AD-787 732

MULTIVARIATE POLYNOMIAL FACTORIZATION

David R. Musser

Wisconsin University

Prepared for:

National Science Foundation
Army Research Office-Durham

July 1974

AR 70-31 Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D AD787732

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Mathematics Research Center
University of Wisconsin, Madison, Wis. 53706

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
None

3. REPORT TITLE

MULTIVARIATE POLYNOMIAL FACTORIZATION

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Summary Report: no specific reporting period.

5. AUTHOR(S) (First name, middle initial, last name)

David R. Musser

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July 1974	56	18

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
Contract No. DA-31-124-ARO-D-462	1445
b. PROJECT NO. None	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	None

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Army Research Office-Durham, N.C.

13. ABSTRACT

This paper describes algorithms for factoring a polynomial in one or more variables, with integer coefficients, into factors which are irreducible over the integers. These algorithms are based on the use of factorizations over finite fields and "Hensel's Lemma constructions." "Abstract algorithm" descriptions are used in the presentation of the underlying algebraic theory. Included is a new generalization of Hensel's p-adic construction which leads to a practical algorithm for factoring multivariate polynomials. The univariate case algorithm is also specified in greater detail than in the previous literature, with attention to a number of improvements which the author has developed based on theoretical computing time analyses and experience with actual implementations.

DD FORM 1 NOV 1473

Unclassified
Security Classification

AGO 5698A

THE UNIVERSITY OF WISCONSIN-MADISON
MATHEMATICS RESEARCH CENTER

Contract No. DA-31-124-ARO-D-462

MULTIVARIATE POLYNOMIAL FACTORIZATION

David R. Musser

MRC Technical Summary Report #1445
July 1974

Received February 25, 1974

II

Madison, Wisconsin 53706

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

MULTIVARIATE POLYNOMIAL FACTORIZATION

David R. Musser†

Technical Summary Report #1445
July 1974

ABSTRACT

This paper describes algorithms for factoring a polynomial in one or more variables, with integer coefficients, into factors which are irreducible over the integers. These algorithms are based on the use of factorizations over finite fields and "Hensel's Lemma constructions." "Abstract algorithm" descriptions are used in the presentation of the underlying algebraic theory. Included is a new generalization of Hensel's p-adic construction which leads to a practical algorithm for factoring multivariate polynomials. The univariate case algorithm is also specified in greater detail than in the previous literature, with attention to a number of improvements which the author has developed based on theoretical computing time analyses and experience with actual implementations.

---

†Computer Sciences Department and Mathematics Research Center, University of Wisconsin; on leave from University of Texas. This work was sponsored in part by the National Science Foundation under grants GJ239, GJ-30125X and GJ-1069, by the United States Army under Contract No. DA-31-124-ARO-D-462, and by the Wisconsin Alumni Research Foundation.

MULTIVARIATE POLYNOMIAL FACTORIZATION

David R. Musser†

1. Introduction and basic concepts.

This paper presents algorithms for factoring a given polynomial in one or more variables, with integer coefficients, into factors which are irreducible over the integers. These algorithms are based on the use of Berlekamp's algorithm for factoring modulo a prime and "Hensel's Lemma constructions" as suggested by Zassenhaus [ZAS69]. A new generalization of Hensel's construction is given, providing a practical basis for an algorithm for factoring multivariate polynomials.

The algorithm for the univariate case has been implemented by G. E. Collins and the author in the SAC-1 system for algebraic calculation [COL71] and tested thoroughly. A detailed description of this implementation is given in [COL72]. Both the univariate and multivariate algorithms have been implemented in the Pascal language at the University of Texas by R. T. Charleton [CHA73].

Following a brief discussion of notation and basic concepts of factorization and use of homomorphic mappings, we shall define the concept of an abstract algorithm, in order to present concisely the common theory behind the univariate and multivariate algorithms. Section 2 gives an overview of the main steps in factorization, followed by detailed abstract algorithms in Section 3. In Sections 4 and 5 we consider the details of applications of

---

†Computer Sciences Department and Mathematics Research Center, University of Wisconsin; on leave from University of Texas. This work was sponsored in part by the National Science Foundation under grants GJ239, GJ-30125X and GJ-1069, by the United States Army under Contract No. DA-31-124-ARO-D-462, and by the Wisconsin Alumni Research Foundation.

the abstract algorithms to the univariate and multivariate integral polynomials. Some consideration is given to computing times in these sections.

1.1. Polynomial notation

A polynomial $A(x) = a_n x^n + \ldots + a_1 x + a_0$ with coefficients $a_n, \ldots, a_1, a_0$ from a ring R, $a_n \neq 0$, is said to have degree n, leading coefficient a_n, and trailing coefficient (or constant term) a_0; we write

$$\deg(A) = n\ ,\ \mathrm{lc}(A) = a_n,\ \ \mathrm{tc}(A) = a_0.$$

By convention, we define

$$\deg(0) = -\infty,\ \ \mathrm{lc}(0) = 0\ ,\ \ \mathrm{tc}(0) = 0\ .$$

If R has an identity 1, we say $A(x)$ is monic if $\mathrm{lc}(A) = 1$.

1.2. Unique factorization domains

In a commutative ring with identity, zero-divisors are elements y and z such that $y \cdot z = 0$. A unit is a divisor of unity, and a prime is a nonunit element which cannot be expressed as a product of nonunit elements. An integral domain is a commutative ring with identity which contains no zero-divisors. A unique factorization domain (UFD) is an integral domain in which every nonzero element is a unit, or is prime, or has a unique factorization into primes (an expression as a product of a finite number of primes which is unique except for unit factors and the order of factors).

Primes are also called irreducible elements, and a unique factorization into primes is often called a complete factorization.

The integral domain Z of integers is a UFD (Fundamental Theorem of Arithmetic), in which the only units are 1 and -1. Any field F is a UFD in which every nonzero element is a unit and there are no irreducible elements.

According to a theorem of Gauss [VDW49, §23], the polynomial domain $D[x_1,\ldots,x_n]$ is a UFD whenever D is. Thus, for example, $Z[x_1,\ldots,x_n]$ and $F[x_1,\ldots,x_n]$ are UFD's.

1.3. Homomorphic mappings

A mapping h from a ring R into a ring $\bar{R}$ is called a homomorphism if for all $a,b \in R$,

(1) $h(a+b) = h(a) + h(b)$,

(2) $h(ab) = h(a)h(b)$.

The application of homomorphic mappings to factorization is based on the factor preserving property (2). The classical algorithm for factoring polynomials, Kronecker's algorithm [VDW49, §25], is based on the use of evaluation homomorphisms. For any fixed $a \in R$, the mapping e_a of $R[x]$ onto R, defined by $e_a(P) = P(a)$ for all $P(x) \in R[x]$, is homomorphic and is called an evaluation homomorphism. To factor $P(x) \in Z[x]$, for example, Kronecker's algorithm evaluates $P(x)$ at several integers, factors the resulting values in Z, and constructs the factors of $P(x)$ using interpolation.

Another well-known application of homomorphic mappings to polynomial factorization is the use of mod p factorizations, where p is a prime integer. Let $P(x) \in Z[x]$ and p be a prime which does not divide the leading coefficient of P. Let h_p denote the homomorphism of Z onto Z_p, the ring of integers modulo p. Z_p is actually a field, so $Z_p[x]$ is a UFD. If $h_p(P)$ turns out to be irreducible over Z_p, then P is irreducible over Z (except possibly for integer factors). If $h_p(P)$ does factor, then its factorization gives an idea what degrees the factors of P might have, and what residue

classes the coefficients modulo p might belong to. These facts have long been used in the limited number of cases in which $h_p(P)$ is easy to factor, e.g. [VDW49, §25]. More general applications of mod p homomorphisms have become possible since the invention in 1967 by Berlekamp of efficient algorithms for factorization in $Z_p[x]$ ([BER68, Ch. 6], [KNU69, §4.6.2]). A second breakthrough was Zassenhaus' suggestion that a construction based on Hensel's Lemma, from the theory of p-adic fields, could be used to progress from a mod p factorization to a corresponding factorization modulo any power of p [ZAS69]. Taking p^j sufficiently large, we can determine from consideration of all mod p^j factorizations all factorizations over the integers. This "Berlekamp-Hensel" factorization algorithm has been improved and extended in a number of ways, as discussed previously in [BER70] and [MUS71]. [COL73], Section 5, gives an overview of this research. The author's main contributions have been the detailed specifications and implementation of a univariate factoring algorithm, with extensive analysis of maximum computing times, and generalization of Hensel's construction to several moduli as a basis for a new multivariate algorithm. Wang and Rothchild [WAN73] use a different generalization of Hensel's construction, but as yet no comparison of the merits of the two constructions has been made.

1.4. Abstract algorithms and validity proofs

In this paper we shall use "abstract algorithm" descriptions in order to present compactly the common theory behind factoring algorithms for both the univariate and multivariate cases and for a number of coefficient domains. An abstract algorithm is one in which the domains of the inputs and outputs are

abstract sets or algebraic systems such as rings, integral domains, or fields. An example of an abstract algorithm is:

Algorithm D (Division of polynomials over a ring). Let $\mathfrak{R}$ be a commutative ring with identity. Given polynomials $A, B \in \mathfrak{R}[x]$ with $lc(B)$ a unit of $\mathfrak{R}$, this algorithm computes polynomials $Q, R \in \mathfrak{R}[x]$ such that

$$A = BQ + R \text{ and } \deg(R) < \deg(B).$$

(1) Set $Q \leftarrow 0$ and $R \leftarrow A$.

(2) (Now $Q, R \in \mathfrak{R}[x]$ and $A = BQ + R$.) If $\deg(R) < \deg(B)$, exit.

(3) Set $n \leftarrow \deg(R) - \deg(B)$, $T \leftarrow (lc(R)/(lc(B))x^n$, $Q \leftarrow Q + T$, $R \leftarrow R - TB$ (this reduces the degree of R), and go to (2).

In dealing with abstract algorithms we leave open the question of what assumptions are required about the abstract domains involved in order to prove effectiveness of the algorithm. (Such questions have been dealt with elsewhere, e.g. [RAB60].) We shall however require that, under the assumption that each step can be effectively performed, the algorithm will terminate in a finite number of steps. A proof of termination of Algorithm D is indicated in the parenthetical assertion in step (3): by the choice of the term T of the quotient polynomial Q, both R and TB have the same leading coefficient, hence the new value of R, $R_1 = R - TB$, is of smaller degree than that of R, and thus the condition tested in step (2) must eventually be satisfied.

