SIAM J. COMPUT.
Vol. 15, No. 2, May 1986

008

REPRESENTATIONS AND PARALLEL COMPUTATIONS FOR RATIONAL FUNCTIONS*

JOACHIM VON ZUR GATHEN†

Abstract. Representations of univariate rational functions over a given base of polynomials are considered, and a fast parallel algorithm for converting from one base representation to another is given. Special cases of this conversion include the following symbolic manipulation problems: Taylor expansion, partial fraction decomposition, Chinese remainder algorithm, elementary symmetric functions, Padé approximation, and various interpolation problems. If n is the input size, then all algorithms run in parallel time $O(\log^2 n)$ and use $n^{O(1)}$ processors. They work over an arbitrary field.

Key words. parallel processing, algebraic computing, symbolic manipulation, interpolation, Chinese remainder algorithm, Padé approximation

1. Introduction. This work forms part of an endeavour to understand the power of parallelism in symbolic computation: for which problems in algebraic manipulation with a polynomial-time sequential solution do fast parallel algorithms exist? Answers to this question may help to understand the power of parallelism in general: one may compare the power of different models of concurrent computation by testing on which models these algorithms can be implemented.

In this paper we present fast parallel algorithms for various problems in algebraic computation. It soon became apparent that the algorithms for all the problems considered here would follow the same pattern, namely conversion between different representations of the given rational function. So we start by introducing in § 2 the notion of representations of rational functions in a given "base" of polynomials. This notion encompasses several ways of representing rational functions, which are especially familiar if the rational function is a polynomial: the sequence of coefficients, a set of values, Taylor series, and a general list of values in "Hermite format". It turns out that if numerator and denominator degrees are correctly specified, then usually (but not always) such representations also determine a rational function uniquely.

We describe in § 3 two fast parallel algorithms that convert the standard coefficient representation of a rational function into a "base representation", and vice versa. Combining them we get an algorithm that converts the representation of a rational function in one base into that in another base. Section 4 discusses the existence question for representations. This turns out to be slightly less straightforward than one might expect, and is illustrated by the well-known fact that the rational functions required in Padé approximation or rational interpolation may fail to exist.

Section 5 presents as application of the general conversion algorithms fast parallel methods for the following problems in symbolic manipulation: Taylor expansion, partial fraction decomposition, Chinese remainder algorithm, elementary symmetric functions, Padé approximation, and various interpolation problems. As our model of parallel computation we can take an arithmetic network, which is a directed acyclic graph such that each edge either carries arithmetic values from the ground field or Boolean values, and with the following nodes: arithmetic operations (+, −, ×, /, fetch-

* Received by the editors June 28, 1983, and in final revised form December 10, 1984. An abstract of this paper appeared in the Proceedings of the 24th IEEE Symposium on Foundations of Computer Science, Tucson, Arizona, 1983, pp. 133–137. This work was partially supported by Natural Sciences and Engineering Research Council of Canada grant 3-650-126-40 and Schweizerischer Nationalfonds grant 2175-0.83.

† Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4.

ing a constant), tests ($a = 0$?) of an arithmetic value a, Boolean operations, and selection of one of two arithmetic inputs according to the value of a further Boolean input. The algorithms can also be implemented on an algebraic PRAM. The networks all have depth (= parallel time) $O(\log^2 n)$ and size (= number of processors) $n^{O(1)}$, where n is the input size. They work over an arbitrary ground field.

The representations discussed so far are sufficient to solve the computational problems of § 5. But their theoretical shortcomings warrant a generalization, so that in § 6 "Laurent representations" are introduced. They have the desired feature that rational functions now always have a unique representation.

The dual relationship between evaluation at many points and interpolation has been observed for a long time; also the fact that both computational problems consist in converting the representation of a polynomial from one format to another; see e.g. Strassen [1974]. However, one usually employs two quite different-looking sequential algorithms to solve the two problems. Besides the unification resulting from our approach even for polynomials, the fact of including rational functions into this framework seems to be even more interesting from a computational point of view, making bedfellows of such distinct-looking problems as Hermite interpolation and Padé approximation. We leave as an open question how well this approach translates into the sequential setting.

2. Representations of rational functions. Let F be an arbitrary field. A sequence $B = (b_1, \cdots, b_p)$ of pairwise relatively prime polynomials $b_1, \cdots, b_p \in F[x] \setminus F$ is called a base. A sequence $N = (n_1, \cdots, n_p) \in N^p$ with $n_i \geqq 1$ is called a precision (for B), and $n = \sum_{1 \leqq i \leqq p} n_i \deg b_i$ is the total precision of (B, N). A sequence

$$r = (r_{10}, \cdots, r_{1,n_1-1}; \cdots; r_{p0}, \cdots, r_{p,n_p-1})$$

such that for all i, j with $1 \leqq i \leqq p$ and $0 \leqq j < n_i$

$$r_{ij} \in F[x],$$

$$\deg r_{ij} < \deg b_i,$$

is called a *representative* in base B with precision N, or (B, N)-representative for short. $R(B, N)$ is the set of all (B, N)-representatives. $R(B, N)$ can be identified with F^n.

DEFINITION 2.1. Let $r \in R(B, N)$, and $g, h \in F[x]$ with $h \neq 0$. Then r is called a *weak (B, N)-representation of* (g, h) if for $1 \leqq i \leqq p$ we have

$$g \equiv h \sum_{0 \leqq j < n_i} r_{ij} b_i^j \mod b_i^{n_i}.$$

r is called a *(B, N)-representation of* $f = g/h \in F(x)$ if in addition

$$\gcd(b_1 \cdots b_p, h) = 1. \qquad \square$$

In other words, if we set

$$r_i = \sum_{0 \leqq j < n_i} r_{ij} b_i^j,$$

then $f \equiv r_i \mod b_i^{n_i}$ is given with "precision $n_i \deg b_i$", and r_i is developed according to powers of b_i.

One verifies that for given B, N, f, r the property "r is a (B, N)-representation of f" does not depend on the choice of polynomials g, h such that $f = g/h$ and

$\gcd(b_1 \cdots b_p, h) = 1$. Ideally, we would like our representations to have two properties:

Representability: Given (B, N), every $f \in F(x)$ has a unique (B, N)-representation.

Convertibility: There exists a fast parallel algorithm for converting representations from one base into another base.

As they stand, both properties fail to hold; but fortunately this failure is only marginal. Representability is discussed in § 4 and Remark 5.4, and it turns out that almost all rational functions have a unique representation. The second conversion algorithm of § 3 may return a pair of polynomials of which the input representation is only a "weak representation"; this is the reason for introducing the latter. Again, for almost all inputs this weakness does not appear. Note that it does not make sense to speak of weak representations of a rational function, since the above congruence alone would allow any (B, N)-representative r to represent any rational function $f = g/h \in F(x)$, by writing $f = (bg)/(bh)$ with $b = b_1^{n_1} \cdots b_p^{n_p}$. The gcd condition rules these unwanted representations out.

In § 6, we introduce the more general "Laurent representations". These then have both desired properties, and thus are the appropriate objects for this theory. Though inappropriate from a theoretical point of view, there are two reasons for introducing the representations and weak representations as above: Firstly, they are easier to define and work with, and secondly, they are sufficient to discuss the applications in § 5.

If $N = (n_1, \cdots, n_p)$ and $M = (m_1, \cdots, m_p)$ are two precisions for B, $m_i \leqq n_i$ for all i, and $r \in R(B, N)$ is a representation of f, then one obtains a representation in $R(B, M)$ of f by truncating the r_{ij}'s with $j \geqq m_i$.

Before proceeding with the general development, let us look at a few familiar representations that fit into this framework. As examples we will use $n = 5$ and the two rational functions

$$f_1 = x^4 - x^3 + 2x^2 - 3x - 2 \in F[x], \qquad f_2 = \frac{-7x^2 + x + 2}{x^2 + x - 1} \in F(x).$$

When B and N are given, we will write $r = \rho(f)$ if r is a (B, N)-representation of f.

1. *Sequence of coefficients.* In this most familiar of all representations we have $p = 1$, $B = (x)$ and $N = (n)$. For a polynomial $f = \sum_{0 \leqq j} f_j x^j \in F[x]$, the (B, N)-representation $\rho(f) = (f_0, \cdots, f_{n-1})$ is simply the sequence of the first n coefficients (irrespective of the degree of f). For a rational function f (with nonzero constant term in the denominator polynomial), $\rho(f)$ is an initial segment of the power series expansion of f. In the example, $N = (5)$ and $\rho(f_1) = \rho(f_2) = (-2, -3, 2, -1, 1)$.

2. *List of values.* Given are $a_1, \cdots, a_n \in F$ pairwise distinct, and $p = n$, $B = (x - a_1, \cdots, x - a_n)$ and $N = (1, \cdots, 1)$. Then $r_i = f(a_i)$ is the value of f at a_i. For the example, let $(a_1, \cdots, a_5) = (-2, -1, 0, 1, 2)$ (and assume that $\operatorname{char} F \neq 5$, so that $\gcd(x - 2, x^2 + x - 1) = 1$). Then $B = (x + 2, x + 1, x, x - 1, x - 2)$, $N = (1, 1, 1, 1, 1)$, $\rho(f_1) = (36, 5, -2, -3, 8)$, and $\rho(f_2) = (-28, 6, -2, -4, \frac{-24}{5})$.

3. *Taylor expansion.* Here some $a \in F$ is given, and $p = 1$, $B = (x - a)$, $N = (n)$, and $r = (r_0, \cdots, r_{n-1})$ where

$$f \equiv \sum_{0 \leqq j < n} r_j (x - a)^j \mod (x - a)^n,$$

so that $r_0, \cdots, r_{n-1}$ are the first n coefficients of the Taylor expansion of f at a. (It is assumed that f can be written with a denominator that does not vanish at a.) If

char $F=0$, then

$$r_j=\frac{1}{j!}\left(\frac{d^jf}{dx^j}\right)(a).$$

Note that these Taylor coefficients are well-defined for any field F, whereas the expression on the right-hand side above does not make sense if char $F\leqq j$.

