INFORMATION AND COMPUTATION **91**, 142–154 (1991)

Boolean Circuits versus Arithmetic Circuits*

JOACHIM VON ZUR GATHEN[†]

Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4

AND

GADIEL SEROUSSI[‡]

Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, California 94304

We compare the two computational models of Boolean circuits and arithmetic circuits in cases where they both apply, namely the computation of polynomials over the rational numbers or over finite fields. Over $\mathbb{Q}$ and finite fields, Boolean circuits can simulate arithmetic circuits efficiently with respect to size. Over finite fields of small characteristic, the two models are equally powerful when size is considered, but Boolean circuits are exponentially more powerful than arithmetic circuits with respect to depth. Most of the technical results given in this synopis are taken from the literature. © 1991 Academic Press, Inc.

1. INTRODUCTION

The most natural model for the computation of polynomials over a field is the *arithmetic circuit* (or *computation* or *straight-line program*), with the following operations: constants from the field, variables (=inputs), and the binary operations +, −, *, /. There are three measures associated with such an arithmetic circuit: its size, its depth, and the degree of the polynomial that it computes.

When the field elements can be represented over some finite alphabet, then it makes sense to also consider Boolean circuits for computing such

* A preliminary version of this paper appeared in the "Proc. 6th Int. Conf. Computer Science, Santiago, Chile, 1986," pp. 171–184.

[†] Partly supported by Natural Sciences and Engineering Research Council of Canada, Grant A2514, by Schweizerischer Nationalfonds, Grant 2175-0.83, by DFG Grant SFB 124, and by Fundación Andes, beca C-10246. Parts of the first author's work were done during visits to Universität Zürich, Switzerland, to Universität des Saarlandes, Saarbrücken, Germany, and to Pontificia Universidad Católica de Chile, Santiago, Chile, and as a Visiting Fellow at the Computer Science Laboratory, Australian National University, Canberra, Australia.

[‡] This author's work was done at the Department of Computer Science, Technion, Israel Institute of Technology, Haifa, Israel.

polynomials. Again, we consider the measures of size and depth. The fields of interest are $\mathbb{Q}$ and the finite fields.

The goal of this paper is to review the literature with a view to comparing the power of these two model of computations, in the cases where they are both applicable. This is an instance of *structured vs. general models* (Borodin, 1982). Each model defines a class P of problems solvable in polynomial size, and a class NC of problems solvable simultaneously in poly-logarithmic depth and polynomial size.

In Section 2, we give some definitions and consider the case of rational numbers. Here Boolean circuits can—under reasonable conditions—efficiently simulate arithmetic circuits, while no efficient simulation in the other direction is known. In general, we use probabilistic choice in the simulating Boolean circuit; when no division occurs in the arithmetic circuit, then randomness is not required.

Starting in Section 3, we consider finite fields. Boolean circuits can simulate arithmetic ones with at most polynomial increase in size. For the reverse direction, no size-efficient simulation is known; we exhibit specific polynomials that appear hard.

In Section 4, we consider the case of small characteristic p, i.e., the simulations are considered to be *efficient* if the size increase is polynomial in p (rather than $\log p$), and the depth increase is polynomial in $\log p$ (rather than $\log \log p$). Then Boolean circuits (and arithmetic circuits over the prime field) can simulate arithmetic circuits with only polynomial increases in size and depth. The reverse simulation is provably impossible for depth; the exponentiation problem distinguishes between the two models. Thus for parallel computations over finite field of small characteristic, arithmetic circuits are a too restrictive model.

In Section 5, we give an extension of the fast parallel Boolean exponentiation procedure to finite commutative algebras of small characteristic. Section 6 applies this to the problem of factoring polynomials over large finite fields of small characteristic.

A recent development is that algebraic computation in structures of a different type—certain groups and monoids–may help in classifying problems of small parallel complexity (see Mix Barrington and Thérien, 1988, Sect. 8).

2. Arithmetic vs Boolean Circuits over $\mathbb{Q}$

As usual, FP denotes the class of Boolean functions that can be computed by uniform Boolean circuit families of polynomial size. For the complexity class $NC^k(P\text{-uniform})$, we also restrict the depth of the circuits to be $O(\log(\text{inputsize})^k)$, and finally $NC(P\text{-uniform}) =$

$\bigcup_{k \geqslant 0} NC^k$($P$-uniform). (See Cook, 1985, for terminology.) We have taken P-uniform as the appropriate notion of uniformity, so that our classes may conceivably differ from the log-space uniform NC^k and NC considered in Cook (1985); Ruzzo (1981) discusses various notions of uniformity. C (non-uniform) is obtained by removing the uniformity condition, for any of these complexity classes C.

