SIAM J. COMPUT.
Vol. 9, No. 1, February 1980

0097-5397/80/0901-0004 $01.00/0

ON THE COMPLEXITY OF COMPOSITION AND GENERALIZED COMPOSITION OF POWER SERIES*

R. P. BRENT† AND J. F. TRAUB‡

Abstract. Let $F(x)=f_1x+f_2x^2+\cdots$ be a formal power series over a field Δ. Let $F^{[0]}(x)=x$ and for $q=1,2,\cdots$, define $F^{[q]}(x)=F^{[q-1]}(F(x))$. The obvious algorithm for computing the first n terms of $F^{[q]}(x)$ is by the composition analogue of repeated squaring. This algorithm has complexity about $\log_2 q$ times that of a single composition. Brent showed that the factor $\log_2 q$ can be eliminated in the computation of the first n terms of $(F(x))^q$ by a change of representation, using the logarithm and exponential functions. We show the factor $\log_2 q$ can also be eliminated for the composition problem, unless the complexity of composition is quasi-linear.

$F^{[q]}(x)$ can often, but not always, be defined for more general q. We give algorithms and complexity bounds for computing the first n terms of $F^{[q]}(x)$ whenever it is defined.

We conclude the paper with some open problems.

Key words. composition, fast algorithms, formal power series, symbolic computation, generalized composition, functional equations, Schroeder function, iteration, similarity transformations

1. Introduction.

Let

$$F(x)=f_1x+f_2x^2+\cdots \tag{1.1}$$

be a formal power series over a field Δ. Let $F^{[0]}(x)=x$ and for $q=1,2,\cdots$, define the *q-composite* of F by

$$F^{[q]}(x)=F^{[q-1]}(F(x)). \tag{1.2}$$

The q-composite may also be called the q-iterate. Let $H(x)$ be the reversion of $F(x)$, i.e., the power series inverse to $F(x)$ under composition. For $q=1,2,\cdots$, define

$$F^{[-q]}(x)=H^{[q]}(x). \tag{1.3}$$

As we shall see below, the q-composite of F can often (but not always) be defined for more general q. If q is not an integer, we shall call $F^{[q]}(x)$ a *generalized q-composite*. We confine ourselves to the case that $F^{[q]}(x)$ is a power series. One important special case of generalized composition is $q=1/r$, where r is an integer. Then $G=F^{[1/r]}(x)$ is an rth root of F under composition, and satisfies the equation $G^{[r]}(x)=F(x)$.

Let

$$F_n(x)=f_1x+\cdots+f_nx^n, \tag{1.4}$$

$$G(x)=F^{[q]}(x)=g_1x+g_2x^2+\cdots, \tag{1.5}$$

$$G_n(x)=g_1x\cdots+g_nx^n. \tag{1.6}$$

Given q and $F_n(x)$, we want to compute $G_n(x)$.

In this paper we shall give algorithms and complexity bounds for computing $G_n(x)$ whenever it is defined. For integer q these algorithms are asymptotically faster than the obvious algorithms.

---

* Received by the editors May 23, 1978, and in revised form December 18, 1978. This research was supported in part by the National Science Foundation under Grant MCS75-222-55 and the Office of Naval Research under Contract N00014-76-C-0370, NR 044-422. The work of the first author was also supported in part by the National Science Foundation under Grant 1-442427-21164-2 at the University of California at Berkeley.

† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania. Presently at Computer Science Department, Australian National University, Canberra, Australia.

‡ Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

We discuss the last point. Let $\mathrm{COMP}_1(n)$ denote the complexity of computing the first n terms of $F(F(x))$, and let q be a power of two. Then the obvious algorithm for computing $G_n(x)$ is by the composition analogue of "repeated squaring," and has complexity $\mathrm{COMP}_1(n) \lg q$. (We shall denote $\log_2$ by lg.) Can we eliminate the multiplicative factor of $\lg q$?

An analogous problem is that of computing $R_n(x)$, the first n terms of $(F(x))^q$. Asymptotically in n, the complexity of forming $R_n(x)$ is the same as the complexity of a single multiplication of two polynomials of degree n. This follows from the observation that if $A(x)$ is a power series with constant term unity, then $(A(x))^q \equiv \exp(q \ln A(x))$. This may be viewed as a change of representation of $A(x)$ to a new representation where multiplication is replaced by addition, followed by the inverse change of representation. Brent (1976) showed that the change of representation could be computed "fast."

This suggests asking whether there is a change of representation which reduces composition to multiplication. We shall see that there is, at least in the "regular" case (see § 3). Furthermore, the change of representation can be computed "fast." *This enables us to eliminate the multiplicative factor of* lg q (unless the complexity of composition is quasi-linear). In addition we shall show (§§ 4–6) that even in the "nonregular" cases we can still eliminate this factor. A bonus is that our algorithms apply for non-integer q (so long as $F^{[q]}(x)$ is a well-defined power series).

The problem of composition and generalized composition occurs in many applications including branching processes, asymptotic analysis, difference equations, numerical analysis, and dynamical systems. See, for example, Aczél (1966), Cherry (1964), de Bruijn (1970), Feller (1957), Harris (1963), Henrici (1974), Knuth (1969), Kuczma (1968), Levy and Lessman (1961), and Melzak (1973). The study of composition (often called iteration) may be viewed as a major subfield of mathematics. See Aczél (1966), Gross (1972), and Kuczma (1968) for very extensive bibliographies. However, little attention seems to have been given to the development of algorithms for computing $F^{[q]}(x)$ when $F(x)$ is a given power series.

