Parallel Construction of Irreducible Polynomials.

Gudmund S. Frandsen [1] [2]

version September 21, 2004

Abstract

Let arithmetic pseudo-$\mathtt{NC}^k$ denote the problems that can be solved by log space uniform arithmetic circuits over the finite prime field $\mathbf{F}_p$ of depth $O(\log^k(n+p))$ and size $(n+p)^{O(1)}$.

We show that the problem of constructing an irreducible polynomial of specified degree over $\mathbf{F}_p$ belongs to pseudo-$\mathtt{NC}^{2.5}$.

We prove that the problem of constructing an irreducible polynomial of specified degree over $\mathbf{F}_p$ whose roots are guaranteed to form a normal basis for the corresponding field extension pseudo-$\mathtt{NC}^2$-reduces to the problem of factor refinement.

We show that factor refinement of polynomials is in arithmetic $\mathtt{NC}^3$. Our algorithm works over any field and compared to other known algorithms it does not assume the ability to take p'th roots when the field has characteristic p.

CR Categories: F.2.1.

[1] This research was supported by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

[2] Department of Computer Science, Aarhus University, Ny Munkegade, 8000 Aarhus C, Denmark. `gsfrandsen@daimi.aau.dk`

Introduction

We study parallel arithmetic computation over both a general field and over finite prime fields.

We consider problems that have known feasible solutions (sequential or probabilistic). However, the study of deterministic parallel algorithms can give new insight into the structure of a problem.

We focus on minimising the circuit depth (parallel time). Only secondarily, will we consider the exact exponent of the processor bound.

Model of Computation.

We choose log space uniform arithmetic circuits as our model of computation. We allow only gates for the usual arithmetic operations $(+, \cdot, -, /)$ and the constants $0, 1$. By excluding arbitrary constants, we need not define how a log space TM should represent fancy constants in connection with uniformity requirements. Other constants (in the prime field concerned) can be built explicitly from $0, 1$. See [Ebe89] for a discussion of constants and uniformity.

Boolean operations may be simulated arithmetically (using field constants 0 and 1). However, inputs and outputs will be field elements only, and these are treated atomically, i.e. we can not access their possible bit representations (at least not directly). For an overview of different arithmetic models of parallel computation (circuits and PRAM's) see [KaRa90].

We shall use the following complexity classes and notions of reduction:

1. Let $\mathbf{F}$ denote an arbitrary field. We define arithmetic $\mathtt{NC}^k$ to consist of those problems with domain $\mathbf{F}^n$ that are solved by log space uniform circuits of depth $O(\log^k(n))$ and size $n^{O(1)}$.

2. Let $\mathbf{F}_p$ be the unique prime field with p elements. We define pseudo-$\mathtt{NC}^k$ to be those problems with domain $\mathbf{F}_p^n$ that are solved by log space uniform (in n and p) circuits of depth $O(\log^k(n+p))$ and size $(n+p)^{O(1)}$.

   We say that problem A *pseudo-*$\mathtt{NC}^k$*-reduces* to problem B, when we can solve A using log space uniform arithmetic circuits with oracle gates for B such that the circuits have depth $O(\log^k(n+p))$, size $(n+p)^{O(1)}$ and only a constant number of oracle gates occur on any path from an input to an output. In this way, we know that if B belongs to pseudo-$\mathtt{NC}^l$ so does A, provided $l \geq k$.

When restricting computations to finite fields it turns out that deterministic polynomial time solutions for some natural problems (polynomial factorisation and construction of q'th nonresidues) are known only when the characteristic is included in the input size. For parallel computations a similar phenomenon occurs for modular exponentiation. This is the motivation for our definition of pseudo-$\mathtt{NC}^k$.

Problems considered.

For the first two problems, we only consider polynomials over finite prime fields. Let $\mathbf{F}_p$ be the finite prime field with p elements for a prime p.

1. *Irreducible Polynomial Construction* is the following problem: Given $n \in \mathbf{N}$ find an irreducible monic $f \in \mathbf{F}_p[x]$ of degree n.

   A solution to the problem gives *normal basis guarantee* if the roots of f form a basis for $\mathbf{F}_{p^n}$ over $\mathbf{F}_p$.

2. *Polynomial Factorisation* is the following problem: Given monic $f \in \mathbf{F}_p[x]$ of degree n, find the unique minimal set of monic irreducible polynomials $\{g_1, g_2, ..., g_k\} \subset \mathbf{F}_p[x]$ such that $f = \prod_{i=1}^{k} g_i^{n_i}$ for suitable $n_i \in \mathbf{N}$.

For the last problem, we consider polynomials over a general field $\mathbf{F}$.

3. Polynomial *Factor Refinement* is the following problem: Given monic $f_1, f_2, ..., f_k \in \mathbf{F}[x]$, find monic $g_1, g_2, ..., g_l \in \mathbf{F}[x]$ such that (i) $\gcd(g_i, g_j) = 1$ for $i \neq j$ and (ii) for all i, $f_i = \prod_{j=1}^{l} g_j^{n_{ij}}$, for some integer exponents n_{ij}. The refinement is *parsimonious* if (iii-a) for all j, $\gcd(n_{1j}, n_{2j}, ..., n_{kj}) = 1$. The refinement is *square free* if (iii-b) for all j, g_j is square free.

Summary of technical results.

Our primary result is:

1. The problem of constructing an irreducible polynomial is in arithmetic pseudo-$\mathtt{NC}^{2.5}$

The bottleneck of the construction is factor refinement, by means of which we can also give a normal basis guarantee:

2. The problem of irreducible polynomial construction with normal basis guarantee pseudo-$\mathtt{NC}^2$-reduces to the problem of factor refinement.

As a byproduct, we also prove

3. The problem of polynomial factorisation pseudo-$\mathtt{NC}^2$-reduces to the problem of factor refinement.

We solve the problem of factor refinement for general fields:

4. The problem of parsimonious factor refinement is in arithmetic $\mathtt{NC}^3$.

Our algorithm can be used for integers as well as for polynomials, and we prove

5. If the gcd of a set of integers can be found in Boolean $\mathtt{NC}^k$ ($k \geq 2$), then the parsimonious factor refinement of two integers can be found in Boolean $\mathtt{NC}^{k+1}$.

Related work.

We do not know of any earlier parallel algorithm for the construction of irreducible polynomials. The first deterministic algorithm for the problem appears in [Sho90a]. Probabilistic algorithms are not difficult to construct, since irreducible polynomials are rather abundant, see e.g [Ben81]. The problem of constructing normal bases were considered by [Lün85, BDS90].

Deterministic algorithms for polynomial factorisation appeared already in [Ber67]. [Gat84] shows that the problem of polynomial factorisation over $\mathbf{F}_p$ is in probabilistic pseudo-$\mathtt{NC}^2$. The newer results of [Mul87] and [BKR86, KKS90] actually implies that the probabilistic parts can be made deterministic (by increasing the depth to $O(\log^3(n+p))$).

Factor refinement has been considered by [Lün85, BDS90], who both provide sequential solutions. The combined work of [Gat84, BKR86, KKS90] gives a parallel solution for square free factor refinement that in addition to arithmetic operations assumes the ability to extract p'th roots, when the characteristic of the field is p.

Open questions.

1. Is polynomial factor refinement in arithmetic $\mathtt{NC}^2$?

2. Can we construct irreducible polynomials with normal basis guarantee in arithmetic $\mathtt{NC}^2$?

   - Theorem 3.(1-2) gives an affirmative answer for restricted degrees.

3. Do the problems that we consider have parallel solutions of cost efficiency comparable to the best sequential solutions ?

