Single-Factor Hensel Lifting and its Application to the Straight-Line Complexity of Certain Polynomials*

Erich Kaltofen

Rensselaer Polytechnic Institute
Department of Computer Science
Troy, New York 12180-3590
Arpa-Net: kaltofen@csv.rpi.edu

Abstract

Three theorems are presented that establish polynomial straight-line complexity for certain operations on polynomials given by straight-line programs of unbounded input degree. The first theorem shows how to compute a higher order partial derivative in a single variable. The other two theorems impose the degree of the output polynomial as a parameter of the length of the output program. First it is shown that if a straight-line program computes an arbitrary power of a multivariate polynomial, that polynomial also admits a polynomial bounded straight-line computation. Second, any factor of a multivariate polynomial given by a division-free straight-line program with relatively prime co-factor also admits a straight-line computation of length polynomial in the input length and the degree of the factor. This result is based on a new Hensel lifting process, one where only one factor image is lifted back to the original factor. As an application we get that the greatest common divisor of polynomials given by a division-free straight-line program has polynomial straight-line complexity in terms of the input length and its own degree.

1. Introduction

The construction of straight-line programs for certain multivariate polynomials, such as the irreducible factors or the GCD of polynomials given by straight-line programs, was discovered to be feasible within the past three years [5], [6], and [7]. The program transformations are in random polynomial-time in the input size and the total degrees of the inputs. If the degrees of the input polynomials are allowed to be exponential in the straight-line program length, some basic constructions such as determining the coefficient of a single variable power can become #P-hard with respect to the straight-line program size alone [18], [6], §5 (see also §2 below). In this article we study how the input degree restriction can be weakened while retaining polynomial straight-line complexity for the answers.

The first theorem shows how to compute a higher order partial derivative in a single variable of a rational function given by a straight-line program. The simple construction is based on the Leibniz rule for higher derivatives of products and can even be carried out in polynomial-time. The next two theorems impose the degree of the output polynomial rather than the input polynomial as a parameter of the length of the output program. Polynomial-time constructibility of the

* 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.

answer programs is lost, however, in part due to the need of scalars derived from the input polynomials of possibly exponential degree, which thus can be exponential in size. We show first that if a straight-line program computes an arbitrarily high power of a polynomial. that base polynomial also admits a polynomial bounded straight-line computation, provided the characteristic of the scalar field does not divide the exponent. Our construction is based on taking roots of univariate power series. Second, we prove that any factor of a polynomial given by a division-free straight-line program with relatively prime co-factor admits a straight-line computation of length polynomial in the input length and its own degree. The proof of this result introduces a new Hensel lifting procedure [22], one where only one factor image is restored back to its original factor.

The assumption of co-primeness in our factorization result can be enforced in the setting of computing the greatest common divisor of polynomial by Hensel lifting, the so-called EZ-GCD procedure [12]. Therefore, we get as an application that the GCD of polynomials given by a division-free straight-line program has polynomial straight-line complexity in terms of the input length and the degree of the GCD. Furthermore, we also can derive from this results another solution to Strassen's problem [16] on the straight-line complexity of the reduced numerator and denominator of a multivariate rational function given by a straight-line program. The new approach does not make use of Padé approximations at all, as was the basis for our first solution to this problem [8], §4.

Notation: Generally, F, $\bar{F}$, K denote fields, char(.) their charcteristic, x, y, z, α indeterminates, f, g, h (multivariate) polynomials, a, b field elements, P, Q straight-line programs. Permitted operands in the instruction sequence of straight-line programs are field elements, indeterminates, and previous program variables. Permitted operators are $+$, $-$, $\times$, and $\div$. If the latter does not occur we call the program division-free. For a precise definition of the straight-line model we refer to [6], §2, or to [15]. By $\mathrm{ldcf}_{x_1}(f) \in F[x_2, \ldots, x_n]$ we denote the coefficient of the highest power of x_1 in $f \in (F[x_2, \ldots, x_n])[x_1]$. $M(d)$ denotes a function for the asymptotic complexity of d-degree polynomial multiplication, at best $M(d) = d \log(d) \log(\log(d))$ [14], [4]. Finally, we use := and =: as a shorthand to indicate the introduction of symbols, the new symbols occurring on the side of the colon.

2. Higher Derivatives

We now show how to compute the k-th order derivative in a single variable of a straight-line program. For polynomials of bounded degree this problem can be solved by computing straight-line programs for the coefficients in the single variable [18], [6]. The following solution, based on the Leibniz formula of higher derivatives, is much simpler and works also for rational functions and does not depend on the input degrees.

Theorem 1: *Let $f \in F(x_1, \ldots, x_n)$ be given by a straight-line program P of length l. Then $\partial^k f / \partial x_1^k$ can be computed by a straight-line program Q of length* $\mathrm{len}(Q) = O(k^2 l)$.

