A verified factorization algorithm for integer polynomials with polynomial complexity*

Jose Divasón Sebastiaan Joosten René Thiemann
Akihisa Yamada
April 20, 2020

Abstract
Short vectors in lattices and factors of integer polynomials are related. Each factor of an integer polynomial belongs to a certain lattice. When factoring polynomials, the condition that we are looking for an irreducible polynomial means that we must look for a small element in a lattice, which can be done by a basis reduction algorithm. In this development we formalize this connection and thereby one main application of the LLL basis reduction algorithm: an algorithm to factor square-free integer polynomials which runs in polynomial time. The work is based on our previous Berlekamp–Zassenhaus development, where the exponential reconstruction phase has been replaced by the polynomial-time basis reduction algorithm. Thanks to this formalization we found a serious flaw in a textbook.

Contents
1 Introduction 2
2 Factor bound 5
3 Executable dvdm operation 6
   3.1 Uniqueness of division algorithm for polynomials . . . . . . 7
   3.2 Executable division operation modulo $m$ for polynomials . . 12
4 The LLL factorization algorithm 13
5 Correctness of the LLL factorization algorithm 16
   5.1 Basic facts about the auxiliary functions . . . . . . . . . . . . 16
   5.2 Facts about Sylvester matrices and norms . . . . . . . . . . . . 17

*Supported by FWF (Austrian Science Fund) project Y757. Jose Divasón is partially funded by the Spanish project MTM2017-88804-P.
1 Introduction

In order to factor an integer polynomial \( f \), we may assume a modular factorization of \( f \) into several monic factors \( u_i \): \( f \equiv \text{lcf}(f) \cdot \prod u_i \) modulo \( m \) where \( m = p^l \) is some prime power for user-specified \( l \). In Isabelle, we just reuse our verified modular factorization algorithm [1] to obtain the modular factorization of \( f \).

We briefly explain how to compute non-trivial integer factors of \( f \). The key is the following lemma [2, Lemma 16.20].

**Lemma 1 ([2, Lemma 16.20])** Let \( f, g, u \) be non-constant integer polynomials. Let \( u \) be monic. If \( u \) divides \( f \) modulo \( m \), \( u \) divides \( g \) modulo \( m \), and \( |f|^{\deg(g)} \cdot |g|^{\deg(f)} < m \), then \( h = \gcd(f, g) \) is non-constant.

Let \( f \) be a polynomial of degree \( n \). Let \( u \) be any degree-\( d \) factor of \( f \) modulo \( m \). Now assume that \( f \) is reducible, so \( f = f_1 \cdot f_2 \) where w.l.o.g., we assume that \( u \) divides \( f_1 \) modulo \( m \) and that \( 0 < \deg(f_1) < n \). Let us further assume that a lattice \( L_{u,k} \) encodes the set of all polynomials of
degree below \( d + k \) (as vectors of length \( d + k \)) which are divisible by \( u \) modulo \( m \). Fix \( k = n - d \). Then clearly, \( f_1 \in L_{u,k} \).

In order to instantiate Lemma 1, it now suffices to take \( g \) as the polynomial corresponding to any short vector in \( L_{u,k} \): \( u \) will divide \( g \) modulo \( m \) by definition of \( L_{u,k} \) and moreover \( \deg(g) < n \). The short vector requirement will provide an upper bound to satisfy the assumption 
\[
\|f\|^{\deg(g)} \cdot \|g\|^{\deg(f)} < m.
\]

\[
\|g\| \leq 2^{(n-1)/2} \cdot \|f_1\| \leq 2^{(n-1)/2} \cdot 2^{n-1} \|f\| = 2^{3(n-1)/2} \|f\| \tag{1}
\]

\[
\|f\|^{\deg(g)} \cdot \|g\|^{\deg(f)} \leq \|f\|^{n-1} \cdot (2^{3(n-1)/2} \|f\|)^n = \|f\|^{2n-1} \cdot 2^{3n(n-1)/2} \tag{2}
\]

Here, the first inequality in (1) is the short vector approximation \( (f_1 \in L_{u,k}) \). The second inequality in (1) is Mignotte’s factor bound \( (f_1 \) is a factor of \( f \)).

Finally, (1) is used as an approximation of \( \|g\| \) in (2).

Hence, if \( l \) is chosen large enough so that \( m = p^l > \|f\|^{2n-1} \cdot 2^{3n(n-1)/2} \) then all preconditions of Lemma 1 are satisfied, and \( h = \gcd(f,g) \) will be a non-constant factor of \( f \). Since the degree of \( h \) will be strictly less than \( n \), \( h \) is also a proper factor of \( f \), i.e., in particular \( h \notin \{1, f\} \).

The textbook [2] also describes the general idea of the factorization algorithm based on the previous lemma in prose, and then presents an algorithm in pseudo-code which slightly extends the idea by directly splitting off irreducible factors [2, Algorithm 16.22]. We initially implemented and tried to verify this pseudo-code algorithm (see files Factorization_Algorithm_16_22.thy and Modern_Computer_Algebra_Problem.thy). After some work, we had only one remaining goal to prove: the content of the polynomial \( g \) corresponding to the short vector is not divisible by the chosen prime \( p \). However, we were unable to figure out how to discharge this goal and then also started to search for inputs where the algorithm delivers wrong results. After a while we realized that Algorithm 16.22 indeed has a serious flaw as demonstrated in the upcoming example.

**Example 1** Consider the square-free and content-free polynomial \( f = (1 + x) \cdot (1 + x + x^3) \). Then according to Algorithm 16.22 we determine

- the prime \( p = 2 \)
- the exponent \( l = 61 \)
  (our new formalized algorithm uses a tighter bound which results in \( l = 41 \))
- the leading coefficient \( b = 1 \)
- the value \( B = 96 \)
- the factorization mod \( p \) via \( h_1 = 1 + x \), \( h_2 = 1 + x + x^3 \)
the factorization mod $p^l$ via $g_1 = 1 + x$, $g_2 = 1 + x + x^3$

$f^* = f$, $T = \{1, 2\}$, $G = \emptyset$.

we enter the loop and in the first iteration choose

$u = 1 + x + x^3$, $d = 3$, $j = 4$

we consider the lattice generated by $(1, 1, 0, 1)$, $(p^l, 0, 0, 0)$, $(0, p^l, 0, 0)$, $(0, 0, p^l, 0)$.

now we obtain a short vector in the lattice: $g^* = (2, 2, 0, 2)$.

Note that $g^*$ has not really been computed by Algorithm 16.10, but it satisfies the soundness criterion, i.e., it is a sufficiently short vector in the lattice.

To see this, note that a shortest vector in the lattice is $(1, 1, 0, 1)$.

$$||g^*|| = 2 \cdot \sqrt{3} \leq 2 \cdot \sqrt{2} \cdot \sqrt{3} = 2^{(j-1)/2} \cdot ||(1, 1, 0, 1)||$$

So $g^*$ has the required precision that was assumed by the short-vector calculation.

the problem at this point is that $p$ divides the content of $g^*$. Consequently, every polynomial divides $g^*$ mod $p$. Thus in step 9 we compute $S = T$, $h = 1$, enter the then-branch and update $T = \emptyset$, $G = G \cup \{1 + x + x^3\}$, $f^* = 1$, $b = 1$.

Then in step 10 we update $G = \{1 + x + x^3, 1\}$ and finally return that the factorization of $f$ is $(1 + x + x^3) \cdot 1$.

More details about the bug and some other wrong results presented in the book are shown in the file Modern_Computer_Algebra_Problem.thy.