If we do not require effectiveness in our abstract algorithms, the reader may well ask, by what criteria do we construct them? For we could in some steps of our algorithms merely cite the existence of some quantity without any indication of a method of constructing the quantity. However, all of

the algorithms to be presented have been written with the purpose of generalizing methods which are known not just to be effective in particular domains, but to be "very effective" or "efficient" methods. This is meant in the sense that each step of the abstract algorithm is of sufficient simplicity that there are known to be efficient algorithms for carrying it out in at least one particular domain. In Algorithm D, for example, each step involves only simple arithmetic operations for which efficient algorithms are known when R is the ring of integers, or the rational number field, or a finite field.

Besides the proof of termination, we are also interested in proving the validity of the algorithm: that when applied to inputs which satisfy the input assumptions, the algorithm produces outputs which satisfy the output assertions. The method of proof to be used is based on the method of "inductive assertions" described in [FLO67] and [KNU68, §1.2.1]. The basic idea of the method is to associate with some or all of the steps or substeps of the algorithm assertions about the current state of the computation, and to prove that each assertion is true each time control reaches the corresponding step, under the assumption that the previously encountered assertions are true. If this can be done in such a way that the assertions associated with the first step are the input assumptions and those associated with the terminal step(s) are the output assertions, then the algorithm is necessarily valid, by induction on the number of steps performed.

In applying the method we have usually not attempted to list all of the assertions which actually hold at each step; in general we have tried to maintain about the same degree of explicitness as is usual in a conventional proof

of a theorem. In Algorithm D, we have included only two assertions, in step (2), for the purpose of proving validity (the assertion in step (3) was included for the sake of proving termination, as discussed previously). It is trivial that these assertions are true the first time step (2) is executed. Assuming them true at a given execution of step (2), they may be shown to be true at the next execution as follows: Let $Q_1 = Q + T$ and $R_1 = R - TB$; since $lc(B)$ is a unit, $T \in \mathfrak{R}[x]$, hence so are Q_1 and R_1; also $BQ_1 + R_1 = B(Q + T) + R - TB = BQ + R = A$; since Q is set to Q_1 and R to R_1 in step (3), the assertions $Q, R \in \mathfrak{R}[x]$ and $A = BQ + R$ still hold when step (2) is reached again.

The abstract algorithm concept may be easily formalized in terms of conventional set theory, and in fact such a formalization is given by Knuth in his initial formal definition of algorithms [KNU68, pp. 7-8]. (Knuth goes on to modify this definition to include the property of effectiveness.) The inductive assertion method is also easily formalized in terms of Knuth's model, as shown in [MUS71].

2. Overview.

In Section 3, we shall state the basic algorithms for factorization in $D[x]$ where D is any UFD. In Sections 4 and 5 we consider separately the cases that $D = Z$ and $D = Z[v_1, \ldots, v_n]$. Assume we are given a polynomial $C(x) \in D[x]$ to be factored, i.e. we are given a representation

$$C(x) = c_m x^m + c_{m-1} x^{m-1} + \ldots + c_0 \, , \; c_i \in D$$

and we must determine the factors of $C(x)$ which are irreducible over D. The following are the essential steps of the overall algorithm:

1. First eliminate proper factors of degree zero and repeated factors by means of greatest common divisor calculations in D and $D[x]$. (These steps are sufficient to satisfy some of the assumptions made in later phases of the algorithm, particularly Hensel's construction.) Thus we have

$$C(x) = \prod_{i=1}^{q} F_i(x)$$

where the F_i are distinct irreducible polynomials of positive degree, and our task is to determine these F_i.

2. Choose $p_1, \ldots, p_m$ in D such that factorization in $E[x]$ is possible, where $E = D/(p_1, \ldots, p_m)$. (With $D = Z$, we will have $m = 1$, choosing a single prime integer p and E will be $Z_p = GF(p)$, the Galois field of order p. With $D = Z[v_1, \ldots, v_n]$ we will have $m = n+1$, choosing a prime p and linear polynomials $v_1 - a_1, \ldots, v_n - a_n$ as the moduli; again $E = GF(p)$.)

3. Obtain a factorization

$$C \equiv c \prod_{k=1}^{r} G_k \pmod{p_1, \ldots, p_m}$$

$$c = \mathrm{lc}(C), \quad G_k \in D[x]$$

not necessarily complete, but such that each F_i corresponds to a product of one or more of the G_k; i.e. there is a partition of $\{G_1, \ldots, G_r\}$ into subsets $\mathcal{S}_1, \ldots, \mathcal{S}_q$ such that

$$F_i \equiv f_i \prod_{G \in \mathcal{S}_i} G \pmod{p_1, \ldots, p_m}$$

$$f_i = \mathrm{lc}(F_i) .$$

4. Using Hensel's construction, lift the G_k to corresponding $H_k \in D[x]$ such that

$$C \equiv c \prod_{k=1}^{r} H_k \pmod{p_1^{j_1}, \ldots, p_m^{j_m}}$$

for sufficiently large positive integers $j_1, \ldots, j_m$.

5. Partition the H_k into subsets $\mathcal{V}_i$ such that

$$F_i \equiv f_i \prod_{H \in \mathcal{V}_i} H \pmod{p_1^{j_1}, \ldots, p_m^{j_m}}$$

thereby determining the F_i.

In order to simplify the presentation, we shall confine the discussion in Section 3 to the case of a single modulus and defer generalizations to several moduli to Section 5.

3. Abstract factoring algorithms

3.1. Reduction to a primitive polynomial. If $C(x) = c_0 \in D$ then we merely factor c_0 in D; we assume the existence of an algorithm for this factorization. Otherwise, we compute the greatest common divisor d of $c_m, \ldots, c_0$ (called the content of C in D) and divide $C(x)$ by d, thereby obtaining a primitive polynomial $C^*(x)$, i.e. one whose coefficients are relatively prime. Thus $C^*(x)$, called the primitive part of $C(x)$ (denoted $pp(C)$) has no proper factors of degree zero, and this property simplifies the task of factoring $C^*(x)$. We proceed to factor d and $C^*(x)$ and combine the two lists of factors to produce the list of factors of $C(x)$.

3.2. Reduction to squarefree polynomials. Given a primitive polynomial $C(x)$ over D, we proceed to factor it into squarefree polynomials, i.e. having no repeated factors. Using greatest common divisor calculations we obtain a factorization

$$C = Q_1 Q_2^2 \cdots Q_t^t \qquad (1)$$

where Q_i is the product of all the irreducible factors of C with multiplicity i. We then factor each Q_i, putting i copies of each factor on the list of factors of C.

The algorithm for producing the factorization (1) is based on Theorem S below. For its statement we require two definitions:

Elements x and y in a ring D are said to be associates if $x = uy$ for some unit u of D. We write $x \sim y$ (this is an equivalence relation).

The characteristic of a ring D is the smallest positive integer n such that $nx = 0$ for all x in D, or zero if no such integer exists. (If D is an integral domain, the characteristic is prime if it is not zero.)

Theorem S. Let D be a UFD, C be a nonconstant, primitive polynomial over D, and $B = \gcd(C, C')$ where C' denotes the derivative of C. Let $C = P_1^{e_1} \cdots P_n^{e_n}$ be a complete factorization of C.

a. If $\deg(B) = 0$ then C is squarefree.

b. If D has characteristic zero, then $B \sim P_1^{e_1-1} \cdots P_n^{e_n-1}$.

c. If D has characteristic zero and C is squarefree, then $B \sim 1$.

d. If D has characteristic zero, then $C/B \sim P_1 \cdots P_n$, the greatest squarefree divisor of C.

Proof: a. Suppose C is not squarefree; thus $C = P^2Q$ for some P and Q over D, $\deg(P) > 0$. Then $C' = P^2Q' + 2PP'Q$ is a multiple of P, hence $P|B$, hence $\deg(B) > 0$. Thus $\deg(B) = 0$ implies C is squarefree.

b. Since $B|C$, $B \sim P_1^{\delta_1} \cdots P_n^{\delta_n}$, where $0 \le \delta_i \le e_i$, $1 \le i \le n$. To show that $\delta_i = e_i - 1$, let $P = P_i$, $e = e_i$ and $Q = C/P^e$. Then $C = P^eQ$ and $C' = P^eQ' + eP^{e-1}P'Q$, hence $P^{e-1}|B$. Suppose $P^e|B$. Then $P^e|C'$, hence $P^e|eP^{e-1}P'Q$, and since D is an integral domain, $P|eP'Q$. But P and Q are relatively prime, so $P|eP'$. Since the characteristic of D is zero, $eP' \neq 0$, hence $\deg(eP') \ge \deg(P)$, a contradiction. Thus $P^e \nmid B$, while $P^{e-1}|B$, so $\delta_i = e - 1 = e_i - 1$.

c,d. Obvious from b.

Thus to factor C one could compute the greatest squarefree divisor $A = C/\gcd(C, C')$ and factor it to obtain the P_i, then divide C by P_i as many times as possible, to determine the e_i. However, we can do better than this if C is not already squarefree, for we will show that we can then partially factor A and determine the e_i by means of further gcd calculations.

Let $Q_i = \prod_{j \in E_i} P_j$, where $E_i = \{j : e_j = i\}$. ($Q_i = 1$ when E_i is empty.) Then, for $t = \max\{e_1, \ldots, e_n\}$ we have

$$C = Q_1 Q_2^2 \cdots Q_t^t, \quad Q_i \text{ squarefree},$$

$$\deg(Q_t) > 0, \ \gcd(Q_i, Q_j) \sim 1 \ \text{ for } \ i \neq j . \tag{2}$$

We call (2) a squarefree factorization of C, since each Q_i is either unity or a squarefree polynomial of positive degree. The Q_i are uniquely determined by the conditions in (2), except for unit factors.

By Theorem S, if $B = \gcd(C, C')$ and $A = C/B$ then $B \sim Q_2 Q_3^2 \cdots Q_t^{t-1}$ and $A \sim Q_1 Q_2 \cdots Q_t$. If $D = \gcd(A, B)$ then $D \sim Q_2 Q_3 \cdots Q_t$, hence $Q_1 \sim A/D$. The following algorithm shows how we can continue, computing $Q_2, \ldots, Q_t$:

Algorithm S (Squarefree factorization). Let D be a UFD of characteristic zero. Given a primitive polynomial C of positive degree, let $C = Q_1 Q_2^2 \cdots Q_t^t$ be a squarefree factorization of C. This algorithm computes t and $A_1 \sim Q_1, \ldots, A_t \sim Q_t$.