The following conventions are adopted below. We deal with formal power series; that is, we do not concern ourselves with convergence. Power series are denoted by upper case letters such as $A(x)$ or simply A, with coefficients denoted by the corresponding lower case letters such as a_i. If $A(x) = a_k x^k + a_{k+1}x^{k+1} + \cdots$, $a_k \neq 0$, then $\operatorname{ord}(A) = k$. It is convenient to define $\operatorname{ord}(0) = \infty$. If $\operatorname{ord}(B - C) \geqq k$ we write $B = C + O(x^k)$. The polynomial $b_0 + b_1 x + \cdots + b_{k-1}x^{k-1}$ is denoted either by $B(x) \bmod x^k$ or by $B_{k-1}(x)$. It is convenient to define $\gamma(n, q) = O(\delta(n, q))$ to mean $|\gamma(n, q)| \leqq K|\delta(n, q)|$ for all sufficiently large integer n for all q under consideration.

We summarize the remainder of the paper. Our complexity model is specified in § 2. In § 3 we study the "regular" case when the multiplier f_1 is such that $f_1 \neq 0$, $f_1^m \neq 1$, $m = 1, 2, \cdots$. In the following three sections we consider the cases $f_1 = 0$; $f_1 = 1$; $f_1^m = 1$, integer $m > 1$, but $f_1 \neq 1$, respectively.

In each of §§ 3, 4, and 5 we define an "auxiliary" function, demonstrate it can be computed fast by "divide and conquer," and show how it can be used to compute $F^{[q]}$. The case studied in § 6 can be reduced to that of § 5. In the concluding section we state a theorem (Theorem 7.1) summarizing our results, state the defining equations for all cases, and mention some open problems.

2. Complexity model. In this section we state our complexity model and summarize the complexity results needed below. We assume that scalar arithmetic operations are performed exactly and have unit cost. Thus our time bounds are invalid if, for

example, exact rational arithmetic is used. However, our algorithms should still be useful in this case.

Given power series $A(x)$ and $B(x)$, the time required to compute $A(x)B(x) \bmod x^n$ is denoted by MULT (n). If ord $(B) \geqq 1$, the time required to compute $A(B(x)) \bmod x^n$ is denoted by COMP (n). We assume that MULT (n) and COMP (n) satisfy certain plausible regularity conditions (see Brent and Kung (1978, § 1)). Then Brent and Kung (1978) show

$$\text{COMP}(n) = O(\min(n^{(1+r)/2}, (n \lg n)^{1/2}\,\text{MULT}(n))), \tag{2.1}$$

if matrix multiplication has complexity $O(n^r)$. If the field Δ is such that fast algorithms like the FFT are available, then

$$\text{MULT}(n) = O(n \lg n) \tag{2.2}$$

(see Borodin and Munro (1975)), and it follows from (2.1) that

$$\text{COMP}(n) = O((n \lg n)^{3/2}). \tag{2.3}$$

The bounds in this paper will be expressed in terms of the complexity function

$$\text{COMP}_2(n) = \sum_{j=0}^{\lfloor \lg n \rfloor} 2^j\,\text{COMP}(\lceil 2^{-j} n \rceil). \tag{2.4}$$

Assume (with notation as in Knuth (1976))

$$\text{COMP}(n) = \Theta(n^\alpha s(n)), \tag{2.5}$$

where $\alpha \geqq 1$, and $s(n)$ is a monotonic increasing positive function. (For example, $s(n)$ might be $(\lg n)^\beta$ for some constant $\beta \geqq 0$.) Then, $s(n) = O(n^\varepsilon)$ for all $\varepsilon > 0$.

$$\text{COMP}_2(n) = \begin{cases} O(\text{COMP}(n)), & \text{if } \alpha > 1, \\ O(\text{COMP}(n) \lg n), & \text{if } \alpha = 1. \end{cases} \tag{2.6}$$

If the field Δ is such that (2.3) holds, then $\alpha \leqq \frac{3}{2}$. If $\alpha > 1$, then $\text{COMP}_2(n)$ may be replaced by $O(\text{COMP}(n))$ in our bounds.

If $\alpha = 1$, we say COMP (n) is quasi-linear. If $\alpha = 1$ and q is a fixed integer, then "repeated squaring" is asymptotically faster than our algorithms. Of course, if q is not an integer, then "repeated squaring" is not an alternative to our algorithms. If $\alpha > 1$, our result (that we can eliminate the multiplicative factor of $\lg q$) holds for all fields of characteristic zero and all finite fields of characteristic p greater than n.

If f_1^q is defined, we denote the complexity of computing f_1^q by POWER (q). If q is a positive integer, then POWER $(q) = O(\lg q)$. To eliminate POWER (q) terms we sometimes assume that f_1^q is given.

In Brent (1976) it is shown that the complexity of computing $\ln(1 + A(x)) \bmod x^n$ is $O(\text{MULT}(n))$ for any power series A, ord $(A) > 0$. Using Brent's results it can be shown that the complexity of computing $(B(x))^q \bmod x^n$ is $O(\text{MULT}(n) + \text{POWER}(q))$. By Brent and Kung (1978, Lemma 4.2) MULT $(n) = O(\text{COMP}(n))$, so we can absorb MULT (n) into COMP (n) in our analyses.

Recall that $\text{COMP}_1(n)$ was defined as the complexity of computing the first n terms of $F(F(x))$. It can be shown, by means similar to the proof of Brent and Kung (1978) that the complexity of reversion and composition are asymptotically equal, that $\text{COMP}(n) = O(\text{COMP}_1(n))$.

3. The regular case. In this section we study the computation of $F^{[q]}(x)$ when $f_1 \neq 0$, $f_1^m \neq 1$, $m = 1, 2, \cdots$. We call this the *regular* case. Define the Schroeder

function $S(x)$ by

$$S(F(x)) = f_1 S(x) \qquad \text{ord}\,(S) = 1, \qquad s_1 = 1. \tag{3.1}$$

$S(x)$ exists and is unique (Schroeder (1871), Kuczma (1968, Chap. 6)). See also Parker (1977). It is easy to prove that, for all integer q,

$$F^{[q]}(x) = S^{[-1]}(f_1^q S(x)). \tag{3.2}$$

$S(x)$ and $S^{[-1]}(x)$ play the role that the logarithm and exponential functions play in computing $(F(x))^q$ fast. They reduce self-composition to scalar powering.