Let F be a field, $n \geqslant 0$, and $x_1, ..., x_n$ indeterminates over F. An arithmetic circuit α over F with n inputs $x_1, ..., x_n$ has 0-ary operations from $F \cup \{x_1, ..., x_n\}$, and binary operations from $\{+, -, *, /\}$. We denote by $s(\alpha)$ and $d(\alpha)$ the size and depth of α, respectively. (See Strassen, 1972, with slightly different terminology.) To each node of α is associated in a natural way a rational function from $F(x_1, ..., x_n)$. We say that α *computes* any of these functions. We assume that no division by the rational function zero occurs. A sequence $(\alpha_n)_{n \in \mathbb{N}}$ is called a circuit family. A polynomial family $(f_n)_{n \in \mathbb{N}}$ consists of polynomials $f_n \in F[x_1, ..., x_n]$.

FP_F, NC_F^2, and NC_F consist of those polynomial families that can be computed by (P-uniform) arithmetic circuit families of polynomial size, and simultaneously $O(\log^2 n)$ and $(\log n)^{O(1)}$ depth, respectively, for input size n. (Note that, in general, an arithmetic circuit computes a rational function; we only consider polynomials here.) If we drop the uniformity condition, we obtain FP_F (non-uniform), etc. Eberly (1989), and von zur Gathen (1986) discuss uniformity for arithmetic circuits. It is usually—at least over infinite fields—reasonable to restrict the degrees of the polynomials:

$$D_F = \{\text{polynomial families } (f_n)_{n \in \mathbb{N}} : \deg(f_n) = n^{O(1)}\}.$$

Then $FP_F \cap D_F$ (non-uniform) consists of the *p-computable families* of Valiant (1982). This class does not change if we disallow divisions (Strassen, 1973). While it is conjectured that $NC^2 \neq P$ (see Cook, 1985), we have the surpirising result that

$$FP_F\ (\text{non-uniform}) \cap D_F = NC_F^2\ (\text{non-uniform}) \cap D_F$$

for all F (Valiant *et al.*, 1983; Miller *et al.*, 1988; see Borodin *et al.*, 1982, for finite fields).

Elements of $\mathbb{Q}$ or finite fields can be represented over a finite alphabet, and we can consider both arithmetic and Boolean circuits for the computation of polynomials. The goal of this paper is to compare the power of the two models. More precisely, for a polynomial $f \in F[x_1, ..., x_n]$ we consider an arithmetic circuit α computing f, and a Boolean circuit β which, on input of a representation (e.g., in binary) of inputs $a_1, ..., a_n \in F$, computes a representation of $f(a_1, ..., a_n)$. If F is infinite (and $n \geqslant 1$), α will work for infinitely many inputs, but β only for a finite set of inputs. For a finite field

F we consider a binary description of the field size q to be part of the input, so that *polynomial in the input size* means $(n \log q)^{O(1)}$, and *poly-logarithmic* means $(\log n + \log \log q)^{O(1)}$. We are mainly interested in exponential gaps, and identify size and depth functions that are polynomially related. In particular, it will turn out that for a finite field F of small characteristic, the problem of computing large powers (with an exponent of polynomial binary length) is in NC, but not in NC_F; i.e., it can be efficiently computed by Boolean circuits, but not by arithmetic circuits over F.

We consider $F = \mathbb{Q}$ in this section. Everything carries over to the case of algebraic number fields.

In one direction, arithmetic circuits of polynomial size can compute outputs of exponential binary length, so that, trivially, $FP_{\mathbb{Q}}$ is not a subset of (non-uniform) FP. However, if one allows size polynomial in both input and output size (which may be more than polynomial in the input size for this question) and also allows random choice in the Boolean circuit, then probabilistic Boolean circuits can simulate arithmetic circuits over $\mathbb{Q}$ in polynomial size (von zur Gathen, 1985, Corollary 6.9). If we do not insist on uniformity, then we can use deterministic Boolean circuits for this simulation (Adleman, 1978). To compute the value modulo an integer of an arithmetic circuit without division at integer inputs, randomness is not required (von zur Gathen, 1985); Jung (1985) gives a very efficient simulation with a depth increase from d to $O(d \log^* n)$, where n is the number of inputs, and $\log^*$ is the iterated logarithm. If tests for zero and division with remainder of integers are allowed, then one can factor integers by arithmetic circuits over $\mathbb{Q}$ of linear size (Shamir, 1979); many people hope that one cannot do this with (probabilistic) Boolean circuits of polynomial size.

For the other direction, the answer is not clear. On the one hand, it seems hopeless to give a general simulation of Boolean circuits (computing a polynomial) by arithmetic circuits with at most polynomial increase in size, although no specific apparently hard example is known. On the other hand, no superpolynomial lower bounds for arithmetic circuits computing polynomials of polynomial degree are known. Some polynomials are conjectured to be hard for both models: the Boolean version of the permanent is $\#P$-complete (Valiant, 1979a), and the arithmetic version is p-complete (Valiant, 1979b).