Once we realized the problem, we derived another algorithm based on Lemma 1, which also runs in polynomial-time, and prove its soundness in Isabelle/HOL. The corresponding Isabelle statement is as follows:

**Theorem 1 (LLL Factorization Algorithm)**

assumes square_free (f :: int poly)

and degree f \neq 0

and LLL_factorization f = gs

shows f = prod_list gs

and \forall g_i \in set_list gs. irreducible g_i

Finally, we also have been able to fix Algorithm 16.22 and provide a formal correctness proof of the the slightly modified version. It can be seen as an implementation of the pseudo-code factorization algorithm given by Lenstra, Lenstra, and Lovász [3].
2 Factor bound

This theory extends the work about factor bounds which was carried out in the Berlekamp-Zassenhaus development.

theory Factor-Bound-2
imports Berlekamp-Zassenhaus.Factor-Bound
  LLL-Basis-Reduction.Norms
begin

lemma norm-1-bound-mignotte: norm1 f ≤ 2^(degree f) * mahler-measure f
proof (cases f = 0)
  case f0: False
  have cf: coeff f = map (λ i. coeff f i) [0 ..< Suc (degree f)] unfolding coeff-def
    using f0 by auto
  have real-of-int (∑ i≤degree f. real (degree f choose i) * mahler-measure f)
    unfolding sum-distrib-right[symmetric] by auto
  also have ... = real (∑ i≤degree f. real (degree f choose i) * mahler-measure f)
    unfolding sum-distrib-right[symmetric] by auto
  finally show ?thesis unfolding norm1-def .
qed (auto simp: mahler-measure-ge-0 norm1-def)

lemma mahler-measure-l2norm: mahler-measure f ≤ sqrt (of-int ∥ f ∥^2)
  using Landau-inequality-mahler-measure[of f] unfolding sq-norm-poly-def
  by (auto simp: power2-eq-square)

lemma sq-norm-factor-bound:
  fixes f h :: int poly
  assumes dvd: h dvd f and f0: f ≠ 0
  shows ∥h∥^2 ≤ 2 ^ (2 ∗ degree h) * ∥f∥^2
proof –
  let ?r = real-of-int
  have h21: ?r ∥h∥^2 ≤ (?r (norm1 h))^2 using norm2-le-norm1-int[of h]
    by (metis of-int-le-iff of-int-power)
  also have ... ≤ (2^(degree h) * mahler-measure h) ^ 2
    unfolding power-mono[OF norm-1-bound-mignotte[of h], of 2]
    by (auto simp: norm1-ge-0)
  also have ... = 2 ^ (2 ∗ degree h) * (mahler-measure h) ^ 2
    unfolding power-mono[of f] unfolding power-ge-0
    by (simp add: power-even-eq power-mult-distrib)
  also have ... ≤ 2 ^ (2 ∗ degree h) * (mahler-measure f) ^ 2
    by (rule mult-left-mono[OF power-mono], auto simp: mahler-measure-ge-0)
  finally show ?thesis
qed
also have \( \ldots \leq 2^2 (2 \ast \text{degree } h) \ast \|f\|^2 \)

proof (rule mult-left-mono)
  have \( \|f\|^2 \geq 0 \) by auto
  from real-sqrt-pow2[OF this]
  show \((\text{mahler-measure } f)^2 \leq \|f\|^2\)
    using power-mono[OF mahler-measure-l2norm[of f], of 2]
    by (auto simp: mahler-measure-ge-0)
qed auto

finally show \( \|h\|^2 \leq 2^2 (2 \ast \text{degree } h) \ast \|f\|^2 \)
unfolding of-int-le-iff .

qed

end

3 Executable dvd operation

This theory contains some results about division of integer polynomials
which are not part of Polynomial_Factorization.Dvd_Int_Poly.thy.

Essentially, we give an executable implementation of division modulo \( m \).

theory Missing-Dvd-Int-Poly
imports
  Berlekamp-Zassenhaus.Poly-Mod-Finite-Field
  Berlekamp-Zassenhaus.Polynomial-Record-Based
  Berlekamp-Zassenhaus.Hensel-Lifting
  Subresultants.Subresultant
  Perron-Frobenius.Cancel-Card-Constraint
begin

lemma degree-div-mod-smult:
  fixes \( g :: \text{int poly} \)
  assumes \( g : \text{degree } g \leq j \)
  and \( r : \text{degree } r \leq d \)
  and \( u : \text{degree } u = d \)
  and \( q1 : g = q \ast u + \text{smult } m \ast r \)
  and \( q : q \neq 0 \) and \( m\text{-not0: } m \neq 0 \)
  shows \( \text{degree } q < j - d \)

proof
  have u-not0: \( u \neq 0 \) using \( u \) \( r \) by auto
  have d-ug: \( d \leq \text{degree } (u \ast q) \) using \( u \) \( \text{degree-mult-right-le[of } q \) by auto
  have j: \( j > \text{degree } (q \ast u + \text{smult } m \ast r) \) using \( q1 \) \( g \) by auto
  have degree \((\text{smult } m \ast r) \leq d \) using degree-smult-eq m-not0 \( r \) by auto
  also have \( \ldots \leq \text{degree } (u \ast q) \) using d-ug by auto
  finally have deg-mr-ug: \( \text{degree } (\text{smult } m \ast r) < \text{degree } (q \ast u) \)
    by (simp add: mult.commute)
  have j2: \( \text{degree } (q \ast u + \text{smult } m \ast r) = \text{degree } (q \ast u) \)
    by (rule degree-add-eq-left[of deg-mr-ug])

end
also have ... = degree q + degree u
by \( \text{rule degree-mult-eq[OF q u-not0]}\)
finally have degree q = degree g - degree u using g1 by auto
thus \?thesis
using j j2 : degree (q * u) = degree q + degree u; u
by linarith
qed

3.1 Uniqueness of division algorithm for polynomials

lemma uniqueness-algorithm-division-poly:
fixes f::'a::{comm-ring,semiring-1-no-zero-divisors} poly
assumes f1: f = g * q1 + r1
and f2: f = g * q2 + r2
and g: g \neq 0
and r1: r1 = 0 \lor degree r1 < degree g
and r2: r2 = 0 \lor degree r2 < degree g
shows q1 = q2 \land r1 = r2
proof -
have 0 = g * q1 + r1 - (g * q2 + r2) using f1 f2 by auto
also have ... = g * (q1 - q2) + r1 - r2
by (simp add: right-diff-distrib)
finally have eq: g * (q1 - q2) = r2 - r1 by auto
have q-eq: q1 = q2
proof (rule ccontr)
assume q1-not-q2: q1 \neq q2
hence nz: g * (q1 - q2) \neq 0 using g by auto
hence degree (g * (q1 - q2)) \geq degree g
by (simp add: degree-mult-right-le)
moreover have degree (r2 - r1) < degree g
using eq nz degree-diff-less r1 r2 by auto
ultimately show False using eq by auto
qed
moreover have r1 = r2 using eq q-eq by auto
ultimately show ?thesis by simp
qed

lemma pdivmod-eq-pdivmod-monic:
assumes g: monic g
shows pdivmod f g = pdivmod-monic f g
proof -
obtain q r where qr: pdivmod f g = (q,r) by simp
obtain Q R where QR: pdivmod-monic f g = (Q,R) by (meson surj-pair)
have g0: g \neq 0 using g by auto
have f1: f = g * q + r
by (metis Pair-inject mult-div-mod-eq qr)
have r: r=0 \lor degree r < degree g
by (metis Pair-inject assms degree-mod-less leading-coeff-0-iff qr zero-neq-one)
have f2: f = g * Q + R

by (simp add: QR assms pdivmod-monic(1))

have R: R=0 ∨ degree R < degree g
  by (rule pdivmod-monic[OF g QR])

have q=Q ∧ r=R by (rule uniqueness-algorithm-division-poly[OF f1 f2 g0 r R])

thus ?thesis using qr QR by auto

qed

definition pdivmod2 f g = (if Mp g = 0 then (0, f)
else let ilc = inverse-p m ((lead-coeff (Mp g)));
    h = Polynomial.smult ilc (Mp g); (q, r) = pseudo-divmod (Mp f) (Mp h)
in (Polynomial.smult ilc q, r))

context poly-mod
begin

definition pdivmod2 f g = (if Mp g = 0 then (0, f)
else let ilc = inverse-p m ((lead-coeff (Mp g)));
    h = Polynomial.smult ilc (Mp g); (q, r) = pseudo-divmod (Mp f) (Mp h)
in (Polynomial.smult ilc q, r))

end

definition pdivmod2 f g = (if Mp g = 0 then (0, f)
else let ilc = inverse-p m ((lead-coeff (Mp g)));
    h = Polynomial.smult ilc (Mp g); (q, r) = pseudo-divmod (Mp f) (Mp h)
in (Polynomial.smult ilc q, r))