Equation (3.1) has an interesting matrix interpretation. To the formal power series $F(x)$ and $S(x)$, we may associate infinite matrices M_F and M_S, respectively (see, for example, Henrici (1974, p. 45)). Then $S^{[-1]}(x)$ is associated with M_S^{-1}. It is easy to show that (3.1) corresponds to a matrix similarity transformation which transforms M_F to a diagonal matrix with diagonal elements f_1^k, $k = 1, 2, \cdots$. The conditions $f_1 \neq 0$, $f_1^m \neq 1$ ensure that the eigenvalues of M_F are all distinct.

If q is not an integer but q and the scalar f_1 are such that f_1^q is defined, then (3.2) may be used to define $F^{[q]}$. We shall use the "divide and conquer" strategy to compute $S(x)$ fast and then show how to compute $F^{[q]}$ from (3.2) in total time $O(\text{COMP}_2(n) + \text{POWER}(q))$.

Although we wish to solve the functional equation (3.1), to make the "divide and conquer" strategy work we embed (3.1) in the more general linear functional equation

$$A(x)W(F(x)) - B(x)W(x) - C(x) = 0, \tag{3.3}$$

where W is the unknown. Note that this equation includes reversion as a special case. The "divide and conquer" algorithm introduced to solve (3.3) may therefore be used to revert power series. This algorithm is different from the one derived by Newton iteration and given in Brent and Kung (1978).

Lemma 3.1 gives the basis for a "divide and conquer" algorithm for solving (3.3). The proof is by substitution. Lemma 3.2 gives sufficient conditions for the existence of a formal solution, and Lemma 3.3 gives an upper bound on the time required to compute an approximate solution.

LEMMA 3.1. *If n, p are nonnegative integers,* $\text{ord}\,(F) \geqq 1$,

$$A(x)U(F(x)) - B(x)U(x) - C(x) = x^n R(x) \tag{3.4}$$

and

$$A(x)(F(x)/x)^n V(F(x)) - B(x)V(x) + R(x) = O(x^p), \tag{3.5}$$

then

$$A(x)W(F(x)) - B(x)W(x) - C(x) = O(x^{n+p}) \tag{3.6}$$

where

$$W(x) = U(x) + x^n V(x). \tag{3.7}$$

Remark 3.1. If Lemma 3.1 is applied for $n = p = 2^j$, $j = 0, 1, 2, \cdots$, we have an algorithm for approximating $W(x)$ which is quadratically convergent in the sense of Kung and Traub (1978).

LEMMA 3.2. *If* $\text{ord}\,(F) \geqq 1$,

$$a_0 f_1^m \neq b_0 \quad \textit{for } m = 1, 2, 3, \cdots \tag{3.8}$$

and

$$a_0 = b_0 \quad \textit{implies} \quad c_0 = 0 \tag{3.9}$$

then there is a formal power series W, *satisfying* (3.3), *with* ord $(W) = 0$ *unless* $c_0 = 0$ *and* $a_0 \neq b_0$.

Proof. We shall construct $w_0, w_1, \cdots$ such that $W(x) = \sum_{j=0}^{\infty} w_j x^j$ satisfies (3.3). We let

$$W_m(x) = \sum_{j=0}^{m} w_j x^j \tag{3.10}$$

and show by induction on m that, for some power series $R_{m+1}(x)$,

$$A(x)W_m(F(x)) - B(x)W_m(x) - C(x) = x^{m+1}R_{m+1}(x) = O(x^{m+1}). \tag{3.11}$$

Let

$$w_0 = \begin{cases} 1 & \text{if } a_0 = b_0, \\ c_0/(a_0 - b_0) & \text{otherwise.} \end{cases} \tag{3.12}$$

Then (3.11) holds for $m = 0$, starting the induction. Assuming that (3.11) holds for $m \geqq 0$, we define

$$w_{m+1} = \frac{R_{m+1}(0)}{b_0 - a_0 f_1^{m+1}} \tag{3.13}$$

and apply Lemma 3.1 (with $n = m+1$, $p = 1$, $U = W_m$, $V = w_{m+1}$) to deduce that (3.11) holds with m replaced by $m+1$. Thus, the result follows by induction on m. □

LEMMA 3.3. *Suppose that* $w_0, \cdots, w_{n-1}$ *can be found in time* $t(n)$ *whenever the conditions of Lemma* 3.2 *apply. Then*

$$t(2n) \leqq 2t(n) + \text{COMP}\,(2n) + O(\text{MULT}\,(n)). \tag{3.14}$$

Proof. In time $t(n)$ we find $u_0, \cdots, u_{n-1}$ such that (3.4) holds for some power series $R(x)$, where $U(x) = \sum_{j=0}^{n-1} u_j x^j$. Compute $U(F(x)) \bmod x^{2n}$ in time COMP $(2n)$, and then find

$$R(x) = \frac{A(x)U(F(x)) - B(x)U(x) - C(x)}{x^n} \bmod x^n \tag{3.15}$$

in time $O(\text{MULT}\,(n))$. [Note: MULT $(2n) = O(\text{MULT}\,(n))$.]

