i = 0$; otherwise $x_i = 1$. This follows from the fact that in $GF(p^s)$, $a^{p^s-1} = 1$ for all $a \neq 0$.)

Segment 2. For $i = 1, 2, \ldots, n$, write the code to perform the following task:

$\bar{x}_i \leftarrow 1 - x_i$

(If $x_i = 0$, then $\bar{x}_i = 1$; if $x_i = 1$, then $\bar{x}_i = 0$.)

Segment 3. For $i = 1, 2, \ldots, m$, if $C_i = l_{i_1} + l_{i_2} + l_{i_3}$, each l_{i_j} in $\{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$, then write the code to perform the following task:

$z_i \leftarrow (1 - l_{i_1})(1 - l_{i_2})(1 - l_{i_3})$
$z_i \leftarrow 1 - z_i$

(If C_i is satisfied, i.e., one of l_{i_1}, l_{i_2}, or l_{i_3} is 1, then $z_i = 1$; if C_i is not satisfied, then $z_i = 0$.)

Segment 4

```
$z \leftarrow 1$
$z \leftarrow z * z_1$
.
.
.
$z \leftarrow z * z_m$
```

It is straightforward to verify that F computes the zero-function if and only if L is not satisfiable. Moreover, F can be constructed in polynomial time. $\square$

Note. The proof of Theorem 3.2 above implies that the nonzero-equivalence problem for polynomial expressions over finite fields is NP-complete. This settles (positively) an open problem in [18].

4. *A Boundary Between Probabilistically Decidable Problems and NP-Hard Problems*

We have shown that the zero-equivalence problem for $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-programs over infinite fields is probabilistically decidable in polynomial time (Section 2). On the other hand, over finite fields the problem is NP-hard (Section 3). In this section we look at what happens when the size of the field (over which the zero-equivalence of programs is to be decided) is a function of the length of the program. A special instance of this problem was considered implicitly in [4], where equivalence of free Boolean graphs was shown to be probabilistically decidable in polynomial time. The proof in [4] was essentially a reduction to the zero-equivalence problem for a restricted class of straight-line programs over finite fields, where the size of the field is a function of the sizes of the graphs.

We shall show that if the function which maps the length of the program to the size of the field grows fast, then zero-equivalence is probabilistically decidable in polynomial time. If the function grows slowly, then the problem is NP-hard, and there is a sharp boundary between these two cases. Thus in certain cases we can show a "gap" between the class R of sets which are probabilistically decidable in polynomial time (see [2, 14] for a precise definition of R) and the class NPC of NP-complete sets. This result is of special interest in light of [15], which gives strong evidence that the class NP of nondeterministic polynomial-time languages consists of an infinite hierarchy of accepting density classes (in the sense of [1]), the higher classes in the hierarchy corresponding to sets which have smaller accepting density.

Definition. For each nonnegative rational number t, let H_t be the problem of deciding for an arbitrary $\{z \leftarrow 1, z \leftarrow x + y, z \leftarrow x - y, z \leftarrow x * y\}$-program F over $\mathrm{GF}(2^{\lfloor r^t \rfloor})$, $r = \mathrm{length}(F)$, whether F computes a nonzero-function.

THEOREM 4.1

(*a*) *If* $t \geq 1$, *then* H_t *is probabilistically decidable in polynomial time.*
(*b*) *If* $t < 1$, *then* H_t *is NP-complete.*

PROOF

(a) Let F be a program of length r. By Lemma 2.1, $\deg(P_F) \leq 2^r$. Clearly, if $\deg(P_F) = 2^r$, then P_F has one of two forms: $P_F = x^{2^r}$ or $P_F = x^{2^{r-1}}y^{2^{r-1}}$, where x and y are input variables. It is easy to check whether $\deg(P_F) = 2^r$. If $\deg(P_F) = 2^r$, then $F \not\equiv 0$ over any nontrivial field. So assume that $\deg(P_F) < 2^r$. Since there are at least 2^r elements in $\mathrm{GF}(2^{\lfloor r^t \rfloor})$ for $t \geq 1$, $F \equiv 0$ over $\mathrm{GF}(2^{\lfloor r^t \rfloor})$ if and only if P_F is the zero polynomial, and hence if and only if $F \equiv 0$ over any infinite field which contains $GF(2^{\lfloor r^t \rfloor})$. This last problem can be decided probabilistically in polynomial time as in Lemma 2.11.

(b) To show that H_t is NP-hard, let $L = C_1 \cdots C_m$ be a Boolean formula in CNF over variables $x_1, \ldots, x_n$, where each C_i contains exactly three literals. We shall construct a program F of length r over $GF(2^{\lfloor r^t \rfloor})$ such that F computes the zero-function if and only if L is not satisfiable. The construction is given below.

Stage 1. Define $r = \lceil (8(m+n))^{1/(1-t)} \rceil$ and $d = \lfloor r^t \rfloor$. Hence $Q = GF(2^d)$.