We note that Boolean circuits can solve problems that arithmetic circuits are not adapted to. For example, in Boolean size $m^{O(1)}$, one can decide on input of two integers a, b with binary length at most m and $b \neq 0$ whether $r = a/b \in \mathbb{Z}$ or not. Given such an $r \in \mathbb{Q}$, it seems that one cannot decide this by *arithmetic Boolean circuits* (allowing tests "$a = 0$?") over $\mathbb{Q}$ of size $m^{O(1)}$, but only in size $2^{O(m)}$ (by generating all pairs (a, b)) (see von zur Gathen, 1986, for a description of these circuits (which are called *arithmetic networks* there)).

3. Finite Fields of Large Characteristic

Consider a finite field $F = \mathbb{F}_p$, where p is a (large) prime. Let $t = \lceil \log p \rceil$. An element a of F can be represented by the unique binary representation $(b_0, ..., b_{t-1}) \in \{0, 1\}^t$ of $b = \sum_{0 \leqslant i < t} b_i 2^i$ with $0 \leqslant b < p$ and $a = (b \bmod p) \in F = \mathbb{Z}/(p)$. (We distinguish between the objects $b \in \mathbb{Z}$ and $b \bmod p \in \mathbb{F}_p$, of different type.)

Consider the function $B_i: F \to \{0, 1\}$, with $B_i(a) = b_i$ for $0 \leqslant i < t$. One direction of the question of relative strength of arithmetic and Boolean circuits can be posed as follows: can the functions B_i be computed efficiently by arithmetic circuits over F? A positive answer would imply that arbitrary Boolean circuits can be simulated efficiently. As any function from F to $\{0, 1\}$ (considering $\{0, 1\} \subseteq F$), every B_i can be expressed as a polynomial of degree at most $p-1$ over F; e.g., $B_0(a) = \sum_{0 \leqslant i < p/2} (1 - [a - (2i+1) \bmod p]^{p-1})$ for any $a \in F$. Thus, $O(p)$ is an upper bound on the sequential arithmetic complexity of each B_i, but no upper bound polynomial in the input size $\log p$ is known.

For the other direction of comparison, we note the following well-known fact.

Proposition 3.1. *Let p be a prime, $F = \mathbb{F}_p$, and α an arithmetic circuit over F of size s and depth d.*

(i) *α can be simulated by a Boolean circuit of size $O(s \log p \log^2 \log p)$ and depth $O(d \log p \log^2 \log p)$.*

(ii) *If α has no divisions, then α can be simulated by a Boolean circuit of size $s(\log p)^{O(1)}$ and depth $O(d \log \log p)$.*

Proof. (ii) follows from the fact that one addition or multiplication of integers modulo p can be performed on (P-uniform) Boolean circuits of size $(\log p)^{O(1)}$ and depth $O(\log \log p)$ (Beame *et al.*, 1986). For (i), we note that with the Extended Euclidean Scheme, one can compute one inverse modulo p in size $O(\log p \log^2 \log p)$ (see Aho *et al.*, 1974, Sect. 8.11).

Since in our arithmetic complexity classes, the field is considered constant, we trivially have $FP_F \subseteq FP$ and $NC_F^k \subseteq NC^k$ for all k. However, taking field size into consideration, only (ii) gives a satisfactory simulation, while the depth of (i) may be very large. (As mentioned in the previous sentence, our complexity classes like NC_F^k do not capture this fact. Von zur Gathen, 1986, introduces *c-universal* circuits to remedy this situation, but we will not use this notion.) Let δ be the degree of the polynomial computed by an arithmetic circuit α. Typically, we think of a single input size parameter N such that $\log p$, s, and δ are polynomial in

N, and d is polynomial in $\log N$. Then the depth in (i) may increase to polynomial in N. Fixing $N = s\delta \log p$, there are two possibilities for avoiding this:

(i) One can eliminate the divisions by a parallel version of Strassen's (1973) method. This involves possibly extending the ground field, then making random choices from the larger field; see Borodin *et al.* (1982) and von zur Gathen (1985) for details. The individual steps in Strassen's method are trivial to parallelize. The results are "random polynomial-time uniform" Boolean circuits of size $s^{O(1)}$ and depth $d^{O(1)}$. By the parallelization method of Valiant *et al.* (1983), we can achieve depth $O((\log N)^2)$ without any assumptions about the depth of the input circuit, but then lose even this weak uniformity property.