Since ord $(F) \geqq 1$, $F(x)/x$ is a power series, and by an algorithm given in Brent (1976) we can compute $(F(x)/x)^n \bmod x^n$, and thus

$$\tilde{A}(x) = A(x)(F(x)/x)^n \bmod x^n, \tag{3.16}$$

in time $O(\text{MULT}\,(n))$. Now (3.5) with $p = n$ is just

$$\tilde{A}(x)V(F(x)) - B(x)V(x) + R(x) = O(x^n),$$

so we can find $v_0, \cdots, v_{n-1}$ in time $t(n)$. Using Lemma 3.1, we take

$$w_j = \begin{cases} u_j & \text{if } 0 \leqq j < n, \\ v_{j-n} & \text{if } n \leqq j < 2n, \end{cases}$$

and the result follows. □

COROLLARY 3.1. *With the notation of Lemma* 3.3,

$$t(n) = O(\text{COMP}_2\,(n)). \tag{3.17}$$

Proof. This follows from Lemma 3.3, the definition of $\mathrm{COMP}_2(n)$, and the fact that $\mathrm{MULT}(n) = O(\mathrm{COMP}(n))$. □

COROLLARY 3.2. *If* $\mathrm{ord}(F) = 1$ *and* $f_1^m \neq 1$ *for* $m = 1, 2, \cdots$, *then we can compute the first n coefficients,* $s_0, \cdots, s_{n-1}$ *of the Schroeder function* $S(x)$ *satisfying* (3.1) *in time* $O(\mathrm{COMP}_2(n))$.

Proof. We solve a special case of (3.3), namely

$$(F(x)/x)W(F(x)) - f_1 W(x) = 0, \tag{3.18}$$

to obtain $w_0, \cdots, w_{n-2}$ by the method of Lemma 3.2. Then $S(x) = xW(x)$ satisfies (3.1) mod x^n, so $s_0 = 0$ and $s_j = w_{j-1}$ for $j = 1, \cdots, n-1$. □

THEOREM 3.1. *Assume* $\mathrm{ord}(F) = 1$, $f_1^m \neq 1$ *for* $m = 1, 2, \cdots$. *Let* f_1^q *be defined and let*

$$G(x) = F^{[q]}(x). \tag{3.19}$$

Then $g_0, \cdots, g_{n-1}$ *can be computed in time*

$$O(\mathrm{COMP}_2(n) + \mathrm{POWER}(q)). \tag{3.20}$$

Proof. Using the method of Corollary 3.2, we compute $S_{n-1}(x) = \sum_{j=1}^{n-1} s_j x^j$ such that $s_1 \neq 0$ and

$$S_{n-1}(F(x)) = f_1 S_{n-1}(x) + O(x^n) \tag{3.21}$$

in time $O(\mathrm{COMP}_2(n))$. Now

$$S_{n-1}(G(x)) = f_1^q S_{n-1}(x) + O(x^n), \tag{3.22}$$

and thus

$$G(x) = S_{n-1}^{[-1]}(f_1^q S_{n-1}(x)) + O(x^n). \tag{3.23}$$

Using the method of Brent and Kung (1978), we can compute $S_{n-1}^{[-1]}(x)$ mod x^n in time $O(\mathrm{COMP}(n))$, and the $g_0, \cdots, g_{n-1}$ are obtained from (3.23) in time $\mathrm{COMP}(n) + \mathrm{POWER}(q)$. The result follows. □

Remark 3.2. The condition $f_1^m \neq 1$ is necessary so that the divisor in (3.13) is nonzero. Thus, we need only assume that $f_1^m \neq 1$ for $m = 1, 2, \cdots, n-2$. If F is a formal power series over a finite field with characteristic p, then it is necessary to assume $n \leqq p$.

The proofs above are constructive and give the following two algorithms.

ALGORITHM 3.1. The algorithm $\mathcal{A}(A, B, C, F, W, m)$ finds $w_0, \cdots, w_{m-1}$ such that $W(x)$ satisfies (3.3). It is defined recursively by:

if $m = 1$ **then** {use equation (3.12) to define w_o} **else**

{$n \leftarrow \lceil m/2 \rceil$;

$\mathcal{A}(A, B, C, F, U, n)$;

Compute R using equation (3.15);

Compute $\tilde{A}$ using equation (3.16);

$\mathcal{A}(\tilde{A}, B, -R, F, V, n)$;

for $j \leftarrow 0$ step 1 **until** $n-1$ **do** $\{w_j \leftarrow u_j;\ w_{n+j} \leftarrow v_j\}$}.

ALGORITHM 3.2. The following algorithm computes $G(x) = F^{[q]}(x)$ if the conditions of Theorem 3.1 apply:

1. Take $A(x) = F(x)/x$, $B(x) = f_1$, $C(x) = 0$ and find $w_0, \cdots, w_{n-2}$ such that $W(x)$

satisfies (3.3) by applying $\mathcal{A}(A, B, C, F, W, n-1)$ (see Algorithm 3.1).

2. Let $s_0 = 0$, $s_i = w_{i-1}$ for $j = 1, \cdots, n-1$, and compute $S^{[-1]}(f_1^q S(x)) \bmod x^n$ using the composition and reversion algorithms of Brent and Kung (1978).

4. Multiplier zero. In this section we study the case $f_1 = 0$. Since the problem is trivial if $F(x) \equiv 0$, we can assume ord $(F) = k$, $1 < k < \infty$. We define auxiliary power series $S(x)$ by

$$S(F(x)) = f_k(S(x))^k, \qquad \text{ord}\,(S) = 1, \qquad s_1 = 1. \tag{4.1}$$

This reduces to Schroeder's equation (3.1) if $k = 1$. By induction on q we have, for all positive integer q,

$$F^{[q]}(x) = S^{[-1]}\{f_k^{(k^q-1)/(k-1)}[S(x)]^{k^q}\}. \tag{4.2}$$

Remark 4.1. The restriction to positive integer q is essential here. For example, take $F = x^3$. Then $F^{[q]}$ does not exist as a power series for $q = -1$ or $q = \frac{1}{2}$.