Stage 2. Carry out the reduction given in the proof of Theorem 3.2. Let the program obtained be F' and $r' = \text{length}(F')$. Referring to the proof of Theorem 3.2, we see that segment 1 requires at most $2n(\lfloor r^t \rfloor + 1)$ instructions, since $x_i \leftarrow x_i^{2^d-1}$ can be coded as

$$\begin{array}{l} y \leftarrow y - y \\ y \leftarrow y + x_i \\ x_i \leftarrow 1 \\ x_i \leftarrow x_i * y \\ \left.\begin{array}{l} \left.\begin{array}{l} y \leftarrow y * y \\ x_i \leftarrow x_i * y \end{array}\right\} \\ \quad\vdots \\ \left.\begin{array}{l} y \leftarrow y * y \\ x_i \leftarrow x_i * y \end{array}\right\} \end{array}\right\} d-1 \text{ pairs} \end{array}$$

Segments 2, 3, and 4 require at most $n + 1$, $6m + 1$, and $m + 1$ instructions, respectively. Hence, $r' \le 2n(\lfloor r^t \rfloor + 1) + n + 1 + 6m + 1 + m + 1 \le 8(m+n)\lfloor r^t \rfloor$. Since $8(m+n) \ge r^{1-t}$, $r' \le r$.

Stage 3. Pad F' with $r - r'$ additional instructions of the form $x_1 \leftarrow 1$. Let the program obtained be F Then $\text{length}(F) = r$ and F computes the zero-function over $GF(2^{\lfloor r^t \rfloor})$ if and only if L is not satisfiable. Moreover, F can be constructed in polynomial time.

To show that H_t is in NP, we use the following nondeterministic algorithm, given a program $F(x_1, \ldots, x_n)$:

(1) Compute $d = \lfloor r^t \rfloor$, where $r = \text{length}(F)$.
(2) Nondeterministically find an irreducible polynomial of degree d over GF(2). This can be done by guessing a polynomial of degree d over GF(2) and checking (in deterministic polynomial time using Berlekamp's algorithm [12]) that it is irreducible.
(3) Using the irreducible polynomial found in (2), generate nondeterministically field elements $a_1, \ldots, a_n$ in $GF(2^{\lfloor r^t \rfloor})$ and compute $F(a_1, \ldots, a_n)$.
(4) Output 'yes' if $F(a_1, \ldots, a_n) \ne 0$; otherwise output 'no.' □

REFERENCES

(Note. References [6, 8] are not cited in the text)

1 Adleman, L Two theorems on random polynomial time Proc. 19th Ann. IEEE Symp. on Foundations of Computer Science, Ann Arbor, Mich., 1978, pp 75–83.

2. Adleman, L., and Manders, K. Reducibility, randomness, and intractability. Proc. 9th Ann. ACM Symp. on Theory of Computing, Boulder, Colo, 1977, pp. 151–163

3 Berlekamp, E. *Algebraic Coding Theory.* McGraw-Hill, New York, 1968.

4. Blum, M, Chandra, A, and Wegman, M. Equivalence of free Boolean graphs can be decided probabilistically in polynomial time. *Inf. Proc. Lett 10* (1980), 80–82.

5 Cook, S The complexity of theorem-proving procedures Proc. 3rd Ann. ACM Symp. on Theory of Computing, Shaker Heights, Ohio, 1971, pp 151–158.

6 Garey, M, and Johnson, D *Computers and Intractability A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

7 Hardy, G, and Wright, E. *The Theory of Numbers,* 4th ed. Oxford University Press, 1960.

8. HOPCROFT, J., AND ULLMAN, J.D. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, Mass., 1979.

9. HYAFIL, L. On the parallel evaluation of multivariate polynomials. Proc. 10th Ann. ACM Symp. on Theory of Computing, San Diego, Calif., 1978, pp. 193–195.

10. IBARRA, O., AND LEININGER, B. On the simplification and equivalence problems for straight-line programs To appear in *J. ACM* (available as Tech. Rep. 79-21, Computer Science Dep., Univ. of Minnesota, Minneapolis, Minn., 1979).

11 IBARRA, O., ROSIER, L., AND MORAN, S. Probabilistic algorithms and straight-line programs for some rank decision problems. *Inf. Proc. Lett. 12* (1981), 227–232.

12. KNUTH, D. *The Art of Computer Programming, Vol. 2. Seminumerical Algorithms.* Addison-Wesley, Reading, Mass., 1969.

13. MEYER, A., AND STOCKMEYER, L. Word problems requiring exponential time. Proc. 5th Ann. ACM Symp. on Theory of Computing, Austin, Texas, 1973, pp 1–9

14. MILLER, G. Riemann's hypothesis and tests of primality. Ph D. Dissertation, Univ. of California, Berkeley, Calif, 1975.

15. MORAN, S. On the accepting density hierarchy in NP (submitted for publication) A more complete version is available as Tech. Rep. 79-29, Computer Science Dep., Univ. of Minnesota, Minneapolis, Minn, 1979.

16. SCHWARTZ, J. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM 27*, 4 (Oct. 1980), 701–717.

17 VALIANT, L. Completeness classes in algebra. Proc. 11th Ann ACM Symp on Theory of Computing, Atlanta, Ga., 1979, pp. 249–261

18. YEMINI, Y. Some theoretical aspects of position-location problems. Proc. 20th Ann. IEEE Symp on Foundations of Computer Science, San Juan, Puerto Rico, 1979, pp. 1–8.

19. YEMINI, Y. On some randomly decidable geometrical problems. Submitted for publication.

RECEIVED APRIL 1980, REVISED MAY 1981; ACCEPTED FEBRUARY 1982