   - Theorem 4 gives an affirmative answer in a very restricted case.

Arithmetic $\mathtt{NC}^k$.

In this section, we consider arithmetic computations over an arbitrary field $\mathbf{F}$. We summarise some known results about arithmetic $\mathtt{NC}^k$ membership. The results on linear algebra will be used extensively later. In addition we present a new result on parsimonious factor refinement.

Fact 1.

1. Let M be an $n \times n$ matrix over $\mathbf{F}$. Then the problems of computing the determinant, the inverse (if exists) and the rank of M all belong to arithmetic $\mathtt{NC}^2$. (The rank, an integer, is represented in unary.)

2. Let $f_1, f_2, ..., f_k \in \mathbf{F}[x]$ and let n be the degree of $\prod_{i=1}^{k} f_i$. Then the problems of computing $\gcd(f_1, f_2, ..., f_k)$ and computing the quotient and remainder of the division f_1/f_2 both belong to arithmetic $\mathtt{NC}^2$.

The computation of determinant and inverse was proven to be in arithmetic $\mathtt{NC}^2$ by [Csa76] (for fields of characteristic 0) and by [BGH82] and [Ber84] (for general fields). The latter result was inspired by [VSBR83]. A simple and uniform construction for general fields appeared in [Chi85].

[Mul87] reduced the computation of matrix rank to a determinant computation.

Fast parallel circuits for polynomial division were presented in [Ebe89]. For our purposes a reduction to matrix determinant and inversion suffices [Gat84].

The problem of computing the gcd of two polynomials was reduced to matrix inversion/determinant in [BGH82] and the computation of the gcd of many polynomials was reduced to the computation of matrix rank in [Gat84] combined with [BGH82].

Theorem 1.

The problem of parsimonious factor refinement is in arithmetic $\mathtt{NC}^3$.

Proof

We call a set of polynomials F *relatively square free*, if the parsimonious refinement of $F = \{f_1, ...f_k\}$ into $G = \{g_1, ..., g_l\}$, where $f_i = \prod_{j=1}^{l} g_j^{n_{ij}}$ satisfies that $n_{ij} \in \{0, 1\}$ for all i, j.

We shall show that given F then the problem of computing a relatively square free set F' such that F and F' have identical parsimonious refinements belongs to $\mathtt{NC}^3$.

The remaining problem of refining F' into factors that are pairwise prime belongs to $\mathtt{NC}^3$ by the algorithm in [KKS90]. The latter algorithm is presented in a context where it gets only square free inputs, but it works also for a relatively square free set of inputs.

Consider the following parallel algorithm. When the algorithm is input a set $F = F_0 \subseteq \mathbf{F}[x]$ it will iteratively compute $F_1, F_2, ...$ where F_{i+1} is identical to F_i except that zero or more polynomials in F_i have been split into two or more factors in F_{i+1}. No factors are lost, i.e. if a polynomial can be expressed as a product of powers of elements in F_i then the same is true for F_{i+1}.

Algorithm 1

input: $F = \{f_1, f_2, ..., f_k\}$

output: $H = \{h_1, h_2, ..., h_l\}$ such that

i. H is relatively square free.

ii. The parsimonious refinements of F and H are identical.

method:

```
procedure rsq($F$);
    $F_0 := F$;
    $i := 0$;
    repeat
        $F_{i+1} := \cup_{e \in F_i} \text{split}(e, F_i)$;
        $i := i + 1$;
    until $F_i = F_{i+1}$;
    RETURN($F_i$);
end;
```

where

```
procedure split($e, \{f_1, f_2, ..., f_k\}$);
    m := deg($e$);
    for $i \in [1, ..., k]$ in parallel do
        $d_i := \mathrm{gcd}(e, f_i) \cdot e/\mathrm{gcd}(e, f_i^m)$;
    od;
    $c := \mathrm{gcd}(d_1, d_2, ..., d_k)$;
    for $i \in [0, ..., m]$ in parallel do
        $b_i := \mathrm{gcd}(e, c^i)$;
    od;
    for $i \in [1, ..., m]$ in parallel do
        $a_i := b_i/b_{i-1}$;
    od;
    RETURN($\{a_1, a_2, ..., a_m\} \setminus \{1\}$);
end;
```

We shall argue that the algorithm is partially correct. Let $\{g_1, ..., g_l\}$ be the parsimonious factor refinement of F. Define $N_{ij} = \{n | \exists f \in F_i.\ g_j \text{ divides } f \text{ precisely } n \geq 1 \text{ times}\}$. We can express the correct termination of the algorithm by the condition

(a) $F_{i+1} = F_i$ if and only if $N_{ij} = \{1\}$ for all j.

To prove (a), we follow the execution of a single phase in detail.

Consider the call split$(e, \{f_1, ..., f_k\})$ and assume $e = \prod g_j^{m_j}$, $f_i = \prod g_j^{n_{ij}}$. In the first step, we compute

$$d_i = \prod_{\{j | n_{ij} = 0\}} g_j^{m_j} \cdot \prod_{\{j | n_{ij} > 0\}} g_j^{\min\{m_j, n_{ij}\}}$$

Let $n_j = \min_{\{i | n_{ij} > 0\}} \{n_{ij}\}$. Then the next step computes

$$c = \prod_{\{j | m_j \neq 0\}} g_j^{n_j}$$

If n_j is less than m_j for some j's, e will be split into nontrivial factors:

$$b_i = \prod_j g_j^{\min\{m_j, i n_j\}}$$

and

$$a_i = \prod_{\{j|m_j \geq in_j\}} g_j^{n_j} \cdot \prod_{\{j|in_j > m_j \geq (i-1)n_j\}} g_j^{m_j \bmod n_j}$$

This combined with $\gcd(N_{ij}) = 1$ for all i, j proves (a), and we have demonstrated the partial correctness of the program. In addition, we have also proved

(b) $N_{i+1,j} = \{n \bmod \min(N_{ij}) | n \in N_{ij}\} \cup \{\min(N_{ij})\} \setminus \{0\}$.

¿From (b) it follows that if $N_{ij} \neq N_{i+1,j} \neq N_{i+2,j}$ then $\min(N_{ij}) \geq \min(N_{i+1,j}) + \min(N_{i+2,j})$. By induction, we may thus prove that if k iterative calls of *split* are needed before $N_{ij} = \{1\}$ then $n \geq \min(N_{0,j}) \geq \phi_k$, where ϕ_k is the kth Fibonacci number. Since ϕ_k grows exponentially in k, we conclude that the algorithm stops within $O(\log(n))$ iterative calls of *split*. Each execution of *split* can be handled in depth $O(\log^2(n))$ by fact 1. This proves the depth bound of the theorem.

□

Algorithm 1 can also be used for computation on integers in the usual binary representation. We need only replace each reference to *degree* with a reference to *number-of-bits*. Since division and iterated multiplication of integers is known to be in Boolean $\mathtt{NC}^2$ (If $\mathtt{P}$-uniformity suffices then circuits of depth $O(\log(n))$ and size $n^{O(1)}$ do the job [BCH86]), we have in fact proved:

Corollary 1.

If the problem of computing the gcd of a set of integers is in Boolean $\mathtt{NC}^k$ $(k \geq 2)$, then the problem of computing the parsimonious factor refinement of a set of integers is in Boolean $\mathtt{NC}^{k+1}$.

Representations of Finite Extension Fields.