(1) Set $B \leftarrow \gcd(C, C')$, $A \leftarrow C/B$, $j \leftarrow 1$.

(2) (At this point $B \sim Q_{j+1} Q_{j+2}^2 \cdots Q_t^{t-j}$ and $A \sim Q_j Q_{j+1} \cdots Q_t$) .
If $B \sim 1$ then set $t \leftarrow j$, $A_t \leftarrow A$, and exit.

(3) Set $D \leftarrow \gcd(A, B)$, $A_j \leftarrow A/D$. (Then $D \sim Q_{j+1} Q_{j+2} \cdots Q_t$ and $A_j \sim Q_j$.)

(4) Set $B \leftarrow B/D$, $A \leftarrow D$, $j \leftarrow j+1$, and go to (2).

The reader may easily verify the inductive assertions in the algorithm.

Algorithm S is based on an algorithm presented by Horowitz in [HOR69, pp. 58-60, 69-70], which in turn was based on an algorithm due to Robert Tobey. Horowitz' version is equivalent to Algorithm S with steps (3) and (4) replaced by:

(3') Set $E \leftarrow \gcd(B, B')$, $D \leftarrow B/E$, $A_j \leftarrow A/D$. (Then $E \sim Q_{j+2} Q_{j+3}^2 \cdots Q_j^{t-j-1}$, $D \sim Q_{j+1} Q_{j+2} \cdots Q_t$, $A_j \sim Q_j$.)

(4') Set $B \leftarrow E$, $A \leftarrow D$, $j \leftarrow j+1$, and go to (2).

Note that D and $E = B/D$ are computed in both versions, but in different ways. Algorithm S appears to require slightly less computation than Horowitz' version, but its main virtue seems to be that it can be easily adapted for squarefree factorization over finite fields (which are of prime rather than zero characteristic), whereas it appears to be rather difficult to adapt Horowitz' version for this problem. Algorithms for the finite field case are discussed in [MUS71]. These algorithms are, however, not necessary in the application to factoring integral polynomials, as will be seen in the following section.

3.3. Choice of modulus. Now assume $C(x)$ is primitive and squarefree. We next choose a modulus p such that factorization in $E[x]$ is possible, where $E = D/(p)$. We shall see in Section 3.7 that in order to apply the Hensel construction to lift a given factorization

$$C \equiv \bar{A}\bar{B} \pmod{p} \tag{1}$$

to a corresponding factorization

$$C \equiv AB \pmod{p^j}$$

it is necessary to also have $\bar{S}$, $\bar{T}$ such that

$$\bar{A}\bar{S} + \bar{B}\bar{T} \equiv 1 \pmod{p} . \tag{2}$$

Sufficient conditions for the existence of $\bar{S}$, $\bar{T}$ are that E be a field and $\bar{A}$ and $\bar{B}$ be relatively prime over E, for then the Extended Euclidean Algorithm yields $\bar{S}$ and $\bar{T}$. Let us assume that we can find p such that E is a field and C has the same degree and remains squarefree mod p (i.e. when regarded as a polynomial over E). Then, in (1), $\bar{A}$ and $\bar{B}$ must be relatively prime, and thus (2) is satisfiable.

As we saw in Theorem S, part a, if we compute $B = \gcd(C, C')$ in $E[x]$ and find $\deg(B) = 0$ then this guarantees that C is squarefree in $E[x]$.

In the case $D = Z$, we choose a prime integer p, obtaining $E = GF(p)$, the Galois field of order p. We shall see in Section 4.1 that there are only a finite number of primes p for which C can fail to be squarefree mod p. There are a number of other considerations in the choice of primes in Z as we shall discuss in Section 4.1.

3.4. Factorization mod p. Since we have chosen p so that E is a field, $E[x]$ is a UFD. We assume the existence of an algorithm for factoring in $E[x]$, but not necessarily one which obtains the complete factorization of $C \bmod p$. A partial factorization

$$C \equiv g \prod_{k=1}^{r} G_k \pmod{p} \tag{1}$$

will suffice, provided the G_k are distinct and there exists a partition of the G_k into subsets corresponding to the irreducible factors F_i of C, as described in Section 2, step 3. Ideally, we would like to find the factorization in which each G_k is the image of some F_i, but generally there is no a priori way of satisfying the partition requirement other than obtaining the complete factorization in $E[x]$. We shall see, however, in Section 5 a very important case in which it can be satisfied with a partial factorization.

Since E is a field, it is convenient to assume the G_k are monic. Then $g \equiv \mathrm{lc}(C) \pmod{p}$.

Of course, if we find that $r = 1$ then since $C(x)$ has the same degree modulo p, it must be irreducible over D, and we are done. Otherwise we have to continue with the following steps.

3.5. Determining modulus size. In choosing our modulus p, we gave no consideration in Section 3.3 to its "size." If $D = Z$ and p is sufficiently large, then the set $\{-\lfloor p/2 \rfloor, \ldots, 0, \ldots, \lfloor p/2 \rfloor\}$ of residues of p contains the coefficients of any factor of C, and we

could proceed directly to determine the true factors of C using the mod p factorization. For large p, however, it may be very difficult to obtain the mod p factorization (this point is discussed further in Section 4.1). Hensel's construction provides the alternative of using a small prime p and lifting a mod p factorization to a mod p^j factorization for sufficiently large j.

In the case of an abstract domain D, our assumption at this point is that we can algorithmically determine a positive integer j and a complete set of residues R of p^j, such that R is a <u>factoring set</u> for C. In general, we define a factoring set for C to be any subset of D which contains the coefficients of any factor A^* of $C^* = lc(C) \cdot C$ for which $deg(A^*) \leq \lfloor deg(C)/2 \rfloor$ and $lc(A^*) \mid lc(C)$. These requirements may seem odd, but will become clear when we examine the operation of the algorithm for finding true factors, in Section 3.8.

3.6. <u>Lifting a factorization (Hensel's construction for several factors)</u>. At this point we have a primitive, squarefree polynomial $C \in D[x]$, $p \in D$ such that $E = D/(p)$ is a field and C has the same degree and is squarefree mod p, a positive integer j and monic polynomials $G_1, \ldots, G_r \in D[x] (r \geq 2)$ such that

$$C \equiv lc(C) G_1 \cdots G_r \pmod{p} .$$

The goal now is to lift this factorization to a corresponding one mod p^j, i.e.

$$C \equiv \mathrm{lc}(C) H_1 \cdots H_r \pmod{p^j}$$

$$\left.\begin{array}{r} H_i \equiv G_i \pmod p \\ \deg(H_i) = \deg(G_i) \\ H_i \text{ is monic} \end{array}\right\} \quad i = 1, \ldots, r .$$

This is done by repeated application of Hensel's construction to pairs of factors in which one factor is G_i and the other is $G_{i+1} \cdots G_r$.

1. Set $\bar{C} \leftarrow C \bmod p$, $i \leftarrow 1$.

2. (Now we have

   a. $C_0 \equiv C H_1 \cdots H_{i-1} \pmod{p^j}$ where C_0 was the initial value of C;

   b. $H_k \equiv G_k \pmod p$, $\deg(H_k) = \deg(G_k)$, and H_k is monic for $k = 1, \ldots, i-1$;

   c. $\bar{C} \equiv C \equiv \mathrm{lc}(C) G_i G_{i+1} \cdots G_r \pmod p$;

   d. $\bar{C}$ is squarefree mod p.)

   Set $\bar{A} \leftarrow G_i$, $\bar{B} \leftarrow \bar{C}/\bar{A}$ (division mod p). (Thus $C \equiv \bar{A}\bar{B} \pmod p$, $\bar{A}$ is monic, and $\bar{A}$ and $\bar{B}$ are relatively prime mod p, by d.)

3. Using the Extended Euclidean Algorithm, obtain $\bar{S}, \bar{T} \in D[x]$ such that $\bar{A}\bar{S} + \bar{B}\bar{T} \equiv 1 \pmod p$.

4. Apply Algorithm Q (Hensel's construction, as described in Section 3.7) to $p, j, C, \bar{A}, \bar{B}, \bar{S}, \bar{T}$, obtaining $A, B, S, T \in D[x]$

such that

$$C \equiv AB \pmod{p^j}$$

$$A \equiv \bar{A} \pmod{p}$$

$$B \equiv \bar{B} \pmod{p}$$

$$\deg(A) = \deg(\bar{A})$$

$$A \text{ is monic .}$$

5. Set $H_i \leftarrow A$, $C \leftarrow B$, $\bar{C} \leftarrow \bar{B}$, $i \leftarrow i + 1$. If $i \leq r$, go to step 2. Otherwise, exit.

3.7. Quadratic Hensel construction. This "quadratic" construction, so-called because it progresses through factorizations modulo $p, p^2, p^4, p^8, \ldots$ in successive iterations, will be given in essentially the form discussed by Knuth [KNU69, pp. 398, 546]. This version differs somewhat from the construction proposed by Zassenhaus [ZAS69], although the latter is also quadratic in nature. (Hensel's original construction, in the theory of p-adic fields, was linear [VDW 49, pp. 248-250].)

Algorithm Q (Quadratic Hensel Algorithm). Let D be a commutative ring with identity. The inputs are an element p of D; a positive integer j; and polynomials $C, \bar{A}, \bar{B}, \bar{S}, \bar{T} \in D[x]$ such that

$$C \equiv \bar{A}\bar{B} \pmod{p},\ \bar{A}\bar{S} + \bar{B}\bar{T} \equiv 1 \pmod{p},\ \bar{A} \text{ is monic.}$$

The outputs are $q = p^i$ where $i \geq j$ and $A, B, S, T \in D[x]$ satisfying

$$\left.\begin{aligned} &C \equiv AB \pmod{q},\ AS + BT \equiv 1 \pmod{q} \\ &A \equiv \bar{A},\ B \equiv \bar{B},\ S \equiv \bar{S},\ T \equiv \bar{T} \pmod{p}, \\ &\deg(A) = \deg(\bar{A}) \text{ and } A \text{ is monic.} \end{aligned}\right\} \qquad (1)$$

1. Set $i \leftarrow 1$, $q \leftarrow p$, $A \leftarrow \bar{A}$, $B \leftarrow \bar{B}$, $S \leftarrow \bar{S}$, $T \leftarrow \bar{T}$.