Proof: For every program variable v_λ, $1 \leqslant \lambda \leqslant l$, in P we introduce k additional variables $v_\lambda^{(\kappa)}$, $1 \leqslant \kappa \leqslant k$, $v_\lambda^{(0)} = v_\lambda$, such that $v_\lambda^{(\kappa)}$ computes the κ-th derivative of the function computed in v_λ. Now let an individual assignment be $w \leftarrow u \circ v$. The following relations apply:

$$\circ = +:\ w^{(\kappa)} = u^{(\kappa)} + v^{(\kappa)}$$

$$\circ = \times:\ w^{(\kappa)} = \sum_{\mu=0}^{\kappa} \binom{\kappa}{\mu} u^{(\kappa-\mu)} v^{(\mu)}$$

$$\circ = \div:\ w^{(\kappa)} = \Big(u^{(\kappa)} - \sum_{\mu=1}^{\kappa} \binom{\kappa}{\mu} w^{(\kappa-\mu)} v^{(\mu)}\Big) / v^{(0)}.$$

Therefore, clearly $w^{(\kappa)}$ can be determined from $u^{(0)}, \ldots, u^{(\kappa)}, v^{(0)}, \ldots, v^{(\kappa)}, w^{(0)}, \ldots, w^{(\kappa-1)}$ in $O(\kappa)$ assignments. Notice that if an operand, u say, is an indeterminate or scalar, then $u^{(j)} = 0$ except if $u = x_1$, then $u^{(1)} = 1$. Overall, each assignment in P gets expanded into at most $O(k^2)$ assignments and the result follows. □

The construction of Q from P can be accomplished in polynomial-time, that is the transformation is uniform in the sense of [6]. That means that we have another Monte-Carlo solution for finding the degree of a polynomial $f \in F[x_1, \ldots, x_n]$, namely by testing programs for

$$\frac{\partial f(yx_1, \ldots, yx_n)}{\partial y}, \ldots, \frac{\partial^\kappa f(yx_1, \ldots, yx_n)}{\partial y^\kappa}, \cdots$$

for zero by random evaluation. Our first solution used an interpolation approach [6], §5, and is in general more efficient. Notice that if the first k derivatives do not evaluate to zero, the input is either a polynomial of higher degree or a non-polynomial rational function. In the latter case, a more general test is available, cf. [8], Corollary 4.1.

There is evidence that it is impossible to compute the k-th derivative in $(l \log(k))^{O(1)}$ assignments. Consider the example from [18],

$$g(y_1, \ldots, y_n, z_{1,1}, \ldots, z_{n,n}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{n} y_j z_{i,j} \right).$$

Then the coefficient of the monomial $y_1 \cdots y_n$ in g is the permanent of the matrix $[z_{i,j}]_{1 \le i,j \le n}$. Notice that in contrast to theorem 1,

$$\frac{\partial^n g}{\partial y_1 \cdots \partial y_n} = \operatorname{perm}([z_{i,j}]_{1 \le i,j \le n}),$$

which makes taking partial derivatives in mixed variables #P-hard. Now performing a Kronecker substitution $x^{(n+1)^{i-1}}$ for y_i this permanent appears as the coefficient $c_k(z_{1,1}, \ldots, z_{n,n})$ of x^k for

$$k = 1 + (n+1) + (n+1)^2 + \cdots + (n+1)^{n-1},$$

in

$$f(x, z_{1,1}, \ldots, z_{n,n}) = g(x, x^{n+1}, \ldots, x^{(n+1)^{n-1}}, z_{1,1}, \ldots, z_{n,n}).$$

Therefore

$$\left. \frac{\partial^k f(x, z_{1,1}, \ldots, z_{n,n})}{\partial x^k} \right|_{x=0}$$

computes $k!\ \operatorname{perm}([z_{i,j}])$, whereas f can be computed by a division-free program of length $O(n^2)$. One may argue that divisions in the program for the k-th derivative can make the substitution $x = 0$ invalid, but we do not believe that divisions endow the straight-line complexity model with exponentially more power (see §6, problem 5). Also the exponentially sized constant $k!$ should not enable an exponential speed up of the computation of the permanent.

3. Large Exponent Roots

Our next theorem shows, for instance, that a very high power of a hard polynomial, such as the n by n permanent raised to the power 2^n, cannot have a straight-line computation of polynomial length. It is based on the fact that the d-degree root of a polynomial only depends on its $d+1$ low order coefficients, provided the constant coefficient is non-zero [9], §4.7. Our proof, however, will be based on the asymptotically faster Newton iteration.

Theorem 2: *Let $f \in F[x_1, \ldots, x_n]$ be given by a straight-line program P of length l,* $\operatorname{card}(F) > 2^{l+1}$, *and let $g \in F[x_1, \ldots, x_n]$, $d = \deg(g)$, be such that $g^e = f$, e not divisible by* $\operatorname{char}(F)$. *Then g can be computed by a straight-line program Q of length* $\operatorname{len}(Q) = O(l\, M(d))$.

Proof: By working with the translated polynomial

$$\bar{f}(x_1, \ldots, x_n, y) := f(x_1 y + a_1, \ldots, x_n y + a_n).$$

$a_i \in F$, we can find the Taylor series coefficients $c_0, \ldots, c_d \in F[x_1, \ldots, x_n]$ of

$$\bar{f}(x_1, \ldots, x_n, y) \equiv c_0(x_1, \ldots, x_n) + c_1(x_1, \ldots, x_n) y + \cdots + c_d(x_1, \ldots, x_n) y^d \bmod y^{d+1}$$

in $O(l\, M(d))$ assignments [6]. Moreover, the a_i can be chosen such that $c_0 \ne 0$. Here we make use of the assumption that the field is sufficiently large. Let

$$\bar{g}(x_1, \ldots, x_n, y) := \sum_{i=0}^{d} u_i(x_1, \ldots, x_n) y^i := g(x_1 y + a_1, \ldots, x_n y + a_n).$$