The sequence of coefficients is nothing but the Taylor expansion at 0. In the example, if we choose $a=2$ and $n=5$ (and char $F\neq 5$), then

$$\rho(f_1)=(8,25,20,7,1) \quad \text{and} \quad \rho(f_2)=\left(\frac{-24}{5},\frac{-3}{5},\frac{4}{25},\frac{-1}{25},\frac{1}{125}\right).$$

4. *General list of values.* As a common generalization of the last two cases, we get the general list of values of the rational function and some of its derivatives in "Hermite format". We have $a_1,\cdots,a_p\in F$ pairwise distinct, $B=(x-a_1,\cdots,x-a_p)$ and $N=(n_1,\cdots,n_p)$ with $n_1+\cdots+n_p=n$. Then

$$\rho(f)=(r_{10},\cdots,r_{1,n_1-1};\cdots;r_{p0},\cdots,r_{p,n_p-1}),$$

where

$$r_i=\sum_{0\leqq j<n_i} r_{ij}(x-a_i)^j\equiv f \mod (x-a_i)^{n_i}$$

is the initial segment of length n_i of the Taylor expansion of f at a_i. In the example, we choose $p=2$, $a_1=-1$, $a_2=2$, $B=(x+1,x-2)$ and $N=(2,3)$ (and again assume char $F\neq 5$). Then

$$\rho(f_1)=(5,-14;8,25,20) \quad \text{and} \quad \rho(f_2)=\left(6,-21;\frac{-24}{5},\frac{-3}{5},\frac{4}{25}\right).$$

Remark. Note that the standard encoding of a rational function $f=g/h$ by the coefficients of g and h is not a representation in our sense. However, it will play a crucial role in the conversion algorithms of the next section.

3. Conversion algorithms. We now describe two algorithms: the first one converts a rational function that is given as the quotient of two polynomials (and these polynomials are represented by their coefficient sequences) into its representation in base B with precision N, and the second one performs the inverse conversion. Thus the standard representation by coefficients plays a special role: The basic parts r_{ij} are given by their coefficients, and our conversions from one representation to another pass via this standard representation.

We will make use of the Extended Euclidean Scheme of two polynomials $a_0, a_1\in F[x]$, where $a_1\neq 0$:

$$\begin{aligned} a_0&=q_1a_1+a_2, & s_2a_0+t_2a_1&=a_2,\\ &\vdots & &\vdots\\ a_{l-2}&=q_{l-1}a_{l-1}+a_l, & s_la_0+t_la_1&=a_l,\\ a_{l-1}&=q_la_l, & s_{l+1}a_0+t_{l+1}a_1&=a_{l+1}, \end{aligned}$$

where the following conditions are satisfied for $2\leqq k\leqq l+1$: $a_k, q_k, s_k, t_k\in F[x]$, $a_{l+1}=0$, $a_{k-1}=q_ka_k+a_{k+1}$, $\deg a_k<\deg a_{k-1}$, $s_0=1$, $t_0=0$, $s_1=0$, $t_1=1$, $s_k=s_{k-2}-q_{k-1}s_{k-1}$ and $t_k=t_{k-2}-q_{k-1}t_{k-1}$. Thus the q's are the quotients and the a's the remainders of Euclid's algorithm, the s's and t's are the "continuants" or "convergents", and $\gcd(a_0,a_1)$ is

the unique monic scalar multiple of a_l. (By convention, all gcd's of polynomials in $F[x]$ are monic.) Also note that $\gcd(s_k, t_k) = 1$ and $\deg a_{k-1} + \deg t_k = \deg a_0$ for all $k \geqq 1$.

ALGORITHM STANDARD-TO-REPRESENTATION. (Standard coefficients to base representation.)

Input: A base $B = (b_1, \cdots, b_p)$ with a precision $N = (n_1, \cdots, n_p)$ and total precision n, and a pair (g, h) of polynomials in $F[x]$ such that $\gcd(b_1 \cdots b_p, h) = 1$. All input polynomials are given by their coefficient vectors.

Output: A (B, N)-representation r of $f = g/h$.

1. For all i, $1 \leqq i \leqq p$, do steps 2, 3, 4.
2. Compute $s_i, t_i \in F[x]$ such that
   $s_i b_i^{n_i} + t_i h = 1$,
   $\deg t_i < n_i \deg b_i$.
3. Compute $r_i \in F[x]$ such that
   $r_i \equiv g t_i \bmod b_i^{n_i}$,
   $\deg r_i < n_i \deg b_i$.
4. For all j, $0 \leqq j < n_i$, compute $u_{ij}, v_{ij}, r_{ij} \in F[x]$ such that
   $r_i = u_{ij} b_i^j + v_{ij}$,
   $\deg v_{ij} < j \deg b_i$,
   $r_{ij} = u_{ij} - u_{i,j+1} b_i$.
   (Division with remainder of r_i by b_i^j. Use $u_{i0} = r_i$ and $u_{i,n_i} = 0$.)
5. Return
   $r = (r_{10}, \cdots, r_{1,n_1-1}; \cdots; r_{p0}, \cdots, r_{p,n_p-1})$.

ALGORITHM REPRESENTATION-TO-STANDARD. (Base representation to standard coefficients.)

Input: A base $B = (b_1, \cdots, b_p)$ with a precision $N = (n_1, \cdots, n_p)$ and total precision n, a representative $r \in R(B, N)$ and $d \in \mathbf{N}$ with $d < n$. (This number d serves as a bound on the degree of the denominator polynomial.) Again all input polynomials are given by their coefficients.

Output: The coefficients of two polynomials $g, h \in F[x]$ such that r is a weak (B, N)-representation of (g, h), $\deg h \leqq d$, $\deg g < n - d$, h is monic and $\gcd(b_1^{n_1} \cdots b_p^{n_p}, h) = \gcd(g, h)$.

1. For all i, $1 \leqq i \leqq p$, compute $r_i = \sum_{0 \leqq j < n_i} r_{ij} b_i^j$.
2. For all i, $1 \leqq i \leqq p$, compute $v_i, e_i \in F[x]$ such that

$$v_i b_1^{n_1} \cdots b_{i-1}^{n_{i-1}} b_{i+1}^{n_{i+1}} \cdots b_p^{n_p} \equiv 1 \bmod b_i^{n_i},$$

$$\deg v_i < n_i \deg b_i,$$

$$e_i = v_i b_1^{n_1} \cdots b_{i-1}^{n_{i-1}} b_{i+1}^{n_{i+1}} \cdots b_p^{n_p}.$$

3. Compute $\omega \in F[x]$ such that

$$w \equiv \sum_{1 \leqq i \leqq p} r_i e_i \bmod b_1^{n_1} \cdots b_p^{n_p},$$

   $\deg w < \sum n_i \deg b_i = n$.
   (Then for all i, $w \equiv r_i \bmod b_i^{n_i}$.)
4. Compute the length l and the entries a_k, s_k, t_k, where $2 \leqq k \leqq l+1$, of the Extended Euclidean Scheme of $(a_0, a_1) = (b_1^{n_1} \cdots b_p^{n_p}, w)$.
5. Determine k, $1 \leqq k \leqq l+1$, such that $\deg a_k < n - d \leqq \deg a_{k-1}$, the leading coefficient λ of t_k, and return $g = \lambda^{-1} a_k$, $h = \lambda^{-1} t_k$.

THEOREM 3.1. *The algorithms STANDARD-TO-REPRESENTATION and REPRESENTATION-TO-STANDARD work correctly as described. They can be performed on arithmetic networks of depth* $O(\log^2 n)$ *and size* $n^{O(1)}$ *for inputs that have total precision at most* n *and (for STANDARD-TO-REPRESENTATION)* $\deg g + \deg h < n$.

Proof. We first consider algorithm *STANDARD-TO-REPRESENTATION*. The depth and size bounds follow from the results in von zur Gathen [1984b].

To prove correctness, we note that for $1 \leqq i \leqq p$

$$\sum_{0 \leqq j < n_i} r_{ij} b_i^j = \sum_{0 \leqq j < n_i} (u_{ij} - u_{i,j+1} b_i) b_i^j$$

$$= \sum_{0 \leqq j < n_i} (u_{ij} b_i^j - u_{i,j+1} b_i^{j+1}) = u_{i0} - u_{in_i} b_i^{n_i} = r_i,$$

and the polynomial

$$r_{ij} b_i^j = u_{ij} b_i^j - u_{i,j+1} b_i^{j+1} = -v_{ij} + v_{i,j+1}$$

has degree $< (j+1) \deg b_i$, and hence $\deg r_{ij} < \deg b_i$. This shows that $r \in R(B, N)$, and

$$g - hr_i \equiv g - hgt_i \equiv g(1 - ht_i) \equiv 0 \bmod b_i^{n_i}$$

implies that r is a weak (B, N)-representation of (g, h). Thus algorithm *STANDARD-TO-REPRESENTATION* works correctly.

To prove correctness of algorithm *REPRESENTATION-TO-STANDARD*, we first note that v_i as required in step 2 exists, since $b_1, \cdots, b_p$ are pairwise relatively prime. For $1 \leqq i, j \leqq p$ we have

$$e_i \equiv \delta_{ij} \bmod b_j^{n_j}$$

(so that the e_i's are "mutually orthogonal idempotents" in $F[x]/(b_1^{n_1} \cdots b_p^{n_p})$), and $w \equiv r_i \bmod b_i^{n_i}$. (This is the "Lagrange version" of the Chinese remainder algorithm.)

Clearly $s_k b_1^{n_1} \cdots b_p^{n_p} + t_k w = a_k$ implies that

$$g \equiv hw \equiv h \sum_{0 \leqq j < n_i} r_{ij} b_i^j \bmod b_i^{n_i}$$

for all i. Also

$$\gcd(g, h) = \gcd(b_1^{n_1} \cdots b_p^{n_p}, h),$$

since $\gcd(s_k, t_k) = 1$. (For later use, it is convenient to make t_k monic.) Furthermore, $\deg a_{k-1} + \deg t_k = n$ and $n - d \leqq \deg a_{k-1}$ imply the required degree bound $\deg h \leqq d$. The depth estimate $O(\log^2 n)$ follows from von zur Gathen [1984b]. □

Remark 3.2. Division with remainder of two polynomials in $F[x]$ of degree at most n can be performed in depth $O(\log n)$ and size $n^{O(1)}$, and also the iterated product $p = p_1 \cdots p_n$ of polynomials with $\deg p_i \leqq n$. This was proved by Reif [1983] in the case that F supports the Fast Fourier Transform, and by Eberly [1984] in general (for P-uniform networks). Therefore all steps of the two conversion algorithms can be performed in depth $O(\log n)$, except possibly step 2 of *STANDARD-TO-REPRESENTATION*, and steps 2 and 4 of *REPRESENTATION-TO-STANDARD*. In particular, both conversion problems (i.e. computing the output from the input, as stated) are "log-depth reducible" to EES, the problem of computing all entries of the Extended Euclidean Scheme of two univariate polynomials. (The reduction is P-uniform, rather than the usually required log-space uniform.)

