Fast Algorithms for Manipulating Formal Power Series

R. P. BRENT

Australian National University, Canberra, Australia

AND

H. T. KUNG

Carnegie-Mellon University, Pittsburgh, Pennsylvania

ABSTRACT The classical algorithms require order n^3 operations to compute the first n terms in the reversion of a power series or the composition of two series, and order $n^2 \log n$ operations if the fast Fourier transform is used for power series multiplication. In this paper we show that the composition and reversion problems are equivalent (up to constant factors), and we give algorithms which require only order $(n \log n)^{3/2}$ operations. In many cases of practical importance only order $n \log n$ operations are required; these include certain special functions of power series and power series solution of certain differential equations. Applications to root-finding methods which use inverse interpolation and to queueing theory are described, some results on multivariate power series are stated, and several open questions are mentioned.

KEY WORDS AND PHRASES formal power series, reversion of power series, composition of power series, computational complexity, fast algorithms, special functions of power series, power series solution of differential equations, queueing theory, fast Fourier transform

CR CATEGORIES 5.7, 5.15, 5.17

1. *Introduction*

We are interested in the complexity of algorithms for manipulating formal power series. For example, such algorithms may compute the first n terms in the product, quotient, or composition of two given power series. These problems arise in combinatorics and analysis of algorithms, where the desired power series is a generating function, as well as in numerical analysis. See, for example, Knuth [26], Ferguson, Nielsen, and Cook [14], Riordan [35], Gilbert [18], Niven [31], Jackson and Reilly [25], Levy and Lessman [30], Norman [32], and Henrici [20, 21].

Let $\mathscr{P}$ be the integral domain of formal power series $P(s) = p_0 + p_1 s + p_2 s^2 +$ over some field K. "Formal" means that we are not concerned with questions of convergence. If F is a set of indeterminates over K, and E is a finite subset of the extension field $K(F)$, then $L(E \bmod F)$ denotes the number of operations necessary to compute E, starting from $K \cup F$ and working in $K(F)$. Informally, $L(E \bmod F)$ is the number of operations required to compute E, given F.

If $A, B \in \mathscr{P}$ and C is the formal product of A and B, we define $M(n) = L(c_0, \ldots, c_n \bmod a_0, \ldots, a_n, b_0, \ldots, b_n)$. Informally, $M(n)$ is the number of operations required to compute the

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.

Authors' addresses: R. P. Brent, Computer Science Department, Stanford University, Stanford, CA 96305; H. T. Kung, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213.

first $n + 1$ coefficients in the product of two power series. The classical algorithm gives $M(n) = O(n^2)$, but if the field K is capable of supporting the fast Fourier transform (FFT), then $M(n) = O(n \log n)$ (see, e.g. Knuth [26] and Borodin and Munro [5]).

Let $P, Q \in \mathscr{P}$, $p_0 = 0$. The *composition* of Q and P is the formal power series R such that $R(s) = Q(P(s))$ is a formal identity. The *composition problem* is to compute $r_0, \ldots, r_n$ given $p_1, \ldots, p_n$ and $q_0, \ldots, q_n$. We define $\mathrm{COMP}(n) = L(r_0, \ldots, r_n \bmod p_1, \ldots, p_n, q_0, \ldots, q_n)$, so $\mathrm{COMP}(n)$ is the number of operations required to solve the composition problem.

The *functional inverse* or *reversion* of P is the power series $V = P^{(-1)}$ such that $P(V(s)) = s$ or $V(P(s)) = s$ is a formal identity. The *reversion problem* is to compute $v_1, \ldots, v_n$ given $p_1, \ldots, p_n$. It is clear that the problem can be viewed as that of computing the derivatives of the inverse function (see, e.g. Traub [38, App. B]). We define $\mathrm{REV}(n) = L(v_1, \ldots, v_n \bmod p_1, \ldots, p_n)$.

The classical algorithms for both the composition and reversion problems require order n^3 operations (see, e.g. Knuth [26]), or order $n^2 \log n$ operations if the FFT is used for polynomial multiplication as pointed out in Kung and Traub [28, §4]. In fact the classical algorithms give $\mathrm{COMP}(n) = O(nM(n))$ and $\mathrm{REV}(n) = O(nM(n))$. In Section 2 we show that $\mathrm{COMP}(n) = O((n \log n)^{1/2} M(n))$. We also give an $O(\sqrt{n} \cdot \max(M(n), N(\sqrt{n})))$ algorithm where $N(j)$ is the number of operations required to multiply two $j \times j$ matrices. This algorithm is faster than both the $O((n \log n)^{1/2} M(n))$ algorithm and the classical algorithm when the polynomial multiplication algorithm to be used implies that $M(n)$ has order at least $N(\sqrt{n})/\sqrt{(\log n)}$ (e.g. when $M(n) \sim cn^2$ for some constant c) In Section 3 we show that the reversion problem can be solved by Newton's method and composition, so $\mathrm{REV}(n) = O((n \log n)^{1/2} M(n))$ also.

In Section 4 we show that $\mathrm{COMP}(n) = O(\mathrm{REV}(n))$ and $\mathrm{REV}(n) = O(\mathrm{COMP}(n))$, so the composition and reversion problems are essentially equivalent. Thus, in attempting to obtain improved upper or lower bounds one can work with either the composition problem or the reversion problem.

In Section 5 we show that the composition $Q(P(s))$ may be computed in $O(M(n))$ operations if Q satisfies a suitable differential equation. For example, Q could be a Bessel function or a hypergeometric function. We also study the complexity of computing the formal series solution of certain first-order differential equations. In Section 6 we mention several other problems for which $O(M(n))$ algorithms exist, and give an application to the theory of root-finding methods. Most of this paper is restricted to power series in one variable, but the methods extend to dense power series in several variables. Some of our results on multivariate cases are stated in Section 7. The considerations for sparse power series and polynomials in several variables are rather different; see Heindel [19] and Horowitz [23, 24].

In this paper we analyze algorithms under the assumption that all coefficient computations are done in a finite field or in finite-precision floating-point arithmetic. An analysis dealing with variable-precision coefficients is yet to be performed.

Some of the results of this paper were announced in Brent and Kung [9].

Some Regularity Conditions. Let Z^+ be the set of all nonnegative integers and let $G: Z^+ \to Z^+$ be a nondecreasing function. We say that G satisfies *Condition A* if, for some α, $\beta \in (0, 1)$, $G(\lceil \alpha n \rceil) \le \beta G(n)$ for all sufficiently large n. We say that G satisfies *Condition B* if, for some $\alpha', \beta' \in (0, 1)$, $G(\lfloor \alpha' n \rfloor) \ge \beta' G(n)$ for all sufficiently large n. For example, if G is nondecreasing and $C_1 n^\gamma \log^\delta n \le G(n) \le C_2 n^\gamma \log^\delta n$ for positive constants γ, C_1, C_2, and any constant δ, then G satisfies Conditions A and B.

LEMMA 1.1. *If G satisfies Condition A and $\rho \in (0, 1)$, then*

$$\sum G(\lceil \rho^j n \rceil) = O(G(n)), \tag{1.1}$$

where the sum is taken over all integers $j \ge 0$ such that $\rho^j n \ge 1$.

PROOF. It is easy to show that $\sum G(\lceil \rho^j n \rceil)/G(n)$ is bounded by a convergent geometric series for all sufficiently large n. See Brent [7, Lemma 3.4] for details. □

LEMMA 1.2. *If G satisfies Condition B and $\gamma > 1$, then*

$$G(\lfloor \gamma n \rfloor) = O(G(n)). \tag{1.2}$$

PROOF. There exists j such that $\alpha^j \gamma \leq 1$. Thus, for all sufficiently large n,

$$G(\lfloor \gamma n \rfloor) \leq (1/\beta')G(\lfloor \alpha' \gamma n \rfloor) \leq \quad \leq (1/\beta')^j G(n). \qquad \square$$

We say that G satisfies *Condition A_s* if G satisfies *Condition A* with $1 > \alpha \geq \beta > 0$. Clearly Condition A_s is stronger than Condition A. For example, if $G(n) = \lceil n^\delta \rceil H(n)$ for some constant $\delta \geq 1$ and some nondecreasing function $H: Z^+ \to Z^+$, then G satisfies Condition A_s.