Since

$c_0(x_1,\ldots,x_n) = \bar{f}(x_1,\ldots,x_n,0) = f(a_1,\ldots,a_n) \in F$

we have $u_0 \in F$ with $u_0^e = c_0$. We determine $u_1,\ldots,u_d$ by Newton iteration of $z^e - \bar{f} = 0$ on the power series approximation of $\bar{f}$. We now describe this iteration in detail.

```
For $i \leftarrow 1,\ldots,\lceil \log_2(d+1) \rceil$ Do
    At this point we have computed
        $u_0,\ldots,u_j, w_0,\ldots,w_j \in F[x_1,\ldots,x_n]$,
    $j := 2^{i-1} - 1$, such that for
        $\alpha_{i-1}(y) = \sum_{k=0}^{j} u_k y^k,\ \beta_{i-1}(y) = \sum_{k=0}^{j} w_k y^k$,
    we have
        $\alpha_{i-1}\beta_{i-1} \equiv 1 \bmod y^{j+1},\ \alpha_{i-1}^e \equiv \bar{f} \bmod y^{j+1}$.
    We now encode straight-line assignments for $u_{j+1},\ldots,u_{2j-1}, w_{j+1},\ldots,w_{2j-1}$ that satisfy
        $\alpha_i(y) = (1 - \frac{1}{e})\alpha_{i-1}(y)$
            $+ \frac{1}{e}\bar{f}\,\beta_{i-1}(y)^{e-1} \bmod y^{\min(d+1,\,2^i)}$,
    and if $2^i < d+1$,
        $\beta_i(y) = 2\beta_{i-1}(y) - \beta_{i-1}^2(y)\alpha_i(y) \bmod y^{2^i}$.
    Notice that
        $\alpha_i \equiv \alpha_{i-1} - \left.\frac{z^e - \bar{f}}{\partial(z^e - \bar{f})/\partial z}\right|_{z=\alpha_{i-1}} \bmod y^{2^i}$
    and
        $\beta_i \equiv \beta_{i-1} - \left.\frac{1/z - \alpha_i}{\partial(1/z - \alpha_i)/\partial z}\right|_{z=\beta_{i-1}} \bmod y^{2^i}$.
    The amount of straight-line code for this step is $O(\log(e)M(2^i))$ using binary exponentiation for finding $\beta_{i-1}^{e-1} \bmod y^{2^i}$.
```

The total cost for the Newton iteration is $O(\log(e)M(d))$, or with $\log(e) = O(l)$, $O(l\ M(d))$ assignments. □

The construction of Q in the above proof is "almost" in random polynomial-time, all that is required as additional input is $u_0 \in F$ and e (given in binary). If e is polynomial in value, we could have also used the factorization algorithm in [7] to construct a program for g, but then the above method also leads to a random polynomial time algorithm for constructing the program Q. Moreover, this approach is more efficient both in terms of the asymptotic length of the resulting program as well as practicality.

We remark that the cardinality restriction on F is unessential in the above and also in the following theorems. If $\mathrm{card}(F) \leqslant 2^{l+1}$, we can carry out the construction in an algebraic extension $F(\theta)$ of degree $m := [F(\theta):F]$. The point is now that the resulting program Q, which uses scalars in $F(\theta)$, can be transformed to a program $\bar{Q}$ over $F(x_1,\ldots,x_n)$. For future reference, let us formulate the following lemma.

Lemma 1: *Let $f \in F(x_1,\ldots,x_n)$ be given by a straight-line program P over $K(x_1,\ldots,x_n)$, $K := F[\theta]/(g(\theta))$ where $g(\theta) \in F[\theta]$ is irreducible. Furthermore, let $l := \mathrm{len}(P)$ and $m := \deg(g)$. Then g can be computed by a straight-line program Q over $F(x_1,\ldots,x_n)$ of length* $\mathrm{len}(Q) = O(M(m)\log(m)\,l)$.

Proof: The idea is to construct for each function $v_\lambda \in K(x_1,\ldots,x_n)$, $1 \leqslant \lambda \leqslant l$, computed in the λ-th assignment of P a straight-line code segment for $w_{\lambda,\mu} \in F(x_1,\ldots,x_n)$, $0 \leqslant \mu < m$, such that

$$v_i \equiv \sum_{\mu=0}^{m-1} w_{\lambda,\mu}\theta^\mu \bmod g(\theta).$$

Division is the most costly operation and requires $O(M(m)\log(m))$ assignments to determine $w_{\lambda,\mu}$ from the corresponding coefficients of the dividend and divisor, using the extended polynomial version of the Knuth-Schönhage GCD algorithm to invert the divisor modulo $g(\theta)$, see e.g. [2].

If one were to carry out this step constructively, certain program variables in $F(x_1,\ldots,x_n)$ would need to be tested for zero, which can be accomplished by the Monte-Carlo algorithm in [6], §3. These tests can be avoided, however, by computing the coefficients of the inverse modulo g as quotients of minors in the Sylvester matrix [3] employing a division-free determinant program. The cost is then $m^{\omega+2+o(1)}$, where ω is the matrix multiplication exponent, and Q is obtained deterministically in polynomial time in l and m. □

4. Low Degree Factors

In [7] we have established that any factors of a family of multivariate polynomials with polynomially bounded degree and straight-line complexity can themselves be computed by straight-line programs of polynomial length. We now generalize this result by relaxing the degree bound condition on the input polynomials. The additional restrictions in the following theorem needed for our argument are discussed further after its proof.