In this section, we establish the mathematical preliminaries for the algorithmic constructions in the next section. Our construction of irreducible polynomials will be divided into cases according to the multiplicative structure of the degree wanted. The basic idea is borrowed from [Sho90a]. He considers 3 distinct cases:

1. Combining irreducible polynomials of prime power degrees $n_1, n_2, \ldots$, n_k into a single irreducible polynomial of composite degree $n = \prod_{i=1}^{k} n_i$.
2. Constructing irreducible polynomials of prime power degree $n = p^r$, where p is the characteristic of the field.
3. Constructing irreducible polynomials of prime power degree $n = q^r$, where $q \neq p$.

Shoup was not concerned with normal bases. By changing the details of the constructions in the cases (1) and (2), we make it possible to give (preserve) the normal basis guarantee, with no extra computational cost. We have not succeeded to do so in case (3). Instead, we shall call upon the standard method for constructing normal bases from polynomial bases. A major part of case (3) is the construction of field elements that are q'th nonresidues. The method used by [Sho90a] requires iterative polynomial factorisation. Since we have not been able to bound the number of iterations sufficiently, we present a different construction, which in fact is also due to Shoup [Sho90b].

Definitions.

Let p be a prime and let n be a positive integer. We let $\mathbf{F}_{p^n}$ denote the unique degree n extension field over the prime field $\mathbf{F}_p$.

1. Let $f_n \in \mathbf{F}_p[x]$ be irreducible of degree n. Let α be a root of $\mathbf{F}_p$. Then $\mathbf{F}_p[x]/(f_n)$, $\mathbf{F}_p(\alpha)$ and $\mathbf{F}_{p^n}$ are all isomorphic.

   The elements $1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^{n-1}$ are linearly independent and form a *polynomial basis* for $\mathbf{F}_{p^n}$ (over $\mathbf{F}_p$).

   If the elements $\alpha, \alpha^p, \alpha^{p^2}, \ldots, \alpha^{p^{n-1}}$ (all the roots of f_n) are linearly independent, then they form a *normal* basis for $\mathbf{F}_{p^n}$ (over $\mathbf{F}_p$).

   Every finite field $\mathbf{F}_{p^n}$ has at least one normal basis.

2. If $f \in \mathbf{F}_p[x]$ then let $\hat{f} \in \mathbf{F}_p[x]$ be the *linearised* polynomial defined by $\hat{f}(x) = \sum_j c_j x^{p^j}$, when $f(x) = \sum_j c_j x^j$. It is the case that $\widehat{fg} = \hat{f} \circ \hat{g} = \hat{g} \circ \hat{f}$.

   For $\alpha \in \mathbf{F}_{p^n}$ let $\mu_\alpha \in \mathbf{F}_p[x]$ be the minimum degree polynomial such that $\hat{\mu_\alpha}(\alpha) = 0$. If $\beta \in \mathbf{F}_{p^n}$ then $\mu_\beta(x)$ divides $x^n - 1$ and β generates a normal basis precisely, when $\mu_\beta(x) = x^n - 1$.

3. Let $a \in \mathbf{F}_{p^n}$.

   Let q be a prime. a is a q'th *nonresidue* (in $\mathbf{F}_{p^n}$) if the equation $x^q - a = 0$ has no solution over $\mathbf{F}_{p^n}$.

   a is a *primitive root* (for $\mathbf{F}_{p^n}$) if the order of a in the multiplicative group $\mathbf{F}^*_{p^n}$ is $p^n - 1$. It is the case that a is a primitive root if and only if a is a q'th nonresidue for all q that divides $p^n - 1$.

These definitions with many results can all be found in the encyclopedic exposition of [LiNi83].

Fact 2.

Let p be a prime and let n be a positive integer.

1. Let $n = n_1 \cdot n_2$, where $\gcd(n_1, n_2) = 1$ and let $\mathbf{F}_{p^{n_1}} \cong \mathbf{F}_p(\alpha_1)$ and $\mathbf{F}_{p^{n_2}} \cong \mathbf{F}_p(\alpha_2)$. Then $\mathbf{F}_{p^n} \cong \mathbf{F}_p(\alpha)$, where $\alpha = \alpha_1 + \alpha_2$.

2. Let $\alpha_0 = 1$ and let α_r be a root of the polynomial $x^p - x - \prod_{i=0}^{r-1} \alpha_i^{p-1}$. Then $\mathbf{F}_{p^{p^r}} \cong \mathbf{F}_p(\alpha_r)$.

3. Let q be a prime and r a positive integer ($q \neq p$).

   If $q = 2$ and $p = 3 \bmod 4$ then let $m = 2$, otherwise ($q \neq 2$ or $p = 1 \bmod 4$) let m be the order of p modulo q.

   Let α be any qth nonresidue in $\mathbf{F}_{p^m}$, let β be a root of $x^{q^r} - \alpha$ and let $\gamma = \sum_{i=0}^{m-1} \beta^{p^i \cdot q^r}$, then $\mathbf{F}_{p^m} \cong \mathbf{F}_p(\alpha)$, $\mathbf{F}_{p^{m \cdot q^r}} \cong \mathbf{F}_p(\beta)$, and $\mathbf{F}_{p^{q^r}} \cong \mathbf{F}_p(\gamma)$.

4. Let q, m be as in (3). Define s by $p^m - 1 = q^s \cdot t$, where q does not divide t.

Let $f_1(x), ..., f_s(x) \in \mathbf{F}_p[x]$ satisfy that $f_1(x)$ is an irreducible factor of $x^{q-1} + x^{q-2} + ... + x + 1$ and $f_k(x)$ is an irreducible factor of $f_{k-1}(x^q)$ for $2 \leq k \leq s$.

Then $f_k(x)$ has degree m (with the exception that $f_1(x) = x + 1$ when $q = 2$ and $p = 3 \bmod 4$) and any root α of $f_k(x)$ satisfies that $\alpha^{q^k} = 1$ and $\alpha^{q^{k-1}} \neq 1$ for $1 \leq k \leq s$. If α is a root of $f_s(x)$, then α is a q'th nonresidue in $\mathbf{F}_p(\alpha) \cong \mathbf{F}_{p^m}$.

5. Let $f \in \mathbf{F}_p[x]$ have degree n and assume f is the product of k distinct irreducible factors each of which has degree d, $f = \prod_{j=1}^k g_j$. Define $c \in \mathbf{F}_p[x][y]$ by $c(y) = \prod_{i=1}^d (y - x^{p^i})$ and define $h_0, h_1, ..., h_{d-1} \in \mathbf{F}_p[x]$ by $c(y) = y^d + \sum_{i=0}^{d-1} h_i y^i$. Then $\{g_1, g_2, ..., g_m\} =$ parsimonious-factor-refinement$(\cup_{i=0}^{d-1} \cup_{c \in \mathbf{F}_p} \{\gcd(f, h_i - c)\})$.
6. Let α be given such that $\mathbf{F}_p(\alpha) \cong \mathbf{F}_{p^n}$. For $i = 0, 1, 2, ..., n-1$, let $f_i = \mu_{\alpha^i}$. Let $\{g_1, g_2, ..., g_k\} =$ parsimonious-factor-refinement$(\{f_0, f_1, ... f_{n-1}\})$, and let n_{ij} be defined by $f_i = \prod_{j=1}^k g_j^{n_{ij}}$. For $j = 1, 2, ..., k$ choose $b(j) \in \{0, ..., n-1\}$ such that $n_{b(j),j} = \max_i\{n_{ij}\}$. Let $h_j = f_{b(j)}/g_j^{n_{b(j),j}}$, then $\beta = \sum_{j=1}^k \hat{h}_j(\alpha^{b(j)})$ generates a normal basis for $\mathbf{F}_p(\beta) \cong \mathbf{F}_p(\alpha)$.

Fact 2.(1)-(4) are proved in [Sho90a] where these and fact 2.(5) are used for the first efficient deterministic construction of irreducible polynomials.