(ii) We might allow a "redundant representation" of $a \in F$ by the binary representations of $a_1, a_2 \in \mathbb{N}$ with $0 \leqslant a_1, a_2 < p$, $a_2 \neq 0$ and $a = (a_1/a_2 \bmod p)$. This leads to Boolean circuits of size $O(sN^2)$ and depth $O(d \log^2 N)$, using trivial arithmetic.

The upshot is that we have both size- and depth-efficient simulations of arithmetic circuits without divisions by Boolean circuits; with divisions, the situation is less satisfactory. If p is small, say $p \leqslant N$ (or $p = N^{O(1)}$), then we have good simulations also in the presence of divisions.

Both sequential and parallel simulations of Boolean circuits by arithmetic circuits are wide open.

4. Finite Fields of Small Characteristic

The most interesting case for our comparative study is given by finite fields of small characteristic p, i.e., where in our simulations the size is allowed to increase by a factor $p^{O(1)}$, and the depth by $(\log p)^{O(1)}$. Here the exponentiation problem discriminates between arithmetic and Boolean circuits for parallel computation. Instead of Boolean circuits, we consider arithmetic circuits over the prime field $F = \mathbb{F}_p$; those can now be efficiently simulated by Proposition 3.1.

In Section 2, we defined the notion of an arithmetic circuit computing a rational function f. An arithmetic circuit has several outputs in general; for simplicity, we restrict ourselves to arithmetic circuits with one output in the sequel. Over finite fields, it is usually more relevant to consider the associated pointwise mapping. If α is an arithmetic circuit over F with inputs $x_1, ..., x_n$, then its domain of definition $\operatorname{def}(\alpha) \subseteq F^n$ consists of those values $a \in F^n$ for which no division by zero occurs in α. Recall that no division by the rational function zero is allowed, so that $\operatorname{def}(\alpha)$ is a proper algebraic subset of F^n, if F is infinite. α value-computes a function $f\colon F^n \to F$ if α ourputs $f(a)$ for all $a \in \operatorname{def}(\alpha)$. (No condition is imposed for the $a \notin \operatorname{def}(\alpha)$; thus if F is finite and $\operatorname{def}(\alpha) = \varnothing$, then α value-computes any function. See von zur Gathen, 1987, for a discussion.) If α computes f, then α also value-computes f; over finite fields, the reverse implication may fail to hold. In the sequel, log always stands for $\log_2$ unless otherwise specified.

FACT 4.1. *Let p be a prime, $n \geqslant 1$, $q = p^n$, $F = \mathbb{F}_p \subseteq K = \mathbb{F}_q$, $0 < e < q$, and $\pi_K^e\colon K \to K$ with $\pi_K^e(a) = a^e$ for $a \in K$. Then*

(i) *There exist P-uniform arithmetic circuits over F of depth $O(\log(np))$ and size $(n \log p)^{O(1)}$ computing π_K^e.*

(ii) *For any arithmetic circuit over K value-computing π_K^e with depth d we have (assuming $e \leqslant q/2$)*

$$d \geqslant \min\{\log e, \log(q/2 - e + 1), \log q - \log\log q - 1\}.$$

Proof. (i) is in von zur Gathen (1990), building upon the pioneering NC_F^2-result (for small p) by Fich and Tompa (1988), who obtained log-space uniform depth $O(\log n \log(np))$. (ii) is in von zur Gathen (1987). ▌

COROLLARY 4.2. *If $p \leqslant n$ and $p^{n-2} \leqslant e \leqslant p^{n-1}$, then π_K^e can be computed by arithmetic circuits over F of size $n^{O(1)}$ and depth $O(\log n)$. Any arithmetic circuit over K value-computing π_K^e has depth $\Omega(n)$.*

The assumption $e < p^n$ is not a severe restriction, since $a^{p^n} = a$ for all $a \in K$. For $e \geqslant p^n$, one can compute $e' < p^n$ such that $e \equiv e' \bmod (p^n - 1)$ by P-uniform Boolean circuits of size $(\log e)^{O(1)}$ and depth $O(\log\log e)$ (Beame *et al.*, 1986). Then $\pi_K^e = \pi_K^{e'}$.

We can phrase this result so that the arithmetic and Boolean depth complexity of polynomials over one fixed (infinite) field differ exponentially. Let p be a prime number, K an algebraic closure of $\mathbb{F}_p$, $m \in \mathbb{N}$, and $2^{m-1} \leqslant e \leqslant 2^m$. Then we have Boolean circuits of depth $O(\log m)$ that compute π_K^e on inputs with binary length at most m, i.e., from subfields $\mathbb{F}_{p^n} \subset K$ with $\log(p^n) \leqslant m$. Any arithmetic circuit over K value-computing π_K^e (for inputs from these subfields) has depth $\Omega(m)$.

In order to apply the lower bound to situations other than exponentiation, we note the following fact.