Theorem 3: *Let $f \in F[x_1, \ldots, x_n]$ be given by a division-free straight-line program P of length l, and let $g \in F[x_1, \ldots, x_n]$, $d := \deg(g)$, be a factor of f such that $\mathrm{GCD}(g, f/g) = 1$. Furthermore, assume that $\mathrm{card}(F) > 2^l(2d+1)$. Then g can be computed by a division-free straight-line program Q of length*

$$\mathrm{len}(Q) = O(l\, M(d^3) + d^2 M(d^2)).$$

Before we can prove theorem 3 we need to introduce a new approach to Hensel lifting [22], [13], [21]. This new algorithm only lifts the original image of one factor and we hence refer to it by the name single-factor lifting.

Algorithm *Single-Factor Lifting*

Assume $f(x, y) = g(x, y)\, h(x, y)$, $f, g, h \in F[x, y]$, F a field, $d_x := \deg_x(g)$, $d_y := \deg_y(g)$, such that

$$\mathrm{ldcf}_x(f) \in F,\ \mathrm{GCD}(g(x, 0), h(x, 0)) = 1. \quad (1)$$

This algorithm describes a method for lifting the equation

$$g(x, 0)\, h(x, 0) \equiv f(x, y) \bmod y$$

to obtain $g(x, y)$ without accessing $\deg_x(f)$ coefficients. Its inputs are a truncated $g(x, 0)$-adic expansion of $h(x, 0)$ and $f(x, y) \bmod y^{d_y+1}$.

Input: $g_0(x) := g(x, 0)$,

$$h(x, 0) \bmod g_0(x)^{d_y+1} =: \sum_{i=0}^{d_y} \hat{h}_0^{(i)}(x) g_0(x)^i,$$

$\hat{h}_0^{(i)} \in F[x]$, $\deg(\hat{h}_0^{(i)}) < d_x$, and

$$f(x, y) \bmod (g_0(x)^{d_y+1}, y^{d_y+1}) =: \sum_{i=0}^{d_y} \sum_{j=0}^{d_y} \hat{f}_j^{(i)}(x) y^j g_0(x)^i,$$

$\hat{f}_j^{(i)} \in F[x]$, $\deg(\hat{f}_j^{(i)}) < d_x$. Here and in the following the polynomials "with hats" are always in $F[x]$. (It might be unclear at the moment how to obtain the $\hat{f}_j^{(i)}$ without accessing all coefficients of f, but as we will explain later, for f given by division-free straight-line programs this is not difficult.)

Output: $g(x, y)$.

For $k \leftarrow 0, \ldots, d_y$ **Do** Step L. Then **Return** $g(x, y) = \sum_{j=0}^{d_y} \hat{g}_j(x) y^j$.

Step L: This step lifts by one degree in y. For a polynomial $\psi(x, y) \in F[x, y]$ let $\psi_k(x, y) := (\psi(x, y) \bmod y^{k+1})$, $\hat{\psi}_k(x) y^k := \psi_k(x, y) - \psi_{k-1}(x, y)$, $k \geqslant 1$, $\hat{\psi}_0 = \psi_0$. In normal lifting, at this point we have g_k, h_k and determine $\hat{g}_{k+1}$, $\hat{h}_{k+1}$ by

$$h_0(x) \hat{g}_{k+1}(x) + g_0(x) \hat{h}_{k+1}(x) = \frac{f_{k+1}(x, y) - g_k(x, y) h_k(x, y) \bmod y^{k+2}}{y^{k+1}},$$

$\deg(\hat{g}_{k+1}) < d_x$, $\deg(\hat{h}_{k+1}) < \deg(h_0)$. Let

$$\hat{t}_{k+1}(x) y^{k+1} :\equiv f_{k+1} - g_k h_k \bmod y^{k+2},$$

$\hat{t}_{k+1} \in F[x]$, $\deg(\hat{t}_{k+1}) < \deg_x(f)$ by (1). The key identity is

$$h_0 \hat{g}_{k+1} + g_0 \hat{h}_{k+1} = \hat{t}_{k+1}. \quad (2)$$

In this algorithm we determine $\hat{g}_{k+1}$ by

$$\hat{g}_{k+1} = (\hat{t}_{k+1} h_0^{-1} \bmod g_0). \quad (3)$$

We will compute along sufficiently high-order g_0-adic expansions of the h_k's $\in F[x, y]$. For $\psi(x, y) \in F[x, y]$ let

$$\hat{\psi}_k^{(i)} := \frac{\hat{\psi}_k \bmod g_0^{i+1} - \hat{\psi}_k \bmod g_0^i}{g_0^i} \in F[x],$$

$i \geq 0$. Notice that $\hat{\psi}_k^{(i)}$ is the i-th digit in the g_0-adic expansion of $\hat{\psi}_k$. Also $\deg(\hat{\psi}_k^{(i)}) < d_x$.