The following lemmas reduce the solution of (4.1) to problems solved in the previous section.

LEMMA 4.1. *If* ord $(F) = k > 1$ *the equation.*

$$W(F(x)) - kW(x) + \{(k-1) + \ln [F(x)/(f_k x^k)]\} = 0 \tag{4.3}$$

has a solution $W(x)$, *and* $w_0, \cdots, w_{n-1}$ *can be computed in time* $O(\text{COMP}_2(n))$.

Proof. Lemmas 3.1 to 3.3 are applicable to (4.3), so $W(x)$ exists and $w_0, \cdots, w_{n-1}$ can be computed in time $O(\text{COMP}_2(n))$ by the method used in the proof of Lemma 3.3. □

LEMMA 4.2. *If* ord $(F) = k > 1$ *and* $W(x)$ *satisfies* (4.3), *then*

$$S(x) = x \exp(W(x) - 1) \tag{4.4}$$

satisfies (4.1).

Proof. Substitute $W(x) = 1 + \ln(S(x)/x)$ in (4.3). From (3.12), $w_0 = 1$, so $S(x)$ is a power series. □

Using the algorithm of Brent (1976) we can compute the first n coefficients of

$$[S(x)/x]^{k^q} = \exp[k^q(W(x) - 1)]$$

in time $O(\text{MULT}(n))$ once $w_0, \cdots, w_{n-1}$ are known. We can also compute $f_k^{(k^q-1)/(k-1)}$ in time POWER $((k^q - 1)/(k-1))$. Then, using a slight modification of the composition and reversion algorithms of Brent and Kung (1978) we have:

THEOREM 4.1. *Assume* ord $(F) = k > 1$, $q \geqq 1$ *is a positive integer, and*

$$G(x) = F^{[q]}(x)/x^{k^q}. \tag{4.5}$$

Then $g_0, \cdots, g_{n-1}$ *can be computed in time*

$$O(\text{COMP}_2(n) + \text{POWER}((k^q - 1)/(k-1))). \tag{4.6}$$

5. Multiplier unity. Now we consider the case that the multiplier f_1 is equal to unity. We define an auxiliary function T by

$$T(F(x)) = F'(x)T(x), \qquad \text{ord}\,(T) = \text{ord}\,(F(x) - x). \tag{5.1}$$

$T(x)$ exists and is unique up to a scaling factor (Kuczma (1968, Lemma 9.4)). Let $G(x) = F^{[q]}(x)$. Then we show below that $G(x)$ may be computed from the equation

$$T(G(x)) = G'(x)T(x). \tag{5.2}$$

Remark 5.1. T may also exist if $f_1 \neq 1$. If F is such that the Schroeder function S exists, then $T(x) = cS(x)/S'(x)$, where c is a nonzero constant.

Example 5.1. If $F(x) = 2x + x^2$, then $S(x) = \ln(1+x)$, $T(x) = (1+x)\ln(1+x)$, $F^{[q]}(x) = (1+x)^{2^q} - 1$. If $F(x) = x/(1-x)$, then $T(x) = x^2$.

Although we wish to solve the functional equation (5.1), as before we need to embed (5.1) in a more general equation. Throughout this section we define d by $F(x) = x + f_d x^d + \cdots$, $f_d \neq 0$, and let k be any integer greater than d. Then we shall solve

$$x^{1-d}[(F(x)/x)^k Y(F(x)) - F'(x)Y(x)] - A(x) = 0 \tag{5.3}$$

for $Y(x)$.

Lemma 5.1 gives the basis for a "divide and conquer" algorithm for solving (5.3). Lemma 5.2 gives sufficient conditions for the existence of a formal solution, and Lemma 5.3 gives an upper bound on the time required to compute an approximate solution. Lemma 5.4 establishes (5.2) and gives a sufficient condition for G to be uniquely defined.

LEMMA 5.1. *Let n, p be nonnegative integers. If*

$$x^{1-d}[(F(x)/x)^k U(F(x)) - F'(x)U(x)] - A(x) = x^n R(x) \tag{5.4}$$

and

$$x^{1-d}[(F(x)/x)^{k+n} V(F(x)) - F'(x)V(x)] + R(x) = O(x^p), \tag{5.5}$$

then

$$x^{1-d}[(F(x)/x)^k W(F(x)) - F'(x)W(x)] - A(x) = O(x^{n+p}) \tag{5.6}$$

where

$$W(x) = U(x) + x^n V(x). \tag{5.7}$$

Proof. By direct substitution. Note that since $F(x) = x + f_d x^d + \cdots$, the terms in square brackets in (5.4) to (5.6) have ord $\geqq d - 1$. □

LEMMA 5.2. *There is a formal power series $Y(x)$ such that*

$$x^{1-d}[(F(x)/x)^k Y(F(x)) - F'(x)Y(x)] = A(x). \tag{5.8}$$

Proof. We shall construct $y_0, y_1, \cdots$ such that $Y(x) = \sum_{i=0}^{\infty} y_i x^i$ satisfies (5.8). Recall our assumption that $k > d = \text{ord}\,(F(x) - x)$. Take

$$y_0 = \frac{a_0}{(k-d)f_d} \tag{5.9}$$

and let

$$Y_n(x) = \sum_{i=0}^{n} y_i x^i. \tag{5.10}$$

Thus

$$x^{1-d}[(F(x)/x)^k Y_{n-1}(F(x)) - F'(x)Y_{n-1}(x)] - A(x) = x^n R_n(x) \tag{5.11}$$

is true for $n = 1$ (where R_n is some power series). Define

$$y_n = \frac{-R_n(0)}{(k+n-d)f_d} \tag{5.12}$$

for $n \geqq 1$. Using Lemma 5.1 with $p = 1$, it is straightforward to prove that (5.11) holds for all $n \geqq 1$, by induction on n. Thus, the result follows. □