LEMMA 1.3. *If G satisfies Condition A_s and $\rho \in (0, 1)$, then*

$$\sum \rho^{-j} G(\lceil \rho^j n \rceil) = O((\log n) G(n)), \tag{1.3}$$

where the sum is taken over all integers $j \geq 0$ such that $\rho^j n \geq 1$.

PROOF. Assume n is sufficiently large. Since $G(\lceil \alpha n \rceil) \leq \beta G(n) \leq \alpha G(n)$,

$$G(\lceil \alpha^\sigma n \rceil) \leq G(\lceil \alpha^{\lfloor \sigma \rfloor} n \rceil) \leq \alpha^{\lfloor \sigma \rfloor} G(n) \leq (1/\alpha)\alpha^\sigma G(n)$$

for any real $\sigma \geq 0$. Let $\gamma = \log_\alpha \rho$. Then for all $j \geq 0$,

$$G(\lceil \rho^j n \rceil) = G(\lceil \alpha^{\gamma j} n \rceil) \leq (1/\alpha)\alpha^{\gamma j} G(n) = (1/\alpha)\rho^j G(n),$$

and the result follows immediately. $\square$

We assume throughout the paper that M satisfies Condition A_s. Similar conditions are usually assumed, either explicitly (see, e.g. Aho, Hopcroft, and Ullman [2, p. 280] and Fischer and Stockmeyer [17]) or implicitly (see, e.g. Borodin [4]). We shall also assume that COMP and REV satisfy Conditions A and B, respectively One should note, however, that what we really need in this paper are the consequences of these conditions, namely, the properties (1.1), (1.2), and (1.3)

Notation. s and t denote free variables or indeterminates over K. Formal power series over K are denoted by upper-case letters, and the coefficients in the power series by corresponding lower-case letters, e.g. $P(s) = p_0 + p_1 s + \cdots + p_n s^n + \quad$. The formal derivative of P is $P'(s) = p_1 + 2p_2 s + \cdots$, and the formal integral of P is $\int_0^s P(t)dt = p_0 s + \frac{1}{2}p_1 s^2 + \cdots$. For any positive integer k, $P(s) \bmod s^k$ denotes the finite series consisting of all terms of $P(s)$ of degree less than k. To compute $P(s) \bmod s^k$ means to compute $p_0, \ldots, p_{k-1}$. By the notation $Q(s) = P(s) \pmod{s^k}$, we mean $(P(s) - Q(s)) \bmod s^k = 0$, i.e. power series $P(s)$ and $Q(s)$ agree in their terms of degree less than k. Where necessary we assume that the characteristic of K is zero or sufficiently large.

2. *Fast Algorithms for Composition*

Let $P(s) = p_1 s + \quad + p_n s^n$ and $Q(t) = q_0 + \quad + q_n t^n$ be given. In this section we give two algorithms for computing the first $n + 1$ coefficients, $r_0, \ldots, r_n$, in the series $R(s) = Q(P(s))$.

2.1 THE FIRST ALGORITHM. The algorithm is based on the following grouping of terms of $Q(t)$.

$$Q(t) = Q_0(t) + Q_1(t)t^k + Q_2(t)(t^k)^2 + \quad + Q_{k-1}(t)(t^k)^{k-1},$$

where $k = \lceil \sqrt{(n+1)} \rceil$ and $Q_i(t) = \sum_{j=0}^{k-1} q_{ik+j} t^j$, $i = 0, \ldots, k-1$ (assume $q_l = 0$ for $l > n$). A similar idea was used by Paterson and Stockmeyer [33].

ALGORITHM 2.1

1. Compute $P^i(s)$, $i = 2, \ldots, k$.
2. Let $T(s) = P^k(s) \bmod s^{n+1}$ Compute $T^i(s)$, $i = 2, \quad , k-1$
3. Compute $Q_i(P(s))$, $i = 0, \ldots, k-1$, by using the results of step 1
4. Compute $Q_i(P(s))T^i(s)$, $i = 1, \quad , k-1$, by using the results of steps 2 and 3
5. Compute $\sum_{i=0}^{k-1} Q_i(P(s))T^i(s)$ by using the result of step 4

Note that during the computation we always truncate terms of degree higher than n. It is easy to see that steps 1, 2, 4, and 5 each can be done in $O(kM(n)) = O(\sqrt{n}M(n))$ operations. In the following we examine step 3. Let

$$(P(s))^j = \sum_{l=0}^{n} p_l^{(j)} s^l \bmod s^{n+1}, \quad j = 0, \ldots, k.$$

(Note that all $p_l^{(j)}$ are available after step 1.) Then

$$\begin{aligned} Q_i(P(s)) &= \sum_{j=0}^{k-1} q_{ik+j} \sum_{l=0}^{n} p_l^{(j)} s^l \qquad (\text{mod } s^{n+1}) \\ &= \sum_{l=0}^{n} \left(\sum_{j=0}^{k-1} q_{ik+j} p_l^{(j)} \right) s^l \quad (\text{mod } s^{n+1}). \end{aligned}$$

Step 3 amounts to computing $\sum_{j=0}^{k-1} q_{ik+j} p_l^{(j)}$ for $l = 0, \ldots, n$, $i = 0, \ldots, k-1$, given the q_{ik+j} and $p_l^{(j)}$. The computation may be viewed as the following matrix multiplication between an $(n+1) \times k$ and a $k \times k$ matrix:

$$\begin{bmatrix} p_0^{(0)} & p_0^{(1)} & \cdots & p_0^{(k-1)} \\ p_1^{(0)} & & & \\ p_2^{(0)} & & \vdots & \\ \vdots & & & \\ p_n^{(0)} & & & p_n^{(k-1)} \end{bmatrix} \begin{bmatrix} q_0 & q_k & & q_{(k-1)k} \\ \vdots & & & \\ q_{k-1} & & \cdots & q_{k^2-1} \end{bmatrix}.$$

This can clearly be done by performing $\lceil (n+1)/k \rceil$ matrix multiplications between $k \times k$ matrices. Define

$$N(j) = \text{number of operations needed to multiply two } j \times j \text{ matrices.}$$

Then step 3 takes $O((n/k)N(k))$ or $O(\sqrt{n}N(\sqrt{n}))$ operations. Therefore Algorithm 2.1 establishes the following:

THEOREM 2.1 $COMP(n) = O(\sqrt{n} \cdot max(M(n), N(\sqrt{n})))$.

2.2 THE SECOND ALGORITHM. The second algorithm is based on a formal Taylor expansion of Q. Write $P(s) = P_m(s) + P_r(s)$, where $P_m(s) = p_1 s + \cdots + p_m s^m$ and $P_r(s) = p_{m+1}s^{m+1} + p_{m+2}s^{m+2} + \cdots$ for some $m < n$. (The value of m will be determined later.) It can be shown by induction that the following Taylor expansion holds formally:

$$Q(P) = Q(P_m + P_r) = Q(P_m) + Q'(P_m)P_r + \tfrac{1}{2}Q''(P_m)(P_r)^2 + \cdots.$$

Let $l = \lceil n/m \rceil$. Since the degree of any term in $(P_r)^{l+i}$ is at least $n+1$ for any $i > 0$,

$$Q(P(s)) = Q(P_m(s)) + Q'(P_m(s)) \cdot P_r(s) + \cdots + (1/l!)Q^{(l)}(P_m(s)) \cdot P_r^l(s) \quad (\text{mod } s^{n+1}).$$

This equality gives us the following algorithm for computing the first $n+1$ coefficients in the series $R(s) = Q(P(s))$:

ALGORITHM 2.2

1. Compute $Q(P_m(s)) \bmod s^{n+1}$.
2. Compute $Q'(P_m(s)), Q''(P_m(s)), \ldots, Q^{(l)}(P_m(s)) \bmod s^{n+1}$
3. Compute $P_r(s), P_r^2(s), \ldots, P_r^l(s) \bmod s^{n+1}$
4. Compute $(1/i!)Q^{(i)}(P_m(s))\ P_r^i(s) \bmod s^{n+1}$ for $i = 1, \ldots, l$
5. Sum the result obtained from step 4.