As the loop invariant, at this point we have g_k and $\hat{h}_j^{(i)}$, $0 \leq j \leq k$, $0 \leq i \leq d_y - j$. We first find $\hat{t}_{k+1}^{(i)}$, $0 \leq i \leq d_y - k$. This is done by multiplying

$$(z + \sum_{j=1}^{k} \hat{g}_j(x) y^j)(\sum_{i=0}^{d_y-k} \sum_{j=0}^{k} \hat{h}_j^{(i)}(x) y^j z^i)$$
$$=: \sum_{i=0}^{d_y-k} \sum_{j=0}^{k+1} \tilde{w}_j^{(i)}(x) y^j z^i \bmod (z^{d_y-k}, y^{k+2}), \tag{4}$$

$\tilde{w}_j^{(i)} \in F[x]$, $\deg(\tilde{w}_j^{(i)}) < 2d_x - 1$. Here z is a placeholder for g_0. Also by the invariant we must have $\tilde{w}_j^{(i)} = 0$ for all $0 \leq j \leq k$ and all corresponding i.

Then the g_0-adic "digits" and "carries" are determined by division with remainder for $i = 0, \ldots, d_y - k$,

$$\tilde{w}_{k+1}^{(i)}(x) =: \hat{r}_{k+1}^{(i)}(x) + g_0(x) \hat{s}_{k+1}^{(i)}(x),$$

$\deg(\hat{r}_{k+1}^{(i)}) < d_x$. Finally, for $i = 0, \ldots, d - k$, by (2) set

$$\hat{t}_{k+1}^{(i)} = \hat{f}_{k+1}^{(i)} - (\hat{r}_{k+1}^{(i)} + \hat{s}_{k+1}^{(i-1)}),\ \hat{s}_{k+1}^{(-1)} = 0.$$

Thus by (3) $\hat{g}_{k+1} \equiv (\hat{h}_0^{(0)})^{-1} \hat{t}_{k+1}^{(0)} \bmod g_0$. Now the $\hat{h}_{k+1}^{(i)}$, $0 \leq i \leq d - k - 1$, are determined by

$$\sum_{i=0}^{d_y-k-1} \hat{h}_{k+1}^{(i)} g_0^{i+1} \equiv$$
$$\sum_{i=0}^{d_y-k} \hat{t}_{k+1}^{(i)} g_0^i - \hat{g}_{k+1} \sum_{i=0}^{d_y-k} \hat{h}_0^{(i)} g_0^i \bmod g_0^{d-k+1}.$$

Again, "digits and carries" of $\hat{g}_{k+1} \hat{h}_0^{(i)}$ have to be computed by remaindering. □

Lemma 2: Algorithm Single-Factor Lifting requires $O(d_y^2 M(d_x d_y))$ arithmetic operations in F.

Proof: Each iteration in the loop is dominated by the cost of computing the $\tilde{w}_j^{(i)}$ in (4). That is essentially $O(d_y)$ bivariate multiplications of polynomials degree $< d_x$ in x and degree $k \leq d_y$ in y, each of which can be done in $O(M(d_x d_y))$ arithmetic operations. □

We now can prove theorem 3.

Proof of Theorem 3: For a polynomial $\chi \in F[x_1, \ldots, x_n]$ let

$$\bar{\chi}(x_1, \ldots, x_n, y) :=$$
$$\chi(x_1 + a_1, yx_2 + b_2 x_1 + a_2, \ldots, yx_n + b_n x_1 + a_n).$$

Now we choose $a_1, \ldots, a_n, b_2, \ldots, b_n \in F$ such that for $h := f / g$

$$\deg(\bar{f}) = \deg_{x_1}(\bar{f}) \text{ and}$$
$$\mathrm{GCD}(\bar{g}(x_1, 0, \ldots, 0), \bar{h}(x_1, 0, \ldots, 0)) = 1. \tag{5}$$

This means that the points must not be a zero of a certain leading coefficient and resultant. We refer to the analysis of the Factorization algorithm in [8] for more detail. Observe that $\deg(f) \leq 2^l$. The idea is now to interpret $\bar{f}$ as a bivariate polynomial in x_1 and y over the field $\bar{F} := F(x_2, \ldots, x_n)$. The key property that allows us to use the Single-Factor Lifting algorithm is that

$$\bar{\chi}_0 := \bar{\chi}(x_1, \ldots, x_n, 0) \in \bar{F}[x_1]$$

is actually an element in $F[x_1]$. Therefore the coefficients of $\bar{g}_0$ and $\hat{h}_0^{(i)} \in \bar{F}[x_1]$ required as input for single-factor lifting are scalars and most certainly have short computations. The input assumptions (1) to the lifting algorithm are satisfied by (5). In addition we need a straight-line program that computes $\hat{f}_{j,m}^{(i)} \in \bar{F}$, where

$$\sum_{i=0}^{d} \left(\sum_{j=0}^{d} \sum_{m=0}^{d-1} \hat{f}_{j,m}^{(i)}(x_2, \ldots, x_n) x_1^m y^j \right) \bar{g}_0(x_1)^i$$
$$:\equiv \bar{f} \bmod (\bar{g}_0(x_1)^{d+1}, y^{d+1}).$$

