Uniform Closure Properties of P-Computable Functions*

Erich Kaltofen

Rensselaer Polytechnic Institute, Dept. of Computer Science
Troy, New York 12181

and

Mathematical Sciences Research Institute
Berkeley, California 94720

Preliminary Report

1. Introduction

Valiant [24] introduced the notion of a family of *p-computable* polynomials as those multivariate polynomials of polynomially-bounded degree and straight-line computation length. He raised the question of whether p-computable families would be closed under natural mathematical operations and showed that this is true for taking repeated partial derivatives in a single variable, whereas by taking repeated partial derivatives in many variables one can obtain the general permanent from a polynomial-sized formula.

In [12] we have introduced straight-line programs as a means of representing polynomials. Therefore our algorithms require that the p-computable outputs can be computed from the p-computable inputs in at least random polynomial-time. We call families satisfying this additional requirement *uniformly* closed. The main result in [12] can now be stated concisely as that every family of p-computable polynomials is uniformly closed under the greatest common divisor operation. It is easy to show that Valiant's closure properties are also uniform. In this paper we establish uniform closure of families of p-computable polynomials for two more important operations, factorization and extracting the numerator and denominator of a rational function.

The factorization problem of polynomials in straight-line representation was first solved for the case in which the factors were to be produced in sparse format [9]. Unfortunately, even sparse polynomials can have factors with super-polynomially more non-zero terms [7] and therefore those algorithms computing the sparse factorization can take more than polynomially-many steps in the input size and degree. Uniform closure for this problem, of course, guarantees that the straight-line representation of the factors can be found in random polynomial-time in the input degree and program length. The key idea of our solution, in addition to the contributions in [7] and [9], is to employ Hensel lifting but to replace the p-adic expansion of the coefficients by the expansions into homogeneous parts of the minor variables. We thus lift all minor variables simultaneously and avoid the variable by variable lifting loop that would compound programs of exponential size.

It is clear that unformity can be only achieved for coefficient fields over which bivariate polynomial factorization is in polynomial-time. As the algorithms in [15] and [10] for rational coefficients might indicate, uniformity is not any more an easy matter. Another sophisticated tool used to establish uniformity are the effective Hilbert irreducibility theorems [6] and [11]. For rational coefficients we can prove even binary random polynomial-time for our algorithm provided the size of the coefficients of the input polynomial is also polynomially bounded. If the coefficient field is of positive characteristic p and the multiplicity of an irreducible factor is divisible by p, there is an additional problem. We can, however, compute a straight-line computation for the appropriate p^k-th power of such a factor.

* This material is based upon work supported by the National Science Foundation under Grant No. DCR-85-04391 and by an IBM Faculty Development Award. Part of work on §3 and §4 was done while the author was visiting the Tektronix Computer Research Laboratory in Beaverton, Oregon.

Let us for a moment come back to the question of factorizing into sparse polynomials. The examples causing super-polynomial blow-up for the size of the answer have the property that many other factors are very sparse. In general, one may wish to retrieve the sparse factors as such and leave the dense factors in straight-line format. Fortunately, Zippel's conversion algorithm (cf [12],. §6) allows to do just that. More precisely, given a bound t we can now probabilistically determine in polynomial-time in t from the straight-line factorization the sparse format of all irreducible factors with no more than t terms, this without any restriction on characteristic and multiplicities. Moreover, the running time is always polynomial even if we were unlucky in our choice of evaluation points. We think that this finally settles the question of sparse factorization in a very satisfactory manner.

We now turn to the computation of numerator and denominator of p-computable rational functions. We note that our definition of such a family requires that there is a polynomial bound for the length of the straight-line computations, which also contain divisions, and a polynomial bound for the degrees of the reduced numerator and denominator of the rational functions computed. Strassen [22] raised the question whether the numerators and therefore also the denominators were p-computable. Here we show that computations of polynomial-length for the numerator and denominator can be found in random polynomial-time and as one consequence also settle this open problem of more than a decade. The main idea for our solution comes mostly from our uniform closure result for GCDs [12] put together with the theory of Padé approximations. Another important consequence of the p-computability of the numerator and denominator of rational functions is that it can be used to parallelize p-computable rational functions in general. First we note that Hyafil [8] and Valiant et al [25]. have shown how to evaluate p-computable polynomials in parallel, that is, in polynomial-size and poly-logarithmic depth. We now can apply this parallelization to our straight-line program for the numerator and denominator and therefore establish that every p-computable rational function can be evaluated in parallel in polynomial size and poly-logarithmic depth.

This paper is organized as follows. Section 2 contains the result on polynomial factorization. Section 3 introduces the properties of Padé approximants used in section 4, which contains the construction for numerator and denominator. Section 5 concludes by raising open questions.

Notation: We use the same notation as in [12] and [9], but for the convenience of the reader we shall repeat it here. By **Q** we denote the the rational numbers and by GF(q) the finite field with q elements. F usually denotes a field and char(F) its characteristic. A polynomial $f \in F[x_1, \ldots, x_n]$ is homogeneous of degree d if

$$f(x_1, \ldots, x_n) = \sum_{e_1+\cdots+e_n=d} c_{e_1,\ldots,e_n} x_1^{e_1} \cdot \cdots \cdot x_n^{e_n},$$

$c_{e_1,\ldots,e_n} \in F$. The coefficient of the highest power of x_1 in $f \in (F[x_2, \ldots, x_n])[x_1]$ is referred to as the leading coefficient of f in x_1, $\mathrm{ldcf}_{x_1}(f)$. Two polynomials f_1 and f_2 are associates, $f_1 \sim f_2$, if $f_1 = cf_2$ with $0 \neq c \in F$. For $F = \mathbf{Q}$ the binary size of the monomial coefficients of f as fractions of integers with a common denominator, the combined coefficient size, is denoted by

cc-size(f).