end

definition pdivmod2 f g = (if Mp g = 0 then (0, f)
else let ilc = inverse-p m ((lead-coeff (Mp g)));
    h = Polynomial.smult ilc (Mp g); (q, r) = pseudo-divmod (Mp f) (Mp h)
in (Polynomial.smult ilc q, r))

end

context poly-mod-prime-type
begin

lemma dvdm-iff-pdivmod0:
  assumes f: (F :: 'a mod-ring poly) = of-int-poly f
  and g: (G :: 'a mod-ring poly) = of-int-poly g
  shows g dvdm f = (snd (pdivmod F G) = 0)
proof
  have [transfer-rule]: MP-Rel f F unfolding MP-Rel-def
    by (simp add: Mp-f-representative f)
  have [transfer-rule]: MP-Rel g G unfolding MP-Rel-def
    by (simp add: Mp-f-representative g)
  have (snd (pdivmod F G) = 0) = (G dvd F)
    unfolding dvd-eq-mod-eq-0 by auto
  from this [untransferred] show ?thesis by simp
qed

lemma of-int-poly-Mp-0[simp]: (of-int-poly (Mp a) = (0:: 'a mod-ring poly)) = (Mp a = 0)
by (auto, metis Mp-f-representative map-poly-0 poly-mod-Mp-Mp)

lemma uniqueness-algorithm-division-of-int-poly:
  assumes g0: Mp g ≠ 0
  and f: (F :: 'a mod-ring poly) = of-int-poly f
  and g: (G :: 'a mod-ring poly) = of-int-poly g
  and F: F = G * Q + R
  and R: R = 0 ∨ degree R < degree G
  and Mp-f: Mp f = Mp g * q + r
  and r: r = 0 ∨ degree r < degree (Mp g)
  shows Q = of-int-poly q ∧ R = of-int-poly r
proof (rule uniqueness-algorithm-division-poly[OF f - - R])
  have f: Mp f = to-int-poly F unfolding f
    by (simp add: Mp-f-representative)
have \( g' : \text{Mp} \ g = \text{to-int-poly} \ G \) unfolding \( g \)
by (simp add: \( \text{Mp-f-representative} \))

have \( f'' : \text{of-int-poly} \ (\text{Mp} \ f) = F \)
by (metis (no-types, lifting) \( \text{Dp-Mp} \) eq \( \text{Mp-f-representative} \)
\( \text{Mp-smult-m-0} \) add-cancel-left-right \( f \) \map\text{-poly-zero} \( \text{of-int-hom} \) \map\text{-poly-hom-add}

to-int-mod-ring-hom.hom-zero to-int-mod-ring-hom.injectivity)

have \( g'' : \text{of-int-poly} \ (\text{Mp} \ g) = G \)
by (metis (no-types, lifting) \( \text{Dp-Mp} \) eq \( \text{Mp-f-representative} \)
\( \text{Mp-smult-m-0} \) add-cancel-left-right \( g \) \map\text{-poly-zero} \( \text{of-int-hom} \) \map\text{-poly-hom-add}

to-int-mod-ring-hom.hom-zero to-int-mod-ring-hom.injectivity)

have \( \text{of-int-poly} \ (\text{Mp} \ g) = G \)
by (metis (no-types, lifting) \( \text{Dp-Mp} \) eq \( \text{Mp-f-representative} \)
\( \text{Mp-smult-m-0} \) add-cancel-left-right \( g \) \map\text{-poly-zero} \( \text{of-int-hom} \) \map\text{-poly-hom-add}

finally show \( F = \text{of-int-poly} \ (\text{Mp} \ g \ast q + r) \) using \( \text{Mp-f} \) \( f'' \) by auto

show \( \text{of-int-poly} \ r = 0 \lor \text{degree} \ (\text{of-int-poly} \ r :: \text{a mod-ring poly}) < \text{degree} \ G \)

proof (cases \( r = 0 \))
case True
hence \( \text{of-int-poly} \ r = 0 \) by auto
then show \( \text{thesis} \) by auto

next
case False
have \( \text{degree} \ (\text{of-int-poly} \ r :: \text{a mod-ring poly}) \leq \text{degree} \ (r) \)
by (simp add: \( \text{degree-map-poly-le} \))
also have \( ... < \text{degree} \ (\text{Mp} \ g) \) using \( r \) False by auto
also have \( ... = \text{degree} \ G \) by (simp add: \( g'' \))
finally show \( \text{thesis} \) by auto

qed
show \( G \neq 0 \) using \( g0 \) unfolding \( g'' \)\[\text{symmetric} \] by simp

qed

corollary uniqueness-algorithm-division-to-int-poly:
assumes \( g0 : \text{Mp} \ g \neq 0 \)
and \( f : (F :: \text{a mod-ring poly}) = \text{of-int-poly} \ f \)
and \( q : (G :: \text{a mod-ring poly}) = \text{of-int-poly} \ g \)
and \( F : F = G \ast Q + R \)
and \( R : R = 0 \lor \text{degree} \ R < \text{degree} \ G \)
and \( \text{Mp-f} : \text{Mp} \ f = \text{Mp} \ g \ast q + r \)
and \( r : r = 0 \lor \text{degree} \ r < \text{degree} \ (\text{Mp} \ g) \)
shows \( \text{Mp} \ q = \text{to-int-poly} \ Q \land \text{Mp} \ r = \text{to-int-poly} \ R \)
using uniqueness-algorithm-division-of-int-poly[of \( \text{assms} \)]
by (auto simp add: \( \text{Mp-f-representative} \))

lemma uniqueness-algorithm-division-Mp-Rel:
assumes \( \text{monic-Mpg} : \text{monic} \ (\text{Mp} \ g) \)
and \( f : (F :: \text{a mod-ring poly}) = \text{of-int-poly} \ f \)
and \( g : (G :: \text{a mod-ring poly}) = \text{of-int-poly} \ g \)
and \( q r : \text{pseudo-divmod} \ (\text{Mp} \ f) \ (\text{Mp} \ g) = (q,r) \)
and QR: pseudo-divmod F G = (Q,R)

**shows** MP-Rel q Q ∧ MP-Rel r R

**proof** (unfold MP-Rel-def, rule uniqueness-algorithm-division-to-int-poly[OF - f g])

- **show** f-gq-r: Mp f = Mp g * q + r
  - by (rule pdivmod-monic(1)[OF monic-Mpg], simp add: pdivmod-monic-pseudo-divmod qr monic-Mpg)
  - have monic-G: monic G using monic-Mpg
    - using Mp-f-representative g by auto
  - show F = G * Q + R
    - by (rule pdivmod-monic(1)[OF monic-G], simp add: pdivmod-monic-pseudo-divmod QR monic-G)
  - show Mp g ≠ 0 using monic-Mpg by auto
  - show R = 0 ∨ degree R < degree G
    - by (rule pdivmod-monic(2)[OF monic-G], auto simp add: pdivmod-monic-pseudo-divmod monic-G intro: QR)
  - show r = 0 ∨ degree r < degree (Mp g)
    - by (rule pdivmod-monic(2)[OF monic-Mpg], auto simp add: pdivmod-monic-pseudo-divmod monic-Mpg intro: qr)

**qed**

**definition** MP-Rel-Pair A B ≡ (let (a, b) = A; (c, d) = B in MP-Rel a c ∧ MP-Rel b d)

**lemma** pdivmod2-rel[transfer-rule]:

(MP-Rel ===⇒ MP-Rel-Pair) (pdivmod2) (pdivmod)