Fact 2.(2) originates earlier in [AdLe86].

Fact 2.(5) is proved in [Sho90c] where it is used for one of the most efficient deterministic factorisation algorithms. The idea of constructing *separation* sets that allow complete factorisation by a final factor refinement goes back to [Ber67].

Fact 2.(6) have been used by several sources for constructing normal bases [Lün85, BDS90].

To support parallelism and to some extent avoid factor refinement in our construction of normal bases (fact 2.(6)), we also use the following results:

Theorem 2.

Let p be a prime and let n be an integer.

1. Let $n = n_1 \cdot n_2$, where $\gcd(n_1, n_2) = 1$ and let $\mathbf{F}_{p^{n_1}} \cong \mathbf{F}_p(\alpha_1)$ and $\mathbf{F}_{p^{n_2}} \cong \mathbf{F}_p(\alpha_2)$. Then $\mathbf{F}_{p^n} \cong \mathbf{F}_p(\alpha)$, where $\alpha = \alpha_1 \cdot \alpha_2$. If α_1, α_2 generate normal bases for $\mathbf{F}_p(\alpha_1)$ and $\mathbf{F}_p(\alpha_2)$ respectively, then α generates a normal basis for $\mathbf{F}_p(\alpha)$.

2. Let $\alpha_0 = 1$ and let α_r be a root of the polynomial $x^p - \alpha_{r-1}^{p-1} x - 1$ for $r \geq 1$. Then $\mathbf{F}_{p^{p^r}} \cong \mathbf{F}_p(\alpha_r)$.

   Let $\beta_r = \alpha_r^{-1}$ and let $\gamma_r = \alpha_r^{p-1}$. Then β_r and γ_r both generate normal bases for $\mathbf{F}_{p^{p^r}} \cong \mathbf{F}_p(\beta_r) \cong \mathbf{F}_p(\gamma_r)$. In addition β_r is a root of $x^p + (\sum_{i=0}^{r-1} \beta_i) x^{p-1} - 1$ and γ_r is a root of $x^p + \sum_{i=1}^{p-1} (-1)^i \gamma_{r-1}^{p-i} x^i - 1$.

3. Let α be given such that $\mathbf{F}_{p^n} \cong \mathbf{F}_p(\alpha)$. Let q be a prime such that q divides $p^n - 1$. If l satisfies that $p^l > n^2$ then the set $\{\alpha^l + \sum_{i=0}^{l-1} c_i \alpha^i | c_i \in \mathbf{F}_p\}$ contains a q'th nonresidue.

Proof

1. $\mathbf{F}_p(\alpha) \cong \mathbf{F}_{p^{n_1 \cdot n_2}}$: Assume that $\mathbf{F}_p(\alpha)$ is contained in some proper subfield $\mathbf{F}_{p^{n_1 \cdot n_2 / r}}$ (r prime) of $\mathbf{F}_{p^{n_1 \cdot n_2}}$. without loss of generality, we can assume that $r | n_1$. Hence both α_2 and $\alpha_1 \cdot \alpha_2$ (and therefore also α_1) is in $\mathbf{F}_{p^{n_1 \cdot n_2 / r}}$ contradicting the fact that $\mathbf{F}_p(\alpha_1, \alpha_2) \cong \mathbf{F}_{p^{n_1 \cdot n_2}}$ (because $\gcd(n_1, n_2) = 1$).

   α generates a normal basis: We must prove that $\sum_{i=0}^{n-1} c_i \alpha^{p^i} = 0$ implies that $c_i = 0$ for all i. We represent the numbers $i \in \{0, 1, 2, ..., n-1\}$ as $i = (j, k)$, where $j = i \bmod n_1$, $k = i \bmod n_2$ and the equation becomes $0 = \sum_{j=0}^{n_1-1} \sum_{k=0}^{n_2-1} c_{(j,k)} (\alpha_1 \alpha_2)^{p^{(j,k)}} = \sum_{j=0}^{n_1-1} (\sum_{k=0}^{n_2-1} c_{(j,k)} \alpha_2^{p^k}) \alpha_1^{p^j}$. We may conclude that $c_{(j,k)} = 0$ for all (j, k) if (i) α_1 generates a normal basis for $\mathbf{F}_p(\alpha_1, \alpha_2)$ over $\mathbf{F}_p(\alpha_2)$ and if (ii) α_2 generates a normal basis for $\mathbf{F}_p(\alpha_2)$ over $\mathbf{F}_p$. (ii) is given by assumption, so we need only worry about (i).

   Proof of (i): We must prove that the elements of $N = \{\alpha_1^{p^{in_2}} | i = 0, 1, ..., n_1 - 1\} = \{\alpha_1^{p^i} | i = 0, 1, 2, ..., n_1 - 1\}$ are linearly independent over $\mathbf{F}_p(\alpha_2)$. By assumption N is a a normal basis over $\mathbf{F}_p$. Hence the polynomial basis $P = \{1, \alpha_1, \alpha_1^2, ..., \alpha_1^{n_1 - 1}\}$ can be expressed as an linear combination of the elements of N. But P is a basis for $\mathbf{F}_p(\alpha_1, \alpha_2)$ over $\mathbf{F}_p(\alpha_2)$ and therefore, so is N.

2. The proof will use induction on r.

   $\mathbf{F}_p(\alpha_r) \cong \mathbf{F}_{p^r}$: By definition α_r is a root of $x^p - \alpha_{r-1}^{p-1}x - 1$, which implies that $\alpha_r^p = \alpha_{r-1}^{p-1} \cdot \alpha_r + 1$. By induction on k, we find that $\alpha_r^{p^k} = \alpha_r \cdot \alpha_{r-1}^{p^k-1} + \sum_{i=1}^k \alpha_{r-1}^{p^k-p^i} = \alpha_{r-1}^{p^k}(\alpha_r \cdot \alpha_{r-1}^{-1} + \sum_{i=1}^k (\alpha_{r-1}^{-1})^{p^i})$. If we can prove that $k = p^r$ is the smallest k for which the equation $\alpha_r^{p^k} = \alpha_r$ is possible, then we will have proved that $x^p - \alpha_{r-1}^{p-1}x - 1$ is irreducible over $\mathbf{F}_p(\alpha_{r-1})$ and $\mathbf{F}_p(\alpha_r) \cong \mathbf{F}_{p^{p^r}}$. First, we deduce that such a minimal k must be a multiple of p^{r-1}. This follows because $\gamma_{r-1} = (\alpha_r^p - 1)/\alpha_r$ and by inductive assumption $\mathbf{F}_{p^{p^{r-1}}} \cong \mathbf{F}_p(\gamma_{r-1})$. We know that $\alpha_{r-1}^{p^{p^{r-1}}} = \alpha_{r-1}$ and $\sum_{i=1}^{p^{r-1}} (\alpha_{r-1}^{-1})^{p^i} = c \neq 0$, since $\alpha_{r-1}^{-1} = \beta_{r-1}$ generates a normal basis for $\mathbf{F}_{p^{p^{r-1}}}$ by inductive assumption. Hence, $\alpha_r^{p^{p^{r-1}}} = \alpha_r + c\alpha_{r-1}$ and $\alpha_r^{p^{jp^{r-1}}} = \alpha_r + jc\alpha_{r-1}$, where the term $(jc\alpha_{r-1})$ does not vanish unless j is a multiple of p.