On the other hand, also EES is log-depth reducible to the problem of computing the conversion from representation to standard coefficients: given $a_0, a_1 \in F[x]$ with

$0 \leqq \deg a_1 < \deg a_0 \leqq n$, we can compute for all d, $0 \leqq d < n$, polynomials u_d, b_d such that

$$u_d a_1 \equiv b_d \bmod a_0,$$

$$\deg b_d < n - d, \qquad \deg u_d \leqq d, \quad u_d \text{ monic},$$

by the conversion from the $((a_0), (1))$-representation $r = (a_1)$ of a_1 to standard representation with degree bound d. Then we can compute $v_d \in F[x]$ such that

$$v_d a_0 + u_d a_1 = b_d.$$

It follows from von zur Gathen [1984b], Lemma 2.2, that the nonzero b_d of minimal degree is a scalar multiple of a_l, and together with those b_d for which $b_d \prec b_{d-1}$ we get—up to scalar multiples—the remainders of the Extended Euclidean Scheme of (a_0, a_1), and v_d, u_d are the "convergents". It is easy then to also compute the quotients required for EES, and the correct scalar multipliers in depth $O(\log n)$ (see von zur Gathen [1984b]).

Thus the two problems EES and conversion from representation to standard coefficients are log-depth equivalent. In particular, the continued fraction decomposition $(q_1, \cdots, q_l)$ of $a_0/a_1 \in F(x)$ can be calculated via this conversion problem.

Remark 3.3. In this paper, we only consider algebraic complexity, using the model of arithmetic networks over an arbitrary field F. When F is $\mathbf{Q}$ or a finite field, it also makes sense to ask for Boolean circuits implementing the algorithms, with inputs now represented in some standard fashion over the alphabet $\{0, 1\}$, say. It follows from the results in Borodin, Cook, Pippenger [1983] (see also Eberly [1984]) that both conversion algorithms can be performed on log-space uniform families of Boolean circuits of depth $O(\log^2 n)$ and size $n^{O(1)}$. Note, however, that at the present time no $(\log n)^{O(1)}$ depth method is known to compute modular inverses (or even the gcd) of n-bit integers. When F is $\mathbf{Q}$ or some $\mathbf{Z}_p$, then the above Boolean circuits use a "redundant notation" u/v for field elements, with $u, v \in \mathbf{Z}$, $v > 0$, but not necessarily $\gcd(u, v) = 1$. If $F = \mathbf{Z}_p$, then we can also enforce $0 \leqq u, v < p$, but we do not know how to replace (u, v) by the unique w such that $w \equiv (u/v) \bmod p$ and $0 \leqq w < p$.

4. Representability. In this section, we discuss the questions of Representability and Convertibility as mentioned in § 2. We first provide a (large) subset $S(B) \subseteq F(x)$ such that every $f \in S(B)$ has a unique (B, N)-representation, and then a subset $T_{B,n,d} \subseteq F(x)$ such that on $T_{B,n,d}$, algorithms *STANDARD-TO-REPRESENTATION* and *REPRESENTATION-TO-STANDARD* compute functions that are inverse to each other.

Fix a base B and a precision N. If $f = g/h$ with $g, h \in F[x]$, $\gcd(g, h) = 1$ and $\gcd(b_1, h) \neq 1$, then f has of course no representation in base B. Namely, if r were a representation, then

$$\gcd(b_1, h) | g - hr_1,$$

and hence

$$\gcd(b_1, h) | g,$$

contradicting the assumption.

The example $(bg)/(bh)$ of § 2 showed that the congruence condition in Definition 2.1 is too weak for unique representation. To rule out this example, we introduced the gcd condition, which corresponds to

$$S(B) = \{f \in F(x): \exists g, h \in F[x] \text{ such that } f = g/h \text{ and } \gcd(b_1 \cdots b_p, h) = 1\}.$$

(This semilocal ring $S(B)$ is the intersection in $F(x)$ of all localizations $F[x]_{(q)}$, with

q running through the irreducible factors of $b_1 \cdots b_p$.) We now show that Representability holds for the elements of $S(B)$.

THEOREM 4.1. *Let B be a base, N a precision for B, $g, h \in F[x]$ with* $\gcd(g, h) = 1$, *and $f = g/h \in F(x)$. Then the following are equivalent:*

(i) *(g, h) has a unique weak (B, N)-representation,*

(ii) *f has a unique (B, N)-representation,*

(iii) $\gcd(b_1 \cdots b_p, h) = 1$,

(iv) $f \in S(B)$.

Proof. We will prove "(i)⇒(iv)⇒(iii)⇒(i)" and "(ii)⇔(iii)".

For "(i)⇒(iv)", assume $f \notin S(B)$, and that (g, h) has a weak (B, N)-representation $r = (r_{10}, \cdots, r_{p,n_p-1}) \in R(B, N)$. Then $\gcd(b_1 \cdots b_p, h) \neq 1$, and we take i and an irreducible polynomial $q \in F[x]$ such that $q | \gcd(b_i, h)$. Now change r to $\bar{r} \in R(B, N)$ by replacing r_{i,n_i-1} with $\bar{r}_{i,n_i-1} = r_{i,n_i-1} + b_i/q$. Then

$$h r_{i,n_i-1} b_i^{n_i-1} \equiv h \bar{r}_{i,n_i-1} b_i^{n_i-1} \bmod b_i^{n_i},$$

and $\bar{r} \neq r$ is also a weak (B, N)-representation of (g, h).

For "(iv)⇒(iii)", let $f \in S(B)$, and write $f = \bar{g}/\bar{h}$ with $\bar{g}, \bar{h} \in F[x]$ and $\gcd(b_1 \cdots b_p, \bar{h}) = 1$. Now assume that $\gcd(b_1 \cdots b_p, h) \neq 1$, say that $q | \gcd(b_1 \cdots b_p, h)$ for some irreducible $q \in F[x]$. Then from $\bar{g}h = g\bar{h}$ we get $q | g\bar{h}$ and hence $q | g$, a contradiction.

For "(iii)⇒(i)", we have seen in Theorem 3.1 that algorithm *STANDARD-TO-REPRESENTATION* with input (B, N, g, h) computes a (B, N)-representation of f, which is of course also a weak (B, N)-representation of (g, h). So now suppose that (g, h) has two weak (B, N)-representations $r, \bar{r} \in R(B, N)$, and let $r_i = \sum_{0 \leq j < n_i} r_{ij} b_i^j$, and similarly $\bar{r}_i$. Then for all i, $1 \leq i \leq p$, we have

$$b_i^{n_i} | h(r_i - \bar{r}_i), \qquad \gcd(b_i, h) = 1,$$

$$\deg(r_i - \bar{r}_i) < n_i \deg b_i,$$

which implies that

$$b_i^{n_i} | r_i - \bar{r}_i,$$

and hence $r_i = \bar{r}_i$. It follows that $r = \bar{r}$, and (i) is proven.

For "(iii)⇒(ii)", we first note that, assuming (iii), any $r \in R(B, N)$ is a weak (B, N)-representation of (g, h) if and only if it is a (B, N)-representation of f. Since (iii) implies (i), (ii) also follows.

For "(ii)⇒(iii)", we assume that f has a (unique) (B, N)-representation, and therefore $f = \bar{g}/\bar{h}$ with $\bar{g}, \bar{h} \in F[x]$ and $\gcd(b_1 \cdots b_p, \bar{h}) = 1$. Then $\bar{g}h = g\bar{h}$, and if $q | \gcd(b_i, h)$ for some i and $q \in F[x]$ irreducible, then $q | g$. (iii) is proven. □

Example 4.2. For the second property of algorithms *REPRESENTATION-TO-STANDARD* and *STANDARD-TO-REPRESENTATION* computing inverse functions, consider the example $p = 1$, $B = (x^3 - x)$, $N = (1)$, $d = 1$ and $r = (x^2 + 1)$ (and assume char $F \neq 2$). The output of algorithm *REPRESENTATION-TO-STANDARD* is $(-2x, -x)$. If we now simplify $f = -2x/-x$ to $f = 2$ and apply algorithm *STANDARD-TO-REPRESENTATION* with input B, N, $(2, 1)$, the output is $(2) \neq r$. This example makes it clear that the second property will not hold for all $r \in R(B, N)$. But we will see that it holds for "almost all" r.

Consider the following set $T_{B,n,d}$:

$$T_{B,n,d} = \{(g, h) \in F[x]^2 : \gcd(g, h) = \gcd(b_1 \cdots b_p, h) = 1,$$

$$h \text{ is monic, } \deg g < n - d, \deg h \leqq d\}.$$

Ignoring the data B, N, d, which remain unchanged throughout the algorithms (d does not even appear in *STANDARD-TO-REPRESENTATION*), we now write $\alpha(g, h) = r$ and $\beta(r) = (g, h)$ for the functions computed by the two algorithms *STANDARD-TO-REPRESENTATION* and *REPRESENTATION-TO-STANDARD*. For $(g, h) \in T_{B,n,d}$, we have $f = g/h \in S(B)$, and $r = \alpha(g, h)$ is a (B, N)-representation of f.

THEOREM 4.3. *Let B be a base, N a precision for B with total precision n, and $d < n$. Then the functions α and β are inverse bijections between the set $T_{B,n,d} \subseteq F(x)$ and its image in $R(B, N)$.*

Proof. Let $U = \alpha(T_{B,n,d})$ be the image of $T_{B,n,d}$ in $R(B, N)$. It is sufficient to show that $\beta \circ \alpha(\bar{g}, \bar{h}) = (\bar{g}, \bar{h})$ for any $(\bar{g}, \bar{h}) \in T_{B,n,d}$, since then $\alpha: T_{B,n,d} \to U$ is injective, and surjective by definition. Furthermore, β is a section of α, and hence its inverse.

So let $(g, h) = \beta \circ \alpha(\bar{g}, \bar{h}) = (\lambda^{-1} a_k, \lambda^{-1} t_k)$, using the notation of the algorithms, and write $b = b_1^{n_1} \cdots b_p^{n_p}$. We can assume $\bar{g} \neq 0$, and then $a_k \neq 0$. From

$$\bar{g} \equiv \bar{h} r_i \mod b_i^{n_i}$$

for all i, it follows that

$$\bar{g} \equiv \bar{h} w \mod b,$$

and there exists $s \in F[x]$ such that

$$sb + \bar{h}w = \bar{g},$$

$$\deg \bar{g} + \deg \bar{h} < n = \deg b.$$

By the uniqueness of the Extended Euclidean Scheme (see e.g. von zur Gathen [1984b], Lemma 2.2), there exists $m \in \{1, \cdots, l\}$ and $u \in F[x]$ such that

$$\bar{g} = u a_m, \qquad \bar{h} = u t_m,$$

$$\deg a_m \leqq \deg \bar{g} < \deg a_{m-1}.$$

Since $\deg a_m \leqq \deg \bar{g} < n - d \leqq \deg a_{k-1}$, we have $k \leqq m$. But on the other hand $n - \deg a_{m-1} = \deg t_m \leqq d$ implies that $n - d \leqq \deg a_{m-1}$, and therefore $m \leqq k$. It follows that $m = k$, and

$$\bar{g} = u\lambda^{-1} g, \qquad \bar{h} = u\lambda^{-1} h.$$

Since $\gcd(\bar{g}, \bar{h}) = 1$, $u \in F$. Both h and $\bar{h}$ are monic, therefore $u\lambda^{-1} = 1$. □

5. Applications. We can now reap the fruits of the work spent in setting up the previous notation. By putting together algorithms *REPRESENTATION-TO-STANDARD* and *STANDARD-TO-REPRESENTATION* (using different bases) we obtain a fast parallel algorithm for conversion from one base to another. This yields fast parallel algorithms for a number of important problems in symbolic manipulation (and also clarifies how these problems are related to each other).