**proof** (auto simp add: rel-fun-def MP-Rel-Pair-def)

- interpret pm: prime-field m
  - using m unfolding prime-field-def mod-ring-locale-def by auto
  - have p: prime-field TYPE(′a) m
    - using m unfolding prime-field-def mod-ring-locale-def by auto
  - fix f g a b
  - assume 1[transfer-rule]: MP-Rel f F
    - and 2[transfer-rule]: MP-Rel g G
    - and 3: pdivmod2 f g = (a, b)
  - have MP-Rel a (F div G) ∧ MP-Rel b (F mod G)
  - proof (cases Mp g ≠ 0)
    - case True note Mp-g = True
      - have G: G ≠ 0 using Mp-g 2 unfolding MP-Rel-def by auto
      - have gG[transfer-rule]: pm.mod-ring-rel (lead-coeff (Mp g)) (lead-coeff G)
        - using 2
        - unfolding pm.mod-ring-rel-def MP-Rel-def by auto
      - have [transfer-rule]: (pm.mod-ring-rel ===⇒ pm.mod-ring-rel) (inverse-p m)
        - inverse
          - by (rule prime-field.mod-ring-inverse[OF p])
      - hence rel-inverse-p[transfer-rule]:
        - pm.mod-ring-rel (inverse-p m ((lead-coeff (Mp g))) (inverse (lead-coeff G))
        - using gG unfolding rel-fun-def by auto
let \(?h\) = \((\text{Polynomial.smult} \ (\text{inverse-p} \ m \ (\text{lead-coeff} \ (\text{Mp} \ g)))) \ g)\)
define \(h\) where \(h = \text{Polynomial.smult} \ (\text{inverse-p} \ m \ (\text{lead-coeff} \ (\text{Mp} \ g)))\)

define \(H\) where \(H = \text{Polynomial.smult} \ (\text{inverse} \ (\text{lead-coeff} \ (\text{Mp} \ g)))\)

have \(hH' : \text{MP-Rel} \ ?h \ H\) unfolding \(\text{MP-Rel-def}\) unfolding \(H\) by \((\text{metis} \ (\text{mono-tags, hide-lams}) \ 2 \ \text{MP-Rel-def} \ \text{M-to-int-mod-ring} \ \text{Mp-f-representative})\)

rel-inverse-p functional-relation left-total-\(\text{MP-Rel}\) of-int-hom.map-poly-hom-smult

\(\text{pm.mod-ring-rel-def right-unique-\(\text{MP-Rel}\) to-int-mod-ring-hom.injectivity to-int-mod-ring-of-int-M}\)

have \(\text{Mp} \ (\text{Polynomial.smult} \ (\text{inverse-p} \ m \ (\text{lead-coeff} \ (\text{Mp} \ g)))) \ g\) = \(\text{Mp} \ (\text{Polynomial.smult} \ (\text{inverse-p} \ m \ (\text{lead-coeff} \ (\text{Mp} \ g)))) \ (\text{Mp} \ g)\) by \(\text{simp}\)

hence \(hH : \text{MP-Rel} \ h \ H\) using \(hH' \ h\) unfolding \(\text{MP-Rel-def}\) by \(\text{auto}\)

obtain \(q \ x\) where \(\text{pseudo-fh: pseudo-divmod} \ (\text{Mp} \ f) \ (\text{Mp} \ h) = (q, x)\) by \((\text{meson} \ \text{surj-pair})\)

hence \(\text{lc-G:} \ (\text{lead-coeff} \ (\text{G})) \neq 0\) using \(G\) by \(\text{auto}\)

have \(a : a = \text{Polynomial.smult} \ (\text{inverse-p} \ m \ ((\text{lead-coeff} \ (\text{Mp} \ g)))) \ q\)

using 3 pseudo-fh \(\text{Mp-g}\)

unfolding \(\text{pdivmod2-def}\) \(\text{Let-def} \ h\) by \(\text{auto}\)

have \(b : b = x\) using 3 pseudo-fh \(\text{Mp-g}\)

unfolding \(\text{pdivmod2-def}\) \(\text{Let-def} \ h\) by \(\text{auto}\)

have \(\text{Mp-Rel-FH:} \ \text{MP-Rel} \ q \ (F \ \text{div} \ H) \ \land \ \text{MP-Rel} \ x \ (F \ \text{mod} \ H)\)

proof \((\text{rule} \ \text{uniqueness-algorithm-division-\(\text{MP-Rel}\)})\)

show monic \((\text{Mp} \ h)\)

proof
  have \(\text{aux:} \ (\text{inverse-p} \ m \ (\text{lead-coeff} \ (\text{Mp} \ g))) = \text{to-int-mod-ring} \ (\text{inverse} \ (\text{lead-coeff} \ (\text{G})))\)
    using \(\text{rel-inverse-p}\) unfolding \(\text{pm.mod-ring-rel-def}\) by \(\text{auto}\)

hence \(M : \text{invertible} \ M \ (\text{inverse-p} \ m \ (\text{lead-coeff} \ (\text{Mp} \ g))))\)

by \((\text{simp add:} \ \text{M-to-int-mod-ring} \ \text{Mp-coeff})\)

thus \(?\text{thesis}\) unfolding \(h\) unfolding \(\text{Mp-coeff}\) by \(\text{auto}\)

(by \((\text{metis} \ (\text{no-types, lifting}) \ 2 \ \text{H} \ \text{M-to-int-mod-ring} \ \text{Mp-coeff} \ \text{aux} \ \text{degree-smult-eq} \ gG)\)

\(hH'\)

inverse-zero-imp-zero \(\text{lc-G}\) left-inverse \(\text{pm.mod-ring-rel-def to-int-mod-ring-hom.injectivity}\)

to-int-mod-ring-hom.map-poly-hom.smult

to-int-mod-ring-times)

qed

hence monic-H: monic \(H\) using \(hH \ H\) \(\text{le-G}\) by \(\text{auto}\)

show \(f : F = \text{of-int-poly} \ f\)

using 1 unfolding \(\text{MP-Rel-def}\)

by \((\text{simp add:} \ \text{Mp-f-representative} \ \text{poly-eq-iff})\)

have \(\text{pdivmod} \ F \ H = \text{pdivmod-monic} \ F \ H\)

by \((\text{rule} \ \text{pdivmod-eq-pdivmod-monic} \ [OF \ \text{monic-H}])\)

also have \(...) = \text{pseudo-di} \ F \ H\)

by \((\text{rule} \ \text{pdivmod-monic-pseudo-di} \ [OF \ \text{monic-H}])\)

finally show \(\text{pseudo-di} \ F \ H = (F \ \text{div} \ H, F \ \text{mod} \ H)\) by \(\text{simp}\)

show \(\text{H} = \text{of-int-poly} \ h\)

by \((\text{meson} \ \text{MP-Rel-def} \ \text{Mp-f-representative} \ \text{hH right-unique-\(\text{MP-Rel}\) right-unique-def})\)

11
show pseudo-divmod (Mp f) (Mp h) = (q, x) by (rule pseudo-fh)

qed

hence Mp-Rel-F-div-H: MP-Rel q (F div H) and Mp-Rel-F-mod-H: MP-Rel x (F mod H) by auto

have F div H = Polynomial.smult (lead-coeff G) (F div G)
  unfolding H using div-smult-right[OF lc-G] inverse-inverse-eq
  by (metis div-smult-right inverse-zero)

hence F-div-G: (F div G) = Polynomial.smult (inverse (lead-coeff G)) (F div H)
  unfolding MP-Rel-def a F-div-G Mp-f-representative
  by auto

thus ?thesis using Mp-Rel-F-div-H unfolding MP-Rel-def a F-div-G Mp-f-representative by auto

qed

moreover have MP-Rel b (F mod G)
  using Mp-Rel-F-mod-H b H inverse-zero-imp-zero lc-G
  by (metis mod-smult-right)

ultimately show ?thesis by auto

next

assume Mp-g-0: ¬ Mp g ≠ 0

hence pdivmod2 f g = (0, f) unfolding pdivmod2-def by auto

hence a: a = 0 and b: b = f using 3 by auto

have G0: G = 0 using Mp-g-0 2 unfolding MP-Rel-def by auto

have MP-Rel a (F div G) unfolding MP-Rel-def G0 a by auto

moreover have MP-Rel b (F mod G) using 1 unfolding MP-Rel-def G0 a b
  by auto

ultimately show ?thesis by simp

qed

thus MP-Rel a (F div G) and MP-Rel b (F mod G) by auto

qed

3.2 Executable division operation modulo \( m \) for polynomials

lemma dvdn-iff-Mp-pdivmod2:
  shows \( g \ dvdn f = (M p (snd (pdivmod2 f g)) = 0) \)