   $\mathbf{F}_p(\beta_r) \cong \mathbf{F}_p(\gamma_r) \cong \mathbf{F}_{p^{p^r}}$: Clearly, $\mathbf{F}_p(\beta_r) \cong \mathbf{F}_p(\alpha_r)$. In addition, note that $\alpha_r^p - \alpha_{r-1}^{p-1}\alpha_r - 1 = 0$ implies that $\alpha_r^{p-1} = \alpha_{r-1}^{p-1} + \alpha_r^{-1}$. Hence, $\alpha_r^{p-1} = \sum_{i=0}^r \alpha_i^{-1}$, and we find that $\beta_r = \alpha_r^{-1}$ is a root of the polynomial $x^p + (\sum_{i=0}^{r-1} \beta_i)x^{p-1} - 1$. The equation $\gamma_r = \gamma_{r-1} + \beta_r$ also implies that $\mathbf{F}_p(\gamma_r) \cong \mathbf{F}_{p^{p^r}}$. To find the minimal polynomial for γ_r, we note that $\gamma_r^p = \gamma_{r-1}^p + \beta_r^p = \gamma_{r-1}^p + 1 - \gamma_{r-1}\beta_r^{p-1} = \gamma_{r-1}^p + 1 - \gamma_{r-1}(\gamma_r - \gamma_{r-1})^{p-1} = \gamma_{r-1}^p + 1 - \gamma_{r-1}\sum_{i=0}^{p-1}(-1)^i \gamma_{r-1}^{p-1-i}\gamma_r^i$. When reorganising terms, we find that γ_r is a root of $x^p + \sum_{i=1}^{p-1}(-1)^i\gamma_{r-1}^{p-i}x^i - 1$.

   β_r and γ_r both generate normal bases: In the fields, we work with, normal basis generators can be characterised in a very simple way, viz. $\delta \in \mathbf{F}_{p^{p^r}}$ generates a normal basis for $\mathbf{F}_{p^{p^r}}$ if and only if $\sum_{i=0}^{p^r-1} \delta^{p^i} \neq 0$. To see the truth of this, we use that $\mu_\delta(x)$ divides $x^{p^r} - 1$. But $x^{p^r} - 1 = (x-1)^{p^r}$. Let $g(x) = (x-1)^{p^r-1} = \sum_{i=0}^{p^r-1} x^i$, then δ generates a normal basis if and only if μ_δ does not divide g if and only if $\hat{g}(\delta) \neq 0$.

   Next observe that $\sum_{i=0}^{p^r-1} \gamma_{r-1}^{p^i} = p \cdot \sum_{i=0}^{p^{r-1}-1} \gamma_{r-1}^{p^i} = 0$, which implies that $\sum_{i=0}^{p^r-1} \gamma_r^{p^i} = \sum_{i=0}^{p^r-1} \beta_r^{p^i}$ and we need only prove that β_r generates a normal basis. $\beta_r, \beta_r^{p^{p^{r-1}}}, \beta_r^{p^{2\cdot p^{r-1}}}, ..., \beta_r^{p^{(p-1)\cdot p^{r-1}}}$ are the p distinct roots of the polynomial $x^p + \gamma_{r-1}x^{p-1} - 1$. Hence, $\sum_{i=0}^{p-1} \beta_r^{p^{i\cdot p^{r-1}}} = -\gamma_{r-1}$ and $\sum_{i=0}^{p^r-1} \beta_r^{p^i} = -\sum_{i=0}^{p^{r-1}-1} \gamma_{r-1}^{p^i} \neq 0$ by inductive assumption.

3. In [Sho90b] it is proved that for l satisfying $p^l > c \cdot n^6 \cdot \log^4(p)$ the set

$\{\alpha^l + \sum_{i=0}^{l-1} c_i\alpha^i | c_i \in \mathbf{F}_p\}$ contains a primitive root for $\mathbf{F}_p(\alpha)$, where c is a universal constant. However, when we merely want a q'th nonresidue one may relax the lower bound on l to $p^l > n^2$.

Define $A_l \subseteq \mathbf{F}_p[x]$ to be all monic polynomials of degree l. Let $B_l = \{f(\alpha)|f \in A_l\}$. If $l \geq n$ then $B_l = \mathbf{F}_p(\alpha)$, and clearly contains a q'th nonresidue. Hence, we may assume that $l < n$: In that case $B_l \subseteq \mathbf{F}_p(\alpha)^*$ and we can split B_l in q'th powers $B_{l,q}$ and q'th nonresidues $B_l - B_{l,q}$.

Let $\chi : \mathbf{F}_p(\alpha)^* \to \mathbf{C}$ be a multiplicative character of order q, then

$$\frac{1}{q}\sum_{i=0}^{q-1} \chi^i(\beta) = \begin{cases} 1 & \text{for } \beta \in B_{l,q} \\ 0 & \text{for } \beta \in B_l \setminus B_{l,q} \end{cases}$$

For $\beta \in B_l$, let $f_\beta \in A_l$ be the unique polynomial such that $f_\beta(\alpha) = \beta$. Define $\Lambda(\beta)$ to be the degree of g if f_β is a power of some irreducible $g \in \mathbf{F}_p[x]$ and 0 otherwise. Define $S(l) = \sum_{\beta\in B_l} \Lambda(\beta)$ and $S_q(l) = \sum_{\beta\in B_{l,q}} \Lambda(\beta)$. Then $S(l) = p^l$ and $S_q(l) = \sum_{\beta\in B_l}(\frac{1}{q}\sum_{i=0}^{q-1}\chi^i(\beta))\Lambda(\beta) = \frac{1}{q}S(l) + \frac{1}{q}\sum_{i=1}^{q-1}(\sum_{\beta\in B_l}\chi^i(\beta)\Lambda(\beta))$.

[Sho90b] proves that

(a) $|\sum_{\beta\in B_l} \chi(\beta)\Lambda(\beta)| \leq (n-1)\sqrt{p^l}$ for $l \geq 1$.

Hence, we get $S(l) - S_q(l) \geq \frac{q-1}{q}p^l - \frac{q-1}{q}(n-1)\sqrt{p^l} = \frac{q-1}{q}\sqrt{p^l}(\sqrt{p^l} - (n-1))$, and $B_l - B_{l,q}$ is nonempty, when $\sqrt{p^l} > (n-1)$ and $l \geq 1$.

□

Fact 2.(4) shows how to construct a q'th nonresidue by s iterative factorisations. The best bound on s in terms of p and n that we have managed to prove is the trivial one ($a < \log(p) \cdot n/\log(n)$). That is not good enough for a pseudo-NC-algorithm, and theorem 2.(3) will be the basis of our construction of q'th nonresidues.

The direct normal basis construction of Theorem 2.(1)-(2) will turn out to allow $\mathtt{NC}^2$-networks, whereas the reduction of normal basis construction to factor refinement will only be pseudo-$\mathtt{NC}^2$ and factor refinement requires $\mathtt{NC}^3$ networks to our best knowledge.

Arithmetic pseudo-$\mathtt{NC}^k$.

In this section, we prove the main result of the paper. As far as possible, we try to make our constructions in arithmetic NC rather than pseudo-NC, and we do succeed in two of the three subcases arising when constructing irreducible polynomials (theorem 3.(1-2)). The subproblems that seem to require the power of pseudo-NC are modular polynomial exponentiation, polynomial factorisation and construction of q'th nonresidues.

Contrary to our earlier resolution to disregard processor efficiency, we exhibit a very processor efficient construction of degree 2^r polynomials over $\mathbf{F}_2$.

Definitions.

Let p be a prime.

1. *Modular Polynomial Exponentiation* is the following problem: Given $f, g \in \mathbf{F}_p[x]$ and $r \in \mathbf{N}$, where f has degree n, g has degree $< n$ and $r < p^n$ then compute $(g^r \bmod f)$.

Fact 3.