LEMMA 4.3. *Let $K = \mathbb{F}_q$, $f \in K[x]$ of degree $e < \frac{3}{4}q$, and α an arithmetic circuit of depth d over K value-computing the function $K \to K$ given by f. Then*

$$d \geqslant \min\{\log e, \log(\tfrac{3}{4}q - e + 1), \log q - \log q - 2\}.$$

If $q/2 \geqslant e \geqslant \sqrt{q}$ and $q \geqslant 1898$, then $d \geqslant (\log q)/2$.

Proof. Let $S = \mathrm{def}(\alpha)$. We use von zur Gathen (1987, Lemma 2.3). If $\# S \geqslant \frac{3}{4}q$, then d is greater or equal to the minimum of the first two terms in the claim, and otherwise d is at least as large as the third term. For large enough q (in fact, for $q \geqslant 1898$), a straightforward calculation shows that all three terms are at least $(\log q)/2$. ∎

When K is a finite field of small characteristic, a natural reperesentation of a field element as a vector over the prime field can be computed by an arithmetic circuit over K of polynomial size (Lempel *et al.*, 1982). Corollary 4.5 below shows that it cannot be computed in sublinear depth. We first state a more general result for any *F-linear mapping* $\phi: K \to F$, with $\phi(ua + vb) = u\phi(a) + v\phi(b)$ for $u, v \in F$ and $a, b \in K$.

THEOREM 4.4. *Let p be a prime number, $n \geqslant 11$, and $\phi: K \to F$ be any nonzero F-linear mapping from $K = \mathbb{F}_{p^n}$ onto $F = \mathbb{F}_p$. Then any arithmetic circuit over K that value-computes ϕ has depth at least $(n \log p)/2$.*

Proof. Any nonzero linear transformation ϕ from K onto F can be expressed as $\phi(a) = T(u_\phi a)$, where $T(b) = \sum_{0 \leqslant i < n} b^{p^i}$ is the trace function from K onto F, and $u_\phi \in K \setminus \{0\}$ depends only on ϕ (Lidl and Niederreiter, 1983, Theorem 2.24). Hence, the degree of ϕ as a polynomial over K is $p^{n-1} < \frac{3}{4}p^n$, and the claim now follows from Lemma 4.3 (using $n \geqslant 11$ to ensure $p^n \geqslant 1898$). ∎

Now we consider a generator $\alpha \in K$ of $K = \mathbb{F}_{p^n}$ as F-algebra, say $\alpha = x \bmod f$ if $K = F[x]/(f)$ and $f \in F[x]$ is irreducible of degree n. An element a of K has a unique representation as a vector over $F = \mathbb{F}_p$,

$$a = \sum_{0 \leqslant i < n} \phi_i(a)\, \alpha^i,$$

with $\phi_i(a) \in F$. Each ϕ_i is nonzero and F-linear, since $ua + bv = \sum (u\phi_i(a) + v\phi_i(b))\alpha^i$. Thus we have the following.

COROLLARY 4.5. *Let $n \geqslant 11$, $0 \leqslant i < n$, and $\phi_i: K \to F$ as above. Any arithmetic circuit over K that value-computes ϕ_i has depth at least $(n \log p)/2$.*

These results become slightly stronger when we only consider division-free arithmetic circuits. Then in Lemma 4.3 we have

$$d \geqslant \min\{\log e, \log(q - e + 1)\},$$

and depth at least $(n-1)\log p$ in Theorem 4.4 and Corollary 4.5.

Our framework is the two models of computation for polynomial functions over K: arithmetic circuits over K and arithmetic circuits over F (or, equivalently, Boolean circuits if p is small). In the first model, the input is some $u \in K$, and in the second model, the inputs are the coordinates $\phi_i(u) \in F$ of u. As any function from K to $K \supseteq F$, ϕ_i can be expressed as a polynomial in $K[x]$. Corollary 4.5 says that even these input functions for arithmetic circuits over F cannot be calculated in less than linear depth by arithmetic circuits over K.

The results of this and the previous section can be interpreted in two ways. In defense of arithmetic circuits one may say that only polynomials of *small* (e.g., polynomial) degree should be considered. Otherwise, the conclusion is that arithmetic circuits are a very weak model of parallel computation over large finite fields of small characteristic; lower bounds in this model for a problem may not reflect its true complexity. While the restriction to polynomials of small degree may be reasonable over infinite fields, over finite fields very natural problems, whose formulation does not involve large powers, require the computation of such powers. Apart from Theorem 4.3 and Corollary 4.4, we now mention two important decision problems. In our restricted model of arithmetic circuits, let us say that to decide whether a property holds or does not hold, one has to value-compute the field constants 1 or 0, respectively. The *algebraic computation trees* of Strassen (1983) and the *arithmetic Boolean circuits* of von zur Gathen (1986) (also called arithmetic networks), are better suited to deal with this type of decision problem; however, we have stated the lower bounds only for arithmetic circuits. (In fact, they are also valid for arithmetic Boolean circuits (von zur Gathen, unpublished).)