proof –

let ?F = (of-int-poly f)::'a mod-ring poly

let ?G = (of-int-poly g)::'a mod-ring poly

have a[transfer-rule]: MP-Rel f ?F
  by (simp add: MP-Rel-def Mp-f-representative)
have \( b \) [transfer-rule]: \( MP-Rel \ g \ ?G \)
   by (simp add: MP-Rel-def Mp-f-representative)
have \( MP-Rel-Pair \ (pdivmod2 \ f \ g) \ (pdivmod \ ?F \ ?G) \)
   using pdivmod2-rel unfolding rel-fun-def using a b by auto
hence \( MP-Rel \ (snd \ (pdivmod2 \ f \ g)) \ (snd \ (pdivmod \ ?F \ ?G)) \)
   unfolding MP-Rel-Pair-def by auto
hence \((Mp \ (snd \ (pdivmod2 \ f \ g)) = 0) = (snd \ (pdivmod \ ?F \ ?G) = 0)\)
   unfolding MP-Rel-def by auto
thus \(?thesis\) using dvdm-iff-pdivmod0 by auto
qed

end

lemmas (in poly-mod-prime) dvdm-pdivmod = poly-mod-prime-type.dvdm-iff-Mp-pdivmod2
[unfolded poly-mod-type-simps, internalize-sort 'a :: prime-card, OF type-to-set, unfolded remove-duplicate-premise, cancel-type-definition, OF non-empty]

lemma (in poly-mod) dvdm-code:
  \( g \ dvdm \ f = \text{if \ prime \ } m \ \text{then} \ Mp \ (snd \ (pdivmod2 \ f \ g)) = 0 \)
  else Code.abort (STR "dvdm error: \( m \) is not a prime number") (\( \lambda \ -. \ g \ dvdm \ f \))
   using poly-mod-prime.dvdm-pdivmod[unfolded poly-mod-prime-def]
   by auto

declare poly-mod.pdivmod2-def[code]
declare poly-mod.dvdm-code[code]

end

4 The LLL factorization algorithm

This theory contains an implementation of a polynomial time factorization algorithm. It first constructs a modular factorization. Afterwards it recursively invokes the LLL basis reduction algorithm on one lattice to either split a polynomial into two non-trivial factors, or to deduce irreducibility.

theory LLL-Factorization-Impl
  imports LLL-Basis-Reduction.LLL-Certification
          Factor-Bound-2
          Missing-Dvd-Int-Poly
          Berlekamp-Zassenhaus.Berlekamp-Zassenhaus
begin

hide-const (open) up-ring.coef up-ring.monom
Unique-Factorization.factor Divisibility.factor
Unique-Factorization.factor Divisibility.factor
 Divisibility.prime
definition factorization-lattice where factorization-lattice u k m ≡
map (λi. vec-of-poly-n (u * monom 1 i) (degree u + k)) [k:>..0] @
map (λi. vec-of-poly-n (monom m i) (degree u + k)) [degree u >..0]

fun min-degree-poly :: int poly ⇒ int poly ⇒ int poly
where min-degree-poly a b = (if degree a ≤ degree b then a else b)

fun choose-u :: int poly list ⇒ int poly
where choose-u [] = undefined
| choose-u [gi] = gi
| choose-u (gi # gj # gs) = min-degree-poly gi (choose-u (gj # gs))

lemma factorization-lattice-code[code]:
factorization-lattice u k m = (let n = degree u in
map (λi. vec-of-poly-n (monom-mult i u) (n+k)) [k:>..0]
@ map (λi. vec-of-poly-n (monom m i) (n+k)) [n:>..0])
) unfolding factorization-lattice-def monom-mult-def
by (auto simp: ac-simps Let-def)

Optimization: directly try to minimize coefficients of polynomial u.

definition LLL-short-polynomial where
LLL-short-polynomial pl n u = poly-of-vec (short-vector-hybrid 2 (factorization-lattice
(poly-mod.inv-Mp pl (poly-mod.Mp pl u)) (n - degree u) pl))

locale LLL-implementation =
fixes p pl :: int
begin

function LLL-many-reconstruction where
LLL-many-reconstruction f us = (let
d = degree f;
d2 = d div 2;
f2-opt = find-map-filter
(λ u. gcd f (LLL-short-polynomial pl (Suc d2) u))
(λ f2. let deg = degree f2 in deg > 0 ∧ deg < d)
(filter (λ u. degree u ≤ d2) us)
in case f2-opt of None ⇒ [f]
| Some f2 ⇒ let f1 = f div f2;
(us1, us2) = List.partition (λ gi. poly-mod.dvdm p gi f1) us
in LLL-many-reconstruction f1 us1 @ LLL-many-reconstruction f2 us2)
by pat-completeness auto

termination
proof (relation measure (λ (f,us). degree f), goal-cases)
case (3 f us d d2 f2-opt f2 f1 pair us1 us2)
from find-map-filter-Some\[\text{OF } 3(4)\\{\text{unfolded } 3(3) \text{ Let-def}\}\] 3(1,5)
show \(?\text{case by auto}\)
next
\begin{verbatim}
case (2 \(f \ \text{us d d2 \(f2\)-opt \(f2\) \(f1\) pair \(us1\) \(us2\)})
from find-map-filter-Some\[\text{OF } 2(4)\\{\text{unfolded } 2(3) \text{ Let-def}\}\] 2(1,5)
have \(f: f1 \ast f2\) \text{ and } \(f0: f \neq 0\) \text{ by auto}
and \(\deg: \deg f2 > 0\ \deg f2 < \deg f\) \text{ by auto}
have \(\deg f = \deg f1 + \deg f2\) \text{ using } \(f0\) unfolding \(f\)
by (subst degree-mult-eq, auto)
with \(\deg\) show \(?\text{case by auto}\)
qed auto
\end{verbatim}

function \(\text{LLL-reconstruction}\) where
\(\text{LLL-reconstruction } f \ \text{us} = (\text{let})\)
\begin{verbatim}
\(d = \deg f;\)
\(u = \text{choose-u us};\)
\(g = \text{LLL-short-polynomial pl d u};\)
\(f2 = \text{gcd } f \ g;\)
\(\deg = \deg f2\)
in if \(\deg = 0\ \lor \deg \geq d\) then \([f]\)
else let \(f1 = f \ \text{div } f2;\)
\((\text{us1, us2}) = \text{List.partition (\(\lambda \ gi. \ \text{poly-mod dvdn p gi f1}\)) us}\)
in \(\text{LLL-reconstruction } f1 \ \text{us1} @ \text{LLL-reconstruction } f2 \ \text{us2})\)
by pat-completeness auto
\end{verbatim}
termination
proof (relation measure \((\lambda (f,us). \deg f), \text{goal-cases}\))
\begin{verbatim}
case (2 \(f \ \text{us d u g f2 deg f1 pair \(us1\) \(us2\)})
hence \(f: f1 \ast f2\) \text{ and } \(f0: f \neq 0\) \text{ by auto}
have \(\deg: \deg f = \deg f1 + \deg f2\) \text{ using } \(f0\) unfolding \(f\)
by (subst degree-mult-eq, auto)
from 2 have \(\deg f2 > 0\ \deg f2 < \deg f\) \text{ by auto}
thus \(?\text{case using } \deg\) by auto
qed auto
end
\end{verbatim}
declare \(\text{LLL-implementation.LLL-reconstruction.simps[code]}\)
declare \(\text{LLL-implementation.LLL-many-reconstruction.simps[code]}\)

definition \(\text{LLL-factorization :: int poly } \Rightarrow \text{ int poly list}\) where
\(\text{LLL-factorization } f = (\text{let})\)
\begin{verbatim}
— find suitable prime
\(p = \text{suitable-prime-bz } f;\)
— compute finite field factorization
\((*, \text{fs}) = \text{finite-field-factorization-int } p \ f;\)
— determine exponent \(l\) and \(B\)
\(n = \deg f;\)
\(no = ||f||^2;\)
\(B = \text{sqrt-int-ceiling } (2^*(5 \ast (n - 1) \ast (n - 1)) \ast no^*(2 \ast (n - 1)))\);
\end{verbatim}
\( l = \text{find-exponent } p \ B; \)
— perform hensel lifting to lift factorization to mod \( p^l \)
\( us = \text{hensel-lifting } p \ l \ f \ fs; \)
— reconstruct integer factors via LLL algorithm
\( pl = p^l \)
in LLL-implementation.LLL-reconstruction \( p \ pl \ f \ us \))