Notice that

$$\deg_{x_1}(\bar{g}) = d \text{ and } \deg_y(\bar{g}) \leq d.$$

We determine $\hat{f}_{j,m}^{(i)}$ by finding the corresponding polynomials for each program variable in $\bar{P}$, where $\bar{P}$ is the straight-line program for $\bar{f}$. We illustrate this process for the assignment $w \leftarrow u \times v$. Assume $\hat{u}_{j,m}^{(i)}$, $\hat{v}_{j,m}^{(i)}$ compute the coefficients of $x_1^m y^j$ in the i-th digit of the $\bar{g}_0$-adic expansion of the polynomials computed in u and v, respectively. First we find the tri-variate convolutions

$$\tilde{w}_{j,m}^{(i)} = \sum_{i_1+i_2=i} \sum_{j_1+j_2=j} \sum_{m_1+m_2=m} \hat{u}_{j_1,m_1}^{(i_1)} \hat{v}_{j_2,m_2}^{(i_2)},$$

$0 \leqslant i \leqslant d, 0 \leqslant j \leqslant d, 0 \leqslant m \leqslant 2d-2$. This we can do in $O(M(d^3))$ assignments per multiplication in $\bar{P}$. We now take care of the "carries with radix $\bar{g}_0$." We encode the division with remainder for all $0 \leqslant i, j \leqslant d$,

$$\sum_{m=0}^{2d-2} \tilde{w}_{j,m}^{(i)} x_1^m =: \sum_{m=0}^{d-1} \hat{r}_{j,m}^{(i)} x_1^m + \bar{g}_0(x_1) \sum_{m=0}^{d-2} \hat{s}_{j,m}^{(i)} x_1^m.$$

There are $O(d^2)$ divisions each of which can be carried out in $O(M(d))$ assignments. Finally we set

$$\hat{w}_{j,m}^{(i)} \leftarrow r_{j,m}^{(i)} + s_{j,m}^{(i-1)}, \quad s_{j,m}^{(-1)} = 0,$$

$0 \leqslant m \leqslant d-1, 0 \leqslant i, j \leqslant d$.

Additive assignments in $\bar{P}$ are a much simpler affair, and the overall cost for computing $\hat{f}_{j,m}^{(i)}$ is $O(l\ M(d^3))$. We like to point out that it is here that we must exclude divisions from P. The reason is that we cannot necessarily invert all functions modulo $\bar{g}_0$ by which is divided. It appears that this problem cannot be handled by translating the input.

Now we have straight-line computations for all elements in $\bar{f}$ needed as inputs to the Single-Factor Lifting algorithm. It remains to encode the arithmetic operations performed during this algorithm at an additional cost of $O(d^2 M(d^2))$ assignments (see lemma 2). Notice that we obtain f by setting $y = 1$ and performing the proper back-translations. □

If $\deg(f) = l^{O(1)}$ the result in [8] is obviously stronger than this theorem, so let us suppose that $\deg(f)$ is super-polynomial in l. Our proof methods based on the Single-Factor Hensel Lifting procedure above then do not permit an unconditionally uniform, that is random polynomial-time, construction of Q from P alone, although if d is significantly smaller than $\deg(f)$, the Single-Factor Lifting algorithm may prove more efficient than standard lifting even in practice.

The assumption that $\mathrm{GCD}(g, f/g) = 1$ is essentially equivalent to stating that g be irreducible and its multiplicity e in f be small, that is $e = l^{O(1)}$. Unfortunately, we do not know how to eliminate this condition on e (see §6, problem 3). Notice first, however, that if $\deg(f/g^e) = l^{O(1)}$, a straight-line program of length $(l\ d)^{O(1)}$ for g could still be constructed. For we could apply theorem 3 to f/g^e in place of g and find a straight-line program with divisions for g^e. The construction of g then follows by Theorem 2. Second, observe that the elimination of the multiplicity bound e of g in f would also eliminate the assumption that P be division-free. This follows by replacing each straight-line assignment in P with assignments that compute the unreduced polynomial numerator and denominator of the rational function corresponding to that assignment. Clearly the irreducible g would be a factor of the numerator corresponding to the assignment that computes the polynomial f.

5. Low Degree Greatest Common Divisors

Two interesting corollaries follow from theorem 3. The first concerns the determination of a straight-line program for the degree bounded GCD of polynomials given by a division-free straight-line program. Surprisingly, the Euclidean algorithm does not enter in its proof, instead it is based on the so-called EZ-GCD method [12]. It is not even clear to us how an equivalent statement can be derived by using a remainder sequence.

Theorem 4: *Let $f_i \in F[x_1, \ldots, x_n]$, $1 \leqslant i \leqslant r$, be given by a division-free straight-line program P of length l, $g := \mathrm{GCD}(f_1, \ldots, f_r)$, $d := \deg(g)$, $\mathrm{card}(F) > 2^l(2d+1)$. Then g can be computed by a straight-line program Q of length*

$$\mathrm{len}(Q) = O(l\ M(d^3) + d^2 M(d^2)).$$

Proof: We observe that if $\mathrm{card}(F) > d$ there exist $a_i \in F$, $1 \leqslant i \leqslant r$, such that

$$\mathrm{GCD}(g, a_1\frac{f_1}{g} + \cdots + a_r\frac{f_r}{g}) = 1. \qquad (6)$$