2. (Now $q = p^i$; $A, B, S, T \in D[x]$ and the conditions (1) are satisfied.) If $i \geq j$, exit.

3. Set $U \leftarrow (C - AB)/q$. (Since $C \equiv AB \pmod{q}$ we know $U \in D[x]$.) Using Algorithm S, which is described below, with inputs A, B, S, T, U, solve the congruence $AY + BZ \equiv U \pmod{q}$ for $Y, Z \in D[x]$ such that $\deg(Z) < \deg(A)$.

4. Set $A^* \leftarrow A + qZ$, $B^* \leftarrow B + qY$. (Thus

$$\begin{aligned} C - A^*B^* &= C - AB - q(AY + BZ) - q^2YZ \\ &= q(U - AY - BZ) - q^2YZ \\ &\equiv 0 \pmod{q^2}; \end{aligned}$$

furthermore $A^* \equiv A \equiv \bar{A} \pmod{p}$ and $B^* \equiv B \equiv \bar{B} \pmod{p}$; and, since $\deg(Z) < \deg(A)$, $\deg(A^*) = \deg(A) = \deg(\bar{A})$ and $\mathrm{lc}(A^*) = \mathrm{lc}(A)$, so A^* is monic.)

5. Set $U_1 \leftarrow (A^*S + B^*T - 1)/q$. Using Algorithm S with inputs A, B, S, T, U_1, solve the congruence $AY_1 + BZ_1 \equiv U_1 \pmod{q}$ for $Y_1, Z_1 \in D[x]$ such that $\deg(Z_1) < \deg(A)$.

6. Set $S^* \leftarrow S - qY_1$, $T^* \leftarrow T - qZ_1$. (Thus

$$\begin{aligned} A^*S^* + B^*T^* &= A^*(S - qY_1) + B^*(T - qZ_1) \\ &= A^*S + B^*T - q(A^*Y_1 + B^*Z_1) \\ &= 1 + q(U_1 - A^*Y_1 - B^*Z_1) \\ &\equiv 1 + q(U_1 - AY_1 - BZ_1) \pmod{q^2} \\ &\equiv 1 \pmod{q^2}.) \end{aligned}$$

7. Replace i, q, A, B, S, T by $2i, q^2, A^*, B^*, S^*, T^*$ and go to step 2.

Note that we did not need to assume D was a UFD, but only a commutative ring with identity. The algorithm of Section 3.6 also works under this weaker assumption. In Section 5 we shall see applications of these algorithms when D is commutative with identity but fails to be an integral domain.

Algorithm S (Solution of a polynomial equation). Let E be a commutative ring with identity. Given $A, B, S, T, U \in E[x]$ such that $\mathrm{lc}(A)$ is a unit of E and $AS + BT = 1$, this algorithm computes $Y, Z \in E[x]$ such that $AY + BZ = U$ and $\deg(Z) < \deg(A)$.

1. Set $V \leftarrow TU$.
2. Using Algorithm D of Section 1.4, compute $Q, Z \in E[x]$ such that $V = AQ + Z$, $\deg(Z) < \deg(A)$.
3. Set $Y \leftarrow SU + BQ$ and exit. (Then $AY + BZ = A(SU + BQ) + B(TU - AQ) = (AS + BT)U = U$).

In Section 3.9 we shall prove two theorems concerning the uniqueness of the outputs of Algorithms S and Q.

3.8. Finding true factors. Having obtained the mod $m = p^j$ factors $H_1, \ldots, H_r$ of C from the algorithm of Section 3.6, we must now consider each combination of these factors, testing by trial division whether its modulo m product is a true factor. Since we do not know the leading coefficient of the factor, it is necessary to form the factor with leading

coefficient $c = lc(C)$ and attempt to divide it into $C^* = c \cdot C$. If this division successfully yields a factor A^* of C^* then $pp(A^*)$ is a factor of C. Only those combinations with $deg(A^*) \leq \lfloor deg(C)/2 \rfloor$ need be considered. From these considerations we can now see the motivation for the definition given in Section 3.5 of a "factoring set" R for C: a set which contains the coefficients of any factor A^* of C^* for which $deg(A^*) \leq \lfloor deg(C)/2 \rfloor$ and $lc(A^*) | c$.

Algorithm T (Finding true factors by combining modulo m factors). Let D be a UFD, m be an element of D and R be a complete set of residues of m in D. Given a primitive polynomial $C \in D[x]$ and a list of monic polynomials $H_1, \ldots, H_r \in D[x]$, such that

$$C \equiv lc(C) H_1 \cdots H_r \pmod{m}$$

this algorithm obtains irreducible $F_1, \ldots, F_q \in D[x]$ comprising the complete factorization of C:

$$C = F_1 \cdots F_q .$$

The following conditions are assumed to be satisfied.

a. R is a factoring set for C.

b. For each factor F_k there is an index set $I_k \subset \{1, \ldots, r\}$ such that

$$F_k \equiv lc(F_k) \prod_{i \in I_k} H_i \pmod{m} .$$

Remark: In Section 3.9 we shall show that the mod $m = p^j$ factors $H_1, \ldots, H_r$ produced by the algorithm of Section 3.6 satisfy assumption b.

1. Set $q \leftarrow 1$, $d \leftarrow 1$. (d will run through the integers $1, 2, \ldots, \lfloor \deg(C)/2 \rfloor$).
2. Set $c \leftarrow \mathrm{lc}(C)$, $C^* \leftarrow c \cdot C$.
3. (Now we have a, b and
   c. $C_0 \equiv C \prod_{i=1}^{q-1} F_i \pmod m$ where C_0 = initial value of C;
   d. $F_1, \ldots, F_{q-1}$ are irreducible;
   e. $c = \mathrm{lc}(C)$, $C^* = cC$;
   f. C has no factor B such that $0 < \deg(B) < d$.)
   If $d > \lfloor \deg(C)/2 \rfloor$, set $F_q \leftarrow C$ and exit.
4. For each $I \subset \{1, \ldots, r\}$ such that $\sum_{i \in I} \deg(F_i) = d$:
   a. Set $A^* \leftarrow c \prod_{i \in I} H_i \bmod m$, with coefficients in R.
   b. If $A^* \mid C^*$, set $B^* \leftarrow C^*/A^*$ and go to step 6.
5. Set $d \leftarrow d + 1$ and go to step 3.
6. Set $A \leftarrow \mathrm{pp}(A^*)$, $F_q \leftarrow A$, $q \leftarrow q + 1$, $C \leftarrow B^*/\mathrm{lc}(A)$, and delete from $H_1, \ldots, H_r$ those H_i with $i \in I$ (this changes r).

3.9. Correctness of the overall algorithm. To establish the correctness of the overall algorithm we have to prove that the modulo $m = p^j$ factorization obtained by the algorithm of Section 3.6 satisfies the input assumption b of Algorithm 3.8T, namely that to each true factor F_k of C there is some corresponding set of factors H_i in the modulo m factorization.

To this end we first prove:

Theorem S. Under the assumptions of Algorithm 3.7S, the polynomials Y and Z are uniquely determined.

Proof: Let $AY_1 + BZ_1 = U$ with $\deg(Z_1) < \deg(A)$. Then $AY_1 + BZ_1 = AY + BZ$, which may be written

$$A(Y_1 - Y) = B(Z - Z_1). \tag{1}$$

Upon multiplying both sides by T and adding $AS(Z - Z_1)$ to both sides, we obtain

$$A[S(Z - Z_1) + T(Y_1 - Y)] = (AS + BT)(Z - Z_1) = Z - Z_1.$$

Unless the polynomial in brackets is zero, the degree of the product on the left side is $\geq \deg(A)$, since $\mathrm{lc}(A)$ is a unit. But $\deg(Z - Z_1) < \deg(A)$, so we conclude that $Z = Z_1$ and by (1) we then have $A(Y_1 - Y) = 0$, which, with the fact that $\mathrm{lc}(A)$ is a unit, implies $Y_1 = Y$.

The following theorem concerns the uniqueness of the polynomials computed by Algorithm 3.7Q:

Theorem Q. Let D be a commutative ring with identity, p be an element of D which is not a zero-divisor, and j be a positive integer. Let $A, B, A_1, B_1, S, T \in D[x]$ satisfy

a. $A_1 B_1 \equiv AB \pmod{p^j}$

b. $\deg(A_1) = \deg(A)$, $\mathrm{lc}(A_1) = \mathrm{lc}(A) = 1$;

c. $A_1 \equiv A$ and $B_1 \equiv B \pmod{p}$;

d. $AS + BT \equiv 1 \pmod{p}$.

Then $A_1 \equiv A$ and $B_1 \equiv B \pmod{p^j}$.

Proof: From c we have the conclusion when $j = 1$. Let $j > 1$. From a, we have $A_1 B_1 \equiv AB \pmod{p^{j-1}}$, so we may assume by induction that $A_1 \equiv A$ and $B_1 \equiv B \pmod{p^{j-1}}$. Hence there exist $Y, Z \in D[x]$ such that $A_1 = A + p^{j-1}Z$, $B_1 = B + p^{j-1}Y$. Thus

$$A_1 B_1 = AB + p^{j-1}(AY + BZ) + p^{2j-2}YZ,$$

$$0 \equiv p^{j-1}(AY + BZ) \pmod{p^j}.$$

From this congruence and the assumption that p is not a zero-divisor follows

$$AY + BZ \equiv 0 \pmod{p}.$$

Also, by b we have $\deg(Z) < \deg(A)$. Hence by Theorem S applied to the ring $D/(p)$ we have $Y \equiv Z \equiv 0 \pmod{p}$, from which we obtain the conclusion of the theorem.