LEMMA 5.3. *Suppose that $y_0, \cdots, y_{n-1}$ can be found in time $t_2(n)$ whenever the conditions of Lemma 5.2 apply. Then*

$$t_2(2n) \leqq 2t_2(n) + \text{COMP}(2n+d-1) + O(\text{MULT}(n)). \tag{5.13}$$

Proof. In time $t_2(n)$ we find $u_0, \cdots, u_{n-1}$ such that (5.4) holds for some power series $R(x)$, if $U(x) = \sum_{j=0}^{n-1} u_j x^j$. Compute $U(F(x)) \bmod x^{2n+d-1}$ and then $R(x) \bmod x^n$ from (5.4). Then find $v_0, \cdots, v_{n-1}$ such that $V(x)$ satisfies (5.5) with $p = n$ (this takes time $t_2(n) + O(\text{MULT}(n))$). From Lemma 5.1 we can take

$$y_j = \begin{cases} u_j & \text{if } 0 \leqq j < n, \\ v_{j-n} & \text{if } n \leqq j < 2n, \end{cases}$$

so we get $y_0, \cdots, y_{2n-1}$ in time $2t_2(n) + \text{COMP}(2n+d-1) + O(\text{MULT}(n))$ as required. □

COROLLARY 5.1. *With the notation of Lemma 5.3, $t_2(n) = O(\text{COMP}_2(n))$.*

COROLLARY 5.2. *There exists a formal power series $T(x)$ such that ord $(T) = d$ and*

$$T(F(x)) = F'(x)T(x). \tag{5.14}$$

Moreover, $t_d, \cdots, t_{n-1}$ can be found in time $O(\text{COMP}_2(n))$.

Proof. If

$$A(x) = x^{-2d}[F'(x)x^d - (F(x))^d] = (f_{d+1} - f_2^2) + \cdots \tag{5.15}$$

and

$$x^{1-d}[(F(x)/x)^{d+1} Y(F(x)) - F'(x)Y(x)] = A(x) \tag{5.16}$$

then

$$T(x) = x^d + x^{d+1} Y(x) \tag{5.17}$$

satisfies (5.14). Thus, the result follows from Lemma 5.2 and Corollary 5.1. □

LEMMA 5.4. *Let q be an integer, T satisfy (5.14), and*

$$G(x) = F^{[q]}(x). \tag{5.18}$$

Then

$$T(G(x)) = G'(x)T(x), \tag{5.19}$$

and the power series $G(x)$ is uniquely determined by (5.19) and the condition

$$\text{ord}\,(G(x) - x - qf_d x^d) > d. \tag{5.20}$$

Proof. It is easy to prove (5.19) by induction for positive q, and the result for negative q then follows. It is also easy to prove by induction that (5.20) holds if G is defined by (5.18). From Lemma 9.4 of Kuczma (1968) the solution of (5.19) satisfying (5.20) is unique, so the result follows. □

One $T(x)$ is known, we can solve (5.19) for $G(x)$, using the "initial condition" (5.20). Since (5.19) is a nonlinear differential equation for G, we can use a Newton-type method as described in Brent and Kung (1978). The algorithms are given below. First we summarize the result:

THEOREM 5.1. *Assume $f_1 = 1$ and let $G = F^{[q]}(x)$. Then $g_0, \cdots, g_{n-1}$ can be computed in time $O(\text{COMP}_2(n))$.*

Proof. First find $t_d, \cdots, t_{n-1}$ such that $T(x)$ satisfies (5.1), as in Corollary 5.2, in time $O(\text{COMP}_2(n))$. Then solve (5.19) and (5.20) by Algorithm 5.3 below (in time $O(\text{COMP}(n))$ to find $g_0, \cdots, g_{n-1}$. □

Remark 5.2. Note that q need not be an integer in Theorem 5.1. Kuczma (1968, Thm. 9.15] considers the question of when $F^{[q]}(x)$ is analytic. See also Baker (1964) and Szekeres (1964).

ALGORITHM 5.1. The algorithm $\mathfrak{B}(A, F, Y, k, d, n)$ finds $y_0, \cdots, y_{n-1}$ such that $Y(n)$ satisfies (5.8). It is assumed that $n > 0$, $a_0, \cdots, a_{n-1}$ and $f_1, \cdots, f_{d+n-1}$ are given, and that the conditions stated after Example 5.1 are satisfied. $\mathfrak{B}(A, F, Y, k, d, n)$ is defined recursively by:

if $n = 1$ **then** {define y_0 by (5.9)}

else {$p \leftarrow \lceil n/2 \rceil$;

$\mathfrak{B}(A, F, U, k, d, p)$;

Compute $U(F(x)) \bmod x^{d+2p-1}$;

Compute $R(x) \bmod x^p$ from (5.4) with n replaced by p;

$\mathfrak{B}(-R, F, V, k+p, d, p)$;

for $j \leftarrow 0$ step 1 **until** $j-1$ **do**

$\{y_j \leftarrow u_j;\ y_{p+j} \leftarrow v_j\}\}$.

ALGORITHM 5.2. The algorithm $\mathfrak{S}(F, T, d, n)$ finds $t_d, \cdots, t_{n-1}$ such that $T(x)$ satisfies (5.14). It is assumed that $f_1, \cdots, f_{n-1}$ are given and that the conditions of Corollary 5.2 are satisfied.

$Y \leftarrow 0$; $T \leftarrow 0$; $t_d \leftarrow 1$;

if $n > d+1$ **then** {compute $A(x) \bmod x^{n-d-1}$ from (5.15);

$\mathfrak{B}(A, F, Y, d+1, d, n-d-1)$;

for $j \leftarrow d+1$ **until** $n-1$ **do** $t_j \leftarrow y_{j-d-1}\}$.