PROPOSITION 4.6. *Let $F = \mathbb{F}_q$, with $q \geq 1898$ odd, and α an arithmetic circuit over F.*

(i) *If α decides whether the input $u \in F$ is a square, then $d \geq (\log q)/2 - 2$.*

(ii) *If α decides whether a quadratic input polynomial from $F[x]$ is irreducible, then $d \geq (\log q)/2 - 3$.*

Proof. (i) For any nonzero $u \in F$ we have $u^{(q-1)/2} = \pm 1$, and u is a square if and only if $u^{(q-1)/2} = 1$. By assumption, on input some $u \in \operatorname{def}(\alpha)$, α computes $v = 1 \in F$ if u is a square, and $v = 0$ otherwise. We now obtain a circuit of depth $d+2$ which value-computes $u^{(q-1)/2} = 2v - 1$ (for $u \neq 0$). The claim follows from Lemma 4.3.

(ii) On input $u_1, u_0 \in F$, α decides whether $f = x^2 + u_1 x + u_0 \in F[x]$ is irreducible or not. With $u_1 = 0$, f is irreducible if and only if $-u_0$ is a square. The claim follows by (i). ▌

If $q = p^n$, then both problems of this proposition can be solved by arithmetic circuits over $\mathbb{F}_p$ of depth $O(\log(np))$ and size $(n \log p)^{O(1)}$.

5. Powers in Commutative Algebras of Small Characteristic

Fich and Tompa (1988) present an arithmetic circuit over $\mathbb{F}_p$ for exponentiation in $\mathbb{F}_{p^n}$ of log-space uniform depth $O(\log n \log(np))$; based on their approach, this was improved by von zur Gathen (1990) to depth $O(\log(np))$, but only P-uniformly (Fact 4.1) (see also Litow and Davida, 1988). We now describe a more general setting in which the Fich and Tompa method still works, while the other approach seems to break down. Let p be a power of a prime, $F = \mathbb{F}_p$, $n \in \mathbb{N}$, and R a commutative n-dimensional algebra over F. (See Herstein, 1968 for a general background on algebras.) We assume that $u_1, ..., u_n \in R$ form an F-basis $(u_1, ..., u_n)$ of R, that an element $a \in R$ is represented by its coordinates $a_1, ..., a_n \in F$ such that $a = \sum_{1 \leqslant i \leqslant n} a_i u_i$, and that the coordinates of the product ab of $a, b \in R$ can be computed by an arithmetic circuit over F with $2n$ inputs, n outputs, size s_R and depth d_R. If we are given the structure constants $v_{ijk} \in F$ such that

$$u_i u_j = \sum_{1 \leqslant k \leqslant n} v_{ijk} u_k \qquad \text{for} \quad 1 \leqslant i, j \leqslant n,$$

then we can choose $s_R = O(n^3)$ and $d_R = O(\log n)$. However, we do not insist on having these structure constants and only assume some circuit with costs s_R, d_R. We consider the problem of computing large powers in R.

Example 5.1. Let y be an indeterminate over F, $g \in F[y]$ of degree n (not necessarily irreducible) and $R = F[y]/(g)$. We can choose $s_R = n^{O(1)}$ and $d_R = O(\log n)$ (Eberly, 1989). This includes, of course, the case $R = \mathbb{F}_{p^n}$.

Example 5.2. With n, y, g, R as above, let $r \in \mathbb{N}$, z an indeterminate over R, $h \in R[z]$ of degree r (not necessarily irreducible), and $T = R[z]/(h)$. We can choose $s_T = (rn)^{O(1)}$ and $d_T = O(\log(rn))$.

To talk about asymptotic complexity, we think of our algebra R as being part of some infinite family; for fixed R, s_R and d_R are $O(1)$. E.g., in Example 5.1, "multiplication in R" is the problem of computing the coefficients $c_0, ..., c_{n-1} \in F$ of

$$c = \left(\sum_{0 \leqslant i < n} a_i y^i\right) \cdot \left(\sum_{0 \leqslant i < n} b_i y^i\right) \bmod \sum_{0 \leqslant i \leqslant n} g_i y^i,$$

given the input $a_0, ..., a_{n-1}, b_0, ..., b_{n-1}, g_0, ..., g_n$.