LEMMA 2.1. *If* $P(s) = p_1 s + \cdots + p_m s^m$, $Q(t) = q_0 + \cdots + q_j t^j$ *with* $m, j \le n$ *and if* $R(s) = Q(P(s)) = r_0 + r_1 s + \cdots$, *then*

$$L(r_0, \ldots, r_n \bmod p_1, \ldots, p_m, q_0, \ldots, q_j) = O((jm/n)(\log n)M(n)).$$

PROOF. We may assume that j is a power of 2. Write $Q(P) = Q_1(P) + P^{j/2} \cdot Q_2(P)$, where Q_1 and Q_2 are polynomials of degree $j/2$. This relation gives us a recursive procedure

for computing $Q(P)$. During the computation we always truncate terms of degree higher than n. Note that $\deg P^j \le jm$ and $\deg Q(P(s)) \le jm$ when $\deg Q = j$. Assume that $T(j)$ operations are needed to compute both $P^{j/2} \bmod s^{n+1}$ and $Q(P) \bmod s^{n+1}$ with $\deg Q = j$. Then by the recursive procedure, we have

$$T(j) \le 2T(j/2) + O(M(\min(jm, n))).$$

Let r be the largest integer k such that $jm/2^k \ge n$. We can assume $r \ge 0$. We have

$$\begin{aligned} T(j) &= O(M(n) + 2M(n) + \cdots + 2^r M(n)) + 2^{r+1}T(j/2^{r+1}) \\ &\le O((jm/n)M(n)) + (2jm/n)T(j/2^{r+1}). \end{aligned}$$

Since $jm/2^{r+1} < n$,

$$\begin{aligned} T(j/2^{r+1}) &= O(M(jm/2^{r+1}) + 2M(jm/2^{r+2}) + \cdots) \\ &= O(M(n) + 2M(\lceil n/2 \rceil) + 4M(\lceil n/4 \rceil) + \quad) \\ &= O((\log n)M(n)) \end{aligned}$$

by Lemma 1.3. Hence $T(j) = O((jm/n)(\log n)M(n))$. □

LEMMA 2.2. *Let $U(s) = P(s)/Q(s)$ with $q_0 \neq 0$. Then*

$$L(u_0, \ldots, u_n \bmod p_0, \ldots, p_n, q_0, \ldots, q_n) = O(M(n)).$$

PROOF. Use a Newton-like method as in Kung [27]. (See also Sieveking [36].) □

LEMMA 2.3. *Let $P(s) = p_1 s + p_2 s^2 + \cdots$, $Q(t) = q_0 + q_1 t + \cdots$, and let $Q'(t) = q_1 + 2q_2 t + \cdots$, the formal derivative of $Q(t)$ with respect to t. If $R(s) = Q(P(s))$ and $D(s) = Q'(P(s))$, then*

$$L(d_0, \ldots, d_n \bmod r_1, \ldots, r_{n+1}, p_1, \ldots, p_{n+1}) = O(M(n)).$$

PROOF. By the chain rule, $R'(s) = Q'(P(s)) \cdot P'(s)$. Hence $D(s) = R'(s)/P'(s)$, and the result follows from Lemma 2.2. □

By Lemma 2.1, step 1 of Algorithm 2.2 can be done in $T_1 = O(m(\log n)M(n))$ operations. By Lemma 2.3, step 2 of Algorithm 2.2 can be done in $T_2 = O(lM(n)) = O((n/m)M(n))$ operations, since $Q^{(i)}(P_m(s))$, $i = 1, \ldots, l$, can be computed successively and each of them takes $O(M(n))$ operations. It is easy to check that steps 3, 4, and 5 can all be done in $O(T_2)$ operations. Hence the total number of operations needed by Algorithm 2.2 is $O(T_1 + T_2)$. Choose $m \sim (n/\log n)^{1/2}$. Then $O(T_1 + T_2) = O((n \log n)^{1/2}M(n))$. We have shown the following:

THEOREM 2.2. $COMP(n) = O((n \log n)^{1/2}M(n))$.

2.3 REMARKS. As stated in Theorems 2.1 and 2.2, the number of operations required by Algorithms 2.1 and 2.2 depends upon $M(n)$ and $N(j)$. There are many algorithms for polynomial multiplication. For example, the classical algorithm gives $M(n) = O(n^2)$, binary splitting multiplication gives $M(n) = O(n^{1.585})$, and FFT multiplication gives $M(n) = O(n \log n)$ (see, e.g. Fateman [13]). Likewise there are various algorithms for matrix multiplication. For example, the classical algorithm gives $N(j) = O(j^3)$ and Strassen's algorithm gives $N(j) = O(j^{2.81})$ (Strassen [37]). Either Algorithm 1.2 or Algorithm 2.2 can take fewer operations asymptotically, depending upon which polynomial or matrix multiplication algorithms are used. The following results are easy consequences of Theorems 2.1 and 2.2.

(i) Suppose that $N(j) = O(j^\alpha)$ for some $\alpha \ge 2$. Then Algorithm 2.2 takes fewer operations than Algorithm 2.1 asymptotically if $M(n) = o(n^{\alpha/2}/\sqrt{(\log n)})$.

(ii) Suppose that the classical polynomial and matrix multiplication algorithms are used, i.e. $M(n) = O(n^2)$ and $N(j) = O(j^3)$. Then Algorithm 2.1 gives $\mathrm{COMP}(n) = O(n^{5/2})$, while the classical algorithm for composition takes $O(n^3)$ operations.

(iii) Suppose that the FFT multiplication is used, so $M(n) = O(n \log n)$. Then Algorithm 2.2 gives $\mathrm{COMP}(n) = O((n \log n)^{3/2})$, which is the best asymptotic bound known for the composition problem.

3. *Fast Algorithms for Reversion*

Let

$$P(s) = p_1 s + p_2 s^2 + \cdots, \quad p_1 \neq 0, \tag{3.1}$$

be given. The functional inverse or reversion of P is the power series $V = P^{(-1)}$ such that $P(V(s)) = s$ or $V(P(s)) = s$ is a formal identity. The facts that V exists and that $v_1, \ldots, v_n$ depend only upon $p_1, \ldots, p_n$ are well known. The reversion problem is to compute $v_1, \ldots, v_n$ given $p_1, \ldots, p_n$. In this section we show that the reversion problem can be solved efficiently by using the fast algorithms for composition presented in Section 2.

Define a function $f: \mathscr{P} \to \mathscr{P}$ by $f(x) = P(x) - s$. Since $P(V(s)) = s$, V is the zero of f. Hence the reversion problem can be viewed as a zero-finding problem. We shall use a Newton-like method to find the zero of f; other iterations can also be used successfully. (See Kung [27] for a similar technique for computing the reciprocals of power series, and also Brent [8, Sec. 13].) The iteration function given by the method is

$$\varphi(x) = x - f(x)/f'(x) = x - (P(x) - s)/P'(x).$$

Since $p_1 \neq 0$, one can easily check that φ maps $\mathscr{P}^*$ into $\mathscr{P}^*$, where $\mathscr{P}^*$ is the set of power series with $p_0 = 0$ and $p_1 \neq 0$. Using the Taylor expansions of P and P', we have

$$\begin{aligned}\varphi(x) - V(s) &= x - V(s) - \frac{(P(V(s)) + P'(V(s))(x - V(s)) + \cdots) - s}{P'(V(s)) + P''(V(s))(x - V(s)) + \cdots} \\ &= \frac{P''(V(s))}{2P'(V(s))}(x - V(s))^2 + \left[\frac{P'''(V(s))}{3P'(V(s))} - \frac{1}{2}\left(\frac{P''(V(s))}{P'(V(s))}\right)^2\right](x - V(s))^3 + \cdots.\end{aligned}$$

Since $p_1 \neq 0$, the expansions of $P''(V(s))/P'(V(s))$, $P'''(V(s))/P'(V(s))$, etc., have no negative powers in s. Thus,

$$\varphi(x) - V(s) = A(s)(x - V(s))^2, \tag{3.2}$$

where $A \in \mathscr{P}$. Suppose that the first k coefficients, $v_1, \ldots, v_k$, of $V(s)$ have already been computed. Substituting $V_k(s) = v_1 s + \cdots + v_k s^k$ for x in (3.2), we have $\varphi(V_k(s)) = V(s) \pmod{s^{2k+2}}$. Hence by computing the first $2k + 1$ coefficients of $\varphi(V_k(s))$, we obtain the first $2k + 1$ coefficients of the reversion $V(s)$. This leads to the following algorithm for computing the first n coefficients, $v_1, \ldots, v_n$, of $V(s)$. Note that the first coefficient of $V(s)$ is $1/p_1$.