We are now prepared to prove:

Theorem T. Let D be a UFD and p be an element of D for which $D/(p)$ is a field. Let $C \in D[x]$ for which $p \nmid \mathrm{lc}(C)$ and C

is squarefree mod p. Suppose C has the factorizations

$$C = F_1 \cdots F_q,\ F_i \text{ distinct, irreducible;}$$

$$C \equiv cG_1 \cdots G_r \pmod{p}$$

$$C \equiv cH_1 \cdots H_r \pmod{p^j}$$

$$c = \mathrm{lc}(C);\ G_i,\ H_i \text{ monic;}$$

$$H_i \equiv G_i \pmod{p};$$

and that $\{1, \ldots, r\}$ is the disjoint union of index sets $I_1, \ldots, I_q$ such that

$$F_k \equiv f_k \prod_{i \in I_k} G_i \pmod{p}$$

$$f_k = \mathrm{lc}(F_k)\ .$$

Then also

$$F_k \equiv f_k \prod_{i \in I_k} H_i \pmod{p^j}\ .$$

Proof: Since $p \nmid \mathrm{lc}(C)$, also $p \nmid f_k$ and f_k^{-1} exists mod p. It is easy to show that f_k^{-1} exists mod $m = p^j$ also. Put

$$A \equiv f_k^{-1} F_k \pmod{m}\ ,$$

$$B \equiv C/A \pmod{m}\ ,$$

$$A_1 \equiv \prod_{i \in I_k} H_i \pmod{m}\ ,$$

$$B_1 \equiv c \prod_{i \in I - I_k} H_i \pmod{m}\ , \qquad I = \{1, \ldots, r\}\ .$$

Then we have

a. $A_1 B_1 \equiv AB \pmod{m}$

b. $\deg(A_1) = \deg(A)$, $\mathrm{lc}(A_1) = \mathrm{lc}(A) = 1$

c. $A_1 \equiv A$ and $B_1 \equiv B \pmod{p}$.

d. Since C is squarefree mod p, A and B are relatively prime mod p and there exist S, T such that $AS + BT \equiv 1 \pmod{p}$.

Therefore, by Theorem Q, $A_1 \equiv A$ and $B_1 \equiv B \pmod{m}$ and therefore

$$F_k \equiv f_k \prod H_i \pmod{m},$$

as was to be shown.

4. Application to univariate polynomials over the integers

To obtain algorithms for factoring univariate polynomials over the integers, we take $D = Z$ and choose a prime integer p as the modulus, so that the mod p factorizations are factorizations of polynomials over $E = Z/(p) = GF(p)$ the Galois field of order p. $GF(p)$, and more generally, $Z/(m)$ with $m = p^i$ is conveniently represented by the integers $0, 1, 2, \ldots, m-1$, with arithmetic performed modulo m. The "symmetric residues" $-\lfloor m/2 \rfloor, \ldots, 0, \ldots \lfloor m/2 \rfloor$ (m odd) could also be used. Division of a by b, with b relatively prime to m, can be performed using the Extended Euclidean algorithm to compute multipliers s, t such that $bs + mt = 1$, so that $bs \equiv 1 \pmod m$ and $as = a/b \pmod m$. Arithmetic modulo m is discussed further in [KNU69, §4.6.1] and [COL69].

4.1. Choice of a prime. The first consideration in the choice of a prime p is that the squarefree polynomial $C(x)$ must remain squarefree modulo p. Since C is squarefree, the discriminant of C, $\mathrm{discr}(C)$, is a non-zero integer. Let $\bar{C} = C \bmod p$. If p does not divide $\mathrm{lc}(C)$ then $\mathrm{discr}(\bar{C}) = \mathrm{discr}(C) \bmod p$, so if p is not a divisor of $\mathrm{discr}(C)$ then $\mathrm{discr}(\bar{C}) \neq 0$ and $\bar{C}$ is squarefree. Hence $C \bmod p$ is squarefree for all but a finite number of primes; and in fact, for a given p, $C \bmod p$ is squarefree with probability $1 - 1/p$ [KNU69, Ex. 4.6.2-2]. We can efficiently test whether $\bar{C}$ is squarefree by testing whether $\gcd(\bar{C}, \bar{C}') = 1$ in $GF(p)[x]$, where $\bar{C}'$ is the derivative of $\bar{C}$.

The size of primes used is an important factor in the choice of an algorithm for factoring over GF(p). Trying large primes would reduce the chances of encountering primes for which C mod p has repeated factors, and would also reduce the number of Hensel's construction iterations required to make p^j sufficiently large. However, Berlekamp's 1967 algorithm for complete factorization over GF(p) is efficient only for small primes. The algorithm has two phases. In the first phase the number, r, of irreducible monic factors of the input $\bar{C}$ is determined. If $r > 1$, the second phase is performed to determine the actual factors. The computing time for the first phase is dominated by $n^3(\log p)^2 + n^2(\log p)^3$, where $n = \deg(\bar{C})$ and the time for the second phase by $n^2rp(\log p)^2$. [KNU69, §4.6.2].

A newer algorithm devised by Berlekamp [BER70] is more efficient for large primes, at least in terms of average computing time. This algorithm has an average time dominated by a polynomial function of n and log p but the maximum computing time may be codominant with $n^3p(\log p)^3$.

By contrast, the <u>maximum</u> time for the quadratic Hensel construction to lift a mod p factorization to a mod $m = p^j$ factorization is dominated by $n^2(\log m)^2 + n(\log m)\log c$, where c is the maximum size of coefficients of C [MUS71]. Thus use of Berlekamp's original algorithm with a small prime p followed by Hensel's construction is probably much more efficient than to use Berlekamp's newer algorithm with a large prime.

Another consideration is that the newer algorithm is considerably more complex than the earlier algorithm.

The third major consideration in the choice of a prime p is the number r of factors in the complete mod p factorization. If C is irreducible, but splits into $r > 1$ irreducible factors mod p, then r factors are also obtained mod $m = p^j$ and a total of 2^{r-1} subsets of factors are considered in Algorithm 3.8T. The time for all other phases of the overall algorithm is dominated by a polynomial function of n and $\log c$ (this is shown in [MUS71]), but since r can be as large as n, the time for Algorithm 3.8T can be an exponential function of n. The <u>average</u> time for randomly chosen inputs C (which are almost always irreducible [KNU69, Ex. 4.6.2-27]), is a polynomial function of n and $\log c$, since it can be shown that the average value of r is about $\log n$ and the average value of 2^{r-1} is about $(n+1)/2$. However, the variance of 2^{r-1} is quite large, about $(n^3-n)/24$, causing a large variance in the overall computing time.

In order to reduce this variance one can factor modulo several small primes for which $C \bmod p$ is squarefree and choose a p which yields the smallest number of irreducible factors. Unfortunately, no matter how many primes are used, the maximum computing time will still be exponential in n: H. P. F. Swinnerton-Dyer has shown (see [BER70]) that for any n which is a power of 2, there is an irreducible integral polynomial of degree n which has at least $n/2$ irreducible

factors modulo p for every prime p. By considering the size of the coefficients of these polynomials, the author has established that the computing time of any Berlekamp-Hensel algorithm for factoring integral polynomials cannot be dominated by a polynomial function of n and $\log c$. The same result can also be established using a certain class of cyclotomic polynomials which have more than $(\log n)^2$ factors for every prime.

Finding an algorithm for factoring integral polynomials with a polynomial dominated maximum computing time (or proving no such algorithm can exist) is a very interesting open problem. Nevertheless, it may not be a problem of great practical importance since the average time of the Berlekamp-Hensel algorithm is polynomial dominated.

There are other advantages to be gained from performing factorizations modulo several different primes. By considering the possible <u>degrees</u> of factors in these factorizations, we obtain important information about the degrees of true factors. If C has r irreducible mod p factors, then in a time dominated by rn we can compute the <u>degree set</u> D_p, the set of degrees of all mod p factors. Since the chosen primes do not divide $lc(C)$, the degree set of C must be contained in D_p for any prime p and therefore must be contained in $D_{p_1} \cap D_{p_2} \cap \cdots \cap D_{p_\nu}$, where $p_1, p_2, \ldots, p_\nu$ are the primes tried. In general, this set can be used to eliminate many of the cases that would otherwise have to be considered in Algorithm 3.8T; and in

particular, when C is irreducible, we will often find $D_{p_1} \cap D_{p_2} \cap \cdots = \{0,n\}$ after only a few primes have been tried, thus proving irreducibility of C without having to use the Hensel construction or Algorithm 3.8T at all. The author has verified, by empirical tests, simulations and theoretical analysis, that the average number of primes which must be tried before proving irreducibility is less than 5 for all $n \leq 200$.

This "degree testing algorithm" is a much more efficient means of proving irreducibility than merely searching for a prime p for which C is irreducible mod p. For the probability that a random polynomial C is irreducible modulo a given prime p is only about $1/n$ [KNU69, Ex. 4.6.2-4] and by the Chinese remainder theorem these mod p "trials" are independent, so an average of about n trials would be required to prove irreducibility.

In order to compute the degree set D_p it is not necessary to factor completely mod p. One can use the "distinct degree factorization" algorithm described in [KNU69, p. 389] and [COL69]. Given a monic squarefree polynomial A over GF(p), this algorithm produces a list $L = ((d_1, A_1), \ldots, (d_s, A_s))$ where the d_i are positive integers, $d_1 < d_2 < \cdots < d_s$ and A_i is the product of all monic irreducible factors of A which are of degree d_i. Thus $A = A_1 \cdots A_s$ and this is a complete factorization just in case no two irreducible factors of A have the same degree.

From the list L it is easy to construct a list $\Delta = \{\delta_1, \delta_2, \ldots, \delta_r\}$ of the degrees of all irreducible factors of A, and from Δ one constructs the degree set D of degrees of all factors as follows: put $D = \{0\}$ and for $i = 1, \ldots, r$ replace D by $D \cup \{d + \delta_i : d \in D\}$. As remarked above, the time to compute the degree set from Δ is dominated by rn.

Another way of finding Δ without performing a complete factorization of A would be to perform only the first phase of Berlekamp's newer algorithm, in which a matrix of polynomials is computed in a block diagonal form. If A has r_i irreducible factors of degree i, then there is an $r_i \times r_i$ block of polynomials of degree i. One could compute the determinant of the block, which is the product of all irreducible factors of degree i, and thus obtain the distinct degree factorization. But Δ is determinable directly from the matrix. The computing time of this phase of the algorithm has not been analyzed, but it is possibly faster than the distinct-degree factorization algorithm, whose time is dominated by $n^3(\log p)^2 + n^2(\log p)^3$.

Of course, if the degree tests fail to establish irreducibility, then a prime p is selected among those which yield the smallest number of irreducible factors, and the complete factorization must be obtained for this prime. With the distinct degree factorization, this is accomplished by applying Berlekamp's algorithm to each A_i for which $d_i \neq \deg(A_i)$, the other A_i being irreducible already. If the first phase

of Berlekamp's newer algorithm has been used, then one merely continues with the rest of the algorithm.