In order to compute the eth power of $a=\sum_i a_i u_i \in R$ for any $e \in \mathbb{N}$, we consider the Frobenius mapping

$$\phi : R \to R,$$
$$a \mapsto a^p.$$

This is an F-linear mapping. We consider e as fixed, or the p-ary representation $(e_0, ..., e_t)$ of e, with $t=\lceil \log_p e \rceil$, $e=\sum_{0 \leqslant j < t} e_j p^j$ and $0 \leqslant e_j < p$ for all j, as *hard-wired* into the following circuit over F.

ALGORITHM. Power in a Commutative Algebra.
Input. Coordinates $a_1, ..., a_n \in F$ of $a=\sum_{1 \leqslant i \leqslant n} a_i u_i \in R$.
Output. Coordinates of $a^e \in R$.

1. For all i, $1 \leqslant i \leqslant n$, compute the coordinates of u_i^p. This gives the $n \times n$-matrix representing ϕ.
2. For all j, $0 \leqslant j \leqslant t$, compute ϕ^j, $a^{p^j} = \phi^j(a_1, ..., a_n)^t$, and $a^{e_j p^j} = (a^{p^j})^{e_j}$.
3. Return $a^e = \prod_{0 \leqslant j < t} a^{e_j p^j}$.

THEOREM 5.3. *The above algorithm computes the coordinates of a^e. It can be implemented on a log-space uniform arithmetic circuit over F of size $O(s_R(n \log p + \log e) + n^3 \log e)$ and depth $O(d_R \log p + (d_R + \log n) \log\log e)$.*

Proof. Since ϕ is the matrix of the F-linear map $a \mapsto a^p$, ϕ^j is the matrix of the map $a \mapsto a^{p^j}$, i.e., $\phi^j(a_1, ..., a_n)^t$ are indeed the coordinates of a^{p^j}. This proves correctness of the algorithm.

Step 1 can be implemented in size $O(ns_R \log p)$ and depth $O(d_R \log p)$. For step 2, size $O(tn^3 + ts_R \log p)$ and depth $O(\log t \log n + d_R \log p)$ are sufficient, and for step 3, size $O(ts_R)$ and depth $O(d_R \log t)$. Note that $t \log p = O(\log e)$. ∎

In the special case of a simply generated algebra $R = \mathbb{F}_p[x]/(h)$, von zur Gathen (1990) applies the results of Eberly (1989) to obtain the following.

Fact 5.4. Let $e, m, n \in \mathbb{N}$, $K = \mathbb{F}_{p^n}$, $h \in K[z]$ of degree m, and $R = K[z]/(h)$. Then $\pi_R^e : R \to R$ can be computed by a P-uniform arithmetic circuit over $F = \mathbb{F}_p$ of size $(mn)^{O(1)} \cdot \log p$ and depth $O(\log(mnp))$, and on Boolean circuits of depth $O(\log(mnp))$ and size $(mn \log p)^{O(1)}$.

The method of Fich and Tompa (1988) yields depth $O(\log(mn) \cdot \log(mnp))$ in this case. It seems unlikely that one could similarly obtain optimal depth for the general case of Theorem 5.3.

6. Factoring Polynomials

We consider the problem of factoring a polynomial $f \in K[x]$ of degree m, where $K = \mathbb{F}_{p^n}$ is a finite field. A careful analysis of the probabilistic method in von zur Gathen (1984) shows that the algorithm (involving tests "$a \neq 0$?" and branching) can be implemented over $F = \mathbb{F}_p$ in depth $O(\log^2(mn) + \log(mnp))$ and size $\log p(mn)^{O(1)}$, where f is assumed to be squarefree. A deterministic version of the algorithm works in depth $O(\log^2(mn)\log(mp))$ and size $p(mn)^{O(1)}$, using the parallel rank algorithm of Mulmuley (1987). If f is not squarefree, then its squarefree decomposition has to be computed first. The algorithm presented in von zur Gathen (1984) for squarefree decomposition requires the computation of large powers in K and has depth $O(\log^2(mn) + T_K)$, where T_K is the depth required to compute $a^p, a^{p^2}, ..., a^{p^{n-1}}$ for $a \in K$. For a few years, this seemed to be a bottleneck for parallel factorization. However, by Fact 5.4, $T_K = O(\log(mnp))$. Hence, the whole factoring algorithm for an arbitrary polynomial can be done probabilistically in depth $O(\log^2(mn) + \log(mnp))$ over F. The deterministic version has depth $O(\log^2(mn)\log(mp))$. If p is small, this is poly-logarithmic depth.

Acknowledgments

We thank Martin Tompa and an anonymous referee for valuable suggestions.

Received May 12, 1989; final manuscript received October 27, 1989

References

Adleman, L. (1978), Two theorems on random polynomial time, *in* "Proceedings, 19th IEEE Symp. Foundations of Computer Science, Los Angeles CA, pp. 75–83.

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1974), "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, MA.