INTERPOLATION (n) has as input a pair (a, r) where $a = (a_1, \cdots, a_n)$ and $r = (r_1, \cdots, r_n)$ have entries from F, and $a_i \neq a_j$ for $1 \leqq i < j \leqq n$. The output are the coefficients of the unique $f \in F[x]$ such that $\deg f < n$ and $f(a_i) = r_i$ for $1 \leqq i \leqq n$. This function is nothing but the conversion from $((x - a_1, \cdots, x - a_n), (1, \cdots, 1))$-representation to $((x), (n))$-representation. (In order to obtain the unique monic $g \in F[x]$ of degree n such that $g(a_i) = r_i$ for all i, one uses the interpolation polynomial f for the values $r_i - a_i^n$, and $g = x^n + f$.)

TAYLOR EXPANSION (n) has as input $a \in F$ and the coefficients $(g_0, \cdots, g_{n-d-1}; h_0, \cdots, h_d)$ of

$$f = \frac{\sum_{0 \leqq j < n-d} g_j x^j}{\sum_{0 \leqq j \leqq d} h_j x^j} \in F(x)$$

where $\sum h_j a^j \neq 0$. The output are the Taylor coefficients $f_0, \cdots, f_{n-1} \in F$ of f at a, so that $f \equiv \sum_{0 \leqq j < n} f_j (x-a)^j \bmod (x-a)^n$. This is the conversion from coefficients to $((x-a), (n))$-representation.

HERMITE INTERPOLATION (n) has an input of the form $(a, r_1, \cdots, r_p)$, where $a = (a_1, \cdots, a_p)$, $r_i = (r_{i0}, \cdots, r_{i,n_i-1})$, all $a_i, r_{ij} \in F$, and $a_i \neq a_j$ for $1 \leqq i < j \leqq p$, and $n_1 + \cdots + n_p = n$. The output are the coefficients of the unique polynomial $f \in F[x]$ such that $\deg f < n$ and

$$f \equiv \sum_{0 \leqq j < n_i} r_{ij} (x-a_i)^j \bmod (x-a_i)^{n_i}$$

for $1 \leqq i \leqq p$.

Thus the first n_i coefficients of the Taylor expansion of f at a_i are prescribed. If $\operatorname{char} F = 0$, this is equivalent to prescribing the values of the first n_i derivatives of f at a_i as

$$\left(\frac{d^j f}{dx^j}\right)(a_i) = j! r_{ij}.$$

HERMITE INTERPOLATION (n) is the conversion from $((x-a_1, \cdots, x-a_p), (n_1, \cdots, n_p))$-representation to $((x), (n))$-representation. INTERPOLATION is a special case of this problem. Again, one can also compute the unique monic interpolation polynomial of degree n.

CHINESE REMAINDER ALGORITHM (n) has as input a sequence $(b_1, \cdots, b_p, r_1, \cdots, r_p)$ of polynomials from $F[x]$ such that $\deg (b_1 \cdots b_p) = n$, $\deg r_i < \deg b_i$, the b_i's are nonconstant and pairwise relatively prime. The output are the coefficients of the unique $f \in F[x]$ such that $f \equiv r_i \bmod b_i$ for $1 \leqq i \leqq p$ and $\deg f < n$. This is the conversion from $((b_1, \cdots, b_p), (1, \cdots, 1))$-representation to $((x), (n))$-representation.

ELEMENTARY SYMMETRIC FUNCTIONS (n) has as input a sequence $(c_1, \cdots, c_n)$ with $c_i \in F$. Output are the elementary symmetric functions $s_j = \sigma_j(c_1, \cdots, c_n)$ for $1 \leqq j \leqq n$. Thus

$$t^n - s_1 t^{n-1} + \cdots + (-1)^n s_n = (t-c_1) \cdots (t-c_n),$$

where t is an indeterminate. This can be viewed as a special case of the monic version of HERMITE INTERPOLATION: Set $C = \{c_1, \cdots, c_n\}$, $p = \# C$, $\{a_1, \cdots, a_p\} = C$, and $r_{ij} = 0$ for $0 \leqq j < n_i$, where a_i occurs exactly n_i times among $c_1, \cdots, c_n$. The inverse function—root-finding—cannot be computed by a rational algorithm.

PARTIAL FRACTION DECOMPOSITION (n) has as input the coefficients of polynomials $b_1, \cdots, b_p$ and q, and $n_1, \cdots, n_p \in \mathbf{N}$ such that $b_1, \cdots, b_p$ are nonconstant and pairwise relatively prime, and $\deg q < \deg (b_1^{n_1} \cdots b_p^{n_p}) = n$. Output are the coefficients of the unique polynomials s_{ij} $(1 \leqq i \leqq p, 1 \leqq j < n_i)$ such that

$$\frac{q}{b_1^{n_1} \cdots b_p^{n_p}} = \sum_{\substack{1 \leqq i \leqq p \\ 1 \leqq j \leqq n_i}} \frac{s_{ij}}{b_i^j},$$

$$\deg s_{ij} < \deg b_i.$$

(For a slightly more general case, one would not assume $\deg q > n$; then one first has to perform a division with remainder to get a polynomial summand plus a problem of the above format.) We note that for all i

$$q \equiv \sum_{\substack{1 \leqq i \leqq p \\ 1 \leqq j \leqq n_i}} b_1^{n_1} \cdots b_{i-1}^{n_{i-1}} b_{i+1}^{n_{i+1}} \cdots b_p^{n_p} \; s_{ij} b_i^{n_i - j} \bmod b_i^{n_i}.$$

We first convert the $((x),(n))$-representation of q to the $((b_1, \cdots, b_p), (n_1, \cdots, n_p))$-representation r, then for all i, $1 \leqq i \leqq p$, take r_i and v_i as in steps 1 and 2 of *REPRESENTATION-TO-STANDARD*, and compute the $((b_i),(n_i))$-representation $(s_{i0}^*, \cdots, s_{i,n_i-1}^*)$ of $v_i r_i$. Then $s_{ij} = s_{i,n_i-j}^*$ for $1 \leqq j \leqq n_i$.

PADÉ APPROXIMATION (n) has as input a polynomial $f \in F[x]$ of degree $< n$, and $d \in \mathbf{N}$ with $0 \leqq d < n$. The output consists of the coefficients of polynomials $g, h \in F[x]$ such that $g \equiv fh \bmod x^n$, $\deg g < n - d$, $\deg h \leqq d$, and h is monic. Thus $g/h \equiv f \bmod x^n$ is a Padé approximant to f (provided $h(0) \neq 0$). This is the conversion from the $((x),(n))$-representation $(f_0, \cdots, f_{n-1})$ of $f = \sum f_i x^i$ to standard representation. (For sequential Padé computations, see Gragg [1972], Geddes [1979], Brent, Gustavson, Yun [1980], and for an overview of the theory Baker, Graves-Morris [1981].)

CAUCHY INTERPOLATION (n) has as input $d \in \mathbf{N}$ with $0 \leqq d < n$ and a pair (a, r) where $a = (a_1, \cdots, a_n)$, $r = (r_1, \cdots, r_n) \in F^n$, and $a_i \neq a_j$ for $1 \leqq i < j \leqq n$. The output consists of the coefficients of polynomials $g, h \in F[x]$ such that $g(a_i) = h(a_i) r_i$ for $1 \leqq i \leqq n$, $\deg g < n - d$, $\deg h \leqq d$, and h is monic. Thus $f = g/h$ is a rational function with prescribed denominator degree that interpolates the given values r_i at a_i, i.e. $f(a_i) = r_i$ (provided $h(a_i) \neq 0$). Cauchy [1821] had considered this problem and given an explicit solution by a closed formula similar to the Lagrange interpolation formula. For a modern treatment, see Gustavson, Yun [1979] and Knuth [1981, Exercise 4.7–13]. Algorithm *REPRESENTATION-TO-STANDARD* with base $B = (x - a_1, \cdots, x - a_n)$, precision $N = (1, \cdots, 1)$, and degree bound d computes a solution.

RATIONAL HERMITE INTERPOLATION (n) has an input of the form $(d, a, r_1, \cdots, r_p)$ where $0 \leqq d < n$, $a = (a_1, \cdots, a_p)$, $r_i = (r_{i0}, \cdots, r_{i,n_i-1})$, all $a_i, r_{ij} \in F$, $a_i \neq a_j$ for $1 \leqq i < j \leqq p$, and $n_1 + \cdots + n_p = n$. The output are the coefficients of polynomials $g, h \in F[x]$ such that

$$g \equiv h \cdot \sum_{0 \leqq j < n_i} r_{ij} (x - a_i)^j \bmod (x - a_i)^{n_i},$$

$\deg g < n - d$, $\deg h \leqq d$, and h is monic. Thus the initial segments of the Taylor expansion of the rational function $f = g/h$ at a_i are prescribed, i.e.

$$f \equiv \sum_{0 \leqq j < n_i} r_{ij} (x - a_i)^j \bmod (x - a_i)^{n_i}$$

(provided $h(a_i) \neq 0$). This problem simultaneously generalizes HERMITE INTERPOLATION (which has $d = 0$), PADÉ APPROXIMATION (which has $p = 1$ and $a_1 = 0$), and CAUCHY INTERPOLATION (which has $n_1 = \cdots = n_p = 1$). It can be computed by algorithm *REPRESENTATION-TO-STANDARD* with input $B = (x - a_1, \cdots, x - a_p)$, $N = (n_1, \cdots, n_p)$ and $r = (r_1; \cdots; r_p)$.

The following theorem tells us that all the above problems have a fast parallel solution.