A straight-line program over a domain D is formally a quadruple $P = (X, V, C, S)$ where X is the set of inputs, V the set of program variables, C the computation sequence, and S the set of scalars occurring in the computation sequence. The length of C is the length of P, len(P). Each program variable v computes an element in D. A polynomial $f \in F[x_1, \ldots, x_n]$, or a rational function $f/g \in F(x_1, \ldots, x_n)$, is given by a straight-line program P if $P = (\{x_1, \ldots, x_n\}, V, C, S)$ computes f or f/g over $F(x_1, \ldots, x_n)$ and $S \subset F$. The program P is defined at $\phi: \{x_1, \ldots, x_n\} \to F$ if no zero-division occurs when evaluating P at $\phi(x_i)$ in place of x_i. The element size of P, el-size(P), denotes the number of bits it takes to represent all elements in S.

By $M(d)$ we denote a function dominating the time for multiplying polynomials in $F[x]$ of maximum degree d. Notice that for arbitrary fields the best known upper bound for $M(d)$ is $O(d \log(d) \log\log(d))$ [20]. The cardinality of a set R is denoted by card(R). We note that for a non-zero polynomial f the probability

$$\text{Prob}(f(a_1, \ldots, a_n) = 0 \mid a_i \in R) \leq \frac{\deg(f)}{\text{card}(R)},$$

see [21].

2. Straight-Line Factorization

We now describe the algorithm for finding the straight-line factors of a p-computable polynomial. The algorithm is derived from the One-Variable Lifting algorithm in [9], with the homogeneous parts of the minor variables replacing the monomials of the single variable with respect to which is lifted. We will compute those homogeneous parts by straight-line programs. The main reason why the answer is polynomial in length is that we only need to add on to the intermediate programs. This is because subsequent homogeneous parts can be computed from previous ones and Strassen's technique of obtaining a homogeneous program need not be applied at each iteration.

Algorithm *Factorization*

Input: $f \in F[x_1, \ldots, x_n]$ be given by a straight-line program P of length l, a bound $d \geq \deg(f)$, and an allowed failure probability $\varepsilon \ll 1$.

Output: Either "failure", that with probability $< \varepsilon$, or $e_i \geq 1$ and irreducible $h_i \in F[x_1, \ldots, x_n]$, $1 \leq i \leq r$, given by a straight-line program Q of length

$$\text{len}(Q) = O(d^2 l + d\, M(d^2) \log d)$$

such that with probability $> 1 - \varepsilon$, $f = \prod_{i=1}^{r} h_i^{e_i}$. In case $p = \text{char}(F)$ divides any e_i, that is $e_i = p^{\hat{e}_i}\, \bar{e}_i$ with $\bar{e}_i$ not divisible by p, we return $\bar{e}_i$ in place of e_i and Q will compute $h_i^{p^{\hat{e}_i}}$.

Step R (Random Points Selection): From a set $R \subset F$ with

$$\text{card}(R) > \frac{6}{\varepsilon} \max(2^{l+1}, 4d\,2^d + d^3)$$

select random elements $a_1, \ldots, a_n, b_2, \ldots, b_n$. If $F = \text{GF}(q)$ with q small we can instead work over $\text{GF}(q^p)$, p a prime integer $> d$. By Theorem 6.1 in [6] no additional factors occur.

Test whether P is defined at $\phi(x_i) = a_i$, $1 \le i \le n$. For $F = \mathbf{Q}$ we call algorithm Zero-Division Test in [12] such that the probability of "failure" even if P were defined at ϕ is less than $\varepsilon/6$. If P turns out to be (probably) undefined at ϕ we return "failure". Otherwise, P is definitely defined at ϕ and we compute the dense representation of

$$f_2 = f(x_1 + a_1,\ x_2,\ b_3 x_1 + a_3, \ldots, b_n x_1 + a_n).$$

This can be done by evaluation and interpolation similarly to the Sparse Conversion algorithm in [12]. If $F = \mathbf{Q}$, a bound for the cc-size(f) must be added to the input parameters and we must again make the probability that "failure" occurs due to the use of modular arithmetic during evaluation less than $\varepsilon/6$.

Step F (Factorization): Factor

$$f_2 = \prod_{i=1}^{r} \tilde{g}_{i,2}^{e_i},$$

$\tilde{g}_{i,2} \in F[x_1, x_2]$ irreducible and pairwise not associated. Notice that by theorem 2.1 of [9] f and f_2 have with high probability the same factorization pattern, that is irreducible factors of f map to pairwise non-associated irreducible factors of f_2 of the same total degrees. For the remainder of the algorithm we will assume that this is the case.

If $\text{char}(F) = p > 0$ divides any of the e_i, say $e_i = p^{\hat{e}_i}\,\bar{e}_i$ with $\bar{e}_i$ not divisible by p, we replace e_i by $\bar{e}_i$ and $\tilde{g}_{i,2}$ by $\tilde{g}_{i,2}^{p^{\hat{e}_i}}$. This replacement guarantees that none of the multiplicities are divisible by the characteristic.

Now set

$$g_{i,0}(x_1) \leftarrow \tilde{g}_{i,2}(x_1,\ b_2 x_1 + a_2) \in F[x_1].$$

Check whether $\text{GCD}(g_{i,0},\ g_{j,0}) \sim 1$ for $1 \le i < j \le r$ and whether $\deg(\tilde{g}_{i,2}) = \deg(g_{i,0})$ for $1 \le i \le r$. If not return "failure".