1. Modular Polynomial Exponentiation is in pseudo-$\mathtt{NC}^2$.

[FiTo88] proves that polynomial modular exponentiation is in Boolean $\mathtt{NC}^2$, provided that p is bounded by $n^{O(1)}$. Their construction does, however, lead to an arithmetic pseudo-$\mathtt{NC}^2$ network when p is unbounded.

Theorem 3.

Let p be a prime and let n be a positive integer.

1. Let $n = \prod_{i=1}^{k} n_i$, where $\gcd(n_i, n_j) = 1$ for $i \neq j$. The problem of constructing an irreducible $f \in \mathbf{F}_p[x]$ of degree n, when given irreducible $f_1, f_2, ..., f_k \in \mathbf{F}_p[x]$ of degrees $n_1, n_2, ..., n_k$ respectively is in arithmetic $\mathtt{NC}^2$.

   If the roots of $f_1, f_2, ..., f_k$ are known to form normal bases for their respective field extensions, then so will the roots of f.

2. Let $n = p^r$. The problem of constructing an irreducible $f \in \mathbf{F}_p[x]$ of degree p^r is in arithmetic $\texttt{NC}^2$.

   The roots of f is guaranteed to form a normal basis for $\mathbf{F}_p[x]/(f)$.

3. Let $n = q^r$, where $q \neq p$ is a prime. The problem of constructing an irreducible $f \in \mathbf{F}_p[x]$ of degree q^r pseudo-$\texttt{NC}^2$-reduces to the problem of polynomial factorisation.

4. The problem of polynomial factorisation pseudo-$\texttt{NC}^2$-reduces to the problem of parsimonious polynomial factor refinement.

5. The problem of constructing an irreducible $f \in \mathbf{F}_p[x]$ of degree n, such that the roots of f form a normal basis for $\mathbf{F}_{p^n}$, pseudo-$\texttt{NC}^2$-reduces to the problem of parsimonious factor refinement.

6. The problem of irreducible polynomial construction lies in pseudo-$\texttt{NC}^{2.5}$.

Proof:

1. We shall use theorem 2.(1). Assume that we have irreducible polynomials $f_1, ..., f_k$ of degrees $n_1, ..., n_k$ with roots $\alpha_1, ..., \alpha_k$. Let $\alpha = \prod_{i=1}^k \alpha_i$. We know that $\mathbf{F}_p(\alpha_1, \alpha_2, ...\alpha_i)$ is a degree n_i extension over $\mathbf{F}_p(\alpha_1, \alpha_2, ..., \alpha_{i-1})$ since $\gcd(n_i, n_j) = 1$ for $i \neq j$. Hence, the set $M = \{\alpha_1^{l_1} \alpha_2^{l_2} \cdot ... \cdot \alpha_k^{l_k} | l_i = 0, 1, 2, ..., n_i - 1 \text{ for } i = 1, 2, ..., k\}$ forms a basis for $\mathbf{F}_p(\alpha)$ over $\mathbf{F}_p$. We want to find the unique monic polynomial $f(x) = x^n + \sum_{i=0}^{n-1} c_i x^i$ such that $f(\alpha) = 0$. The coefficients c_i can be found in arithmetic $\texttt{NC}^2$ by solving a system of linear equations, if we know the representation of $1, \alpha, \alpha^2, ..., \alpha^n$ in terms of the basis M. We note that $\alpha^i = \alpha_1^i \alpha_2^i ... \alpha_k^i$. The representation of each α_j^i in terms of the polynomial basis $\{1, \alpha_j, \alpha_j^2, ..., \alpha_j^{n_j - 1}\}$ is found in arithmetic $\texttt{NC}^2$ by a single modulo operation. Finally, we multiply out (also in arithmetic $\texttt{NC}^2$) to find the representation of α^i in terms of M.

2. Let α_r be defined as in theorem 2.2. We wish to find the unique monic $g \in \mathbf{F}_p[x]$ of degree p^r such that $g(\alpha_r) = 0$. We shall later show how to get a polynomial f with normal basis guarantee from g. We know that $\mathbf{F}_p(\alpha_k)$ is a degree p extension over $\mathbf{F}_p(\alpha_{k-1})$ for all $k \geq 1$. Hence the set $M = \{\alpha_1^{l_1} \alpha_2^{l_2} \cdot ... \cdot \alpha_r^{l_r} \mid l_k = 0, 1, 2, ..., p - 1 \text{ for } k = 1, 2, ..., r\}$ forms a basis for $\mathbf{F}_p(\alpha_r)$ over $\mathbf{F}_p$. If we can find the representation of

$1, \alpha_r, \alpha_r^2, ..., \alpha_r^{p^r}$ in terms of M, then we may find the coefficients of g in arithmetic $\mathtt{NC}^2$ by the strategy used in the proof of part (1) of this theorem.

We still need to show how to find the representation of α_r^l in terms of M. Let $l = \sum_{i=0}^{r} a_i p^i$ be the base p representation of the integer $l < p^{r+1}$. Then

(a) $\alpha_k^l = \prod_{i=0}^{r} (\alpha_k^{p^i})^{a_i}$.

In the proof of theorem 2.(2) we found that $\alpha_k^{p^i} = \alpha_k \cdot \alpha_{k-1}^{p^i-1} + \sum_{j=1}^{i} \alpha_{k-1}^{p^i-p^j}$ for $i \geq 0$. We may insert this expression into (a) and in depth $O(\log^2(n))$ multiply out to find numbers $c_{ij} \in \mathbf{F}_p$ such that

(b) $\alpha_k^l = \sum_{i=0}^{T(l)} \sum_{j=0}^{l-i} c_{ij} \alpha_k^i \alpha_{k-1}^j$, where $T(l) = \sum_{i=0}^{r} a_i$.

By using (b) for a substitution into the right hand side of it self, and possibly repeating this process a few times, we find numbers $d_{ij} \in \mathbf{F}_p$ such that

(c) $\alpha_k^l = \sum_{i=0}^{p-1} \sum_{j=0}^{l-i} d_{ij} \alpha_k^i \alpha_{k-1}^j$.

The computation of (c) from (b) takes at most depth $O(\log^*(n) \log(n))$, since $T(T(T(...T(l)...))) \leq p-1$ for at most $O(\log^*(n))$ iterations of T.

For $k = 1, 2, ..., r$ we iteratively find expressions for α_k^l in the base $\{\alpha_1^{l_1} \alpha_2^{l_2} \cdot ... \cdot \alpha_k^{l_k} \mid l_1, l_2, ..., l_k = 0, 1, 2, ..., p-1\}$. In the beginning, we simply insert $\alpha_0 = 1$ into (c) to get the expression for α_1^l by collecting terms. Each step in the iteration consists in substituting the result from the last step into the right hand side of (c) and multiplying out. There are r steps, each of which can be done in depth $O(\log(n))$. Hence, the complete construction of the irreducible polynomial g is in arithmetic $\mathtt{NC}^2$.

Our remaining concern is the normal basis guarantee. g has the root α_r. We want to find f that has the root $\beta_r = \alpha_r^{-1}$. If $g(x) = x^n + \sum_{i=0}^{n-1} b_i x^i$ then $f(x) = x^n + \sum_{i=0}^{n-1} (b_{n-i}/b_0) x^i$.