definition \texttt{LLL-many-factorization :: int poly \Rightarrow int poly list} where
\texttt{LLL-many-factorization } \( f = (\text{let} \)
— find suitable prime
\( p = \text{suitable-prime-bz } f; \)
— compute finite field factorization
\( (-, fs) = \text{finite-field-factorization-int } p \ f; \)
— determine exponent \( l \) and \( B \)
\( n = \text{degree } f; \)
\( \text{no} = \| f \|^2; \)
\( B = \text{sqrt-int-ceiling } (2^5 \times (n \div 2) \times (n \div 2)) \times \text{no} \times (2 \times (n \div 2)); \)
\( l = \text{find-exponent } p \ B; \)
— perform hensel lifting to lift factorization to mod \( p^l \)
\( us = \text{hensel-lifting } p \ l \ f \ fs; \)
— reconstruct integer factors via LLL algorithm
\( pl = p^l \)
in LLL-implementation.LLL-many-reconstruction \( p \ pl \ f \ us \))
end

5 Correctness of the LLL factorization algorithm

This theory connects short vectors of lattices and factors of polynomials. From this connection, we derive soundness of the lattice based factorization algorithm.

theory \texttt{LLL-Factorization}
imports
    \texttt{LLL-Factorization-Impl}
    \texttt{Berlekamp-Zassenhaus.Factorize-Int-Poly}
begin

5.1 Basic facts about the auxiliary functions

lemma \texttt{nth-factorization-lattice}:
fixes \( u \) and \( d \)
defines \( n \equiv \text{degree } u \)
assumes \( i < n + d \)
shows \( \text{factorization-lattice } u \ d \ m \ ! \ i = \text{vec-of-poly-n } (\text{if } i < d \text{ then } u \times \text{monom } 1 \ (d - \text{Suc } i) \text{ else } \text{monom } m \ (n + d - \text{Suc } i)) \ (n + d) \)
using \texttt{assms}
by (\texttt{unfold factorization-lattice-def, auto simp: nth-append smult-monom Let-def not-less})

\begin{verbatim}
lemma length-factorization-lattice[simp]:
  shows length \((\text{factorization-lattice } u \ d \ m)\) = degree \(u + d\)
by (auto simp: factorization-lattice-def Let-def)

lemma dim-factorization-lattice:
  assumes \(x < \text{degree } u + d\)
  shows \(\text{dim-vec } (\text{factorization-lattice } u \ d \ m \ ! x) = \text{degree } u + d\)
  unfolding factorization-lattice-def using assms nth-append
  by (simp add: nth-append Let-def)

lemma dim-factorization-lattice-element:
  assumes \(x \in \text{set } (\text{factorization-lattice } u \ d \ m)\)
  shows \(\text{dim-vec } x = \text{degree } u + d\)
  using assms
  by (auto simp: factorization-lattice-def Let-def)

lemma set-factorization-lattice-in-carrier[simp]: \(\text{set } (\text{factorization-lattice } u \ d \ m) \subseteq \text{carrier-vec } (\text{degree } u + d)\)
  using dim-factorization-lattice
  by (auto simp: factorization-lattice-def Let-def)

lemma choose-u-Cons: \(\text{choose-u } (x \# xs) = \)
  (if \(xs = []\) then \(x\) else \(\text{min-degree-poly } x \ (\text{choose-u } xs)\))
  by (cases xs, auto)

lemma choose-u-member: \(xs \neq [] \implies \text{choose-u } xs \in \text{set } xs\)
  by (induct xs, auto simp: choose-u-Cons)

declare \text{choose-u.simps}[simp del]
\end{verbatim}

5.2 Facts about Sylvester matrices and norms

\begin{verbatim}
lemma (in LLL) lattice-is-span[simp]: \(\text{lattice-of } xs = \text{span-list } xs\)
by (unfold lattice-of-def span-list-def lincomb-list-def image-def, auto)

lemma sq-norm-row-sylvester-mat1:
  fixes \(f \ g :: 'a :: \text{conjugatable-ring poly}\)
  assumes \(i: i < \text{degree } g\)
  shows \(
  \|
  \text{row } (\text{sylvester-mat } f \ g) \ i\n  \|^2
  = \|f\|^2
  \)
  proof (cases \(f = 0\))
  case True
  thus \(?\text{thesis}\)
  by (auto simp add: sylvester-mat-def row-def sq-norm-vec-def o-def
  inter-sum-list-conv-sum-set-nat i intro!: sum-list-zero)

  next
  case False note \(f = \text{False}\)
  let \(?f = \lambda j. \text{if } i \leq j \land j - i \leq \text{degree } f \text{ then } \text{coeff } f \ (\text{degree } f + i - j) \text{ else } 0\)
  let \(?h = \lambda j. j + i\)
\end{verbatim}
let \( \delta \cdot \text{vec} (\text{degree } f + \text{degree } g) \cdot \delta \)
let \( \gamma = \lambda j. \text{degree } f - j \)
have \( \text{image-g: } \exists g \cdot \{0..<\text{Suc } (\text{degree } f)\} = \{0..<\text{Suc } (\text{degree } f)\} \)
  by (auto simp add: image-def)
  (metis (no-types, hide-lams) Nat.add-diff-assoc add.commute add-diff-cancel-left)
also have \( \ldots \) by (rule arg-cong)
by (auto simp add: inj-on-def)

lemma \( \text{sq-norm-row-sylvester-mat2:} \)
fixes \( f \cdot g \cdot a \cdot a \cdot \text{conjugatable-ring poly} \)
assumes \( i1: \text{degree } g \leq i \) and \( i2: i < \text{degree } f + \text{degree } g \)
shows \( \| \text{row } (\text{sylvester-mat } f \cdot g) \| ^2 = \| g \| ^2 \)
proof -
let \( \delta f = \lambda j. \text{if } i - \text{degree } g \leq j \land j \leq i \text{ then coeff } g (i - j) \text{ else } 0 \)
let \( \delta \cdot \text{vec} (\text{degree } f + \text{degree } g) \cdot \delta \)
let \( \delta h = \lambda j. j + i - \text{degree } g \)
let \( \delta g = \lambda j. \text{degree } g - j \)
have \( \text{image-g: } \exists g \cdot \{0..<\text{Suc } (\text{degree } g)\} = \{0..<\text{Suc } (\text{degree } g)\} \)
  by (auto simp add: image-def)
also have \( \ldots \) by (auto simp add: inj-on-def)
(metis atLeastLessThan_iff diff-diff-cancel diff-le-self less-Suc-le zero-le)

have \( x: x - (i - \text{degree } g) \leq \text{degree } g \) if \( x < \text{Suc } i \) for \( x \) using \( x \) by auto

have bij-h: bij-betw ?h \( \{0..<\text{Suc } (\text{degree } g)\} \{i - \text{degree } g..<\text{Suc } i\} \)

unfolding bij-betw_def inj-on_def using \( i2 \) unfolding image-def
by (auto, metis (no-types) Nat.add-diff-assoc atLeastLessThan_iff \( x \) less-Suc-le)

less-eq-nat.simps(1) ordered-cancel-comm-monoid-diff-class.diff-add)

have \( \|\text{row } (\text{sylvestermat } f \ g) \ i\| = \|\text{row } i\|^2 \)
  by (rule arg-cong[of - - sq-norm-vec], insert i1 i2,
    auto simp add: row-def sylvestermat_def sylvestermat-sub-def)

also have \( \ldots = \sum \text{list } (\text{map } (\text{sq-norm } o \ ?f) [0..<\text{degree } f + \text{degree } g]) \)
  unfolding sq-norm-vec_def by auto

also have \( \ldots = \sum (\text{sq-norm } o \ ?f) \{0..<\text{degree } f + \text{degree } g\} \)
  unfolding interw-sum-list-conv-sum-set-nat by auto

also have \( \ldots = \sum (\text{sq-norm } o \ ?f) \{i - \text{degree } g..<\text{Suc } i\} \)
  by (rule sum_mono_neutral_right, insert i2, auto)

also have \( \ldots = \sum (\text{sq-norm } o \ ?f) \{0..<\text{Suc } (\text{degree } g)\} \)
  by (unfold o_def, rule sum.reindex-bij_betw[symmetric, OF bij-h])

also have \( \ldots = \sum (\lambda j. \text{sq-norm } (\text{coeff } g (\text{degree } g - j))) \{0..<\text{Suc } (\text{degree } g)\} \)

by (rule sum.cong, insert i1, auto)

also have \( \ldots = \sum (\lambda j. \text{sq-norm } (\text{coeff } g j)) \circ \ ?g \{0..<\text{Suc } (\text{degree } g)\} \)
  unfolding o_def ..