4.2. Computing a bound on the coefficients of factors. The height of a polynomial $C(v_1, \ldots, v_s)$ with complex coefficients is defined to be the maximum of the absolute values of the coefficients. We saw in Section 3.5 that in order to fully determine the true factors of $C(x)$ from its modulo p^j factorization it was necessary to have $p^j/2$ larger than the height of any factor of $C(x)$. Thus an a priori bound on the maximum height of factors of $C(x)$ is required. Typically, the height of every factor of $C(x)$ is no larger than the height of $C(x)$ itself, but there exist polynomials with factors of larger height, e.g. $x^3 + x^2 - x - 1 = (x^2+2x+1)(x-1)$. An excellent bound on the height of factors which is based only on the height and degree of $C(x)$ is given by A. O. Gelfond in [GEL60, pp. 135-140]. In fact, Gelfond establishes the bound for multivariate polynomials $C(v_1, \ldots, v_s) = C_1(v_1, \ldots, v_s) \cdots C_m(v_1, \ldots, v_s)$: if n_k is the degree of C in v_k, $n = n_1 + \cdots + n_s$, and $H(C)$ denotes the height of C, then

$$H(C_1) \cdots H(C_m) \leq \left[\frac{(n_1 + 1) \cdots (n_s + 1)}{2^s}\right]^{\frac{1}{2}} 2^n H(C) . \qquad (1)$$

Gelfond shows that this bound is essentially realizable.

In the univariate case, a number of other bounds have been used, as discussed in [MUS71, Section 3.4] and [MIG74]. The bounds discussed in [MUS71], however, require more computation with the coefficients of $C(x)$

than (1), yet generally give a much larger bound. Mignotte [MIG74] improves an earlier theorem of Gelfond which gives a bound similar to the univariate case of (1).

4.3. Hensel's construction. In Algorithm Q, if the coefficients of the initial values of A, B, S, T are chosen in the symmetric residue set $\{-\lfloor p/2 \rfloor, \ldots, 0, \ldots, \lfloor p/2 \rfloor\}$ and in Algorithm S the coefficients of Y and Z are chosen in the symmetric residue set $R_i = \{-\lfloor p^i/2 \rfloor, \ldots, \lfloor p^i/2 \rfloor\}$ then it is easy to show that the coefficients of A, B, S, T lie in R_i. Upon termination, we have $i = 2^k \geq j > 2^{k-1}$ for some k, and if $i > j$, the coefficients of A, B, S, T are larger than necessary, making computations in Algorithm T more expensive than necessary. This may be corrected by modifying steps 2-4 as follows:

2. [Done?] If $i = j$, exit. (This exit is taken only if $j = 1$.)

3. [Compute Y, Z.] if $2i > j$, set $\tilde{q} \leftarrow p^j/q$, $\tilde{A} \leftarrow A \bmod \tilde{q}$, $\tilde{B} \leftarrow B \bmod \tilde{q}$, $\tilde{S} \leftarrow S \bmod \tilde{q}$, $\tilde{T} \leftarrow T \bmod \tilde{q}$, taking the coefficients of $\tilde{A}, \tilde{B}, \tilde{S}, \tilde{T}$, in $R_{\tilde{i}}$. Otherwise, just put $\tilde{q} \leftarrow q$, $\tilde{A} \leftarrow A$, $\tilde{B} \leftarrow B$, $\tilde{S} \leftarrow S$, $\tilde{T} \leftarrow T$. Set $U \leftarrow (C \leftarrow AB)/q$ and apply Algorithm S to $\tilde{A}, \tilde{B}, \tilde{S}, \tilde{T}, U$, obtaining $Y, Z \in Z[x]$ such that $\tilde{A}Y + \tilde{B}Z \equiv U \pmod{\tilde{q}}$ with coefficients in $R_{\tilde{i}}$ and $\deg(Z) < \deg(A)$.

4. [Compute A^*, B^* and check for end.] Set $A^* \leftarrow A + qZ$,

$B^* \leftarrow B + qY$. (Then $C \equiv A^*B^* \pmod{q\tilde{q}}$, $A^* \equiv \bar{A} \pmod{p}$, $B^* \equiv \bar{B} \pmod{p}$, $|lc(A^*)| < p/2$ and the coefficients of A^* and B^* are bounded by $q\tilde{q}/2$.) If $2i > j$ (in which case $q\tilde{q} = p^j$), set $A \leftarrow A^*$, $B \leftarrow B^*$ and exit.

This modification also avoids computing S and T at the last iteration, since they are not used in further computations.

4.4. Finding true factors. There are two modifications of Algorithm 3.8T which can be of very significant benefit in the application to factoring univariate polynomials over the integers. First, one should add as an extra input the set $\mathcal{D} = D_{p_1} \cap D_{p_2} \cap \cdots \cap D_{p_\nu}$ computed from the mod p_i degree sets and append the test "If $d \in \mathcal{D}$, go to step 5." to step 3. This may greatly reduce the number of cases considered.

Secondly, a "trailing coefficient test" should be inserted in step 4a:

"a. Set $t \leftarrow c \prod_{i \in I} tc(H_i) \bmod m$, $t \in R$; if $t \nmid tc(C^*)$ then continue to the next index set. Otherwise, set $A^* \leftarrow \cdots$".

Thus, if t fails to divide $tc(C^*)$, we know A^* cannot divide C^* and the computation of A^* and the trial division of C^* by A^* are skipped.

Except in rare cases, this trailing coefficient test will in fact eliminate most of the computations of A^* that are not true factors and will thus greatly reduce the average computing time of the algorithm.

5. Application to multivariate polynomials over the integers

Suppose now we are given a polynomial $C \in Z[v_1, \ldots, v_n, x]$ to be factored. The most direct way of applying the abstract algorithms of Section 3 to this problem is to take $D = Z[v_1, \ldots, v_n]$ and the modulus $p = v_n - a_n$ for some integer a_n. That is, we evaluate C at $v_n = a_n$, thereby obtaining a polynomial $\bar{C} \in E[x]$, where $E = Z[v_1, \ldots, v_{n-1}]$ ($E = Z$ if $n = 1$). We recursively factor $\bar{C}$, resorting to the univariate algorithm of Section 4 when all of the v_i are eliminated. We then try to lift the factorization of $\bar{C}$ to a corresponding factorization of C modulo $(v_n - a_n)^{j_n}$ where j_n is chosen to exceed the degree of C in v_n. Unfortunately this is not directly possible, since E is not a field and we cannot necessarily find, corresponding to a factorization $\bar{C} = \bar{A}\bar{B}$, multipliers $\bar{S}$ and $\bar{T} \in E[x]$ such that $\bar{A}\bar{S} + \bar{B}\bar{T} = 1$, as is required in the Hensel construction. However, if we back up and take $D = Q(v_1, \ldots, v_n)$, the field of rational functions of $v_1, \ldots, v_n$, we still have $C \in D[x]$, $p = v_n - a_n \in D$ and $\bar{C} = C(v_1, \ldots, v_{n-1}, a_n) \in E[x]$, where now $E = Q(v_1, \ldots, v_{n-1})$ is a field and the Hensel construction can be applied.

The problem with this direct approach is that it requires many rational function computations, which are generally much more expensive than computations with polynomials (because of the gcd computations required to keep results in lowest terms).

Another, probably better approach would be to restrict the coefficient computations to $D = Z[v_1, \ldots, v_n]$ by using a "trial Hensel construction." This construction uses polynomials $\bar{S}, \bar{T} \in E[x]$ and $r \in E = Z[v_1, \ldots, v_{n-1}]$ for which $\bar{A}\bar{S} + \bar{B}\bar{T} = r$, in an attempt to find a factorization $C \equiv AB \pmod{(v_n - a_n)^{j_n}}$, $A, B \in D[x]$ corresponding to a factorization $C \equiv \bar{A}\bar{B} \pmod{v_n - a_n}$. The attempt may fail, but it is not difficult to arrange the computation so that the construction is guaranteed to succeed if $\bar{A}$ and $\bar{B}$ correspond to actual factors of C. This approach has the drawback that the polynomials $\bar{S}$ and $\bar{T}$ must be obtained independently (by a version of the Extended Euclidean Algorithm).

The algorithm to be discussed in this section avoids these problems by using a generalization of Hensel's construction which works simultaneously with several moduli. We take $D = Z[v_1, \ldots, v_n]$ and moduli p (a prime integer) and $v_1 - a_1, \ldots, v_n - a_n$. Thus $\bar{C} = C(a_1, \ldots, a_n, x) \bmod p$ is a univariate polynomial in $E[x]$, where $E = GF(p)$ as in the univariate case. A factorization $\bar{C} = \bar{A}\bar{B}$ can be lifted to a corresponding factorization $C \equiv AB \pmod{p^j, (v_1 - a_1)^{j_1}, \ldots, (v_n - a_n)^{j_n}}$, $A, B \in D[x]$ by the generalized Hensel construction. This construction works entirely with polynomials with integer coefficients (no rational function operations) and increases the moduli quadratically.

The abstract algorithms of Section 3 have been stated only for a single modulus, but we shall describe in this section the changes and additional algorithms necessary to generalize to several moduli.

5.1. Choice of evaluation points. Given $C \in Z[v_1, \ldots, v_n, x]$ we first reduce to the case in which C is primitive and squarefree as in Sections 3.1 and 3.2. Note that for the assumed algorithm for factoring the content of C in $D = Z[v_1, \ldots, v_n]$ we just apply our multivariate algorithm recursively with one fewer variable.

Next we choose integers $a_1, \ldots, a_n$ such that the univariate integral polynomial $\tilde{C}(x) = C(a_1, \ldots, a_n, x)$ has the same degree in x as C and is squarefree. The following algorithm first chooses a_n trying $0, \pm 1, \pm 2, \ldots$ until finding a value such that $A = C(v_1, \ldots, v_{n-1}, a_n, x)$ has the same degree in x as C and is squarefree. In the same way, $a_{n-1}, \ldots, a_1$ are chosen so that the evaluated polynomial remains of the same degree and squarefree at each stage.

1. [Initialize.] Set $\tilde{C} \leftarrow C$, $k \leftarrow n$.

2. [Prepare to choose a_k.] Set $a \leftarrow 0$, $\tilde{c} \leftarrow \mathrm{lc}(\tilde{C})$.