3. Next, let us construct a polynomial of degree q^r, where $q \neq p$. Find m according to fact 2.(3). Find $f_1(x)$ as an irreducible factor of $x^{q-1} +$

$... + x + 1$ using polynomial factorisation. Let $f(x) = f_1(x)$ (except for $q = 2$ and $p = 3 \bmod 4$ in which case $f(x) = f_2(x) = x^2 + 1$). Then $f(x)$ has degree m. Let α be a root of f. An element $\beta \neq 0$ in $\mathbf{F}_p(\alpha)$ is a q'th nonresidue if and only if $\beta^{(p^m-1)/q} \neq 1$. Hence, we compute $((x^l + \sum_{i=0}^{l-1} c_i x^i)^{(p^m-1)/q} \bmod f(x))$ using theorem 2.(3), for all choices of $c_0, c_1, ..., c_{l-1}$. At least one set of coefficients, e.g. $(d_0, d_1, ...d_{l-1})$ leads to a value $\neq 1$. Then $\beta = \alpha^l + \sum_{i=0}^{l-1} d_i \alpha^i$ is a q'th nonresidue. Find the irreducible polynomial $g(x)$ for β by solving a linear system of equations. Let γ be a root of $g(x^{q^r})$ that is irreducible of degree mq^r (fact 2.(3)). Compute $\delta = \sum_{i=0}^{m-1} \gamma^{p^{iq^r}}$ in terms of γ and find the degree q^r minimal polynomial $h(x)$ of δ by solving yet another linear system of equations.

4. Let f have the complete factorisation $f = \prod_{i=1}^{n} \prod_{j=1}^{n_i} g_{ij}^{n_{ij}}$, where g_{ij} is irreducible of degree i and $g_{ij_1} \neq g_{ij_2}$ when $j_1 \neq j_2$. We wish to find the g_{ij}'s.

   We start by making a distinct degree factorisation (an old technique). The polynomial $x^{p^k} - x$ is the product of all irreducible polynomials in $\mathbf{F}_p[x]$ that have a degree dividing k. Hence, compute

   (a) $a_k = \gcd(f, x^{p^k} - x) = \prod_{i|k} \prod_{j=1}^{n_i} g_{ij}$, for $k = 1, 2, ..., n$.

   (b) $b_k = a_k / \gcd(a_k, \prod_{i=1}^{k-1} a_i) = \prod_{j=1}^{n_k} g_{kj}$, for $k = 1, 2, ..., n$.

   In (a) we first compute $a'_k = x^{p^k} \bmod f$ using fact 3.(1) and next we compute $\gcd(f, a'_k - x)$ using fact 1.(2).

   The remaining problem may be characterised as follows: Factor $b = \prod_{j=1}^{m} g_j$, where each $g_j \in \mathbf{F}_p[x]$ is irreducible of degree d and g_i, g_j are distinct if $i \neq j$.

   We use fact 2.(5) and compute:

   (c) $c(y) = (y - x) \cdot (y - x^p) \cdot ... \cdot (y - x^{p^{d-1}}) \bmod b(x) = h_0(x) + h_1(x) \cdot y + h_2(x) \cdot y^2 + ... + h_{d-1}(x) \cdot y^{d-1} + y^d$

   (d) $\{g_1, g_2, ..., g_m\}$ = parsimonious-factor-refinement$(\cup_{i=0}^{d-1} \cup_{c \in \mathbf{F}_p} \{\gcd(b, h_i - c)\})$

5. Parts (1-4) of this theorem shows how the construction of an irreducible polynomial of degree n pseudo-$\texttt{NC}^2$-reduces to the problem of factor refinement. In (3) we give no guarantee that the roots of the constructed polynomial form a normal basis. However, such a guarantee can be extended using factor refinement (and fact 2.(6)):

   Let $f \in \mathbf{F}_p[x]$ be irreducible of degree n. Let α be a root of f. Find $f_i = \mu_{\alpha^i}$ for $i = 0, 1, ..., n-1$. f_i is found by computing $\alpha^i, \alpha^{ip}, ...\alpha^{ip^{n-1}}$ in the polynomial basis $\{1, \alpha, \alpha^2, ..., \alpha^{n-1}\}$ and solving (checking singularity) of suitable systems of linear equations. This is all in pseudo-$\texttt{NC}^2$ by fact 1.(1) and fact 3.(1). Next, we construct $\{g_1, ..., g_k\}$ by factor refinement of $\{f_0, f_1, ..., f_{n-1}\}$. The remaining part of the construction of a normal basis generating element β in terms of α is in pseudo-$\texttt{NC}^2$ (fact 2.6). Finally, find the irreducible polynomial for β.

6. Theorems 3.(1-5) and 1 combined show that the construction of irreducible polynomials with normal basis guarantee is in pseudo-$\texttt{NC}^3$, and it appears that the bottleneck is factor refinement. Actually, factor refinement occurs in only two connections:

   (a) When factoring $x^{q-1} + ... + x + 1$ to find an irreducible $f(x)$ of degree m.

   (b) When constructing a normal basis from a polynomial basis.

   The (b)-bottleneck is easily avoided by dropping the normal basis guarantee and be satisfied with any irreducible polynomial of the proper degree. The (a)-bottleneck could be avoided by making a recursive call of the whole construction algorithm and in this way find an irreducible polynomial of degree m. However, in the worst case the recursion depth could be $O(\log(n))$, and we would still only have a pseudo-$\texttt{NC}^3$ algorithm.

   The trick lies in combining the two ideas, if m is "small", then we make a recursive call, otherwise ($(q-1)/m$ is "small") and it turns out that we can relatively fast find an irreducible factor of degree m.

   To analyse this approach, we will need to know the following:

   (c) If $f \in \mathbf{F}_p[x]$ of degree n is the product of n/m distinct factors, each of degree m, then an irreducible factor of f can be found in depth $O(\log^2(n+p)\cdot\log(n/m))$

To prove (c), we use fact 2.(5), but we stop before doing the factor refinement, when we have computed the set $F=\cup_{i=0}^{m-1}\cup_{c\in\mathbf{F}_p}\{\gcd(f,h_i-c)\}$. F must contain at least one polynomial f_1 of degree d_1, where $m\le d_1\le n/2$. This is because $\prod_{c\in\mathbf{F}_p}\gcd(f,h_i-c)=f$ for any fixed i, and $\cup_{c\in\mathbf{F}_p}\{\gcd(f,h_i-c)\}$, can not be $\{f,1\}$ for all i, if $m\ne n$. Hence, by $\log(n/m)$ recursive steps each of depth $O(\log^2(n+p))$, we can find an irreducible factor of f.

Next we need the following:

(d) The recurrence relation

$T(n)=1+\max_{1\le m\le n}\min\{\log(n/m),T(m)\}$ for $n\ge 2$ and
$T(1)=\frac{1}{2}$.

is satisfied by

$T(n)=\frac{1}{2}+\sqrt{2\log(n)}$ for $n\ge 1$.

For fixed $n\ge 2$, and $1\le m\le n$, this particular solution satisfies

$\log(n/m)$ is decreasing in m,
$T(m)$ is increasing in m and
$\log(n/m)=T(m)$, when $m=n\cdot 2^{\frac{1}{2}-\sqrt{2\log(n)}}$.

Based on (d), we choose a recursive call of our irreducible polynomial construction algorithm, when (i) $m<n\cdot 2^{\frac{1}{2}-\sqrt{2\log(n)}}$ and otherwise (ii) we find an irreducible factor of $x^{q-1}+...+x+1$. From (d) we know that in case (i) $T(n)$ dominates $1+T(m)$ and in case (ii) $T(n)$ dominates $1+\log(n/m)$. Hence, by (c) we conclude that it is possible to find an irreducible polynomial in total depth $O(\log^2(n+p)\cdot T(n))$, which implies the statement, we were to prove.