ALGORITHM 3.1 (Newton's Method)

1. Set $v_1 \leftarrow 1/p_1$ and $k \leftarrow 1$.
2. Compute $v_{k+1}, \ldots, v_{2k+1}$ such that $v_1, \ldots, v_{2k+1}$ are the first $2k + 1$ coefficients of $V_k(s) - (P(V_k(s)) - s)/P'(V_k(s))$, where $V_k(s) = \sum_{i=1}^{k} v_i s^i$.
3. If $2k + 1 \geq n$, the algorithm terminates.
4. Set $k \leftarrow 2k + 1$, and return to step 2.

The essential work of the algorithm is performed at step 2. Note that in the compositions $P(V_k(s))$ and $P'(V_k(s))$ only the first $2k + 1$ terms are needed. By Lemmas 2.2 and 2.3, the algorithm establishes the following theorem.

THEOREM 3.1. $REV(2k + 1) \leq REV(k) + COMP(2k + 1) + O(M(2k + 1))$.

Using the results stated in Section 2.3 for the composition problem, we give some consequences of Theorem 3.1:

(i) Suppose that the classical polynomial multiplication routine is used, i.e. $M(n) = O(n^2)$. Then $\mathrm{COMP}(n) = O(n^{5/2})$. Algorithm 3.1 gives $\mathrm{REV}(n) = O(n^{5/2})$, while in this case all classical algorithms for reversion require $O(n^3)$ operations (see, e.g. Henrici [21, pp. 45–65], Traub [38, App. B], and Knuth [26, pp. 444–451]).

(ii) Suppose that the FFT is used for polynomial multiplication, i.e. $M(n) = O(n \log n)$. Then $\mathrm{COMP}(n) = O((n \log n)^{3/2})$. Algorithm 3.1 gives $\mathrm{REV}(n) = O((n \log n)^{3/2})$, which is the best asymptotic bound known for the reversion problem.

It is possible to define the reversion of a power series of the form

$$t = P(s) = s^{\sigma}(1 + p_1 s + p_2 s^2 + \quad), \tag{3.3}$$

where $\sigma \in K$ and $\sigma \neq 0$. Indeed, the reversion is of the form

$$s = V(t) = t^{1/\sigma}(1 + v_1 t^{1/\sigma} + v_2 t^{2/\sigma} + \quad).$$

(See, e.g. Chrystal [12, p. 378].) Since by (3.3), $t^{1/\sigma} = s(1 + p_1 s + p_2 s^2 + \quad)^{1/\sigma}$, we can compute $v_1, v_2, \ldots, v_n$ in the following way:

(1) Compute $p_i^{(1/\sigma)}$, $i = 1, \ldots, n$, such that

$$\sum_{i=0}^{n} p_i^{(1/\sigma)} s^i = \left(\sum_{i=0}^{n} p_i s^i \right)^{1/\sigma} \bmod s^{n+1},$$

where $p_0 = 1$.

(2) Find the reversion of the series $s(1 + p_1^{(1/\sigma)} s + p_2^{(1/\sigma)} s^2 + \quad)$.

It will be shown in Lemma 6.2 that step 1 can be done in $O(M(n))$ operations. Hence the reversion of a power series of the form (3.3) can be done in $\mathrm{REV}(n) + O(M(n))$ operations. In Section 4 we shall show that $M(n) = O(\mathrm{REV}(n))$. This implies that the number of operations required to find the reversion of the series (3.1) is the same order of magnitude as that required to find the reversion of the series (3.3).

A Numerical Example. The algorithms for composition and reversion have been implemented in Fortran, and several numerical tests performed. For example, we computed the reversion $V(s) = -\ln(1 - s)$ of $P(s) = 1 - \exp(-s) \bmod s^{n+1}$ for various $n \leq 64$. The correct result is $v_j = 1/j$ for $j \geq 1$. With $n = 64$, the computed values $\tilde{v}_j$ satisfied $|\tilde{v}_j - v_j| < 7 \times 10^{-15}$ for all $j \leq 64$. Computations were performed on a Univac 1108 computer with a 60-bit floating-point fraction.

Thus for this example, our reversion algorithm is stable. A general investigation of the stability of our algorithms has not been carried out.

4. *Equivalence of Composition and Reversion*

In this section we show that the composition problem is linearly equivalent to the reversion problem in the sense of Borodin [4] and Hopcroft [22], i.e.

$$\mathrm{REV}(n) = O(\mathrm{COMP}(n)) \text{ and } \mathrm{COMP}(n) = O(\mathrm{REV}(n)).$$

It is necessary to make some mild regularity assumptions. We assume that COMP satisfies Condition A, and that REV satisfies Condition B. It follows from Theorem 4.1 that both COMP and REV satisfy Conditions A and B.

LEMMA 4.1. *If* $U(s) = P^2(s)$ *and* $S(n) = L(u_0, \ldots, u_n \bmod p_0, \ldots, p_n)$, *then* $M(n) = O(S(n))$.

PROOF. Since $4PQ = (P + Q)^2 - (P - Q)^2$, we have $M(n) \leq 2S(n) + O(n) = O(S(n))$. □

LEMMA 4.2. $M(n) = O(\mathrm{COMP}(n))$.

PROOF. If $Q(s) = s^2$ and $P(s) = p_0 + \tilde{P}(s)$ then $P^2 = Q(\tilde{P}) + 2p_0 P - p_0^2$, so $S(n) \leq \mathrm{COMP}(n) + O(n)$, and the result follows from Lemma 4.1. □

LEMMA 4.3. $M(n) = O(\mathrm{REV}(n))$.

PROOF. Let $A(s) = a_0 + a_1 s + \cdots$, $B(s) = s + s^{n+2} A(s)$, and $C = B^{(-1)}$. Then it is not difficult to show that

$$C(s) = s - s^{n+2} A(s) + s^{2n+3}[sA(s)A'(s) + (n + 2)A^2(s)] \pmod{s^{3n+4}}.$$

Thus, in REV$(3n + 3) + O(n)$ operations we can compute $sA(s)A'(s) + (n + 2)A^2(s)$ mod s^{n+1}. Similarly, by defining $B(s) = s + s^{n+3}A(s)$, one can show that

$$C(s) = s - s^{n+3}A(s) + s^{2n+5}[sA(s)A'(s) + (n + 3)A^2(s)] \pmod{s^{3n+6}},$$

so in REV$(3n + 5) + O(n)$ operations we can compute $sA(s)A'(s) + (n + 3)A^2(s)$ mod s^{n+1}. By subtraction, we get $A^2(s)$ mod s^{n+1}. Hence $S(n) \leq$ REV$(3n + 3)$ + REV$(3n + 5)$ + $O(n)$. The result follows from Lemmas 1.2 and 4.1 and the fact that REV satisfies Condition B. □

THEOREM 4.1. *REV$(n) = O($COMP$(n))$ and COMP$(n) = O($REV$(n))$.*

PROOF. From Theorem 3.1,

$$\text{REV}(2k + 1) \leq \text{REV}(k) + \text{COMP}(2k + 1) + O(M(2k + 1)).$$

Similarly, if only $2k$ coefficients are wanted, we have

$$\text{REV}(2k) \leq \text{REV}(k) + \text{COMP}(2k) + O(M(2k)).$$

Hence for any positive integer n, we have

$$\text{REV}(n) \leq \text{REV}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \text{COMP}(n) + O(M(n)).$$

This implies that

$$\text{REV}(n) = O(\sum \text{COMP}(\lceil 2^{-j}n \rceil)) + O(\sum M(\lceil 2^{-j}n \rceil)),$$

where the sums are taken over all integers $j = 0, \ldots, \lceil \log n \rceil$. Since COMP and M satisfy Condition A, by Lemma 1.1,

$$\text{REV}(n) = O(\text{COMP}(n)) + O(M(n)).$$

The first half of the theorem follows from Lemma 4.2.