In [20] this observation is attributed to D. Spear. Here is a justification. Temporarily define for $f \in F[x_1, \ldots, x_n]$

$$\bar{f} := f(x_1, y_2+z_2x_1, \ldots, y_n+z_nx_1),$$

where z_i are new variables. Let

$$\sigma(\alpha_1, \ldots, \alpha_r) \in F[\alpha_1, \ldots, \alpha_r]$$

be a non-zero coefficient of a monomial in the variables $y_2, \ldots, y_n, z_2, \ldots, z_n$ of

$$\mathrm{resultant}_{x_1}(\bar{g}, \sum_{i=1}^{r}\alpha_i\frac{\bar{f}_i}{\bar{g}}).$$

Since the two arguments of the resultant are relatively prime polynomials such a σ exists. Now

$$\sigma(a_1, \ldots, a_r) \neq 0 \qquad (7)$$

implies that

$$\mathrm{GCD}(\bar{g}, \sum_{i=1}^{r}a_i\frac{\bar{f}_i}{\bar{g}}) = 1 \qquad (8)$$

over $F(y_2, \ldots, z_n)[x_1]$. However, $\mathrm{ldcf}_{x_1}(\bar{g}) \in F(z_2, \ldots, z_n)$ so (8) remains true over $F(z_2, \ldots, z_n)[x_1, \ldots, y_n]$. Furtermore, since the substitution $x_i = y_i + z_i x_1$ is an isomorphism on that domain (7) must imply (6). By $\deg(\sigma) \leqslant \deg_{x_1}(\bar{g}) = d$ the existence of a_i's satisfing (6) is established.

We can now apply theorem 3 to $\sum_{i=1}^{r}a_i f_i$ in place of f and obtain a straight-line program for g. □

The second application provides a new solution to Strassen's problem on computing the numerator and denominator of a rational function. Let P be a straight-line program that computes f/g, $f, g \in F[x_1, \ldots, x_n]$, $\mathrm{GCD}(f, g) = 1$. At issue is to find a straight-line program for f and g. Clearly, we can compute fh and gh for some $h \in F[x_1, \ldots, x_n]$ by carrying the unreduced numerators and denominators of the intermediate rational functions explicitly along. As for the previous theorem we can find $a_1, b_1, a_2, b_2 \in F$ such that

$$\mathrm{GCD}(h, a_1 f + b_1 g) = 1, \quad \mathrm{GCD}(h, a_2 f + b_2 g) = 1, \quad a_1b_2 - a_2b_1 \neq 0.$$

Now using theorem 3 on both $a_i fh + b_i gh$ we can find straight-line programs for $a_i f + b_i g$ of length $(\deg(fg)\mathrm{len}(P))^{O(1)}$, $i = 1, 2$. From those f and g are computed as linear combinations. The length of the straight-line program obtained in such a fashion is asymptotically much longer than the one obtained by the Padé approximation solution for this problem [8]. We feel, however, that this new approach further emphasizes the usefulness of theorem 3 even to programs with divisions. For the record, let us state the following theorem, which extends Corollary 4.3 in [8] in case F is a small finite field.

Theorem 5: *Let P be a straight-line program of length l over $F(x_1, \ldots, x_n)$, F an arbitrary field, that computes f/g, $f, g \in F[x_1, \ldots, x_n]$ relatively prime, and let $d := \max(\deg(f), \deg(g))$. Then there exists a straight-line program Q over $F(x_1, \ldots, x_n)$ of parallel depth $O(\log(d)\log(d\,l))$ and size $(d\,l)^{O(1)}$ that also computes f/g.*

Proof: The construction compounds the following results.

P

$\downarrow$ by the above, or by [8], algorithm Rational Numerator or Denominator

Q_1 over $F(\theta)(x_1, \ldots, x_n)$, θ algebraic over F, that computes f and g separately

$\downarrow$ by [16] (see also [6], Theorem 7.1)

Q_2 over $F(\theta)[x_1, \ldots, x_n]$ that computes f, g

$\downarrow$ by a variation of the Lemma in §3

Q_3 over $F[x_1, \ldots, x_n]$ that computes f, g

$\downarrow$ by [19] or by [10]

Q_4 over $F[x_1, \ldots, x_n]$ that computes f and g in parallel

$\downarrow$ divide f by g

Q □

6. Conclusion

This article proves theorems on polynomial straight-line complexity for higher derivatives, roots, factors, and greatest common divisors derived from polynomials given by straight-line programs that can have arbitrarily high degree. It thus extends the theory of closure properties of p-computable polynomials [18], [8], to polynomials of unbounded input degree. We conclude with a collection of carefully considered open problems in the theory of straight-line complexity of polynomials.

Problem 1 (Strassen [17], §7, Problem 1): Can theorem 1 be combined with the Baur and Strassen result [1], that is given $f \in F(x_1, \ldots, x_n)$ by a straight-line program of length l, can all $\partial^k f / \partial x_i^k$, $1 \leqslant i \leqslant n$, be computed by a straight-line program of length $O(k^2 l)$?

Problem 2 (Moses [11], Strassen [17], §9, Problem 2): Given $f \in \mathbf{C}[x]$, $\mathbf{C}$ the complex numbers, by a division-free straight-line program over $\mathbf{C}[x]$ of length l, can $\int f(x)dx$ be computed by a straight-line program of length $l^{O(1)}$?