Let

$$\bar{f}(x_1,\ldots,x_n) = f(x_1 + a_1,\ x_2 + b_2 x_1 + a_2,\ldots,x_n + b_n x_1 + a_n)$$

$$= \prod_{i=1}^{r} h_i(x_1,\ldots,x_n)^{e_i},$$

and assume that h_i are the irreducible factors that correspond to $\tilde{g}_{i,2}$. Notice that the assumptions

on the preservation of the total degrees of the factors throughout the evaluation process also imply that

$$\mathrm{ldcf}_{x_1}(\bar{f}) \in F. \qquad (*)$$

Furthermore, let $\bar{P}$ be a straight-line program computing $\bar{f}$. We write

$$\bar{f}(x_1,\ldots, x_n) = \sum_{j=0}^{d} \sum_{m=0}^{d} \bar{f}_{j,m}(x_2,\ldots, x_n)\, x_1^m,$$

such that $\bar{f}_{j,m} \in F[x_2 ,\ldots, x_n]$ is homogeneous of degree j. We remark that d can now be set to $\deg(\bar{f})$ rather than a bound for it. We will need a straight-line program computing $\bar{f}_{j,m}$. If we replace x_i by $x_i x_1^{d+1}$, $2 \le i \le n$, in $\bar{P}$ then $\bar{f}_{j,m}$ is the coefficient of $x_1^{j(d+1)+m}$. Therefore by evaluating at x_1 and interpolating as in the Polynomial Coefficients algorithm [12] we can find a straight-line program Q_0 for $\bar{f}_{j,m}$ of length

$$\mathrm{len}(Q_0) = O(d^2 l + M(d^2)\log d).$$

Notice that we need to randomly pick $(d+1)^2$ points at which we interpolate and we must make sure that the straight-line program $\bar{P}$ is defined at those points. If that is not the case, or if for $F = \mathbf{Q}$ we cannot confirm by the Zero-Division Test algorithm [12] that a point is good, that with probability $< \varepsilon/(6(d+1)^2)$, we return "failure". For more details we refer to step P in the cited Polynomial Coefficients algorithm.

Step H (Hensel Lifting Loop): FOR $k \leftarrow 0 ,\ldots, d-1$ DO step L.

Step L (Lift by One Degree): Let

$$h_i(x_1,\ldots, x_n) = \sum_{m=0}^{d_i} \sum_{j=0}^{d_i} c_{i,j,m}(x_2,\ldots, x_n)\, x_1^m,$$

$d_i = \deg(h_i)$, where $c_{i,j,m}(x_2 ,\ldots, x_n) \in F[x_2 ,\ldots, x_n]$ is homogeneous of degree j. At this point we have a straight-line program Q_k over $F(x_2 ,\ldots, x_n)$ that computes $c_{i,j,m}$ for $1 \le i \le r$, $0 \le j \le k$, $0 \le m \le d_i$. Notice that $c_{i,0,m} \in F$ is the coefficient of x_1^m in $g_{i,0}$, and that by (*) $c_{i,j,d_i} = 0$ for $j > 0$. We will extend Q_k to Q_{k+1} that also computes $c_{i,k+1,m}$. It is useful to introduce the following polynomials

$$g_{i,k} = \sum_{j=0}^{k} \sum_{m=0}^{d_i} c_{i,j,m} x_1^m, \quad \hat{g}_{i,k+1} = \sum_{m=0}^{d_i} c_{i,k+1,m} x_1^m.$$

Now consider the congruence

$$\prod_{i=1}^{r} (g_{i,k} + \hat{g}_{i,k+1})^{e_i} \equiv \bar{f} \bmod (x_2,\ldots, x_n)^{k+2}. \qquad (\dagger)$$

Expanding the LHS we get

$$g_{1,0}^{e_1-1} \cdot \cdot \cdot g_{r,0}^{e_r-1} \sum_{i=1}^{r} (e_i \hat{g}_{i,k+1} \prod_{\substack{j=1 \\ j \neq i}}^{r} g_{j,0}) \qquad (\ddagger)$$
$$\equiv \bar{f} - \prod_{i=1}^{r} g_{i,k}^{e_i} \bmod (x_2, \ldots, x_n)^{k+2}.$$

By our loop invariant for Q_k

$$\bar{f} - \prod_{i=1}^{r} g_{i,k}^{e_i} \equiv$$
$$\tau = \sum_{m=0}^{d-1} \tau_m(x_2, \ldots, x_n)\, x_1^m \bmod (x_2, \ldots, x_n)^{k+2},$$

where $\tau_m \in F[x_2, \ldots, x_n]$ is homogeneous of degree $k+1$. Notice that the degree in x_1 is $d-1$ by the assumption (*). We need a program Q_τ over $F(\cdot\,\cdot\,\cdot, c_{i,j,m}, \ldots, \bar{f}_{j,m}, \cdot\,\cdot\,\cdot)$ that computes τ_m. If Q_τ encodes a tree-like multiplication scheme that can be done in

$$\mathrm{len}(Q_\tau) = O(M(d^2) \log d).$$

By our assumption that we lift a true factorization, (‡) is solvable in $\hat{g}_{i,k+1}$ and hence $g_{1,0}^{e_1-1} \cdot \cdot \cdot g_{r,0}^{e_r-1}$ must divide τ. Let

$$\rho = \sum_{m=0}^{d_1+\cdots+d_r-1} \rho_m(x_2, \ldots, x_n)\, x_1^m = \frac{\tau}{g_{1,0}^{e_1-1} \cdot \cdot \cdot g_{r,0}^{e_r-1}},$$