THEOREM 5.1. *The following nine functions can be computed in depth $O(\log^2 n)$ and size $n^{O(1)}$: INTERPOLATION, TAYLOR EXPANSION, HERMITE INTERPOLATION, CHINESE REMAINDER ALGORITHM, ELEMENTARY SYMMETRIC FUNCTIONS, PARTIAL FRACTION DECOMPOSITION, PADÉ APPROXIMATION, CAUCHY INTERPOLATION, RATIONAL HERMITE INTERPOLATION.*

Proof. Use Theorem 3.1 and the fact that all these problems can be solved by algorithms *STANDARD-TO-REPRESENTATION* and *REPRESENTATION-TO-STANDARD.* □

Remark 5.2. Not unexpectedly, the above general result is not the best possible, at least in some cases dealing with polynomial problems. For a polynomial f, TAYLOR EXPANSION can be computed by calculating binomial coefficients and evaluating universal Taylor coefficients of f (see von zur Gathen [1984b, § 6]). This can be performed in depth $O(\log n)$, so that the corresponding statement of Theorem 5.1 is interesting only in the case of rational functions. Reif [1983] provides less obvious methods for interpolation and elementary symmetric functions that use only depth $O(\log n)$ and polynomial size. These methods assume that certain roots of unity are available in the ground field; they have been extended by Eberly [1984] to hold over arbitrary fields.

Remark 5.3. Clearly all our computational problems can be expressed by systems of linear equations, and these are always solvable in depth $O(\log^2 n)$ (Csanky [1976], Borodin, von zur Gathen, Hopcroft [1982], Berkowitz [1984]). However, for singular systems over arbitrary fields only a probabilistic algorithm is known. The above results amount to saying that in the cases considered here it is sufficient to solve nonsingular systems of linear equations; again this is well-known in some cases, such as interpolation by polynomials (Vandermonde matrix).

Remark 5.4. It is well-known that in some cases the rational interpolation problem does not have a solution. Theorem 4.3 shows that most $r \in R(B, N)$ are representations of a rational function $f = g/h$ with $\deg g < n - d$ and $\deg h \leqq d$ (g and h are polynomials). However, in Example 4.2 we have seen that not all $r \in R(B, N)$ are such representations. For a general discussion of this phenomenon, let $B = (b_1, \cdots, b_p)$ be a base, N a precision for B, n the total precision, $0 \leqq d < n$, and $r \in R(B, N)$. Set $a_0 = b_1^{n_1} \cdots b_p^{n_p}$, and let $a_1 \in F[x]$ be a polynomial such that $\deg a_1 < n$ and r is a (B, N)-representation of a_1. By the Chinese Remainder Theorem (or by Theorem 4.1), a_1 exists and is unique. Furthermore, let $a_k, s_k, t_k \in F[x]$ be the (unique) entries of the Extended Euclidean Scheme for (a_0, a_1) with $\deg a_k < n - d \leqq \deg a_{k-1}$. We then have

$$s_k a_0 + t_k a_1 = a_k,$$

$$t_k a_1 \equiv a_k \mod a_0.$$

Now assume that $\gcd(a_0, t_k) = 1$. Then r is indeed the (B, N)-representation of $f = a_k/t_k$, and $\gcd(a_k, t_k) = 1$. The latter property follows using the fact that $\gcd(s_k, t_k) = 1$. Thus "$\gcd(a_0, t_k) = 1$" is a sufficient condition which guarantees that r is the representation of a rational function. Note that it is always satisfied if $d = 0$, and then $k = 1$, $a_k = a_1$, $t_k = 1$.

This sufficient condition holds "almost everywhere" in the following sense. For fixed numbers $d, p, n_1, \cdots, n_p, m_1, \cdots, m_p$ the set of inputs $(b_1, \cdots, b_p, r)$ for algorithm *REPRESENTATION-TO-STANDARD* with $\deg b_i = m_i$ can be considered as a subset of F^m, where

$$m = p + \sum_{1 \leqq i \leqq p} (n_i + 1) m_i.$$

It is a Zariski-open dense subset, the only condition being that $\deg b_i = m_i$ and $\gcd(b_i, b_j) = 1$ for $1 \leqq i < j \leqq p$; the latter condition can be expressed by the nonvanishing of a resultant polynomial. Now there exists a nonzero polynomial P in m variables such that

$$P(b_1, \cdots, b_p, r) \neq 0 \Rightarrow \gcd(a_0, t_k) = 1,$$

using the above notation. Thus the sufficient condition holds "almost everywhere" in the strong sense of algebraic geometry, namely on a Zariski-open dense subset of the input set (assuming that F is infinite). (P can be chosen so that $P(b_1, \cdots, b_p, r) \neq 0$ will imply that the Euclidean Scheme for (a_0, a_1) is normal, i.e. that deg $q_k = 1$ for all k. But in fact, a polynomial P as above exists for every subset of the input space corresponding to some $D(d_1, \cdots, d_{l+1})$, using the notation from Strassen [1983, (5.3)], and such that P does not vanish identically on the subset.)

Example 5.5. Let us now look at a "bad case" for Padé approximation. Let $n \geq 5$ and $r = x^{n-1} - x^{n-2} + 1$. Then $g = h = x^2$ is the only solution of the conditions

$$g, h \in F[x], \qquad g \equiv hr \mod x^n,$$
$$\deg g < n-2, \qquad \deg h \leqq 2, \quad h \text{ monic.}$$

In particular, there does not exist a Padé approximant $f = g/h \in F(x)$ satisfying the above conditions and gcd $(g, h) = 1$. Algorithm *REPRESENTATION-TO-STANDARD* with input $((x), (n))$, $d = 2$, r will output $(g, h) = (x^2, x^2)$. Similarly, Example 4.2 shows that there is no $f = g/h \in F(x)$ such that $f(1) = 2$, $f(0) = 1$, $f(-1) = 2$, and $g, h \in F[x]$, deg g, deg $h \leqq 1$ and gcd $(g, h) = 1$. Algorithm *REPRESENTATION-TO-STANDARD* will compute $(g, h) = (-2x, -x)$ which satisfies the conditions

$$g(1) = 2h(1), \quad g(0) = h(0), \quad g(-1) = 2h(-1).$$

This is a "fake solution" which does not yield an interpolating rational function, while "almost all" other such interpolation problems do have a solution. This phenomenon of nonexistence of solutions to the Padé approximation problem and for rational interpolation was discovered by Kronecker, who illustrated it with an example. In Padé's work [1892], this fundamental limitation does not appear. For further examples, see e.g. Graves-Morris [1980].

Historical remark 5.6. The general idea in this paper is to reduce an algebraic problem to systems of linear equations; for the latter we have fast parallel algorithms. This idea is in fact quite old. Jacobi [1846] reduces the rational interpolation problem (for the general Hermite format) to systems of linear equations, and also explicitly describes the solutions in terms of determinants of the inputs; he has several such descriptions. In particular, Jacobi seems to have been the first to assert the existence of what is now called Padé approximations. Jacobi also realized the close connection between the interpolation problem and general banded systems of linear equations in the Hankel format.

Cauchy [1821] had previously stated an explicit solution to the rational interpolation problem by a closed formula.

Kronecker [1881] has two methods for solving the general interpolation problem, one via continued fractions (i.e. essentially the Euclidean Scheme) and one via systems of linear equations. Both Jacobi and Kronecker are motivated by the elimination problem in algebraic geometry.

In hindsight, it is almost surprising how long it took to rediscover these methods and cast them into modern algorithms for algebraic manipulation: the subresultant algorithms of Collins [1967] and Brown [1971] for gcd's and the Padé computations of Geddes [1979] and Brent, Gustavson, Yun [1980]. The essential ingredients of these algorithms (and of those presented here) are already contained in the classical papers.

6. Laurent representations. The representations defined in § 2 were sufficient to describe a unified fast parallel solution to the problems in § 5. Limitations of these representations have been pointed out: not all rational functions have representations,

and representations may not be unique. In this section, we generalize the notion of representation slightly in order to deal more successfully with these issues.

Two generalizations suggest themselves: one can either allow quotients of polynomials in a representation (which would include the standard representation of a rational function as a quotient of two polynomials), or one can allow representations in a "Laurent format". We pursue the latter approach.

DEFINITION 6.1. Let $B=(b_1, \cdots, b_p) \in F[x]^p$ be a base, consisting of nonconstant pairwise relatively prime polynomials, and $N=(n_1, \cdots, n_p) \in \mathbf{N}^p$ a precision for B, with total precision $n=\sum_{1\leq i\leq p} n_i \deg b_i$. A *Laurent* (B, N)*-representative* is a vector

$$r=(m_1, \cdots, m_p; r_{10}, \cdots, r_{1,n_1-1}; \cdots; r_{p0}, \cdots, r_{p,n_p-1}),$$

where

$$m_i \in \mathbf{Z}, \quad r_{ij} \in F[x], \quad \deg r_{ij} < \deg b_i$$

for all i, j $(1 \leq i \leq p, 0 \leq j < n_i)$. We write $r_i = \sum_{0\leq j<n_i} r_{ij} b_i^j$. Now let $g, h \in F[x]$ with $h \neq 0$. We say that r is a *weak Laurent* (B, N)*-representation of* (g, h) if for all i, $b_i^{m_i}$ divides g (if $m_i \geq 0$), and

$$g b_i^{-m_i} \equiv h r_i \mod b_i^{n_i}.$$

We now present two fast parallel algorithms that perform the conversion between the standard and a Laurent representation of a rational function. Later we will strengthen the conditions on Laurent representations, and obtain a notion with the two crucial properties of Representability and Convertibility. However, for the description of the algorithm it is convenient to work with the weaker notion first.

ALGORITHM STANDARD-TO-LAURENT. (Standard to Laurent representation.)

Input: A base $B=(b_1, \cdots, b_p)$, a precision $N=(n_1, \cdots, n_p)$ for B, and two polynomials $g, h \in F[x]$. All input polynomials are given by their coefficient vectors.

Output: A weak Laurent (B, N)-representation r of (g, h).