To prove the second half, let $P(s) = p_1 s + p_2 s^2 + \cdots$ and $Q(t) = q_0 + q_1 t + \cdots$. We show how to obtain $R(s) = Q(P(s))$ using reversions.

If $p_1 = p_2 = \cdots = p_n = 0$, then $q_0 = R(s)$ mod s^{n+1}. Hence, we may suppose that there exists $k \leq n$ such that $p_k \neq 0$ and that if $k > 1$ then $p_1 = \cdots = p_{k-1} = 0$. Let $\tilde{P}(s) = (P(s)/p_k)^{1/k}$ and $\tilde{Q}(t) = Q(p_k t^k) - q_0$, so $R(s) = \tilde{Q}(\tilde{P}(s)) + q_0$ and $s = \tilde{P}(s)$ mod s^2. By Lemma 6.2 we can compute $\tilde{P}(s)$ mod s^{n+1} in $O(M(n))$ operations. By Lemma 4.3, $M(n)$ is $O($REV$(n))$. Thus, there is no loss of generality in assuming below that $p_1 = 1$ and $q_0 = 0$. Define

$$\begin{aligned} V(t) &= P^{(-1)}(t) \bmod t^{2n+2}, \\ \tilde{V}(t) &= (V(t) - t^{n+1}Q(t)V'(t)) \bmod t^{2n+2}, \\ \tilde{P}(s) &= \tilde{V}^{(-1)}(s) \bmod s^{2n+2}. \end{aligned}$$

We claim that

$$R(s)P^{n+1}(s) = \tilde{P}(s) - P(s) \pmod{s^{2n+2}}. \tag{4.1}$$

To prove this, note that

$$P(\tilde{V}(t)) = P(V(t)) - P'(V(t))t^{n+1}Q(t)V'(t) \pmod{t^{2n+2}}.$$

But $P(V(t)) = t \pmod{t^{2n+2}}$, so $P'(V(t))V'(t) = 1 \pmod{t^{2n+1}}$ and thus

$$P(\tilde{V}(t)) = t - t^{n+1}Q(t) \pmod{t^{2n+2}}. \tag{4.2}$$

Now substituting $\tilde{P}(s)$ for t in (4.2), we obtain

$$P(s) = \tilde{P}(s) - \tilde{P}^{n+1}(s)Q(\tilde{P}(s)) \pmod{s^{2n+2}}. \tag{4.3}$$

Note that $\tilde{P}(s) = P(s) \pmod{s^{n+2}}$ and $\deg \tilde{P}^{n+1} = n + 1$. Thus, (4.3) implies that

$$P(s) = \tilde{P}(s) - P^{n+1}(s)Q(P(s)) \pmod{s^{2n+2}}.$$

We have proved (4.1). Hence

$$s^{n+1}R(s) = (s/P(s))^{n+1}(\tilde{P}(s) - P(s)) \pmod{s^{2n+2}}. \tag{4.4}$$

We can compute $R(s) \bmod s^{n+1}$ by the following algorithm:

1 Compute $V(t)$ and $\tilde{V}(t) \bmod t^{2n+2}$
2. Compute $\tilde{P}(s) \bmod s^{2n+2}$
3. Compute $(s/P(s))^{n+1} \bmod s^{2n+2}$ by the method of Lemma 6 2
4 Compute $R(s) \bmod s^{n+1}$, using (4 4)

Therefore, we have $\mathrm{COMP}(n) \leq 2\ \mathrm{REV}(2n + 1) + O(M(2n + 1))$. Since REV satisfies Condition B, the second half of the theorem follows from Lemmas 1.2 and 4.3. □

5. *Special Functions of Power Series*

Let $P, Q \in \mathscr{P}$, $p_0 = 0$, and $R(s) = Q(P(s))$. In this section we show that $L(r_0, \ldots, r_n \bmod p_1, \ldots, p_n, q_0, \ldots, q_n) = O(M(n))$ if Q satisfies a suitable ordinary differential equation. It is an open problem whether a similar result holds when P rather than Q satisfies a differential equation.

The results given in this section suffice for most practical applications. We do not attempt to state the most general results possible, because this would involve us too deeply in the theory of differential equations.

For completeness, we sketch the result of Brent [8] that log and exp of power series may be computed with $O(M(n))$ operations.

Evaluation of $\log(1 + P(s))$. If $R(s) = \log(1 + P(s))$ then $R'(s) = P'(s)/(1 + P(s))$. Thus we can evaluate the first n terms of $R'(s)$ in $O(M(n))$ operations, and it is easy to deduce the first $n + 1$ terms of $R(s)$.

Evaluation of $\exp(P(s))$. If $R(s) = \exp(P(s))$ then $\log(R(s)) - P(s) = 0$, and this equation may be solved by Newton's method. If

$$R_0(s) = 1 \quad \text{and} \quad R_{i+1}(s) = R_i(s) - R_i(s)(\log(R_i(s)) - P(s)),$$

then $R_i(s) = R(s) \bmod s^{2^i}$. Thus, the number of operations required to find the first $n + 1$ terms of $R(s)$ is $O(M(n) + M(\lceil n/2 \rceil) + M(\lceil n/4 \rceil) + \cdots)$ and, by Lemma 1.1, this is $O(M(n))$.

Reduction to Differential Equation in R. Suppose the differential equation satisfied by $Q(t)$ is $\phi(t, Q(t), Q'(t), \ldots, Q^{(m)}(t)) = 0$. We may substitute $t = P(s)$ and use the chain rule to obtain a differential equation in $R(s) = Q(P(s))$. The number of operations required to make this substitution depends on m and the form of ϕ, but in many cases of practical interest it is only $O(M(n))$. Some examples are given below. Since m is fixed, any method gives $r_1, \ldots, r_{m-1}$ in $O(1)$ operations. Thus, we can assume that $R(s)$ satisfies a given differential equation $\Phi(s, R(s), R'(s), \ldots, R^{(m)}(s)) = 0$, with initial conditions $R(0) = r_0, \ldots, R^{(m-1)}(0)/(m-1)! = r_{m-1}$, and the problem is to compute $r_m, \ldots, r_n$.

5.1 First-Order Linear Equations. It is easy to deal with first-order linear equations of the form $R'(s) + A(s)R(s) = B(s)$, $R(0) = r_0$, where A and B are given power series. The well-known method of integrating factors gives

$$R(s) = (1/J(s))\left(r_0 + \int_0^s B(u)J(u)du\right),$$

where $J(s) = \exp(\int_0^s A(u)du)$. Since we can compute exponentials of power series and perform formal integrations, $R(s) \bmod s^{n+1}$ can be computed in $O(M(n))$ operations.

We also need to consider the equation

$$R'(s) + (\alpha/s + A(s))R(s) = \beta/s + B(s),$$

where $\alpha \neq 0$ and $R(0) = r_0 = \beta/\alpha$. Using the method of integrating factors again, we obtain

$$R(s) = (1/J(s))\left\{s^{-\alpha}\int_0^s u^{\alpha}[B(u)J(u) + \beta((J(u) - 1)/u)]du + \beta/\alpha\right\}.$$

If α is a negative integer, we assume that the coefficient of u^{-1} in the integrand is zero, for otherwise no power series solution exists. Since

$$s^{-\alpha}\int_0^s u^{\alpha}\sum_{j=0}^{\infty} c_j u^j du = \sum_{\substack{j=0 \\ j\neq -(\alpha+1)}}^{\infty} (c_j/(j + \alpha + 1))s^{j+1},$$

provided $c_j = 0$ if $j + \alpha + 1 = 0$, there is no difficulty in performing the formal integration, even if α is not an integer.