$\rho_m \in F[x_2, \ldots, x_n]$. As before, we need a straight-line program Q_ρ over $F(\cdot\,\cdot\,\cdot, \tau_m, \cdot\,\cdot\,\cdot)$ that computes all ρ_m. A simple encoding of polynomial division takes $\mathrm{len}(Q_\rho) = O(M(d))$. Now consider

$$\frac{\rho}{g_{1,0} \cdot \cdot \cdot g_{r,0}} = \frac{e_1 \hat{g}_{1,k+1}}{g_{1,0}} + \cdot\,\cdot\,\cdot + \frac{e_r \hat{g}_{r,k+1}}{g_{r,0}}.$$

It is clear that $e_i c_{i,j,k+1}$ are the coefficients of the univariate partial fraction decomposition of $\rho/(g_{1,0} \cdot \cdot \cdot g_{r,0})$ carried out over the field $F(x_2, \ldots, x_n)$. One way to compute these coefficients by a straight-line program $\hat{Q}_{k+1}$ with $\mathrm{len}(\hat{Q}_{k+1}) = O(M(d))$ is to once and for all find $\hat{g}_{i,0} \in F[x_1]$ with

$$\frac{1}{g_{1,0} \cdot \cdot \cdot g_{r,0}} = \frac{\hat{g}_{1,0}}{g_{1,0}} + \cdot\,\cdot\,\cdot + \frac{\hat{g}_{r,0}}{g_{r,0}}, \quad \deg(\hat{g}_{i,0}) < d_i,$$

and encode the polynomial remaindering

$$\hat{g}_{i,k+1} = \frac{\hat{g}_{i,0}\ \rho \bmod g_{i,0}}{e_i}, \quad 1 \le i \le r.$$

We must be able to divide by e_i and here we need the fact that the multiplicities must not be divisible by $\mathrm{char}(F)$. We finally link the programs Q_k, Q_τ, Q_ρ, and $\hat{Q}_{k+1}$ properly together to obtain the program Q_{k+1}. Notice that

$$\text{len}(Q_{k+1}) \le \text{len}(Q_k) + C\ M(d^2)\log d,$$

where C is an absolute constant. From this relation we can infer the length of Q.

Step T (Final Translation): From Q_d we obtain Q which computes

$$h_i(x_1 - a_1,\ x_2 - b_2(x_1 - a_1) - a_2, \ldots,\ x_n - b_n(x_1 - a_1) - a_n)$$

by adding in front of Q_d instructions for translating the x_i appropriately. □

We now analyze the failure probabilities of the Factorization algorithm. The only way an incorrect program Q can be produced is that the factorization patterns of f and f_2 disagree. By theorem 2.1 in [9] this happens with probability <

$$\frac{4d\,2^d + d^3}{\text{card}(R)} < \varepsilon.$$

"Failure" can occur in six separate circumstances. First, P may be undefined at ϕ, that with probability $< 2^{l+1}/\text{card}(R) < \varepsilon/6$ by an argument similar to that used in Lemma 4.3 of [12]. Second, for $F = \mathbf{Q}$ we might fail to recognize that P is defined at ϕ, but we make this possibility happen with probability $< \varepsilon/6$. Third, for $F = \mathbf{Q}$ the computation of f_2 may fail with probability $< \varepsilon/6$.

Fourth, "failure" can occur if for some $i \ne j$, $\text{GCD}(g_{i,0},\ g_{j,0}) \ne 1$, or $\deg(g_{i,0}) < \deg(\tilde{g}_{i,2})$. Let $\pi_i(\beta_2) = \text{ldcf}_{x_1}(\tilde{g}_{i,2}(x_1,\ \beta_2 x_1 + \alpha_2)$ and let

$$\sigma_{i,j}(\alpha_2,\ \beta_2) = \text{resultant}_{x_1}(\tilde{g}_{i,2}(x_1,\ \beta_2 x_1 + \alpha_2),\ \tilde{g}_{j,2}(x_1,\ \beta_2 x_1 + \alpha_2))$$

over $F[\alpha_2,\ \beta_2,\ x_1]$. It is easy to see that $0 \ne \pi_i\ \sigma_{i,j} \in F[\alpha_2,\ \beta_2]$ and $\pi_i(b_2)\ \sigma_{i,j}(a_2,\ b_2) \ne 0$ implies that the above events are impossible. Now, $\deg(\pi_i) \le d_i$ and $\deg(\sigma_{i,j}) \le 2d_i d_j$ and therefore the probability that the above events occur for any $i \ne j$ is less than

$$\sum_{i=1}^{r} \frac{d_i}{\text{card}(R)} + \sum_{1 \le i < j \le r} \frac{2d_i d_j}{\text{card}(R)}$$

$$< \frac{(d_1 + \cdots + d_r)^2}{\text{card}(R)} < \frac{d^2}{\text{card}(R)} < \frac{\varepsilon}{6}.$$

Notice that if P were division-free, this event would be the only one where failure could occur.