3. [Evaluate $\tilde{c}$ and $\tilde{C}$ at $v_k = a$]. (Now $\tilde{C} \in Z[v_1, \ldots, v_k, x]$, $\deg_x(\tilde{C}) = \deg_x(C)$, $\tilde{C}$ is squarefree, and $\tilde{c} = \mathrm{lc}(\tilde{C}) \in Z[v_1, \ldots, v_k]$.) If $\tilde{c}(v_1, \ldots, v_{k-1}, a) = 0$, go to step 4. Otherwise, set $A \leftarrow \tilde{C}(v_1, \ldots, v_{k-1}, a, x) \in Z[v_1, \ldots, v_{k-1}, x]$ $B \leftarrow \gcd(A, \partial A/\partial x)$. If $\deg_x(B) > 0$ (in which case $\tilde{C}$ is not

squarefree), go to step 4. Otherwise, set $a_k \leftarrow a$, $k \leftarrow k - 1$, $\tilde{C} \leftarrow A$. If $k > 0$, go to step 2; otherwise, exit.

4. [Try again.] If $a > 0$, set $a \leftarrow -a$; otherwise, set $a \leftarrow 1 - a$. Then go back to step 3.

Termination of this algorithm can be shown by considering at step 3 the discriminant of $\tilde{C}$, which is an element of $Z[v_1, \ldots, v_k]$ and can be divisible by only finitely many linear polynomials $v_k - a_k$.

In the ideal case, this algorithm chooses all of the a_i equal to zero and the coefficients of $\tilde{C}$ are just the constant integer terms of the coefficients (in $Z[v_1, \ldots, v_n]$) of C. In case one or more of the a_i cannot be chosen to be zero, we have a problem of potentially serious coefficient growth with each $a_i \neq 0$, and the coefficients of $\tilde{C}$ might be huge. Although this problem has not been analyzed in detail, it seems unlikely that the problem would appear except very rarely.

5.2. Choice of a prime and factorization over GF(p). We can now choose a prime p, put $\bar{C} = \tilde{C} \bmod p$, and factor $\bar{C}$ completely over GF(p), thus obtaining a factorization of C modulo $p, v_1 - a_1, \ldots, v_n - a_n$. In this factorization, however, we again have the problem of the likelihood of there being several factors corresponding to each irreducible factor F_k of C. We can avoid this problem, except in rare cases, by considering the complete factorization of our univariate polynomial $\tilde{C}(x)$. Suppose that the numerical coefficients of each factor F_k of C

are random integers and that the evaluation points $a_1, \ldots, a_n$ result in the factors

$$\tilde{F}_k(x) = F_k(a_1, \ldots, a_n, x)$$

having random integer coefficients. Since almost all polynomials over the integers are irreducible (see [KNU69, Ex. 4.6.2-27]), it is highly probable that

$$\tilde{C} = \tilde{F}_1 \cdots \tilde{F}_q \tag{1}$$

is the complete factorization of $\tilde{C}$. Thus, if we obtain the complete factorization of $\tilde{C}$, we will only rarely have any more factors than in (1), in contrast to the case with complete factorizations over GF(p).

Therefore, our procedure is this: factor $pp(\tilde{C})$ completely, using the univariate case algorithm, obtaining

$$\tilde{C} = \tilde{c} \cdot pp(\tilde{C}) = \tilde{c} \cdot \tilde{C}_1 \cdots \tilde{C}_r, \quad \tilde{c} = \text{content}(\tilde{C}).$$

If $r = 1$, then C is irreducible and we are done. Otherwise, choose a prime p such that $\bar{C} = \tilde{C} \bmod p$ has the same degree as $\tilde{C}$ and is squarefree modulo p. Then, instead of factoring $\bar{C}$ completely over GF(p), just put

$$\begin{aligned} \tilde{G}_k &= \tilde{C}_k \bmod p \\ G_k &\equiv lc(\tilde{G}_k)^{-1} \tilde{G}_k \pmod p \end{aligned}$$

so that

$$\left.\begin{aligned} C &\equiv cG_1 \cdots G_r \pmod{p, v_1 - a_1, \ldots, v_n - a_n} \\ c &= lc(C),\ G_k \text{ monic}, \end{aligned}\right\}. \tag{2}$$

Thus we have only a partial factorization of C modulo $p, v_1 - a_1, \ldots, v_n - a_n$, but one which does satisfy our partition requirement (Section 2, step 3): suppose $\{1, \ldots, q\}$ is the disjoint union of index sets $I_1, \ldots, I_r$ such that

$$F_k \equiv f_k \prod_{i \in I_k} \tilde{C}_i (\text{mod } v_1 - a_1, \ldots, v_n - a_n), \quad f_k \in Z.$$

Then

$$F_k \equiv \text{lc}(F_k) \prod_{i \in I_k} G_i (\text{mod } p, v_1 - a_1, \ldots, v_n - a_n).$$

Thus the factorization (2) can be used as a basis for constructing a mod $\mathcal{M} = (p^j, (v_1 - a_1)^{j_1}, \ldots, (v_n - a_n)^{j_n})$ factorization from which the true factors F_k can be determined, and we can see that by the way (2) was obtained we will usually have only one mod $\mathcal{M}$ factor corresponding to each true factor.

Although the univariate factoring algorithm determined some prime in the process of factoring $\tilde{C}$, it is not necessary to choose p equal to this prime. It is better now to choose a large prime (since we don't have to worry about finding a complete factorization mod p) to reduce the number of Hensel construction iterations. We could, by choosing p large enough, eliminate entirely the phase of the construction which lifts from p to p^j, but this might mean that p would be a multiple precision integer and all of the mod p arithmetic during the other phases of the algorithm would be multiple precision. The best

course would seem to be to choose p as large as possible while constrained to be in single precision integer, on the machine on which the algorithm is implemented.

5.3. Generalized Hensel construction. Algorithm 3.7Q, the Quadratic Hensel construction for a single modulus, may be regarded as an algorithm for lifting a factorization from one residue class ring to another: let

$$E^+ = D/(p^j)$$

$$E = E^+/(p) = D/(p).$$

Given

$$C = \bar{A}\bar{B},\ \bar{A}\bar{S} + \bar{B}\bar{T} = 1,\ \bar{A} \text{ monic in } E[x],$$

Algorithm 3.7Q obtains

$$C = AB,\ AS + BT = 1,\ A \text{ monic in } E^+[x],$$

$$A \equiv \bar{A} \text{ and } B \equiv \bar{B} (\text{mod } p).$$

In this construction D is only required to be a commutative ring with identity, and thus can itself be a residue class ring of the same form as E^+. This suggests we can generalize the construction to any number of moduli. To do so, suppose $p_1, \ldots, p_k \in D$, $j_1, \ldots, j_k$ are positive integers and $m_i = p_i^{j_i}$, $1 \le i \le k$. Define

$$D_0 = D/(p_1, \ldots, p_k)$$

$$D_1 = D/(m_1, p_2, \ldots, p_k)$$

$$D_2 = D/(m_1, m_2, p_3, \ldots, p_k)$$

$$\ldots$$

$$D_k = D/(m_1, \ldots, m_k),$$

and note that $D_i = D_{i+1}/(p_{i+1})$, $0 \le i \le k-1$. Given

$$C = A_0B_0,\ A_0S_0 + B_0T_0 = 1,\ A_0 \text{ monic in } D_0[x],$$

we perform, for $i = 0,1,\ldots,k-1$, Algorithm 3.7Q with $E = D_i$, $E^+ = D_{i+1}$, $p = p_{i+1}$, $j = j_{i+1}$, lifting

$$C = A_iB_i,\ A_iS_i + B_iT_i = 1,\ A_i \text{ monic in } D_i[x],$$

to

$$C = A_{i+1}B_{i+1},\ A_{i+1}S_{i+1} + B_{i+1}T_{i+1} = 1,\ A_{i+1} \text{ monic in } D_{i+1}[x],$$

$$A_{i+1} \equiv A_i \text{ and } B_{i+1} \equiv B_i \pmod{p_{i+1}}.$$

We obtain

$$C = A_kB_k,\ A_kS_k + B_kT_k = 1,\ A_k \text{ monic in } D_k[x]$$

$$A_k \equiv A_0 \text{ and } B_k \equiv B_0 \pmod{p_1, p_2, \ldots, p_k}.$$

To apply this to multivariate factorization, take $D = Z[v_1,\ldots,v_n]$, $k = n+1$, $p_i = v_i - a_i$, $m_i = (v_i - a_i)^{j_i}$ for $1 \le i \le n$, and $p_{n+1} = p$ (prime integer). For simplicity, assume all of the a_i are zero. Thus

$$D_0 = Z[v_1,\ldots,v_n]/(v_1,\ldots,v_n,p) = GF(p)$$

$$D_1 = Z[v_1,\ldots,v_n]/(m_1,v_2,\ldots,v_n,p) = GF(p)[v_1]/(m_1)$$

$$D_2 = Z[v_1,\ldots,v_n]/(m_1,m_2,v_3,\ldots,v_n,p) = GF(p)[v_1,v_2]/(m_1,m_2)$$

$$\ldots$$

$$D_n = Z[v_1,\ldots,v_n]/(m_1,\ldots,m_n,p) = GF(p)[v_1,\ldots,v_n]/(m_1,\ldots,m_n)$$

$$D_{n+1} = Z[v_1,\ldots,v_n]/(m_1,\ldots,m_n,p^j).$$

We thus start with

$$C = A_0B_0,\ A_0S_0 + B_0T_0 = 1,\ A_0 \text{ monic in } GF(p)[x]$$

and finish with

$C = A_{n+1}B_{n+1}$, $A_{n+1}S_{n+1} + B_{n+1}S_{n+1} = 1$, A_{n+1} monic in $Z[v_1, \ldots, v_n]/(m_1, \ldots, m_n, p^j)$

$A_{n+1} \equiv A_0$ and $B_{n+1} \equiv B_0 (\mathrm{mod}\ v_1, \ldots, v_n, p)$.

To simplify the computation at each stage, it is best to reduce C to C_i in $D_i[x]$ initially: Put $C_{n+1} = C \in D_{n+1}[x]$ and for $i = n, \ldots, 0$ let

$$C_i = C_{i+1} \mod p_{i+1} \in D_i[x].$$

Then

$$C_i \equiv C_{i+1} \equiv \cdots \equiv C_{n+1} = C(\mathrm{mod}\ p_{i+1}, \ldots, p_{n+1})$$

hence

$$C_i = C \text{ in } D_i[x],$$

so we can use C_{i+1} in place of C when lifting from

$$C_{i+1} = C_i = A_iB_i \text{ in } D_i[x]$$

to

$$C_{i+1} = A_{i+1}B_{i+1} \text{ in } D_{i+1}[x].$$

For example, it is only necessary to use $C_1 \in D_1[x] = GF(p)[v_1, x]/(m_1)$ when $i = 0$.