5.2 First-Order Nonlinear Equations. It is well known that nonlinear differential equations can be solved by Newton's method if the corresponding linearized equation can be solved. See, for example, Rall [34]. We shall not attempt full generality here, but shall illustrate the idea using the Riccati equation

$$\mathscr{L}R(s) \equiv R'(s) + A(s)R(s) - (R(s))^2 - B(s) = 0,$$

where $A(s)$ and $B(s)$ are given power series, and $R(0) = r_0$.

Since (using Rall's notation) $\mathscr{L}'(R) = d/ds + (A - 2R)I$, the Newton iteration is

$$R_{j+1}(s) = R_j(s) - (1/J_j(s))\int_0^s (R_j(u))J_j(u)du,$$

where $J_j(s) = \exp(\int_0^s (A(u) - 2R_j(u))du)$. To study the convergence property of Newton's method a norm is often used. For our purpose, we use a valuation on $\mathscr{P}$. Then the quadratic convergence of an iteration on $\mathscr{P}$ means that the number of correct terms doubles at each iteration. (See Kung and Traub [29] for details.) Using a Newton-Kantorovich type theorem (see, e.g. Bachman [3, pp. 52–55] and Rall [34, pp. 135–138]), one can easily show that if the initial approximation $R_0(s) = r_0 + \cdots r_l s^l$ is taken to be an initial segment of the solution series with l sufficiently large, then Newton's method converges quadratically. The terms in $R_0(s)$ may be obtained, for example, by equating coefficients. Since l is fixed, any method gives $R_0(s)$ in $O(1)$ operations. Thus to compute $R(s)$ mod s^n, we compute $R_j(s)$ mod s^{2^j+l} and only $\lceil \log_2(n - l)\rceil$ iterations are required. Since $O(M(2^j + l))$ operations are needed at the jth iteration, the number of operations is $O(1) + O(M(n) + M(\lceil n/2\rceil) + \cdots) = O(M(n))$.

The generalization to the Riccati equation in which $A(s)$ is replaced by $\alpha/s + A(s)$ and $B(s)$ by $\beta/s + B(s)$ is straightforward. In fact, the following theorem can be shown by the above argument.

Theorem 5.1. *If a formal power series solution exists for the differential equation*

$$R'(s) = F(s, R(s)), \quad R(0) = r_0,$$

where F is a bivariate rational expression, then the first n terms of the solution series can be computed in $O(M(n))$ operations.

The generalization of Theorem 5.1 to the case where F itself is a bivariate infinite power series or to the case of vector differential equations is straightforward. For example, consider the following differential equation:

$$R'(s) = F(R(s)), \quad R(0) = 0, \tag{5.1}$$

where F is a univariate power series. To compute the first n terms in R we need only the first n terms in F. When we solve (5.1) by Newton's method, the main cost of each iteration is due to composition. Hence the first n terms in R can be obtained in $O(\mathrm{COMP}(n))$ operations. It is instructive to note that if V is the reversion of the P defined by (3.1) then by the chain rule $P'(V(s))\ V'(s) = 1$. Thus V is the power series solution of (5.1) with

$F = 1/P'$. This gives another proof that REV(n) = O(COMP(n)). By the result of Section 4, we have therefore shown that the problem of solving differential equation (5.1), the composition problem, and the reversion problem are all equivalent.

5.3 SECOND-ORDER LINEAR EQUATIONS. Suppose $R''(s) + A(s)R'(s) + B(s)R(s) = C(s)$, where $A(s)$, $B(s)$, and $C(s)$ are given power series, and $R(0) = r_0$, $R'(0) = r_1$. The well-known method of factorization (Burkill [6]) reduces this second-order problem to three first-order problems, one of which is nonlinear. If $\mathscr{D}$ is the differentiation operator, we want power series $S(s)$ and $T(s)$ such that $(\mathscr{D} + S)(\mathscr{D} + T)R = \mathscr{D}^2R + A\mathscr{D}R + BR$, i.e. $S + T = A$ and $T' + ST = B$, which gives $T' + AT - T^2 - B = 0$. This is just the Riccati equation discussed above. The initial condition $T(0) = t_0$ may be chosen arbitrarily. Once T and $S = A - T$ are known, we may solve the first-order linear equations $U' + SU = C$, $U(0) = r_1 + t_0r_0$ and $R' + TR = U$, $R(0) = r_0$ to obtain $U = (\mathscr{D} + T)R$ and then R. Hence $R(s)$ mod s^{n+1} can be computed in $O(M(n))$ operations.

The generalization in which $A(s)$ is replaced by $\alpha/s + A(s)$, etc., is similar, except that $t_0 = \beta/\alpha$ is chosen so that $T(s)$ is a power series.

By repeated application of linearization (i.e. Newton's method) and factorization, the solution of a differential equation of arbitrary order can be reduced to the solution of first-order linear equations. In practice second-order equations are the most common, and we give two examples below.

Hypergeometric Functions of Power Series. As our first example we consider the computation of $R(s) = F(a, b; c; P(s))$, where F is the hypergeometric function

$$F(a, b; c; z) = \sum_{j=0}^{\infty} ((a)_j(b)_j/(c)_j) \cdot z^j/j!.$$

(Here $(a)_j = \Gamma(a + j)/\Gamma(a)$, etc.) By suitable choice of a, b, and c, many elementary functions can be written in this form; see Abramowitz and Stegun [1]. Now $w = F(a, b; c; z)$ satisfies the hypergeometric differential equation

$$z(1 - z)d^2w/dz^2 + [c - (a + b + 1)z]dw/dz - abw = 0,$$

so substituting $z = P(s)$, $w = R(s)$ and using the chain rule gives

$$R'' + \{[c - (a + b + 1)P]P'/[P(1 - P)] - P''/P'\}R' - (ab(P')^2/(P(1 - P)))R = 0,$$

with initial conditions $R(0) = 1$ and $R'(0) = abP'(0)/c$. Thus, we have a second-order linear equation whose power series solution may be obtained as described above, and to compute $R(s)$ mod s^{n+1} requires only $O(M(n))$ operations. Generalized hypergeometric functions of power series may also be computed in $O(M(n))$ operations, using the generalized hypergeometric equation (Henrici [21]) and an obvious generalization of our method.

The algorithm for hypergeometric functions over the real field has been implemented in Fortran. Numerical tests indicate that the effect of rounding errors is usually no worse, and often better, than for the obvious $O(n^3)$ algorithm. However, a rigorous analysis of the numerical properties of our algorithms has not yet been attempted. Special cases which have been tested numerically include $F(1, 1; 2; 1 - e^s) = s/(e^s - 1)$, $F(-a, a; \frac{1}{2}; \sin^2(\sqrt{s})) = \cos(2a\sqrt{s})$, and $F(\frac{1}{2}, \frac{1}{2}; \frac{3}{2}; s^2) = \arcsin(s)/s$.

Bessel Functions of Power Series. Our second example is the computation of $R(s) = J_\nu(P(s))$, where the Bessel function $w = J_\nu(z) = (z/2)^\nu \sum_{k=0}^{\infty} (-\frac{1}{4} z^2)^k/(k!(\nu + k)!)$ satisfies the differential equation

$$d^2w/dz^2 + (1/z)dw/dz + (1 - \nu^2/z^2)w = 0.$$

We may substitute $w = R(s)$ and $z = P(s)$ to obtain a second-order equation for R, and proceed as above. A slight generalization is necessary to deal with the ν^2/z^2 term, but this can be avoided by making the change of variables $w = z^\nu W$, which gives a differential equation

$$d^2W/dz^2 + ((2v + 1)/z)dW/dz + W = 0$$

of the form discussed above.