ALGORITHM 5.3. The following algorithm computes $g_0, \cdots, g_{n-1}$, such that $G(x) = F^{[q]}(x)$. It is assumed that $t_d, \cdots, t_{n-1}$ have been computed using Algorithm 5.2.

$G \leftarrow x + qf_d x^d$;

$k \leftarrow 1$;

while $k + d < n$ **do**

$\{k \leftarrow \min(2k, n-d)$;

$$R \leftarrow \frac{T(G(x)) - G'(x)T(x)}{x^d T(x)} \bmod x^{k-1};$$

$$U \leftarrow \frac{d}{x} - \frac{T'(G(x))}{T(x)} \bmod x^{k-2};$$

$$E \leftarrow \exp\left(\int_0^x U(y)\,dy\right) \bmod x^{k-1};$$

$$V \leftarrow \frac{1}{E(x)} \int_0^x E(y)R(y)\,dy \bmod x^k;$$

$G \leftarrow G + x^d V \bmod x^{k+d}\}$.

Remark 5.3. It can be verified that all the quantities appearing on the lefthand sides in Algorithm 5.3 are indeed power series.

6. Multiplier nontrivial root of unity. In this section we consider the only remaining case: $f_1 \neq 1$, $f_1^m = 1$ for some integer $m > 1$. By Remark 3.2 we may assume $m \leqq n-2$. We also assume q is an integer.

Remark 6.1. The restriction to integer q is essential here. For example, let $F = -x + x^2 + x^3$. There is no formal power series for $F^{[1/2]}(x)$. That is, there is no power series $G(x)$ such that $G^{[2]}(x) = F(x)$ (Kuczma (1968, p. 304)).

In what follows we shall use the following algebraic relations:

$$F^{[p+q]}(x) = F^{[p]}(F^{[q]}(x)), \tag{6.1}$$

$$F^{[pq]}(x) = R^{[p]}(x), \quad \text{where } R(x) = F^{[q]}(x), \tag{6.2}$$

for integer p, q. If q is negative we compose $F^{[-1]}$ instead of F, so without loss of generality we may assume that q is positive. Let

$$q = mr + s, \tag{6.3}$$

where $r \geqq 0$, $0 \leqq s < m$. We can evaluate $M = F^{[m]} = x + \cdots$, and $F^{[s]}$ by the obvious "squaring" method in time $O(\text{COMP}(n) \lg m) = O(\text{COMP}(n) \lg n)$. Then, using the method of § 5, we can evaluate $F^{[mr]} = M^{[r]}$ in time $O(\text{COMP}_2(n))$. Finally, $F^{[q]} = F^{[mr]}(F^{[s]})$ may be evaluated by performing one composition. (An additional reversion is required if $q < 0$.) Thus we have established

THEOREM 6.1. *Assume* ord $(F) = 1$, $f_1 \neq 1$, $f_1^m = 1$ *for some* m *such that* $1 < m \leqq n-2$, q *integer, and let* $G = F^{[q]}$. *Then* $g_1, \cdots, g_{n-1}$ *can be evaluated in time* $O(\text{COMP}(n) \lg m + \text{COMP}_2(n))$.

Remark 6.2. If Δ is the real field (so the only roots of unity are ± 1) then Theorem 6.1 shows that $g_1, \cdots, g_{n-1}$ can be evaluated in time $O(\text{COMP}_2(n))$.

7. Summary and open problems. From Theorems 3.1, 4.1, 5.1, and 6.1 we have

THEOREM 7.1. *Let* $F(x)$ *be a formal power series,* ord $(F) \geqq 1$, *and let* $G(x) = F^{[q]}(x)$. *If* q *satisfies the following conditions*:

(i) *If* ord $(F) > 1$, *then* q *is a positive integer*;

(ii) *If the multiplier* f_1 *is a nontrivial root of unity, then* q *is an integer*;

(iii) f_1^q *is defined*;

and if f_1^q *is given, then* $g_1, \cdots, g_n$ *can be computed in time* $O(\text{COMP}_2(n))$ *and this bound is independent of* q.

Different defining equations are used for the various cases we have had to consider. For the reader's convenience we summarize them here. As before, $G = F^{[q]}$.

I. Regular case: $f_1 \neq 0$, $f_1^m \neq 1$, $m = 1, 2, \cdots$. Define S by $S(F(x)) = f_1 S(x)$, ord $(S) = 1$. Then $G(x) = S^{[-1]}(f_1^q S(x))$.

II. $f_1 = 0$. Define S by $S(F(x)) = f_k (S(x))^k$, ord $(S) = 1$, $s_1 = 1$. Then $G(x) = S^{[-1]}\{f_k^{(k^q-1)/(k-1)} [S(x)]^{k^q}\}$.

III. $f_1 = 1$. Define T by $T(F(x)) = F'(x)T(x)$, and ord $(T) =$ ord $(F(x) - x)$. Then determine $G(x)$ from $T(G(x)) = G'(x)T(x)$ and (5.20).

IV. $f_1 \neq 1$, $f_1^m = 1$ for some integer $m > 1$. This can be reduced to case III.

It is possible to compute G using the same functional equation for cases I–III. Define $U(x)$ by

$$U(F(x)) = \frac{F'(x)}{\text{ord}\,(F)}\, U(x), \qquad \text{ord}\,(U(x)) = \text{ord}\,(F(x) - x). \tag{7.1}$$

$U(x)$ exists and is unique up to a scaling factor. In fact, in cases I and II we have

$$U(x) = cS(x)/S'(x), \tag{7.2}$$

and in case III we have $U(x) = c'T(x)$, for some nonzero constants c and c'. Also, it is easy to prove that G satisfies

$$U(G(x)) = \frac{G'(x)}{[\text{ord}\,(F)]^q}\, U(x). \tag{7.3}$$

Although a unified treatment of cases I–III using (7.1) and (7.3) would be possible, it is simpler to use the Schroeder function $S(x)$ of (3.1) in case I and the generalized Schroeder function of (4.1) in case II, for then G is given explicitly by (3.23) or (4.2) instead of implicitly as a certain solution of (7.3). Also, in proving properties of algorithms for the computation of G by either method, it is natural to consider cases I–III separately.