also have \( \ldots = \sum (\lambda j. \text{sq-norm } (\text{coeff } g j)) (\ ?g \{0..<\text{Suc } (\text{degree } g)\}) \)
  by (rule sum.reindex[symmetric], auto simp add: inj_on_def)

also have \( \ldots = \sum (\text{sq-norm } o \ \text{coeff } g) \{0..<\text{Suc } (\text{degree } g)\} \) unfolding image-g

by simp

also have \( \ldots = \sum \text{list } (\text{map } \text{sq-norm } (\text{coeffs } g)) \)
  unfolding coeffs_def

by (simp add: interw-sum-list-conv-sum-set-nat)

finally show \( \ldots = \text{thesis} \) unfolding sq-norm-poly_def by auto

qed

lemma Hadamard’s-inequality-int:
fixes A::int mat
assumes A: A ∈ carrier-mat n n
shows \( |\text{det } A| \leq \sqrt{\text{of-int } (\text{prod-list } (\text{map } \text{sq-norm } (\text{rows } A)))} \)
proof –

let \( ?A = \text{map-mat real-of-int } A \)

have \( |\text{det } A| = |\text{det } ?A| \) unfolding of-int-hom.hom-det by simp

also have \( \ldots \leq \sqrt{\text{prod-list } (\text{map } \text{sq-norm } (\text{rows } ?A))} \)
  by (rule Hadamard’s-inequality[of \( ?A \) n], insert A, auto)

also have \( \ldots = \sqrt{\text{of-int } (\text{prod-list } (\text{map } \text{sq-norm } (\text{rows } A)))} \) unfolding
of-int-hom.hom-prod-list map-map

by (rule arg-cong[of - - x. \?sqrt (\text{prod-list } x)], rule nth-equalityI, force,
  auto simp: sq-norm-of-int[symmetric] row-def intro!: arg-cong[of - - sq-norm-vec])

finally show \( \ldots = \text{thesis} \) .

qed
lemma resultant-le-prod-sq-norm:
fixes f g::int
defines n ≡ degree f and k ≡ degree g
shows \(|\text{resultant } f \ g| \leq \sqrt{(|f|^2 \cdot k + |g|^2 \cdot n)}\)
proof –
let ?S = sylvester-mat f g
let ?f = sq-norm \circ row ?S
have map-rw1: map ?f [0..<degree g] = replicate k \(|f|^2\)
proof (rule nth-equalityI)
let ?M = map (sq-norm \circ row (sylvester-mat f g)) [0..<degree g]
show length ?M = length (replicate k \(|f|^2\) ) using k-def by auto
show ?M ! i = replicate k \(|f|^2\) ! i if i: i < length ?M for i
proof –
have ik: i<k using i-def by auto
hence i-deg-g: i < degree g using k-def by auto
have replicate k \(|f|^2\) ! i = \(|f|^2\) by (rule nth-replicate[OF ik])
also have ... = (sq-norm \circ row (sylvester-mat f g)) (\(0 + i\))
  using sq-norm-row-sylvester-mat1 ik k-def by force
also have ... = ?M ! i by (rule nth-map-upt[symmetric], simp add: i-deg-g)
finally show ?M ! i = replicate k \(|f|^2\) ! i ..
qed

have map-rw2: map ?f [degree g..<degree f + degree g] = replicate n \(|g|^2\)
proof (rule nth-equalityI)
let ?M = map (sq-norm \circ row (sylvester-mat f g)) [degree g..<degree f + degree g]
show length ?M = length (replicate n \(|g|^2\)) by (simp add: n-def)
show ?M ! i = replicate n \(|g|^2\) ! i if i<n length ?M for i
proof –
have i-n: i<n using n-def that by auto
hence i-deg-f: i < degree f using n-def by auto
have replicate n \(|g|^2\) ! i = \(|g|^2\) by (rule nth-replicate[OF i-n])
also have ... = (sq-norm \circ row (sylvester-mat f g)) (degree g + i)
  using i-n n-def
  by (simp add: sq-norm-row-sylvester-mat2)
also have ... = ?M ! i
  by (simp add: i-deg-f)
finally show ?M ! i = replicate n \(|g|^2\) ! i ..
qed

have p1: prod-list (map ?f [0..<degree g]) = \(|f|^2\)^k
unfolding map-rw1 by (rule prod-list-replicate)

have p2: prod-list (map ?f [degree g..<degree f + degree g]) = \(|g|^2\)^n
unfolding map-rw2 by (rule prod-list-replicate)

have list-rw: [0..<degree f + degree g] = [0..<degree g] \circ [degree g..<degree f + degree g]
  by (metis add.commute upt-add-ev-append zero-le)
have \(|\text{resultant } f \ g| = \det ?S\) unfolding resultant-def ..
also have ... \leq \sqrt{(\text{of-int} \ (\text{prod-list} \ (\text{map} \ sq-norm \ \text{rows} ?S)))}
by (rule Hadamard’s-inequality-int, auto)
also have map sq-norm (rows ?S) = map ?f [0..<degree f + degree g]
  unfolding Matrix.rows-def by auto
also have ... = map ?f ([0..<degree g] @ [degree g..<degree f + degree g])
  by (simp add: list-rw)
also have prod-list ... = prod-list (map ?f [degree g..<degree f + degree g]) by auto
finally show ?thesis unfolding p1 p2 .
qed

5.3 Proof of the key lemma 16.20

lemma common-factor-va-short:
figures f g u :: int poly
defines n ≡ degree f and k ≡ degree g
assumes n0: n > 0 and k0: k > 0
  and monic: monic u and deg-u: degree u > 0
  and uf: poly-mod.dvdm m u f and ug: poly-mod.dvdm m u g
  and short: \[ \|f\|_2^k * \|g\|_2^n < m^2 \]
  and m: m ≥ 0
shows degree (gcd f g) > 0
proof −
interpret poly-mod m .
have f-not0: f ≠ 0 and g-not0: g ≠ 0
  using n0 k0 k-def n-def by auto
have deg-f: degree f > 0 using n0 n-def by simp
have deg-g: degree g > 0 using k0 k-def by simp
obtain s t where deg-s: degree s < degree g and deg-t: degree t < degree f
  and res-eq: \[ \text{resultant } f g \] = s * f + t * g and s-not0: s ≠ 0 and t-not0: t ≠ 0
  using resultant-as-nonzero-poly[OF deg-f deg-g] by auto
have res-eq-modulo: \[ \text{resultant } f g \] = m s * f + t * g using res-eq
  by simp
have u-dvdm-res: u dvdm [:resultant f g:]
proof (unfold res-eq, rule dvdm-add)
  show u dvdm s * f
    using dvdm-factor[OF uf, of s]
    unfolding mult.commute[of f s] by auto
  show u dvdm t * g
    using dvdm-factor[OF ug, of t]
    unfolding mult.commute[of g t] by auto
qed
have res-0-mod: resultant f g mod m = 0
  by (rule monic-dvdm-constant[OF u-dvdm-res monic deg-u])
have res0: resultant f g = 0
proof (rule mod-0-abs-less-imp-0)
  show \[ \text{resultant } f g = 0 \] (mod m) using res-0-mod unfolding cong-def by auto
  have \[ \|\text{resultant } f g\| \leq \text{sqrt} \left( (sq-norm-poly f) \cdot k \cdot (sq-norm-poly g) \cdot n \right) \]
unfolding k-def n-def 
by (rule resultant-le-prod-sq-norm) 
also have \( \ldots < m \) 
proof (rule real-less-lsqrt)  
show \( 0 \leq \text{real-of-int} \left( \|f\|^2 \cdot k \cdot \|g\|^2 \cdot n \right) \)  
by (simp add: sq-norm-poly-ge0)  
show \( 0 \leq \text{real-of-int} \, m \) using m by simp  
show \( \text{real-of-int} \left( \|f\|^2 \cdot k \cdot \|g\|^2 \cdot n \right) < (\text{real-of-int} \, m)^2 \)  
by (metis of-int-less-iff of-int-power short)  
qed  
finally show \(|\text{resultant} \, f \, g| < m\) using of-int-less-iff by blast  
qed 

have \( \neg \text{coprime} \, f \, g \)  
by (rule resultant-zero-imp-common-factor, auto simp add: deg-f res0)  
thus \(?\)thesis  
using res0 resultant-0-gcd by auto  
qed