Fifth, we may not find good interpolation points in order to produce Q_0. If we try at most $(d+1)^4$ points, the probability that at least $(d+1)^2 = d^*$ points are good can be estimated like in the proof of [12], Theorem 5.1. We shall repeat the argument here. An individual point was not picked earlier with probability $\ge 1 - (d+1)^4/\text{card}(R) > 1 - \varepsilon/12$. $\bar{P}$ is not defined at an individual point substituted for x_1 with probability $< 2^{l+1}/\text{card}(R) < \varepsilon/12$. Hence a suitable point can be found in a block of d^* points with probability >

$$1 - (\varepsilon^*)^{d^*} > 1 - \frac{\varepsilon^*}{d^*}, \quad \varepsilon^* = \frac{\varepsilon}{6},$$

because $(1/\varepsilon^*)^{d^*-1} > 2^{d^*-1} \geq d^*$ for $\varepsilon^* < 1/2$. Now the probability that a good point occurs in all of the d^* blocks of points is >

$$(1 - \frac{\varepsilon^*}{d^*})^{d^*} > 1 - \varepsilon^*$$

and therefore failure happens with probability < $\varepsilon/6$.

Sixth and last, for $F = \mathbf{Q}$ we may not recognize that we have good interpolating points, that for all $(d+1)^2$ points together with probability < $\varepsilon/6$. We have established the following theorem.

Theorem 2.1: Algorithm Factorization does not fail with probability > $1 - \varepsilon$. In that case it reduces the problem in polynomially many steps as a function in len(P) and d to factoring bivariate polynomials. Its answer will be correct with probability > $1 - \varepsilon$. It requires polynomially many randomly selected field elements. For $F = \mathbf{Q}$ or $F = \mathrm{GF}(q)$ the algorithm has binary polynomial complexity also in $\log(1/\varepsilon)$, el-size(P), and cc-size(f). □

We now formulate two corollaries to this theorem. The first refers to computing the sparse factorization of f and follows from [12], §6.

Corollary 2.1: If in addition to the input parameters of the Factorization algorithm we are given $t > 0$, for $F = \mathbf{Q}$ or $F = \mathrm{GF}(q)$ we can find in polynomially many binary steps and random bit choices in

$$\text{len}(P),\ d,\ \log(\frac{1}{\varepsilon}),\ \text{el-size}(P),\ \text{cc-size}(f),\ \text{and } t$$

sparse polynomials that with probability > $1 - \varepsilon$ constitute all irreducible factors of f with no more than t monomials. □

Notice that the above running time is always polynomial independently whether the correct sparse factors were produced or whether other factors are dense. This makes this corollary superior to all previous work on sparse factorization. The second corollary deals with possibly non-uniform closure.

Corollary 2.2: Let F be a field of characteristic 0. Then any family of factors of a family of p-computable polynomials over F is p-computable. □

Notice that this corollary applies even to fields in which arithmetic is recursive but over which polynomial factorization is undecidable [4]. It also shows that a polynomial degree bound is necessarily required. We note that $x^{2^d} - 1$ can be computed with $O(d)$ instructions but it is known that over the complex numbers there exist factors that require $\Omega(2^{d/2}/\sqrt{d})$ computation length [16] and [18]. It would be nice to give such an example where the factors are irreducible

polynomials over **Q**.

3. Padé Approximants

We now review those properties of Padé approximants that we need in §4. However, we will not prove any of these properties and instead refer to [1] for an in depth discussion and the references into the literature. Let

$$f(x) = c_0 + c_1 x + c_2 x^2 + \cdots \in F[[x]],\ c_0 \neq 0,\ d, e \geq 0,$$

be given. Going back to Frobenius 1881 a rational function $p(x)/q(x)$ is called a (d, e)-Padé approximant to f if

$$\begin{aligned} \deg(p) \leq d,\ \deg(q) \leq e, \\ f(x)\, q(x) - p(x) \equiv 0 \bmod x^{d+e}. \end{aligned} \qquad (\dagger)$$

It turns out that for any pair (d, e) there always exists a solution to (†) and that furthermore the ratio p/q is unique. This ratio forms the entry in row d and column e of an infinite matrix referred to as Padé table. Already Kronecker 1881 realized that the entries in the $d + e$ anti-diagonal of the Padé table are closely related to the Euclidean remainder sequence of

$$f_{-1}(x) = x^{d+e+1},\ f_0(x) = c_0 + c_1 x + \cdots + c_{d+e} x^{d+e}.$$

Consider the extended Euclidean scheme

$$s_i(x)\, f_{-1}(x) + t_i(x)\, f_0(x) = f_i(x),$$

$$f_i(x) = f_{i-2}(x) \bmod f_{i-1}(x), \quad i \geq 1.$$

Then for the smallest index i with $\deg(f_i) \leq d$ we have $\deg(t_i) \leq e$ and f_i/t_i is the (d, e)-Padé approximant to f. Furthermore, $\mathrm{GCD}(f_i, t_i) = x^k$ for some $k \geq 0$. Thus any algorithm for computing the extended Euclidean scheme results in one for the (d, e)-Padé approximant. Note that the assumption $c_0 \neq 0$ is unessential by changing the lower bound for d.

The classical Euclidean algorithm gives a $O((d + e)^2)$ method for computing the (d, e)-Padé approximant. The ingenious algorithm by Knuth [13] that was improved by Schönhage [19] and applied to polynomial GCDs by Moenck [17] allows to compute the triple (f_i, s_i, t_i) with $\deg(f_i) \leq d$, $\deg(f_{i-1}) > d$, in $O(M(d + e)\log(d + e))$ operations in F.

4. Numerators and Denominators of Rational Functions

We now describe the algorithm that transforms a straight-line computation for a rational function of known degree to one for its (reduced) numerator and denominator. The key idea is that by substituting $x_m + b_m x_1$ for x_m, $2 \leq m \leq n$, we can make the problem a univariate problem in x_1 over the field $F(x_2, \ldots, x_n)$, as was already done in [12]. We then recover the fraction from its Taylor series approximation by computing the Padé approximant in $F(x_2, \ldots, x_n)[[x_1]]$.