1. If $g=0$, return $r=(0, \cdots, 0; 0, \cdots, 0)$ and terminate. Else compute $a=\gcd(g, h)$, the leading coefficient λ of h, and replace (g, h) by $(g/\lambda a, h/\lambda a)$. (By convention, the gcd is monic.)
2. For all i, $1 \leq i \leq p$, do steps 3, 4, and 5.
3. Compute $m_i \in \mathbf{Z}$ as follows. If $\gcd(b_i, h)=1$, then

   $$m_i = \max\{j \in \mathbf{N}: b_i^j | g\}.$$

   Else

   $$\gcd(h, b_i^{-m_i}) = \gcd(h, b_i^{-m_i+1}) \neq \gcd(h, b_i^{-m_i-1}).$$

   (Note that this condition determines $m_i < 0$ uniquely.)
4. Compute

   $$c_i = \gcd(b_i^{|m_i|}, h), \quad g_i = \frac{g b_i^{-m_i}}{c_i}, \quad h_i = \frac{h}{c_i} \in F[x].$$

   (Then $c_i = 1$ if $\gcd(b_i, h)=1$, and $\gcd(g_i, h_i) = \gcd(b_i, h_i) = 1$.)
5. Call algorithm *STANDARD-TO-REPRESENTATION* with input $((b_i), (n_i), (g_i, h_i))$ to return $(r_{i0}, \cdots, r_{i,n_i-1})$.
6. Return

   $$r=(m_1, \cdots, m_p; r_{10}, \cdots, r_{1,n_1-1}; \cdots; r_{p0}, \cdots, r_{p,n_p-1}).$$

ALGORITHM LAURENT-TO-STANDARD. (Laurent representation to standard.)
Input: A base $B=(b_1, \cdots, b_p)$ and a precision $N=(n_1, \cdots, n_p)$ for B, with total precision n, $d \in \mathbf{N}$, $d<n$, and a Laurent (B, N)-representative

$$r=(m_1, \cdots, m_p; r_{10}, \cdots, r_{1,n_1-1}; \cdots; r_{p0}, \cdots, r_{p,n_p-1}).$$

Output: Polynomials $g, h \in F[x]$ such that h is monic, and r is a weak Laurent (B, N)-representation of (g, h).

1. For all i, $1 \leqq i \leqq p$, set $e_i=\max\{0, -m_i\}$, and compute $t, u, s_i, t_i, u_i \in F[x]$ such that

$$t=b_1^{m_1+e_1} \cdots b_p^{m_p+e_p}, \qquad u=b_1^{e_1} \cdots b_p^{e_p},$$

$$t_i=\frac{t}{b_i^{m_i+e_i}}, \qquad u_i=\frac{u}{b_i^{e_i}},$$

$$s_i t_i \equiv u_i \sum_{0 \leqq j<n_i} r_{ij} b_i^j \bmod b_i^{n_i},$$

$$\deg s_i < n_i \deg b_i.$$

2. For all i, $1 \leqq i \leqq p$, compute the $((b_i), (n_i))$-representation $s_i'=(s_{i0}, \cdots, s_{i,n_i-1})$ of s_i as in step 4 of algorithm *STANDARD-TO-REPRESENTATION*. (Then $s_i=\sum_{0 \leqq j<n_i} s_{ij} b_i^j$ and $\deg s_{ij} < \deg b_i$.)
3. Call algorithm *REPRESENTATION-TO-STANDARD* with input $(B, N, (s_1', \cdots, s_p'), d)$ to compute $g', h' \in F[x]$ such that for all i

$$g' \equiv h' s_i \bmod b_i^{n_i},$$

$$\deg g' < n-d, \quad \deg h' \leqq d, \quad h' \text{ monic}.$$

4. Let μ be the leading coefficient of u, and return

$$g=\mu^{-1} g' t,$$

$$h=\mu^{-1} h' u.$$

THEOREM 6.2. *The algorithms STANDARD-TO-LAURENT and LAURENT-TO-STANDARD can be performed in depth $O(\log^2 n)$ and size $n^{O(1)}$ on inputs that have total precision at most n and (for STANDARD-TO-LAURENT) $\deg g + \deg h < n$. They work correctly as described in "Output". In fact, the following relations hold, using the notation of the algorithms, and $r_i=\sum_{0 \leqq j<n_i} r_{ij} b_i^j$:*

(i) In *STANDARD-TO-LAURENT*, if $\gcd(g, h)=1$, then for $1 \leqq i \leqq p$,

$$c_i=\gcd(b_i^{|m_i|}, h), \quad \gcd\left(\frac{h}{c_i}, b_i\right)=1, \quad g b_i^{-m_i} \in F[x],$$

$$g b_i^{-m_i} \equiv h r_i \bmod b_i^{n_i} c_i.$$

(ii) In *LAURENT-TO-STANDARD*, for $1 \leqq i \leqq p$, $g b_i^{-m_i} \in F[x]$, and

$$\deg g < n-d+\sum_{\substack{1 \leqq i \leqq p \\ 0<m_i}} m_i \deg b_i,$$

$$\deg h \leqq d+\sum_{\substack{1 \leqq i \leqq p \\ m_i<0}} |m_i| \deg b_i,$$

$$g b_i^{-m_i} \equiv h r_i \bmod b_i^{n_i+e_i}.$$

Proof. The bounds on the depth and size are clear. For the correctness proof for *STANDARD-TO-LAURENT*, we can assume $g \neq 0$ and $\gcd(g, h) = 1$. Obviously $b_i^{m_i} | g$ if $m_i > 0$. Suppose $q \in F[x]$ is irreducible and divides $\gcd(b_i, h)$, say

$$q^k | b_i,\ q^{k+1} \nmid b_i,\ q^l | h,\ q^{l+1} \nmid h,$$

with $k, l \in \mathbf{N}$ and $k, l > 0$. Then $(-m_i)k \geqq l$ and $q^l | b_i^{-m_i}$, hence $q^l | c_i$ and $q \nmid h/c_i$. Therefore $\gcd(b_i, h_i) = \gcd(b_i, h/c_i) = 1$, and

$$\frac{gb_i^{-m_i}}{c_i} = g_i \equiv h_i r_i = \frac{h}{c_i} r_i \mod b_i^{n_i},$$

using Theorem 3.1. Thus

$$gb_i^{-m_i} \equiv hr_i \bmod b_i^{n_i} c_i,$$

and r is a weak Laurent (B, N)-representation of (g, h).

For *LAURENT-TO-STANDARD*, clearly h is monic, and we have

$$\deg g = \deg g' + \sum_{1 \leqq i \leqq p} (m_i + e_i) \deg b_i < n - d + \sum_{\substack{1 \leqq i \leqq p \\ 0 < m_i}} m_i \deg b_i,$$

$$\deg h = \deg h' + \sum_{1 \leqq i \leqq p} e_i \deg b_i \leqq d + \sum_{\substack{1 \leqq i \leqq p \\ m_i < 0}} (|m_i| \deg b_i).$$

If $m_i > 0$, then $e_i = 0$, and therefore $b_i^{m_i} | t$ and $gb_i^{-m_i} \in F[x]$. With $r_i = \sum_{0 \leqq j < n_i} r_{ij} b_i^j$, we have in any case

$$\mu u_i g b_i^{-m_i} = u_i g' t b_i^{-m_i} = u_i g' t_i b_i^{e_i} \equiv u_i h' s_i t_i b_i^{e_i}$$
$$\equiv u_i h' u_i r_i b_i^{e_i} = u h' u_i r_i = \mu h u_i r_i \bmod b_i^{n_i + e_i}.$$

Since μu_i is a unit mod $b_i^{n_i + e_i}$, it follows that

$$gb_i^{-m_i} \equiv hr_i \bmod b_i^{n_i + e_i}. \qquad \square$$

It is clear that Definition 6.1 of "weak Laurent representation" really is too weak. Under this notion, one function may have many representations. If $B = (x)$, $N = (2)$, $g = 1$, $h = x$, so that $g/h = 1/x = x^{-1}(1 + 0 \cdot x)$, then we would like to have $(-1; 1, 0)$ as (B, N)-representation of (g, h), but in fact $(-1; 1, a)$ is a weak Laurent (B, N)-representation of (g, h), for every $a \in F$. Not even the m_i's are uniquely determined: any $r = (m; a, b)$ with $m \leqq -2$ and $a, b \in F$ is a weak Laurent $((x), (2))$-representation of $(1, x^3)$. Even the somewhat stronger congruence in Theorem 6.2(i) does not completely determine the exponent m_i. As an example, take $p = 1$, $B = (b_1)$, $N = (2)$, $g = b_1 g_1$, $h = 1$ with $\deg g_1 < \deg b_1$. Then $c_1 = 1$, and both $(0; 0, g_1)$ and $(1; g_1, 0)$ are weak Laurent (B, N)-representations of (g, h). Thus some more stringent conditions are necessary to ensure uniqueness of representations.

DEFINITION 6.3. Let B be a base, N a precision, r a Laurent (B, N)-representative as in Definition 6.1, and $g, h \in F[x]$ with $h \neq 0$. We say that r is the *Laurent* (B, N)-*representation of* $f = g/h$ if the following conditions are satisfied:

- (L_1) $\gcd(g, h) = 1$.
- (L_2) For all i, $1 \leqq i \leqq p$, set $c_i = \gcd(b_i^{|m_i|}, h)$. Then $gb_i^{-m_i} \in F[x]$, and $\gcd(b_i, h/c_i) = 1$.
- (L_3) $r_{i0} \neq 0$.
- (L_4) For all i, $1 \leqq i \leqq p$,

$$gb_i^{-m_i} \equiv hr_i \bmod b_i^{n_i} c_i.$$

Clearly this relation between f and r does not depend on the choice of g, h, since (L_1) determines them up to a scalar factor.

LEMMA 6.4. *Let B, N, r, g, h as in Definition 6.3 satisfy (L_1), (L_2), (L_4). Then (L_3) holds if and only if the following condition holds:*

(L_3') *For all i, $1 \leqq i \leqq p$, either $(\gcd(b_i, h) \neq 1$ and $m_i < 0$ and $\gcd(b_i^{-m_i}, h) = \gcd(b_i^{-m_i+1}, h) \neq \gcd(b_i^{-m_i-1}, h))$ or $(\gcd(b_i, h) = 1$ and $m_i \geqq 0$ and $b_i^{m_i} | g$ and $b_i^{m_i+1} \nmid g)$.*

Proof. Note that $r_{i0} \neq 0$ if and only if b_i does not divide r_i.

"$(L_3) \Rightarrow (L_3')$": Fix some i, $1 \leqq i \leqq p$. If $\gcd(b_i, h) = 1$, then $c_i = 1$ and $m_i \geqq 0$, since otherwise $b_i | g b_i^{-m_i}$ and $r_{i0} = 0$. (L_3') follows in this case. If $\gcd(b_i, h) \neq 1$, then $b_i \nmid g$ and $m_i \leqq 0$ by (L_2). Define $l < 0$ by $\gcd(b_i^{-l}, h) = \gcd(b_i^{-l+1}, h) \neq \gcd(b_i^{-l-1}, h))$. Then (L_2) implies that

$$c_i = \gcd(b_i^{|m_i|}, h) = \gcd(b_i^{|l|}, h)$$

and $m_i \leqq l$. If $m_i < l$, then $b_i | b_i^{-m_i}/c_i$, and (L_4) implies that $b_i | (h/c_i) r_i$, hence $b_i | r_i$. Thus $m_i = l$, and (L_3') follows.