6. *Evaluation of Truncated Reversion at a Point*

Let $P \in \mathscr{P}$, $p_0 = 0$, and $V = P^{(-1)}$. In this section we show that $L(v_n \bmod p_1, \ldots, p_n) = O(M(n))$ and $L(V_n(a) \bmod a, p_1, \ldots, p_n) = O(M(n))$, where $V_n(t)$ is the "truncated reversion" of $P(s)$, i.e. $V_n(t) = v_1 t + v_2 t^2 + \cdots + v_n t^n$.

We need some definitions. The quotient field of $\mathscr{P}$ is isomorphic to the field $\mathscr{Q}$ of formal Laurent series over K, i.e. series $\sum_{j=-\infty}^{\infty} a_j t^j$ where $a_j \in K$ and only finitely many a_j are nonzero for negative j. If $A \in \mathscr{Q}$ we define the "residue" of A to be $\operatorname{res}_t[A(t)] = a_{-1}$.

LEMMA 6.1. *$res_t[A(t)] = -res_t[tA'(t)]$.*

LEMMA 6.2 (Brent [8]). *Let $P(s) = p_k s^k + p_{k+1}s^{k+1} + \cdots$ with $p_k \neq 0$ for some $k \geq 0$ be given. Let $R(s) = P^{\sigma}(s)$ for some number $\sigma \neq 0$. If p_k^{σ} is given, then the first n terms in $R(s)$ can be computed in $O(M(n))$ operations.*

PROOF. Define $\tilde{P}(s)$ by $P(s) = p_k s^k[1 + \tilde{P}(s)]$. Then

$$R(s) = p_k^{\sigma} s^{\sigma k}[1 + \tilde{P}(s)]^{\sigma} = p_k^{\sigma} s^{\sigma k} \exp\{\sigma \cdot \log[1 + \tilde{P}(s)]\}.$$

The lemma follows from the preliminary results of Section 5. □

LEMMA 6.3. $v_n = (1/n)\, res_s[P^{-n}(s)] = res_s[sP'(s)/P^{n+1}(s)]$.

PROOF. The first equation follows from the Lagrange-Burmann Theorem (see, e.g. Henrici [21] and Knuth [26]) and the second equation follows from Lemma 6.1. □

THEOREM 6.1. *$L(v_n \bmod p_1, \ldots, p_n) = O(M(n))$.*

PROOF. Note that $\operatorname{res}_s[P^{-n}(s)]$ is the coefficient of s^{n-1} in $[s/P(s)]^n$. Thus, the result follows from Lemmas 6.2 and 6.3. □

LEMMA 6.4.

$$V_n(a) = res_s[(a^{n+1}sP'(s))/(P^{n+1}(s)(a - P(s)))].$$

PROOF. From Lemma 6.3 and the definition of V_n we have

$$V_n(a) = \sum_{j=1}^{n} \operatorname{res}_s[sP'(s)/P^{j+1}(s)]a^j = \operatorname{res}_s\left[\frac{sP'(s)}{P(s)} \sum_{j=1}^{n} (a/P(s))^j\right]$$
$$= \operatorname{res}_s\left[\frac{sP'(s)a(a^n - P^n(s))}{P^{n+1}(s)(a - P(s))}\right].$$

Since $\operatorname{res}_s[sP'(s)a/(P(s)(a - P(s)))] = 0$, the result follows. □

THEOREM 6.2. *$L(V_n(a) \bmod a, p_1, \ldots, p_n) = O(M(n))$.*

PROOF. If $a = 0$ then $V_n(a) = 0$. If $a \neq 0$ then, from Lemma 6.4, $V_n(a)$ is the coefficient of s^{n-1} in $a^{n+1}P'(s)/((P(s)/s)^{n+1}(a - P(s)))$. Thus, the result follows from Lemma 6.2. □

Application to Root Finding. Suppose K is the real or complex field, $f: D \subseteq K \to K$ is a sufficiently smooth function with a simple zero ξ in the interior of D, and x_0 is a sufficiently good approximation to ξ. The direct and inverse polynomial interpolation methods (Traub [38]) may be used to obtain a better approximation $x_1 = \xi + O(|x_0 - \xi|^{n+1})$. Both methods depend on the evaluation of $f(x_0), f'(x_0), \ldots, f^{(n)}(x_0)$. For the direct method, x_1 is chosen to be a sufficiently good approximation to the appropriate zero of the Taylor polynomial $\sum_{j=0}^{n} (x - x_0)^j f^{(j)}(x_0)/j!$. If this zero is found by Newton's method with x_0 as the starting approximation, then $\lceil \log_2(n + 1) \rceil$ iterations are required so the combinatory cost (Kung and Traub [28]) or "overhead" is $O(n \log n)$. Tarjan has shown that this can be reduced to $O(n)$.

If $P(s) = f(x_0 + s) - f(x_0) = \sum_{j=1}^{\infty} s^j f^{(j)}(x_0)/j!$, and $V = P^{(-1)}$ is the reversion of P, then $P(\xi - x_0) = -f(x_0)$, so $\xi = x_0 + V(-f(x_0))$. The inverse polynomial interpolation method avoids the need to find the zero of a polynomial by approximating V rather than P. In fact, the inverse method takes $x_1 = x_0 + V_n(-f(x_0))$, where V_n is the truncated reversion of P (or, equivalently, of $P_n(s) = \sum_{j=1}^{n} s^j f^{(j)}(x_0)/j!$). From Theorem 6.2, the combinatory cost is

$O(M(n)) = O(n \log n)$. Thus, the combinatory cost of both the direct and inverse methods is $O(n \log n)$. This result is mainly of theoretical interest, for in practice n is usually small.

Application to Queueing Theory. By a result of Brockwell [11] and Finch [15, 16] it can be shown that for a $GI/M/1$ queue which is initially empty, the probability that the nth arrival finds more than j customers in the queue is the coefficient of s^{n-j-1} in the generating function $P'(s)/((P(s)/s)^{n+1}(1 - P(s)))$ for some given power series $P(s)$. Hence the method used in the proof of Theorem 6.2 can be applied with small changes for computing the probabilities. The details of this result will be given in a separate paper.

Evaluation of One Coefficient in Composition. Let $P, Q \in \mathscr{P}, p_0 = 0$, and $R(s) = Q(P^{(-1)}(s))$. The following theorem is similar to Theorem 6.1.

THEOREM 6.3. $L(r_n \bmod p_1, \ldots, p_n, q_0, \ldots, q_n) = O(M(n))$.

PROOF. Since $r_0 = q_0$, we may suppose $n > 0$. From the Lagrange-Bürmann theorem,

$$r_n = \operatorname{res}_s[Q'(s)/P^n(s)]/n = \text{coefficient of } s^{n-1} \text{ in } Q'(s)(s/P(s))^n/n,$$

so the result follows from Lemma 6.2. □

It is an open problem whether Theorem 6.3 holds if $R(s) = Q(P(s))$ instead of $Q(P^{(-1)}(s))$.

A Numerical Example. Taking $P(s) = 1 - \exp(-s)$, we evaluated the truncated reversion V_n at a for various n and a by the algorithm establishing Theorem 6.2, using a Univac 1108 computer with a 60-bit floating-point fraction. The effect of rounding errors increased as n increased, but was not excessive for small values of a. (In the root-finding application a should be small.) It seems that the growth in rounding error is due to the ill-conditioning of the problem. Some typical results are given in Table I, where $\tilde{V}_n(a)$ are the computed values.

TABLE I

n	a	$\lvert\tilde{V}_n(a) - V_n(a)\rvert$	n	a	$\lvert\tilde{V}_n(a) - V_n(a)\rvert$
16	0.1	3×10^{-18}	32	0.2	8×10^{-13}
16	0.2	1×10^{-16}	32	0.4	5×10^{-11}
16	0.4	1×10^{-14}	32	0.8	8×10^{-3}
16	0.8	5×10^{-11}	64	0.1	4×10^{-15}
16	1.6	6×10^{-7}	64	0.2	5×10^{-12}
32	0.1	9×10^{-17}	64	0.4	2×10^{-3}

7. *Multivariate Cases*

We have so far dealt with power series in one variable. The results of previous sections in principle can be applied and generalized to power series in several variables, provided that appropriate care is taken to handle various singularity problems associated with multivariate power series. In this section we state some of our results on the composition problem for bivariate power series. For more complete treatment of the multivariate case, the reader is referred to Brent and Kung [10].