Since that approximant is unique it must be the reduced numerator and denominator.

Algorithm *Rational Numerator and Denominator*

Input: A straight-line program P over $F(x_1, \ldots, x_n)$ of length l that computes f/g, $f, g \in F[x_1, \ldots, x_n]$ relatively prime, and $d \geq \deg(f)$, $e \geq \deg(g)$, and a failure allowance $\varepsilon \ll 1$. We shall make the assumption that $d, e \leq 2^l$ since the latter is always a bound.

Output: Either "failure" (that with probability $< \varepsilon$) or a straight-line program Q over $F(x_1, \ldots, x_n)$ of length $O(l\, M(d+e))$ such that Q computes f and g correctly with probability $> 1 - \varepsilon$.

Step T (Translation): From a set $R \subset F$ with

$$\mathrm{card}(R) > \frac{2^{(2C_3+2)\, l\, M(d+e)}}{\varepsilon}$$

select random elements $a_1, \ldots, a_n, b_2, \ldots, b_n$. Here the constant C_3 depends on the polynomial multiplication algorithm used. If F is a finite field with too small a cardinality, we can work in an algebraic extension of F instead. Since the results can be computed by rational operations in F they remain invariant with respect to field extensions.

Test whether P is defined at $\phi(x_m) = a_m$, $1 \leq m \leq n$. For $F = \mathbf{Q}$ we call the algorithm Zero-Division Test in [12], §3, such that the probability of "failure" occurring even if P is defined at ϕ is less than ε. If in this test P turns out to be (probably) undefined at ϕ we return "failure".

Now we translate the inputs of P as $x_1 \leftarrow y_1 + a_1$, $x_m \leftarrow y_m + b_m y_1$, $2 \leq m \leq n$. Let $\bar{P}$ be the straight-line program computing $\bar{f}/\bar{g}$ where

$$\bar{h}(y_1, \ldots, y_n) =$$
$$h(y_1 + a_1, y_2 + b_2 y_1, \ldots, y_n + b_n y_1) \text{ for } h \in F[x_1, \ldots, x_n].$$

Now $\bar{P}$ is defined at $\phi(y_1) = 0$. Also with high probability

$$\mathrm{ldcf}_{y_1}(\bar{f}) \in F. \qquad (*)$$

Step S (Power Series Approximation): Compute a straight-line program Q_1 over $F(y_2, \ldots, y_n)$ such that for the coefficients of the power series

$$\frac{\bar{f}}{\bar{g}} = c_0(y_2, \ldots, y_n) + c_1(y_2, \ldots, y_n) y_1 + \cdots$$

$$+ c_{d+e}(y_2, \ldots, y_n) {y_1}^{d+e} + \cdots, \qquad (\dagger)$$

c_i are computed by Q_1 for all $0 \leq i \leq d+e$. This can be done by directly applying the Polynomial Coefficients algorithm of [12], §4. Notice that $\mathrm{len}(Q_1) \leq C_1\, l\, M(d+e)$, where C_1 is a constant solely depending on the multiplication algorithm used.

Step P (Padé Approximation): Compute a straight-line program Q_2 over $F(y_2, \ldots, y_n)$ that with high probability computes the (d, e)-Padé approximation p/q, $p, q \in F(y_2, \ldots, y_n)[y_1]$ to (†). From §3 we know that this can be accomplished by an extended Euclidean algorithm. Essentially, we perform such an algorithm on the coefficient vectors $(c_i)_{0 \le i \le d+e}$ and that of ${y_1}^{d+e+1}$. In order to test elements in $F(y_2, \ldots, y_n)$ for zero we evaluate the program computing these elements at $\psi(y_m) = a_m$, $2 \le m \le n$. The details of this method for the classical subresultant algorithm can found in [12], §5. If we use the asymptotically fast Knuth-Schönhage procedure (see also [3] for a full description of the algorithm) then

$$\text{len}(Q_2) \le C_1\, l\, M(d+e) + C_2\, M(d+e)\log(d+e)$$

$$\le C_3\, l\, M(d+e), \qquad (‡)$$

where C_2 and C_3 are again constants solely depending on the polynomial multiplication procedure used. Notice that the produced straight-line program may be incorrect (that with small probability) since we may have incorrectly certified an element to be zero.

Once we have a straight-line program for polynomials f_i and $t_i \in F(y_2, \ldots, y_n)[y_1]$ in the extended Euclidean scheme we must find $k \ge 0$ such that $\text{GCD}(f_i, t_i) = {y_1}^k$ over $F(y_2, \ldots, y_n)[y_1]$. This we can again accomplish probabilistically by evaluating the coefficients in y_1 of f_i and t_i.

If we make $\text{ldcf}_{y_1}(p) = 1$, then with high probability p is an associate of $\bar{f}$ in $F[y_1, \ldots, y_n]$. This is because of (*) and because Padé approximants are unique.