"$(L_3') \Rightarrow (L_3)$": Fix some i, $1 \leqq i \leqq p$. If $m_i \geqq 0$, then $\gcd(b_i, h) = 1$ and $b_i \nmid g b_i^{-m_i}$, hence $b_i \nmid r_i$.

If $m_i < 0$, there exists an irreducible polynomial $q \in F[x]$ with multiplicity $l > 0$ in $c_i = \gcd(b_i^{-m_i}, h)$ such that $q^l \nmid b_i^{-m_i-1}$. Choose some such q and l, and let k be the multiplicity of q in b_i. Then $(-m_i - 1) \cdot k < l$, and the multiplicity of q in $b_i^{-m_i}/c_i$ is $(-m_i) \cdot k - l < k$. Since also $q \nmid g$, we have $q^k \nmid g b_i^{-m_i}/c_i$ and therefore $b_i \nmid (h/c_i) r_i$. □

Note that in Definition 6.3, m_i and c_i do not depend on N, and as in § 2, the Laurent (B, M)-representation for $M \leqq N$ is obtained by truncating the Laurent (B, N)-representation.

In order to compare Definitions 2.1 and 6.3, let B, N be as usual, $g, h \in F[x]$ with $\gcd(g, h) = 1$, $f = g/h \in F(x)$, $r = (m_1, \cdots, m_p; r_{10}, \cdots, r_{1,n_1-1}; \cdots; r_{p0}, \cdots, r_{p,n_p-1})$ a Laurent (B, N)-representative. We assume $b_i^{m_i} | g$ if $m_i \geqq 0$, and define $c_i = \gcd(b_i^{|m_i|}, h)$ as in (L_2). Let $b = b_1^{n_1} \cdots b_p^{n_p}$, $c = c_1 \cdots c_p$, and $u \in F[x]$ such that

$$\forall i \quad u \prod_{j \neq i} b_j^{-m_j} \equiv 1 \mod b_i^{n_i} c_i, \qquad \deg u < \deg bc.$$

For simplicity, we assume $\gcd(u, h) = 1$. Set $\bar{g} = (gu/c) \prod_{1 \leqq i \leqq p} b_i^{-m_i}$, $\bar{h} = h/c \in F[x]$, $\bar{f} = \bar{g}/\bar{h} \in F(x)$, and $\bar{r} = (r_{10}, \cdots, r_{1,n_1-1}; \cdots; r_{p0}, \cdots, r_{p,n_p-1}) \in R(B, N)$. Then $\gcd(\bar{g}, \bar{h}) = 1$, and we have:

1. r is a weak Laurent (B, N)-representation of (g, h) and (L_4) holds
   $\Leftrightarrow \bar{r}$ is a weak (B, N)-representation of $(\bar{g}, \bar{h})$,
2. r is the Laurent (B, N)-representation of f
   $\Leftrightarrow \bar{r}$ is a (B, N)-representation of $\bar{f}$ and (L_3') holds.

THEOREM 6.5. *Let B be a base, N a precision for B, and $f \in F(x)$. Then:*

(i) *f has a unique Laurent (B, N)-representation.*

(ii) *Given $g, h \in F[x]$ with $f = g/h$ and $\gcd(g, h) = 1$, algorithm STANDARD-TO-LAURENT computes the Laurent (B, N)-representation of f.*

Proof. (i) For an arbitrary $f \in F(x)$, write $f = g/h$ with $g, h \in F[x]$ and $\gcd(g, h) = 1$. Let r be the weak Laurent (B, N)-representation of (g, h) computed by algorithm *STANDARD-TO-LAURENT* with input B, N, g, h. Theorem 6.2(i) guarantees that (L_1), (L_2), (L_3'), (L_4) hold, hence r is the Laurent (B, N)-representation of f. This also proves claim (ii).

For the uniqueness, let $r^{(1)}$ and $r^{(2)}$ be two Laurent (B, N)-representations of $f = g/h$. Fix some i, $1 \leqq i \leqq p$. First note that the value of m_i is determined by the

condition (L_3'). As usual, we write $r_i^{(k)} = \sum_{0 \le j < n_i} r_{ij}^{(k)} b_i^j$ for $k = 1, 2$. Then

$$hr_i^{(1)} \equiv g b_i^{-m_i} \equiv hr_i^{(2)} \mod b_i^{n_i} c_i$$

by (L_4), and hence

$$\frac{h}{c_i} r_i^{(1)} \equiv \frac{h}{c_i} r_i^{(2)} \mod b_i^{n_i},$$

$$r_i^{(1)} \equiv r_i^{(2)} \mod b_i^{n_i},$$

since $h/c_i \in F[x]$ is a unit mod $b_i^{n_i}$ (using (L_2)). Since $\deg r_i^{(k)} < \deg b_i^{n_i}$, it follows that $r_i^{(1)} = r_i^{(2)}$, hence $r^{(1)} = r^{(2)}$. □

Next we want to prove that the functions computed by the two conversion algorithms are inverses of each other. It turns out that, unlike in Theorem 4.3, we have to impose a condition on the base to guarantee this.

THEOREM 6.6. *Let $B = (b_1, \cdots, b_p)$ be a base. The functions computed by algorithms STANDARD-TO-LAURENT and LAURENT-TO-STANDARD are inverses of each other if and only if each b_i is irreducible.*

Proof. To be more precise, we claim the following for any base B. Let N be a precision for B with total precision n, $0 \le d < n$, $R_N \subseteq \mathbf{Z}^p \times F^n$ the set of Laurent (B, N)-representatives. By Theorem 6.5, we have a mapping $\rho_N: F(x) \to R_N$, which associates to each $f \in F(x)$ the unique Laurent (B, N)-representation $\rho_N(f) \in R_N$ of f. Let

$$T_{B,N,d} = \Bigg\{ (g, h) \in F[x]^2 : \gcd(g, h) = 1,\ h \neq 0 \text{ is monic},$$
$$\rho_N\left(\frac{g}{h}\right) = (m_1, \cdots, m_p; r_{10}, \cdots, r_{1,n_1-1}; \cdots; r_{p0}, \cdots, r_{p,n_p-1}) \in R_N,$$
$$\deg g < n - d + \sum_{0 < m_i} m_i \deg b_i,\ \deg h \le d + \sum_{m_i < 0} |m_i| \deg b_i \Bigg\}.$$

The bounds on the degrees of g and h are the same as those for the output of *LAURENT-TO-STANDARD*. From Theorem 6.5 it is clear that *STANDARD-TO-LAURENT* maps $(g, h) \in T_{B,N,d}$ to $\rho_N(g/h) \in R_N$. Let

$$U_{B,N,d} = \rho_N\left(\left\{ f : \exists (g, h) \in T_{B,N,d} \quad f = \frac{g}{h} \right\}\right) \subseteq R_N.$$

For $(g, h) \in T_{B,N,d}$, we can execute algorithm *STANDARD-TO-LAURENT* with input (B, N, g, h) to get output $r = \rho_N(g/h) \in U_{B,N,d}$. On the other hand, for $r \in U_{B,N,d}$, we can execute algorithm *LAURENT-TO-STANDARD* with input (B, N, d, r) to get output (g, h). The claim of the theorem is that the two conversion algorithms give inverse bijections between $T_{B,N,d}$ and $U_{B,N,d}$ for all N and d if and only if each b_i is irreducible.

For the implication "$\Rightarrow$", we can assume that $b_1 = vw$ with $v, w \in F[x]$ non-constant. Choose $N = (2, 0, \cdots, 0)$ and $d = 1 + \deg v$. With input $g = 1$, $h = v$, *STANDARD-TO-LAURENT* will produce $r = (-1, 0, \cdots, 0; w, 0)$, and *LAURENT-TO-STANDARD* will yield $(w, b_1) \neq (1, v)$.

For the implication "$\Leftarrow$", it is sufficient to show that for any B, N, d, $(g, h) \in T_{B,N,d}$, and $r = \rho_N(g/h)$, algorithm *LAURENT-TO-STANDARD* with input (B, N, d, r) computes (g, h). We use the notation of the algorithm, and set

$$\bar{g} = \mu g t^{-1}, \qquad \bar{h} = \mu h u^{-1}.$$

Our goal is to show that $\bar{g}, \bar{h}$ are equal to g', h' as computed in step 3 of *LAURENT-TO-STANDARD*. First note that $\bar{g}, \bar{h} \in F[x]$: If $m_i \geqq 0$, then $e_i = 0$ and $b_i^{m_i} | g$ by condition (L_2). If $m_i < 0$, then $m_i + e_i = 0$ and $\gcd(b_i^{-m_i}, h) \neq \gcd(b_i^{-m_i - 1}, h)$. Since b_i is irreducible, it follows that $\gcd(b_i^{-m_i}, h) = b_i^{-m_i}$, and $b_i^{e_i} | h$. Therefore $u | h$, and thus $\bar{g}, \bar{h} \in F[x]$. Now for any i, $1 \leqq i \leqq p$, we have $c_i = b_i^{e_i} = \gcd(b_i^{|m_i|}, h)$, and

$$g b_i^{-m_i} \equiv h r_i \bmod b_i^{n_i + e_i},$$

$$u_i t_i b_i^{e_i} \bar{g} = \mu u_i g b_i^{-m_i} \equiv \mu u_i h r_i \equiv \mu h s_i t_i = \bar{h} u s_i t_i = u_i t_i b_i^{e_i} \bar{h} s_i \bmod b_i^{n_i + e_i}.$$

Since u_i and t_i are units mod $b_i^{n_i + e_i}$, it follows that

$$\bar{g} \equiv \bar{h} s_i \bmod b_i^{n_i}. \tag{1}$$

As to the degrees, we have

$$\begin{aligned} \deg \bar{g} &= \deg g + \sum_{1 \leqq i \leqq p} (-m_i - e_i) \deg b_i \\ &< n - d + \sum_{0 < m_i} m_i \deg b_i + \sum_{0 < m_i} (-m_i \deg b_i) = n - d, \\ \deg \bar{h} &= \deg h + \sum_{1 \leqq i \leqq p} (-e_i \deg b_i) \\ &\leqq d + \sum_{m_i < 0} (|m_i| \deg b_i) + \sum_{m_i < 0} (-|m_i| \deg b_i) = d, \end{aligned}$$

since $(g, h) \in T_{B,N,d}$. Now these degree inequalities and (1) also hold for g', h' as computed in *LAURENT-TO-STANDARD*. These were obtained as scalar multiples of certain entries a_k, t_k of the Extended Euclidean Scheme of $(b_1^{n_1} \cdots b_p^{n_p}, w)$, with h' being monic, and $w \equiv s_i \bmod b_i^{n_i}$ for all i. Just as for Theorem 4.3, we now use the uniqueness property of the Extended Euclidean Scheme to get $m \in \{1, \cdots, l\}$ and $v \in F[x]$ such that $\bar{g} = v a_m$, $\bar{h} = v t_m$. Again the inequalities for the degrees force $m = k$, and $\gcd(\bar{g}, \bar{h}) = 1$ implies $\bar{g} = g'$, $\bar{h} = h'$. Therefore the polynomials $\mu^{-1} g' t$ and $\mu^{-1} h' u$ computed by *LAURENT-TO-STANDARD* equal g and h. □

Remark 6.7. The "counterexample" $b_1 = uv$, $g = 1$, $h = u$, where $(v, b_1) \neq (g, h)$ is returned by *LAURENT-TO-STANDARD*, is of course not very convincing, since $v / b_1 = g / h$. The obvious remedy—returning $(g/a, h/a)$ in step 4 of *LAURENT-TO-STANDARD*, with $a = \gcd(g, h)$—does not work as expected. It may happen that $a' = \gcd(g', h') \neq 1$, and that

$$\frac{g'}{a'} \not\equiv \frac{h'}{a'} s_i \bmod b_i^{n_i};$$

see §§ 4 and 5 for examples.