Problem 3: Can the condition $\mathrm{GCD}(g, f/g) = 1$ in theorem 3 be eliminated keeping $\mathrm{len}(Q) = (l\ d)^{O(1)}$? A positive answer to this problem would imply the following: For $f \in \mathbf{C}[x_1, \ldots, x_n]$ consider a zero-test tree of minimal height h for f, which includes straight-line code segements and tests $v_\lambda \mathrel{?=} 0$ at which the computation forks, where v_λ is a previously computed intermediate result. The leaves in the tree output $f = 0$ or $f \neq 0$, which must be true if we execute the tree for any specialization in $\mathbf{C}^n$ of the variables x_i. Then a solution to this problem implies that f can be computed by a straight-line program of length $h \deg(f)^{O(1)}$.

Problem 4: Can theorem 4 be proven with $\mathrm{len}(Q) = l^{O(1)}$, that is for arbitrarily high degrees of the GCD?

Problem 5: If $f \in F[x]$ has straight-line complexity l with divisions, can f be computed by a division-free straight-line program of length $l^{O(1)}$?

Problem 6: In theorem 5, does there hold a lower bound on the depth better than the trivial $\Omega(\log(d))$?

Acknowledgements: Theorem 1 was jointly derived with B. David Saunders. Volker Strassen first indicated to me the consequence of a solution to problem 3, and later Thomas Spencer refreshed my memory. Problem 2 was first brought to my attention by Leslie Valiant, problem 4 by Joos Heintz and Malte Sieveking, and problem 5 by Charles Rackoff. I also thank Lakshman Yagati for his comments.

References

1. W. Baur and V. Strassen, "The complexity of partial derivatives," *Theoretical Comp. Sci.*, vol. 22, pp. 317-330, 1983.
2. R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun, "Fast solution of Toeplitz systems of equations and computation of Padé approximants," *J. Algorithms*, vol. 1, pp. 259-295, 1980.
3. W. S. Brown and J. F. Traub, "On Euclid's algorithm and the theory of subresultants," *J. ACM*, vol. 18, pp. 505-514, 1971.
4. D. G. Cantor and E. Kaltofen, "Fast multiplication of polynomials with coefficients from an arbitrary ring," Manuscript, March 1987.
5. J. von zur Gathen, "Irreducibility of multivariate polynomials," *J. Comp. System Sci.*, vol. 31, pp. 225-264, 1985.
6. E. Kaltofen, "Greatest common divisors of polynomials given by straight-line programs," *Math. Sci. Research Inst. Preprint*, vol. 01918-86, Berkeley, CA, 1986. Expanded version to appear in *J. ACM.* Preliminary version under the title "Computing with polynomials given by straight-line programs I: Greatest common divisors" in *Proc. 17th ACM Symp. Theory Comp.*, pp. 131-142, 1985.

7. E. Kaltofen, "Factorization of polynomials given by straight-line programs," *Math. Sci. Research Inst. Preprint*, vol. 02018-86, Berkeley, CA, 1986. To appear in: "Randomness in Computation," *Advances in Computing Research*, S. Micali ed., JAI Press Inc., Greenwich, CT, January 1987.

8. E. Kaltofen, "Uniform closure properties of p-computable functions," *Proc. 18th ACM Symp. Theory Comp.*, pp. 330-337, 1986.

9. D. E. Knuth, *The Art of Programming, vol. 2, Semi-Numerical Algorithms, ed. 2,* Addison Wesley, Reading, MA, 1981.

10. G. L. Miller, V. Ramachandran, and E. Kaltofen, "Efficient parallel evaluation of straight-line code and arithmetic circuits," *Proc. AWOC '86, Springer Lec. Notes Comp. Sci.*, vol. 227, pp. 236-245, 1986.

11. J. Moses, "Algebraic simplification: A guide for the perplexed," *Commun. ACM*, vol. 14, pp. 548-560, 1971.

12. J. Moses and D. Y. Y. Yun, "The EZ-GCD algorithm," *Proc. 1973 ACM National Conf.*, pp. 159-166, 1973.

13. D. R. Musser, "Multivariate polynomial factorization," *J. ACM*, vol. 22, pp. 291-308, 1975.

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

15. V. Strassen, "Berechnung und Programm I," *Acta Inf.*, vol. 1, pp. 320-335, 1972. (In German).

16. V. Strassen, "Vermeidung von Divisionen," *J. reine u. angew. Math.*, vol. 264, pp. 182-202, 1973. (In German).

17. V. Strassen, "Algebraische Berechnungskomplexität," in *Anniversary of Oberwolfach 1984*, Perspectives in Mathematics, pp. 509-550, Birkhäuser Verlag, Basel, 1984. (In German).

18. L. Valiant, "Reducibility by algebraic projections," *L'Enseignement mathématique*, vol. 28, pp. 253-268, 1982.

19. L. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff, "Fast parallel computation of polynomials using few processors," *SIAM J. Comp.*, vol. 12, pp. 641-644, 1983.

20. P. S. Wang, "The EEZ-GCD algorithm," *SIGSAM Bulletin*, vol. 14/2, pp. 50-60, 1980.

21. D. Y. Y. Yun, "The Hensel lemma in algebraic manipulation," Ph.D. Thesis, M.I.T., 1974. Reprint: Garland Publ., New York 1980.

22. H. Zassenhaus, "On Hensel factorization I," *J. Number Theory*, vol. 1, pp. 291-311, 1969.