Step T (Back-translate): The program Q is obtained by putting assignments for the back-translations

$$y_1 \leftarrow x_1 - a_1, \quad y_m \leftarrow x_m - b_m(x_1 - a_1),\ 2 \le m \le n,$$

in front of Q_2. □

We shall now analyze the overall failure probability of the Rational Numerator and Denominator algorithm. "Failure" is only returned if P is not defined or is not recognized to be defined at ϕ. However, several events must take place in order that the correct answer is returned. First, $\text{ldcf}_{y_1}(\bar{f}) \in F$ that justifies the normalization of p in step P. By lemma 5.1 in [12] this happens with probability ≥

$$1 - \frac{d}{\text{card}(R)} > 1 - \frac{\varepsilon}{4}.$$

Second, all zero-tests performed by evaluating at $\psi(y_m) = a_m$, $2 \le m \le n$, must give the correct answer. This is true if the Knuth-Schönhage algorithm performed over $F(y_2, \ldots, y_n)$ takes the same course as the algorithm performed over F on the elements obtained by evaluating at ψ. In other words, no non-zero element that is tested or by which is divided must evaluate to 0. Since the algorithm takes no more than

$$C_2\, M(d+e)\log(d+e)$$

steps, the degree of any unreduced numerator and denominator of these elements is by (‡) no more than

$$2^{C_3\, l\, M(d+e)}.$$

A (pessimistic) estimate for the number of elements to be tested and by which is divided, including the determination of k, is

$$C_3\, l\, M(d+e) + (d+e) < (C_3+1)\, l\, M(d+e).$$

Therefore the probability that all tests lead to the same result at ψ and that all division are possible at ψ is no less than

$$1 - \frac{(C_3+1)\, l\, M(d+e)\; 2^{C_3\, l\, M(d+e)}}{\mathrm{card}(R)} > 1 - \frac{\varepsilon}{2}.$$

Hence a correct program Q is output with probability $> 1 - 3/4\varepsilon$.

In case $F = \mathbf{Q}$ one more additional possibility of returning an incorrect result must be accounted for, namely that the Zero Test algorithm in [12], §3, might not recognize a non-zero evaluation at ψ properly. However, the probability of such an event can be controlled, say we allow it to happen only with probability no more than

$$\frac{\varepsilon}{4\,(C_2+1)\, M(d+e)\log(d+e)}.$$

Then all tests succeed with probability $> 1 - \varepsilon/4$ and a correct program is output with probability $> 1 - \varepsilon$. In summary, we have the following theorem.

Theorem 4.1: Algorithm Rational Numerator and Denominator does not fail and outputs a program Q that computes f and g with probability $> 1 - 2\varepsilon$. It requires polynomially many arithmetic steps as a function of len(P), d, and e. For $F = \mathbf{Q}$ this is also true for its binary complexity which also depends on el-size(P). The algorithm needs polynomially many randomly selected field elements (bits for $F = \mathbf{Q}$). □

We now formulate three corollaries to the theorem. The first corollary deals with distinguishing straight-line programs that compute polynomials from those that do not. It is clear that if we have the bounds d and e we only need to probabilistically inspect the degree of g after we have a straight-line program for it. But what if we do not have a-priori degree bounds? What we do then is to run our algorithm for

$$d = e = 2^k, \quad k = 1, 2, 3, \cdots.$$

Let f_k and g_k be the numerator and denominator whose computation is produced. For randomly chosen $a_1, \ldots, a_n \in F$ we then probabilistically test whether

$$\frac{f}{g}(a_1,\ldots, a_n) = \frac{f_k(a_1,\ldots, a_n)}{g_k(a_1,\ldots, a_n)}.$$

If the test is positive, with high probability $f = f_k$ and $g = g_k$. We have the following corollary, which also extends the result in [6], Remark 5.6, on probabilistically guessing the degree of a polynomial given by a straight-line program.

Corollary 4.1: Let f/g be given by a straight-line program P over $F(x_1,\ldots, x_n)$. Then we can in random polynomial-time in len(P) and deg(f) + deg(g) determine deg(f) and deg(g) such that with probability $> 1 - \varepsilon$ no failure occurs and the answer is correct. In particular, we can decide whether $f/g \in F[x_1,\ldots, x_n]$. □

For simplicity we state the next corollaries over infinite fields although this can be avoided as mentioned in step D. The next one resolves Strassen's question on computing the numerator and denominator of a rational function without divisions. By

$$L_D(r_1,\ldots, r_m \mid s_1,\ldots, s_n), \quad r_i, s_j \in D,$$

we denote the non-scalar or total complexity of computing r_i from s_j over D, see e.g [22]..

Corollary 4.2: Let F be a infinite field. Then

$$L_{F[x_1,\ldots,x_n]}(f, g \mid x_1,\ldots,x_n) =$$

$$O(M(\deg(fg))^2\ L_{F(x_1,\ldots,x_n)}(f/g \mid x_1,\ldots,x_n)),$$

where $M(d)$ is the corresponding complexity of multiplying d-degree polynomials. In the non-scalar case $M(d) = O(d)$. □

The third corollary concerns the parallelization of a straight-line computation for a rational function. From [25] we get.

Corollary 4.3: Let P be a straight-line program of length l over $F(x_1,\ldots, x_n)$, F infinite, that computes f/g where deg(f), deg(g) $\leq d$. Then there exists a straight-line program Q of depth $O((\log d)(\log d + \log l))$ and size $(l\ d)^{O(1)}$ that also computes f/g. □

5. Conclusion

The question arises what major unresolved problems in the subject of polynomial factorization remain. One theoretical question is to remove the necessity of random choices from any of the problems known to lie within probabilistic polynomial-time, say factorization of univariate polynomials over large finite fields. Another problem is to investigate the parallel complexity of polynomial factorization, say for the **NC** model [2]. Kronecker's reduction from algebraic number coefficients [23] and [14], Berlekamp's factorization algorithm over small finite fields [5], Kaltofen's deterministic Hilbert irreducibility theorem [10], §7, and Weinberger's irreducibility test for **Q**[x] [26] all lead to **NC** solutions by simply applying known **NC** methods for linear

algebra problems. It is open whether factoring in $\mathbf{Q}[x]$ or irreducibility testing in $\mathbf{F}_p[x]$, p large, or in $\mathbf{Q}[x, y]$ can be accomplished in **NC**.