Beame, P. W., Cook, S. A., and Hoover, H. J. (1986), Log depth circuits for division and related problems, *SIAM J. Comput.* **15**, 994–1003.

Borodin, A. (1982), Structured vs. general models in computational complexity, *in* "Logic and Algorithmic," Symposium in honour of Ernst Specker, *Enseign. Math.* **30**, 47–65.

Borodin, A., von zur Gathen, J., and Hopcroft, J. (1982), Fast parallel matrix and GCD computations, *Inform. and Control* **52**, 241–256.

Cook, S. A. (1985), A taxonomy of problems with fast parallel algorithms, *Inform. and Control* **64**, 2–22.

Eberly, W. (1989), Very fast parallel polynomial arithmetic, *SIAM J. Comput.* **18**, 955–976.

Fich, F. E., and Tompa, M. (1988), The parallel complexity of exponentiating polynomials over finite fields, *J. Assoc. Comput. Mach.* **35**, 651–667.

VON ZUR GATHEN, J. (1984), Parallel algorithms for algebraic problems, *SIAM J. Comput.* **13**, 802–824.

VON ZUR GATHEN, J. (1985), Irreducibility of multivariate polynomials, *J. Comput. System Sci.* **31**, 225–264.

VON ZUR GATHEN, J. (1986), Parallel arithmetic computations: A survey, *in* "Proceedings, 12th Int. Symp. Math. Foundations of Computer Science, Bratislava, Lecture Notes in Computer Science, Vol. 233, pp. 93–112, Springer-Verlag, Berlin.

VON ZUR GATHEN, J. (1987), Computing powers in parallel, *SIAM J. Comput.* **16**, 930–945.

VON ZUR GATHEN, J. (1990) Inversion in finite fields, *J. Symbolic Comput.* **9**, 175–183.

HERSTEIN, I. N. (1968), "Noncommutative Rings," Carus Math. Monographs, Vol. 15, Wiley, New York.

JUNG, H. (1985), Depth efficient transformations of arithmetic into Boolean circuits, *in* "Proceedings, Fundamentals of Computation Theory 1985," Lecture Notes in Computer Science, Vol. 199, pp. 167–174, Springer-Verlag, Berlin.

LEMPEL, A., SEROUSSI, G., AND ZIV, J. (1982), On the power of straight-line computations in finite fields, *IEEE Trans. Inform. Theory* **28**, 875–880.

LIDL, R., AND NIEDERREITER, H. (1983), "Finite Fields," Encyclopedia Math. Appl., Vol. 20, Addison-Wesley, Reading, MA, (now distributed by Cambridge Univ. Press).

LITOW, B. E., AND DAVIDA, G. I. (1988), $O(\log(n))$ Parallel time finite field inversion, *in* "Proceedings, Aegean Workshop on Computing," Lecture Notes in Computer Science, Vol. 319, pp. 74–80, Springer-Verlag, Berlin.

MILLER, G. L., RAMACHANDRAN, V., AND KALTOFEN, E. (1988), Efficient parallel evaluation of straight-line code and arithmetic circuits, *SIAM J. Comput.* **17**, 687–695.

MIX BARRINGTON, D. A., AND THÉRIEN, D. (1988), Finite monoids and the fine structure of NC^1, *J. Assoc. Comput. Mach.* **35**, 941–952.

MULMULEY, K. (1987), A fast parallel algorithm to compute the rank of a matrix over an arbitrary field, *Combinatorica* **7**, 101–104.

RUZZO, W. L. (1981), On uniform circuit complexity, *J. Comput. System Sci.* **22**, 365–383.

SHAMIR, A. (1979), Factoring numbers in $O(\log n)$ arithmetic steps, *Inform. Process. Lett.* **8**, 28–31.

STRASSEN, V. (1972), Berechnung und Programm, I, *Acta Inform.* **1**, 320–335.

STRASSEN, V. (1973), Vermeidung von Divisionen, *J. Reine Angew. Math.* **264**, 182–202.

STRASSEN, V. (1983), The computational complexity of continued fractions, *SIAM J. Comput.* **12**, 1–27.

VALIANT, L. G. (1979a), Completeness classes in algebra, *in* "Proceedings, 11th Annu. ACM Symp. Theory of Computing, Atlanta GA, pp. 249–261.

VALIANT, L. G. (1979b), The complexity of computing the permanent, *Theoret. Comput. Sci.* **8**, 189–201.

VALIANT, L. G. (1982), Reducibility by algebraic projections, in "Logic and Algorithmic," Symposium in honour of Ernst Specker, *Enseign. Math.* **30**, 365–380.

VALIANT, L., SKYUM, S., BERKOWITZ, S., AND RACKOFF, C. (1983), Fast parallel computation of polynomials using few processors, *SIAM J. Comput.* **12**, 641–644.

Printed in Belgium