The techniques of §§ 3 and 5 can be applied to far more general nonlinear functional equations. We shall report on this elsewhere.

To conclude we list some open problems suggested by the results of the paper.

1. If the field Δ is such that $\text{MULT}(n) = O(n \lg n)$ then the fastest algorithm known for composition is $O((n \lg n)^{3/2})$. No nontrivial lower bound is known.

a. Is composition harder than multiplication? (It is at least as hard.)

b. Although there are only n inputs and n outputs, the best upper bound known is $O((n \lg n)^{3/2})$. This is comparable to matrix multiplication where there are $2n^2$ inputs and n^2 outputs but the best upper bound known (Pan (1978)) is $O(n^{2.79})$. Can the Brent–Kung upper bound be reduced?

c. Is $\alpha > 1$ in the notation of (2.5)? An affirmative answer would show that $\text{COMP}_2(n) = O(\text{COMP}(n))$.

2. Brent and Kung (1978) showed that, for the reversion problem $R(x) = F^{(-1)}(x)$, the complexity of computing $R_n(x)$ is $O(\text{COMP}(n))$. Consider computing $R_n(x_0)$ for a scalar x_0. This problem has n inputs and one output. Brent and Kung (1978) showed its complexity to be $O(\text{MULT}(n))$. If $G(x) = F^{[q]}(x)$, what is the complexity of computing $G_n(x_0)$? Is it less than the complexity of computing $G_n(x)$?

3. What are the numerical properties of our algorithms? For example, we expect the computation of the Schroeder function to be ill-conditioned if $f_1^{(m)}$ is close to 1 for some $m \leqq n-2$; see (3.13). Cherry (1964) discusses this problem in conjunction with a problem in dynamical systems.

4. What are the complexity bounds for exact arithmetic over the rational field?

Acknowledgment. We are deeply indebted to M. L. Fredman who pointed out to us the critically important idea of using the Schroeder function to effect a change of representation in the regular case. We thank D. E. Knuth who pointed out to us that a single functional equation, (7.1), can be used for three of the cases. Finally, we thank M. Sapsford and M. Sieveking for their careful reading of the manuscript.

REFERENCES

J. ACZÉL (1966), *Lectures on Functional Equations and Their Applications*, Academic Press, New York.

I. N. BAKER (1964), *Fractional iteration near a fixpoint of multiplier 1*, J. Austral. Math. Soc., 4, pp. 143–148.

A. BORODIN AND I. MUNRO (1975), *The Computational Complexity of Algebraic and Numeric Problems*, American Elsevier, New York.

R. P. BRENT (1976), *Multiple-precision zero-finding methods and the complexity of elementary function evaluation*, Analytic Computational Complexity, J. F. Traub, ed., Academic Press, New York, pp. 151–176.

R. P. BRENT AND H. T. KUNG (1978), *Fast algorithms for manipulating formal power series*, Department of Computer Science Report, Carnegie–Mellon University, 1976. Also J. Assoc. Comput. Mech., 25, pp. 581–595.

T. M. CHERRY (1964), *A Singular Case of Iteration of Analytic Functions: A Contribution to the Small-Divisor Problem*, Nonlinear Problems of Engineering, W. F. Ames, ed., Academic Press, New York, pp. 29–50.

N. G. DE BRUIJN (1970), *Asymptotic Methods in Analysis* (Third Edition), North-Holland Publishing Company, Amsterdam.

W. FELLER (1957), *An Introduction to Probability Theory and its Applications*, vol. I, Second Edition, John Wiley, New York.

F. GROSS (1972), *Factorization of Meromorphic Functions*, U.S. Government Printing Office, Washington, DC.

T. E. HARRIS (1963), *The Theory of Branching Processes*, Springer-Verlag, Berlin.

P. HENRICI (1974), *Applied and Computational Complex Analysis*, vol. 1, John Wiley, New York.

D. E. KNUTH (1969), *The Art of Computer Programming*, vol. 2, Addison-Wesley, Reading, MA.

——— (1976), *Big omicron and big omega and big theta*, SIGACT News 8, no. 2, pp. 18–24.

M. KUCZMA (1968), *Functional Equations in a Single Variable*, PWN-Polish Scientific Publishers, Warsaw.

H. T. KUNG AND J. F. TRAUB (1978), *All algebraic functions can be computed fast*, Department of Computer Science Report, Carnegie–Mellon University, 1976. Also J. Assoc. Comput. Mech., 25, pp. 245–260.

H. LEVY AND F. LESSMAN (1961), *Finite Difference Equations*, Pitman, London.

Z. A. MELZAK (1973), *Companion to Concrete Mathematics*, John Wiley, New York.

V. PAN (1978), *Strassen's algorithm is not optimal trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations*, 19th Annual Symposium on Foundations of Computer Science, IEEE Computer Society.

D. S. PARKER, JR, (1977), *Nonlinear recurrences and parallel computation*, High Speed Computer and Algorithm Organization, D. J. Kuck, D. H. Lawrie, and A. H. Sameh, eds., Academic Press, New York, pp. 317–320.

E. SCHROEDER (1871), *Über iterierte Funktionen*, Math. Ann., 3, pp. 296–322.

G. SZEKERES (1964), *Fractional iteration of entire and rational functions*, J. Austral. Math. Soc., 4, pp. 129–142.