Going back to the abstract case, let us write $\mathfrak{D}$ for the domain used in Algorithm 3.7Q, to avoid conflict with our current use of D; thus

$$E^+ = \mathfrak{D}/(p^j)$$

$$E = E^+/(p) = \mathfrak{D}/(p).$$

Now consider the operations performed in Algorithm 3.7Q: these are

operations in $\mathfrak{D}$, so we must consider the structure of $\mathfrak{D}$ in the application of the algorithm with $E^+ = D_{i+1}$, $E = D_i$. We see that we should take

$$\mathfrak{D} = \mathfrak{D}_{i+1} = D/(m_1, \ldots, m_i, p_{i+2}, \ldots, p_k)$$

since this gives

$$\mathfrak{D}_{i+1}/(p_{i+1}) = D/(m_1, \ldots, m_i, p_{i+1}, p_{i+2}, \ldots, p_k) = D_i = E$$

$$\mathfrak{D}_{i+1}/(m_{i+1}) = D/(m_1, \ldots, m_i, m_{i+1}, p_{i+2}, \ldots, p_k) = D_{i+1} = E^+$$

as required.

In the application to $D = Z[v_1, \ldots, v_n]$, we see that we must be prepared to perform Algorithm 3.7Q with coefficient arithmetic in the domains

$$\mathfrak{D}_1 = Z[v_1, \ldots, v_n]/(v_2, \ldots, v_n, p) = GF(p)[v_1]$$

$$\mathfrak{D}_2 = Z[v_1, \ldots, v_n]/(m_1, v_3, \ldots, v_n, p) = GF(p)[v_1, v_2]/(m_1)$$

$$\ldots$$

$$\mathfrak{D}_n = Z[v_1, \ldots, v_n]/(m_1, \ldots, m_{n-1}, p) = GF(p)[v_1, \ldots, v_n]/(m_1, \ldots, m_{n-1})$$

$$\mathfrak{D}_{n+1} = Z[v_1, \ldots, v_n]/(m_1, \ldots, m_n).$$

Since the m_i are powers of the v_i, arithmetic in these domains can be regarded as <u>truncated power series operations</u>. For example, we multiply two elements of $\mathfrak{D}_{n+1}$ with an algorithm which drops terms with degree at least j_i in any variable v_i. This assumes that the evaluation points a_i are all zero; otherwise the reduction $\mathrm{mod}(v_i - a_i)^{j_i}$

would actually require a division by $(v_i - a_i)^{j_i}$. Therefore, if any of the a_i are non-zero, it is probably best to translate the given polynomial $C(v_1, \ldots, v_n, x)$ to

$$C^*(v_1, \ldots, v_n, x) = C(v_1 + a_1, \ldots, v_n + a_n, x) ,$$

and factor C^*. If the complete factorization of C^* is $F_1^* \cdots F_q^*$ then that of C is $F_1 \cdots F_q$, where

$$F_1(v_1, \ldots, v_n, x) = F^*(v_1 - a_1, \ldots, v_n - a_n, x) .$$

5.4. Generalization of other algorithms. The algorithm of Section 3.6, for lifting several factors, may now be generalized to several moduli: all that is required is to substitute the generalized Hensel construction for the single modulus Algorithm Q in step 4. Similarly, Algorithm 3.8T generalizes to several moduli with no difficulty. The theorems of Section 3.9 must also be generalized, but this is also straightforward. The following theorem generalizes Theorem 3.9Q.

Theorem G. Let D be a commutative ring with identity; $p_1, \ldots, p_k$ be elements of D which are not zero-divisors; $m_1 = p_1^{j_1}, \ldots, m_k = p_k^{j_k}$ for some positive integers $j_1, \ldots, j_k$; $p = (p_1, \ldots, p_k)$; $m = (m_1, \ldots, m_k)$. Let $A, B, A_1, B_1 \in D[x]$ satisfy

a. $A_1 B_1 \equiv AB \pmod{m}$;

b. $\deg(A_1) = \deg(A)$ and $\mathrm{lc}(A_1) = \mathrm{lc}(A) = 1$;

c. $A_1 \equiv A$ and $B_1 \equiv B \pmod{p}$;

d. $AS + BT \equiv 1 \pmod{p}$.

Then $A_1 \equiv A$ and $B_1 \equiv B \pmod{m}$.

Proof: By induction on k. If $k = 1$ then the theorem follows from Theorem 3.9Q. Assume $k > 1$ and let

$$D_0 = D/(p_1, \ldots, p_k)$$

$$D_{k-1} = D/(m_1, \ldots, m_{k-1}, p_k)$$

$$D_k = D/(m_1, \ldots, m_k) .$$

Then $D_0 = D_{k-1}/(p_1, \ldots, p_{k-1})$ and by a-d we have, in $D_{k-1}[x]$,

a'. $A_1 B_1 = AB$

b'. $\deg(A_1) = \deg(A)$, $\mathrm{lc}(A_1) = \mathrm{lc}(A) = 1$

c'. $A_1 \equiv A$ and $B_1 \equiv B (\mathrm{mod}\ p_1, \ldots, p_{k-1})$

d'. $AS + BT \equiv 1 (\mathrm{mod}\ p_1, \ldots, p_{k-1})$.

By the induction hypothesis, we therefore have $A_1 = A$ and $B_1 = B$ in $D_{k-1}[x]$. Now since $D_{k-1} = D_k/(p_k)$ we have

$$A_1 \equiv A \quad \text{and} \quad B_1 \equiv B (\mathrm{mod}\ p_k)$$

in $D_k[x]$. The generalized Hensel construction gives S_k and T_k satisfying

$$AS_k + BT_k = 1$$

in $D_k[x]$. We also have a' and b' in $D_k[x]$, so Theorem 3.7Q applies and we conclude that $A_1 = A$ and $B_1 = B$ in $D_k[x]$, as desired.

Now Theorem 3.9T can be generalized by merely substituting $p_1, \ldots, p_k$ and $j_1, \ldots, j_k$ for p and j and $m_1, \ldots, m_k$ for m in its statement and proof, and invoking Theorem G in place of Theorem 3.9Q.

6. Summary and conclusions

By means of abstract algorithms we have presented the algebraic theory underlying the essential steps of a "Berlekamp-Hensel" algorithm for factoring integral polynomials, including squarefree factorization, choice of moduli, Hensel's construction, and searching for true factors. The basic ideas of the univariate case algorithms have appeared previously in the literature but are presented here in greater detail. A practical basis for a multivariate case algorithm is also given in the generalized Hensel construction.

Beyond the algebraic theory, we have gone into detailed consideration of improvements which can be made in the basic algorithms and comparisons between various alternative ways of implementing particular steps. In the univariate case, the most significant of these considerations related to the choice of a prime, the degree compatibility tests, and the trailing coefficient test in the true factor testing algorithm. In the multivariate case, we noted the importance of the translation to make the evaluation points all zero and of the univariate factorization to reduce the number of extraneous factorizations considered.

A number of interesting open problems have been noted, including the existence of an algorithm for factoring integral polynomials with a polynomial bounded computing time, the average computing time of the univariate Berlekamp-Hensel algorithm for classes of reducible polynomials, and the maximum and average computing times of the multivariate algorithm.

Acknowledgements. I am deeply indebted to George E. Collins for his suggestions, guidance and encouragement throughout the course of this work. I wish also to thank the referee for his many detailed suggestions which have led to an improved version of the paper.

REFERENCES

[BER68] E. R. Berlekamp, Algebraic Coding Theory, McGraw-Hill, Inc., New York, 1968.

[BER70] E. R. Berlekamp, Factoring Polynomials over Large Finite Fields, Mathematics of Computation, July, 1970.

[CHA73] R. T. Charleton, A Pascal Implementation of Algorithms for the Factorization and Greatest Common Divisor Calculation of Multivariate Polynomials, Master's Thesis, University of Texas, August, 1973.

[COL69] George E. Collins, L. E. Heindel, E. Horowitz, M. T. McClellan, and D. R. Musser, the SAC-1 Modular Arithmetic System, University of Wisconsin Technical Report No. 10, June, 1969.

[COL71] George E. Collins, the SAC-1 System: an Introduction and Survey, Preceedings of the Second Symposium on Symbolic and Algebraic Manipulation, Los Angeles, March, 1971.

[COL72] George E. Collins and D. R. Musser, the SAC-1 Polynomial Factorization System, Computer Sciences Technical Report No. 157, March, 1972.

[COL73] G. E. Collins, Computer Algebra of Polynomials and Rational Functions, American Mathematical Monthly, Vol. 80, No. 7, August-September, 1973, pp. 725-755.

[FLO67] Robert W. Floyd, Assigning Meanings to Programs, Proceedings of a Symposium in Applied Mathematics, Vol. 19 - Mathematical Aspects of Computer Science (J. T. Schwartz, ed.). American Mathematical Society, Providence, R. I., 1967, pp. 19-32.

[GEL60] A. O. Gelfond, Trancendental and Algebraic Numbers, Dover Publications, New York, 1960.

[HOR69] Ellis Horowitz, Algorithms for Symbolic Integration of Rational Functions, Ph. D. Thesis, Computer Sciences Department, University of Wisconsin, 1969.

[KNU68] Donald E. Knuth, The Art of Computer Programming, Vol. I: Fundamental Algorithms, Addison-Wesley Publishing Co., Reading, Mass., 1968.

[KNU69] Donald E. Knuth, The Art of Computer Programming, Vol. II. Seminumerical Algorithms, Addison-Wesley Publishing Co., Reading, Mass., 1969.

[MIG74] M. Mignotte, An Inequality about Factors of Polynomials, to appear in Mathematics of Computation, 1974.

[MUS71] D. R. Musser, Algorithms for Polynomial Factorization, Computer Sciences Department, Technical Report No. 134 (Ph. D. Thesis), September, 1971.

[RAB60] Michael O. Rabin, Computable Algebra, General Theory and Theory of Computable Fields, Transactions of the American Mathematical Society 95 (1960), pp. 341-360.

[VDW49] B. L. Van der Waerden, Modern Algebra, Vol. 1, trans. by Fred Blum, Frederick Ungar Publishing Co., New York, 1949.

[WAN73] Paul S. Wang and L. Preiss Rothchild, Factoring Multivariate Polynomials Over the Integers, SIGSAM Bulletin No. 28, December, 1973.

[ZAS69] Hans Zassenhaus, On Hensel Factorization, I, Journal of Number Theory 1, 291-311 (1969).