We still need to prove (d), though. Note that $T(m)$ is an increasing function in m and that $\log(n/m)$ is a decreasing function in m (for n fixed). Hence, $\max_{1\le m\le n}\min\{\log(n/m),T(m)\}$ is obtained for an m such that $\log(n/m)=T(m)$. It is now a routine calculation to verify the correctness of the remaining assertions in (d).

□

We have so far not concerned ourselves with optimising the circuit-size/processor-number in our results. All our constructions so far have used matrix inversion or determinant computations, which are very processor expensive operations [KaRa90]. However, we can improve upon the construction of theorem 3.(2) when $p = 2$:

Theorem 4

1. Over the field $\mathbf{F}_2$ an irreducible polynomial of degree $n = 2^r$ may be constructed with log space uniform arithmetic circuits of depth $O(\log^2(n))$ and size $O(n^2)$.

   The roots of the constructed polynomial form a normal basis for $\mathbf{F}_{2^n}$.

Proof

1. Assume α_r is a root of $x^2 + \alpha_{r-1}x + 1$ and $\alpha_0 = 1$, according to theorem 2.(2). We shall iteratively construct $f_1, f_2, ..., f_r$ such that $f_i = g_i + \alpha_{r-i}h_i$, where $g_i, h_i \in \mathbf{F}_p[x]$. It will be the case that $f_i \in \mathbf{F}_p(\alpha_{r-i})[x]$ has degree 2^i and has the roots $\alpha_r^{2^{j2^{r-i}}}$ for $j = 0, 1, ..., 2^i - 1$. Clearly, $f_1(x) = x^2 + \alpha_{r-1}x + 1$ and $f_i(x) = (g_{i-1}(x) + \alpha_{r-i+1}h_{i-1}(x)) \cdot (g_{i-1}(x) + \alpha_{r-i+1}^{2^{2^{r-i}}}h_{i-1}(x))$. The elementary symmetric functions of $\alpha_{r-i+1}^{2^{j2^{r-i}}}$ ($j = 0, 1$) are the coefficients of the polynomial $x^2 + \alpha_{r-i}x + 1$, i.e. $\sigma_1(...) = \alpha_{r-i}$ and $\sigma_2(...) = 1$. Using these facts, we may compute f_i from f_{i-1} in depth $O(\log(2^i))$. We thus find the coefficients of f_r in arithmetic $\mathtt{NC}^2$. Using that $\alpha_0 = 1$, f_r will in fact be the sought for polynomial.

□

Example

To appreciate the simplicity of the construction in the proof of theorem 4, we give a demonstration:

(a) Let $g_0(x) = x$ and $h_0(x) = 1$.

(b) Compute $g_i(x) = g_{i-1}(x)^2 + h_{i-1}(x)^2$, $h_i(x) = g_{i-1}(x)h_{i-1}(x)$ and $k_i(x) = g_i(x) + h_i(x)$ for $i \geq 1$.

The computation will yield the following sequence of polynomials, where $k_i(x)$ is irreducible over $\mathbf{F}_2$ and the roots of $k_i(x)$ form a normal basis for $\mathbf{F}_{2^i}$:

$$g_1(x) = x^2 + 1$$
$$h_1(x) = x$$
$$k_1(x) = x^2 + x + 1$$

$$g_2(x) = x^4 + x^2 + 1$$
$$h_2(x) = x^3 + x$$
$$k_2(x) = x^4 + x^3 + x^2 + x + 1$$

$$g_3(x) = x^8 + x^6 + x^4 + x^2 + 1$$
$$h_3(x) = x^7 + x$$
$$k_3(x) = x^8 + x^7 + x^6 + x^4 + x^2 + x + 1$$

etc.

□

References

[AdLe86] Adleman, L. M. and Lenstra, Jr., H. W., Finding Irreducible Polynomials over Finite Fields. *Proceedings of the 18'th ACM Symposium on Theory of Computing* (1986), pp. 350-355.

[BDS90] Bach, E., Driscoll, J. and Shallit, J., Factor Refinement. *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms* (1990), pp. 201-211.

[BCH86] Beame, P. W., Cook, S. A. and Hoover, H. J., Log Depth Circuits for Division and Related Problems. *SIAM Journal on Computing* 15 (1986), pp. 994-1003.

[Ben81] Ben-Or, M., Probabilistic Algorithms in Finite Fields. *Proceedings of the 22nd Annual IEEE Symposium on Foundations of Computer Science* (1981), pp. 394-398.

[Ber84] Berkowitz, S. J., On Computing the Determinant in Small Parallel Time Using a Small Number of Processors. *Information Processing Letters* 18 (1984), pp. 147-150.

[BKR86] Ben-Or, M., Kozen, D. and Reif, J., The Complexity of Elementary Algebra and Geometry. *Journal of Computer and System Sciences* 32 (1986), pp. 251-264.

[Ber67] Berlekamp, E. R., Factoring Polynomials over Finite Fields. *Bell System Technical Journal* 46 (1967), pp. 1853-1859.

[BGH82] Borodin, A., von zur Gathen, J. and Hopcroft, J., Fast Parallel Matrix and GCD Computations. *Information and Control* 52 (1982), pp. 241-256.

[Chi85] Chistov, A. L., Fast Parallel Calculation of the Rank of Matrices over a Field of Arbitrary Characteristic. In *Fundamentals of Computation Theory, FCT'85.* Lecture Notes in Computer Science 199 (1985), pp. 63-79.

[Csa76] Csanky, L., Fast Parallel Matrix Inversion Algorithms. *SIAM Journal on Computing* 5 (1976), pp. 618-623.

[Ebe89] Eberly, W., Very Fast Parallel Polynomial Arithmetic. *SIAM Journal on Computing* 18 (1989), pp. 955-976.

[FiTo88] Fich, F. E. and Tompa, M., The Parallel Complexity of Exponentiating Polynomials over Finite Fields. *Journal of the ACM* 35 (1988), pp. 651-667.

[Gat84] von zur Gathen, J., Parallel Algorithms for Algebraic Problems. *SIAM Journal on Computing* 13 (1984), pp. 802-824.

[KaRa90] Karp, R. M. and Ramachandran, V., Parallel Algorithms for Shared-Memory Machines. In *Handbook of Theoretical Computer Science A, Algorithms and Complexity.* Elsevier, 1990.

[KKS90] Kaltofen, E., Krishnamoorthy, M. S. and Saunders, B. D., Parallel Algorithms for Matrix Normal Forms. *Linear Algebra and its Applications* 136 (1990), pp. 189-208.

[LiNi83] Lidl, R. and Niederreiter, H., *Finite Fields.* Encyclopedia of Mathematics and its Applications 20, Addison Wesley, 1983.

[Lün85] Lüneburg, H., On a little but useful algorithm. In Lecture Notes in Computer Science 229 (1985), pp. 296-301.

[Mul87] Mulmuley, K., A Fast Parallel Algorithm to Compute the Rank of a Matrix over an arbitrary field. *Combinatorica* 7 (1987), pp. 101-104.

[Sho90a] Shoup, V., New Algorithms for Finding Irreducible Polynomials over Finite Fields. *Mathematics of Computation* 54 (1990), pp. 435-447.

[Sho90b] Shoup, V., Searching for Primitive Roots in Finite Fields. *Proceedings of 22'nd ACM Symposium on Theory of Computing* (1990), pp. 546-554.

[Sho90c] Shoup, V., On the Deterministic Complexity of Factoring Polynomials over Finite Fields. *Information Processing Letters* 33 (1990), pp. 261-267.

[VSBR83] Valiant, L. G., Skyum, S., Berkowits, S. and Rackoff, C., Fast Parallel Computation of Polynomials using Few Processors. *SIAM Journal on Computing* 12 (1983), pp. 641-644.