In connection with the Factorization algorithm presented here, we also mention an open question. Assume that a straight-line program computes a polynomial whose degree is exponential in the length of the program. Are then at least its factors of polynomially bounded degree p-computable? A positive answer to this question would show that testing a polynomial for zero in a suitable decision-tree model is polynomial-time related to computing that polynomial. In general the theory of straight-line manipulation of polynomials may be extendable in part to unbounded input degrees, but even for the elimination of divisions problem [22] the answer is not known.

Nonetheless, in this paper we were able to contribute to Valiant's algebraic counterpart of the theory of **P** vs. **NP** in the positive, that is establish a polynomial upper bound for a major problem in computational algebra. In fact, it comes to us as a small surprise that p-computable polynomials are closed under factorization. And we have, finally, put to rest the problem of computing the sparse factorization of a multivariate polynomial.

Acknowledgement: A conversation with Allan Borodin during this conference last year triggered the construction of the Rational Numerator and Denominator algorithm. Tom Spencer very recently pointed out the possible connection between the factorization question and the relative power of decision vs. computation problems.

References

1. Brent, R. P., Gustavson, F. G., and Yun, D. Y. Y., "Fast solution of Toeplitz systems of equations and computation of Padé approximants," *J. Algorithms* **1**, pp. 259-295 (1980).

2. Cook, S. A., "A taxonomy of problems with fast parallel algorithms," *Inf. Control* **64**, pp. 2-22 (1985).

3. Czapor, S. R. and Geddes, K. O., "A comparison of algorithms for the symbolic computation of Padé approximants," *Proc. EUROSAM '84, Springer Lec. Notes Comp. Sci.* **174**, pp. 248-259 (1984).

4. Fröhlich, A. and Shepherdson, J. C., "Effective procedures in field theory," *Phil. Trans. Roy. Soc., Ser. A* **248**, pp. 407-432 (1955/56).

5. Gathen, J. von zur, "Parallel algorithms for algebraic problems," *SIAM J. Comp.* **13**, pp. 802-824 (1984).

6. Gathen, J. von zur, "Irreducibility of multivariate polynomials," *J. Comp. System Sci.* **31**, pp. 225-264 (1985).

7. Gathen, J. von zur and Kaltofen, E., "Factoring sparse multivariate polynomials," *J. Comp. System Sci.* **31**, pp. 265-287 (1985).

8. Hyafil, L., "On the parallel evaluation of multivariate polynomials," *SIAM J. Comp.* **8**, pp. 120-123 (1979).

9. Kaltofen, E., "Computing with polynomials given by straight-line programs II; Sparse factorization," *Proc. 26th IEEE Symp. Foundations Comp. Sci.*, pp. 451-458 (1985).

10. Kaltofen, E., "Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization," *SIAM J. Comp.* **14**, pp. 469-489 (1985).

11. Kaltofen, E., “Effective Hilbert irreducibility,” *Information and Control* **66**, pp. 123-137 (1985).

12. Kaltofen, E., “Greatest common divisors of polynomials given by straight-line programs,” *J. ACM* **35**(1), pp. 231-264 (1988).

13. Knuth, D. E., “The analysis of algorithms,” *Actes du congr`es international des Mathématiciens* **3**, pp. 269-274, Nice, France (1970).

14. Landau, S., “Factoring polynomials over algebraic number fields,” *SIAM J. Comp.* **14**, pp. 184-195 (1985).

15. Lenstra, A. K., Jr., H. W. Lenstra, and Lovász, L., “Factoring polynomials with rational coefficients,” *Math. Ann.* **261**, pp. 515-534 (1982).

16. Lipton, R. and Stockmeyer, L., “Evaluations of polynomials with superpreconditioning,” *Proc. 8th ACM Symp. Theory Comp.*, pp. 174-180 (1976).

17. Moenck, R. T., “Fast computation of GCDs,” *Proc. 5th ACM Symp. Theory Comp.*, pp. 142-151 (1973).

18. Schnorr, C. P., “Improved lower bounds on the number of multiplications/divisions which are necessary to evaluate polynomials,” *Theoretical Comp. Sci.* **7**, pp. 251-261 (1978).

19. Schönhage, A., “Schnelle Kettenbruchentwicklungen,” *Acta Inf.* **1**, pp. 139-144 (1971). (In German).

20. Schönhage, A., “Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2,” *Acta Inf.* **7**, pp. 395-398 (1977). (In German).

21. Schwartz, J. T., “Fast probabilistic algorithms for verification of polynomial identities,” *J. ACM* **27**, pp. 701-717 (1980).

22. Strassen, V., “Vermeidung von Divisionen,” *J. reine u. angew. Math.* **264**, pp. 182-202 (1973). (In German).

23. Trager, B. M., “Algebraic factoring and rational function integration,” *Proc. 1976 ACM Symp. Symbolic Algebraic Comp.*, pp. 219-228 (1976).

24. Valiant, L., “Reducibility by algebraic projections,” *L’Enseignement mathématique* **28**, pp. 253-268 (1982).

25. Valiant, L., Skyum, S., Berkowitz, S., and Rackoff, C., “Fast parallel computation of polynomials using few processors,” *SIAM J. Comp.* **12**, pp. 641-644 (1983).

26. Weinberger, P. J., “Finding the number of factors of a polynomial,” *J. Algorithms* **5**, pp. 180-186 (1984).