We first extend our mod s^{n+1} notation to bivariate power series. Let $Q(s, t) = \sum_{i,j=0}^{\infty} q_{i,j} s^i t^j$ be a bivariate power series. We define the degree of the term $q_{i,j}s^it^j$ to be $i + j$. $Q(s, t) \bmod (s + t)^{n+1}$ denotes the finite series consisting of all terms of $Q(s, t)$ of degree less than $n + 1$. To compute $Q(s, t) \bmod (s + t)^{n+1}$ means to compute the $q_{i,j}$ for all i, j such that $i + j \le n$.

THEOREM 7.1. *Given a bivariate power series Q and two univariate series P_1, P_2 with no constant terms, $R(s) = Q(P_1(s), P_2(s)) \bmod s^{n+1}$ can be computed in $O(n^2 \log n)$ operations.*

THEOREM 7.2. *Given a univariate power series Q and a bivariate series P with no constant term, $R(s, t) = Q(P(s, t)) \bmod (s + t)^{n+1}$ can be computed in $O(n^{2.5} \log n)$ operations.*

THEOREM 7.3. *Given three bivariate power series P_1, P_2, and Q, where P_1 and P_2 have no constant terms, $R(s, t) = Q(P_1(s, t), P_2(s, t)) \bmod (s + t)^{n+1}$ can be computed in $O(n^{2.5} \log^{1.5} n)$ operations.*

Note that the classical bounds for the composition problems considered in Theorems 7.1, 7.2, and 7.3 are $O(n^4)$, $O(n^5)$, and $O(n^6)$, respectively (or $O(n^3 \log n)$, $O(n^3 \log n)$, and $O(n^4 \log n)$, respectively, if the FFT polynomial multiplication is used).

ACKNOWLEDGMENT. We wish to thank Professors B.D. Craven and J.F. Traub for their comments, and Professor P. Henrici for simplifying the proof of Lemma 6.4.

REFERENCES

1. ABRAMOWITZ, M., AND STEGUN, I A *Handbook of Mathematical Functions.* Nat. Bur Stand., Washington, D C., 1964, chap 15
2. AHO, A V, HOPCROFT, J E, AND ULLMAN, J.D *The Design and Analysis of Computer Algorithms* Addison-Wesley, Reading, Mass, 1974.
3 BACHMAN, G *Introduction to P-Adic Numbers and Valuation Theory* Academic Press, New York, 1964
4 BORODIN, A. On the number of arithmetics to compute certain functions—Circa May 1973. In *Complexity of Sequential and Parallel Numerical Algorithms,* J F Traub, Ed, Academic Press, New York, 1973, pp 149–180.
5 BORODIN, A, AND MUNRO, I *The Computational Complexity of Algebraic and Numeric Problems* American Elsevier, New York, 1975
6 BURKILL, J C *The Theory of Ordinary Differential Equations* Oliver and Boyd, London, 1962, Sec. 9.
7. BRENT, R P The complexity of multiple-precision arithmetic. In *Complexity of Computational Problem Solving,* R S Anderssen & R P Brent, Eds, U of Queensland Press, Brisbane, Australia, 1975, pp. 126–165
8 BRENT, R P Multiple-precision zero-finding methods and the complexity of elementary function evaluation In *Analytic Computational Complexity,* J F Traub, Ed., Academic Press, New York, 1975, pp. 151–176
9. BRENT, R P, AND KUNG, H T. $O((n \log n)^{3/2})$ algorithms for composition and reversion of power series (extended abstract) In *Analytic Computational Complexity,* J F. Traub, Ed, Academic Press, New York, 1976, pp 217–225
10. BRENT, R P, AND KUNG, H.T. Fast algorithms for composition and reversion of multivariate power series Proc. Conf. Theoret Comptr Sci, U. of Waterloo, Waterloo, Ont., Canada, Aug 1977, pp 149–158
11 BROCKWELL, P J The transient behaviour of the queueing system *GI/M/1 J Austral Math Soc 3* (1963), 249–256
12 CHRYSTAL, G. *Textbook of Algebra, Part II* Chelsea, New York, 1964
13 FATEMAN, R J Polynomial multiplication, powers and asymptotic analysis Some comments *SIAM J Compting 3* (1974), 196–213
14 FERGUSON, H R P, NIELSEN, D E, AND COOK, G A partition formula for the integer coefficients of the theta function nome *Math Comput 29* (1975), 851–855
15 FINCH, P D The single server queueing system with non-recurrent input-process and Erlang service time *J Austral Math Soc 3* (1963), 220–236
16 FINCH, P D On partial sums of Lagrange's series with application to theory of queues *J Austral Math Soc 3* (1963), 488–490
17 FISCHER, M J, AND STOCKMEYER, L J Fast on-line integer multiplication *J Comptr Syst Sci 9* (1974), 317–331
18 GILBERT, E N Enumeration of labelled graphs *Canadian J Math 8* (1956), 405–411
19 HEINDEL, L E Computation of powers of multivariate polynomials over the integers *J Comptr Syst Sci 6* (1971), 1–8
20 HENRICI, P Automatic computation with power series *J ACM 3,* 1 (Jan 1956), 10–15
24 HENRICI, P *Applied and Computational Complex Analysis, Vol 1* Wiley-Interscience, New York, 1974, chap 1.
22 HOPCROFT, J E Complexity of computer computations Information Processing 74, North-Holland Pub Co, Amsterdam, 1974, pp 620–626.
23 HOROWITZ, E On the substitution of polynomial forms Proc ACM 1973 Annual Conf, CR No 28686, 1973, pp 153–158
24 HOROWITZ, E The efficient calculation of powers of polynomials *J Comptr Syst Sci 7* (1973), 469–480
25 JACKSON, D M, AND REILLY, J W The enumeration of homeomorphically irreducible labelled graphs *J Combinatorial Theory, Ser B, 19* (1975), 272–286
26 KNUTH, D E *The Art of Computer Programming, Vol 2 Seminumerical Algorithms* Addison-Wesley, Reading, Mass, 1969, Sec 4 7
27 KUNG, H T On computing reciprocals of power series *Numer Math 22* (1974), 341–348
28 KUNG, H.T, AND TRAUB, J F Computational complexity of one-point and multipoint iteration. In *Complexity of Real Computation,* R M Karp, Ed, SIAM-AMS Proc 7, Amer Math Soc, Providence, R I, 1974, pp 149–160
29 KUNG, H T, AND TRAUB, J F All algebraic functions can be computed fast *J ACM 25,* 2 (April 1978), 245–260

30 LEVY, H., AND LESSMAN, F. *Finite Difference Equations* Pitman and Sons, London, 1959.
31 NIVEN, I. Formal power series *Amer. Math. Monthly 76* (1969), 871–889.
32 NORMAN, A. C. Computing with formal power series *ACM Trans. Math. Software 1*, 4 (Dec. 1975), 346–356
33 PATERSON, M. S., AND STOCKMEYER, L. J. On the number of nonscalar multiplications necessary to evaluate polynomials *SIAM J. Comptng. 2* (1973), 60–66
34 RALL, L. B. *Computational Solution of Nonlinear Operator Equations* Wiley, New York, 1969
35 RIORDAN, J. *An Introduction to Combinatorial Analysis* Wiley, New York, 1958
36 SIEVEKING, M. An algorithm for division of power series *Computing 10* (1972), 153–156
37 STRASSEN, V. Gaussian elimination is not optimal *Numer. Math. 13* (1969), 354–356
38 TRAUB, J. F. *Iterative Methods for the Solution of Equations* Prentice-Hall, Englewood Cliffs, N. J., 1964, chap. 5

RECEIVED FEBRUARY 1976, REVISED OCTOBER 1977