Remark 6.8. Do we gain greater generality by dropping the requirement that the base polynomials $b_1, \cdots, b_p$ be relatively prime? The answer is no, not really. For simplicity, we consider in the following $b_i^{n_i}$ as one of the base polynomials (rather than b_i), and thus assume that all exponents n_i are 1. Suppose that $c = \gcd(b_1, b_2) \neq 1$. Then clearly for a representation $r = (r_1; r_2; \cdots)$ of $f \in F(x)$, r_1 and r_2 have to agree mod c. Continuing this process of replacing (b_i, b_j) by $(b_i / c, b_j / c, c)$ if $c = \gcd(b_i, b_j) \neq 1$, one arrives at a pairwise relatively prime basis $c_1, \cdots, c_q$. (Termination is guaranteed by the fact that the sum of the degrees decreases with each such step.) Each c_k is in the "gcd-closure" of $b_1, \cdots, b_p$ (as defined by this process), and there exist exponents $e_{ik} \geqq 0$ such that $b_i = \prod_{1 \leqq k \leqq q} c_k^{e_{ik}}$ for all i. If $a_1, \cdots, a_s$ are the distinct monic irreducible polynomials dividing $b_1 \cdots b_p$, and d_l is the smallest positive multiplicity of a_l in

$b_1, \cdots, b_p$, then each c_k is the product of some $a_l^{d_l}$. In fact, if $S \subseteq \{1, \cdots, s\}$ is nonempty and such that

$$\forall l, m \in S \quad \forall i \leqq p \quad (a_l^{d_l} | b_i \Rightarrow a_m^{d_m} | b_i)$$

and maximal with this property, then $\prod_{l \in S} a_l^{d_l}$ is one of the c_k's. Conversely, every c_k is of this form.

For $i, j \leqq p$, $k \leqq q$, let $u_{ijk} = \min\{e_{ik}, e_{jk}\}$. Then for any $r_1, \cdots, r_p \in F[x]$ we have

$$\exists f \in F[x] \quad \forall i \leqq p \quad f \equiv r_i \bmod b_i \Leftrightarrow \forall i, j, k \quad r_i \equiv r_j \bmod c_k^{u_{ijk}}.$$

(For "$\Leftarrow$", simply interpolate $r_{i_k} \bmod c_k^{e_{i_k,k}}$, $1 \leqq k \leqq q$, with $e_{i_k,k} = \max\{e_{jk} : 1 \leqq j \leqq p\}$.)

It is, however, not clear how to calculate $c_1, \cdots, c_q$ fast in parallel from $b_1, \cdots, b_p$. With a "logarithmic" look at exponents of the a_l's, this problem is related to the following: Given $f_i = (f_{i1}, \cdots, f_{is}) \in \mathbf{N}^s$ for $1 \leqq i \leqq p$, use addition and the coordinate-wise minimum of vectors in $\mathbf{N}^s$ to compute an "orthogonal" basis $g_1, \cdots, g_q \in \mathbf{N}^s$ such that

$$\forall i \quad f_i \in \sum_{1 \leqq k \leqq q} g_k \mathbf{N},$$

$$\forall k \neq m \quad \min(g_k, g_m) = (0, \cdots, 0),$$

$$\forall k \quad g_k \in \text{min-closure of } f_1, \cdots, f_p.$$

(Of course the algorithm would not know the coordinates of the input vectors f_i.)

Remark 6.9. The concepts presented in this paper obviously apply to more general rings than $F[x]$, e.g. a Euclidean valuation ring R, where one has a valuation w and a division-with-remainder property with respect to w (see von zur Gathen [1984a, § 4]). The two prominent examples are our case $R = F[x]$ with $w(f) = 2^{\deg f}$, and $R = \mathbf{Z}$ with $w(a) = |a|$. The "interpolation problem" can be phrased as a "simultaneous approximation problem": finding $g, h \in R$ such that $g \equiv hr_i \bmod b_i^{n_i}$ corresponds to simultaneously approximating each r_i in the b_i-adic valuation with precision n_i (if each b_i is irreducible); the degree condition for polynomials translates into upper bounds for $w(g)$ and $w(h)$. For the general problem, we would be given further valuations $v_1, \cdots, v_p$ on R, precisions $\delta_1, \delta_2, \varepsilon_1, \cdots, \varepsilon_p \in \mathbf{R}$, and $r_1, \cdots, r_p \in R$, and want to compute $g, h \in R$ such that

$$v_i(g - hr_i) \leqq \varepsilon_i,$$

$$w(g) \leqq \delta_1, \qquad w(h) \leqq \delta_2.$$

This question of simultaneous approximation with respect to various valuations subsumes among others the usual Chinese remainder computations, solution of congruences by rational numbers (Miola [1982]), and Padé approximation of power series. If we consider for v_i the absolute value on $R = \mathbf{Z}$ instead of a b_i-adic valuation (and allow $r_i \in \mathbf{R}$), then it also subsumes approximation of a real number by rational numbers.

As of now, no $(\log n)^{O(1)}$ parallel computation for integer Chinese remaindering or the gcd of two n-bit integers is known. But even for sequential algorithms, it would be interesting to have a general approximation algorithm that solves all the above problems.

REFERENCES

G. A. Baker and P. Graves-Morris [1981], *Padé approximants*, Encyclopedia of Mathematics and Its Applications, vols. 13 and 14, Addison-Wesley, Reading, MA.

S. J. BERKOWITZ [1984], *On computing the determinant in small parallel time using a small number of processors*, Inform. Proc. Letters, 18, pp. 147–150.

A. BORODIN, S. COOK AND N. PIPPENGER [1983], *Parallel computation for well-endowed rings and space-bounded probabilistic machines*, Inform. and Control, 58, pp. 96–114.

A. BORODIN, J. VON ZUR GATHEN AND J. HOPCROFT [1982], *Fast parallel matrix and GCD computations*, Inform. and Control, 52, pp. 241–256.

R. P. BRENT, F. G. GUSTAVSON AND D. Y. Y. YUN [1980], *Fast solution of Toeplitz systems of equations and computation of Padé approximants*, J. Algorithms, 1, pp. 259–295.

W. S. BROWN [1971], *On Euclid's algorithm and the computation of polynomial greatest common divisors*, J. Assoc. Comput. Mach., 18, pp. 478–504.

A. CAUCHY [1821], *Cours d'analyse de l'école royale polytechnique (Analyse algébrique)*, in Oeuvres complètes, IIe série, tôme III, pp. 429–433.

G. E. COLLINS [1967], *Subresultants and reduced polynomial remainder sequences*, J. Assoc. Comput. Mach., 14, pp. 128–142.

L. CSANKY [1976], *Fast parallel matrix inversion algorithms*, this Journal, 5, pp. 618–623.

W. EBERLY [1984], *Very fast parallel matrix and polynomial arithmetic*, Proc. 25th Annual IEEE Symposium on Foundations of Computer Science, Singer Island FL, pp. 21–30.

J. VON ZUR GATHEN [1984a], *Hensel and Newton methods in valuation rings*, Math. Comp., 42, pp. 637–661.

——— [1984b], *Parallel algorithms for algebraic problems*, this Journal, 13 (1984), pp. 802–824.

K. O. GEDDES [1979], *Symbolic computation of Padé approximants*, ACM Trans. Math. Software, 5, pp. 218–233.

W. B. GRAGG [1972], *The Padé table and its relation to certain algorithms of numerical analysis*, SIAM Rev., 14, pp. 1–62.

P. R. GRAVES-MORRIS [1980], *Efficient reliable rational interpolation*, Proc. Conf. Padé and Rational Approximation, Theory and Applications, Amsterdam, pp. 28–63.

F. G. GUSTAVSON AND D. Y. Y. YUN [1979], *Fast algorithms for rational Hermite approximation and solution of Toeplitz systems*, IEEE Trans. Circuits and Systems, 26, pp. 750–755.

C. G. J. JACOBI [1846], *Ueber die Darstellung einer Reihe gegebner Werthe durch eine gebrochne rationale Function*, J. Reine Angew. Math., 30, pp. 127–156.

D. E. KNUTH [1981], *The Art of Computer Programming*, Vol. 2, 2nd ed., Addison-Wesley, Reading, MA.

E. KRONECKER [1881], *Zur Theorie der Elimination einer Variabeln aus zwei algebraischen Gleichungen*, Monatsberichte der Akademie der Wissenschaften, Berlin, pp. 535–600.

A. M. MIOLA [1982], *The conversion of Hensel codes to their rational equivalents*, SIGSAM Bull, 16, pp. 24–26.

H. PADÉ [1892], *Sur la représentation approchée d'une fonction par des fractions rationnelles*, Annales Scientifiques de l'Ecole Normale Supérieure, 3e série, 9, Supplément S3-S93.

J. REIF [1983], *Logarithmic depth circuits for algebraic functions*, Proc. 24th Annual IEEE Symposium on Foundations of Computer Science, Tucson AZ, pp. 138–145.

V. STRASSEN [1974], *Some results in algebraic complexity theory*, Proc. International Congress of Mathematicians, Vancouver, pp. 497–501.

——— [1983], *The computational complexity of continued fractions*, this Journal, 12, pp. 1–27.