5.4 Properties of the computed lattice and its connection with Sylvester matrices

lemma factorization-lattice-as-sylvester:  
fixes \( p :: 'a :: \text{semidom poly} \)  
assumes \( \text{dj:} \, d \leq j \) and \( d :: \text{degree} \, p = d \)  
shows \( \text{mat-of-rows} \, j \, (\text{factorization-lattice} \, p \, (j-\, d \, m) \, = \, \text{sylvester-mat-sub} \, d \)  
\((j-\, d) \, p \, [\,: m:]\) \)  
proof (cases \( p = 0 \))  
case True  
have \( \text{deg-p:} \, d = 0 \) using True d by simp  
show \(?\)thesis  
by (auto simp add: factorization-lattice-def True deg-p mat-of-rows-def d)  
next  
case \( p0: \) False  
note \( 1 = \text{degree-mult-eq} [\text{OF} \, p0, \, \text{of monom} \, -, \, \text{unfolded} \, \text{monom-eq-0-iff}], \, \text{OF} \)  
one-neq-zero]  
from \( \text{dj} \) show \(?\)thesis  
apply (cases \( m = 0 \))  
apply (auto simp: mat-eq-iff d[symmetric] 1 coeff-mult-monom sylvester-mat-sub-index mat-of-rows-index nth-factorization-lattice vec-index-of-poly-n degree-monom-eq coeff-const)  
done  
qed

context inj-comm-semiring-hom begin

lemma map-poly-hom-mult-monom \[\text{hom-distribs}]::  
map-poly hom \((p \times \text{monom} \, a \, n)\) \(=\) map-poly hom \(p \times \text{monom} \,(\text{hom} \, a) \, n\)  
by (auto intro!: poly-eql simp:coeff-mult-monom hom-mult)
lemma hom-vec-of-poly-n [hom-distribs]:
  map-vec hom (vec-of-poly-n p n) = vec-of-poly-n (map-poly hom p) n
  by (auto simp: vec-index-of-poly-n)

lemma hom-factorization-lattice [hom-distribs]:
  shows map (map-vec hom) (factorization-lattice u k m) = factorization-lattice
    (map-poly hom u) k (hom m)
  by (auto intro!: arg-cong[of - - λp. vec-of-poly-n p -] simp: list-eq-iff-nth-eq nth-factorization-lattice
    hom-vec-of-poly-n map-poly-hom-mult-monom)

end

5.5 Proving that factorization-lattice returns a basis of the lattice

context LLL
begin

sublocale idom-vec n TYPE(int).

lemma upper-triangular-factorization-lattice:
  fixes u :: 'a :: semidom poly and d :: nat
  assumes d: d ≤ n and du: d = degree u
  shows upper-triangular (mat-of-rows n (factorization-lattice u (n - d) k))
    (is upper-triangular ?M)
  proof (intro upper-triangularI, unfold mat-of-rows-carrier length-factorization-lattice)
    fix i j
    assume ji: j < i and i: i < degree u + (n - d)
    with du have jn: j < n by auto
    show ?M $$ (i,j) = 0$$
    proof (cases u=0)
      case True with ji i show ?thesis
        by (auto simp: factorization-lattice-def mat-of-rows-def)
    next
      case False
      then show ?thesis
        using d ji i
        apply (simp add: du mat-of-rows-index nth-factorization-lattice)
        apply (auto simp: vec-index-of-poly-n[OF jn] degree-mult-eq degree-monom-eq)
        done
    qed

next

lemma factorization-lattice-diag-nonzero:
  fixes u :: 'a :: semidom poly and d
  assumes d: d = degree u
    and dn: d ≤ n

end
and \( u \neq 0 \)
and \( m_0 \cdot k \neq 0 \)
and \( i < n \)
shows \((\text{factorization-lattice } u \cdot (n-d) k)! \cdot i \neq 0\)

proof
have 1: monom \((1::'a) \cdot (n - Suc \cdot (\text{degree } u + i)) \neq 0\)
using \(m_0\) by auto

have 2: \( i < \text{degree } u + (n - d) \)
using \(i\) by auto

let \( ?p = u \cdot \text{monom } 1 \cdot (n - Suc \cdot (\text{degree } u + i))\)

have 3: \( i < n - \text{degree } u \implies \text{degree } (?p) = n - Suc \cdot i\)
using \(\text{assms}\) by (auto simp: \(\text{degree-monom-eq}\))

show \(?\text{thesis}\)
apply (unfold \(\text{nth-factorization-lattice}\))
using \(\text{assms}\)
done

qed

corollary factorization-lattice-diag-nonzero-RAT: fixes \(d\)
assumes \(d = \text{degree } u\)
and \(d \leq n\)
and \(u \neq 0\)
and \(k \neq 0\)
and \(i < n\)
shows \((RAT \cdot (\text{factorization-lattice } u \cdot (n-d) k)! \cdot i \neq 0)\)
using \(\text{factorization-lattice-diag-nonzero-RAT}\) \(\text{assms}\)
by (auto simp: \(\text{nth-factorization-lattice}\))

sublocale gs: \(\text{vec-space TYPE(rat)}\) \(n\).

lemma lin-indpt-list-factorization-lattice: fixes \(d\)
assumes \(d = \text{degree } u\) and \(d \leq n\) and \(u \neq 0\) and \(k \neq 0\)
shows \(\text{gs.lin-indpt-list } (RAT \cdot (\text{factorization-lattice } u \cdot (n-d) k)!) \cdot \text{gs.lin-indpt-list } (RAT \cdot ?vs)\)

proof
have 1: \(\text{rows } (\text{mat-of-rows } n \cdot (\text{map } (\text{map-vec } \text{rat-of-int}) \cdot ?vs)) = \text{map } (\text{map-vec } \text{rat-of-int}) \cdot ?vs\)
using \(\text{dn} \cdot d\)
by (subst \(\text{rows-mat-of-rows}\), auto dest!: \(\text{subsetD}[\text{OF set-factorization-lattice-in-carrier}]\))

note \(2 = \text{factorization-lattice-diag-nonzero-RAT}[OF d \cdot \text{dn} \cdot u \cdot k]\)

show \(?\text{thesis}\)
apply (intro \(\text{gs.upper-triangular-imp-lin-indpt-list}[\text{of mat-of-rows } n \cdot (RAT \cdot ?vs)],\)
unfolded \(1)\)
using \(\text{assms}\) \(2\) by (auto simp: \(\text{diag-mat-def \text{mat-of-rows-index \text{hom-distribs \text{intro!}:upper-triangular-factorization-lattice}}}\))

qed

end
5.6 Being in the lattice is being a multiple modulo

**lemma** (in semiring-hom) hom-poly-of-vec: map-poly hom (poly-of-vec v) = poly-of-vec (map-vec hom v)

by (auto simp add: coeff-poly-of-vec poly-eq-iff)

**abbreviation** of-int-vec ≡ map-vec of-int

context LLL

begin

**lemma** lincomb-to-dvd-modulo:

fixes u d

defines d ≡ degree u

assumes d: d ≤ n and lincomb: lincomb-list c (factorization-lattice u (n − d) k) = g (is ℓ = r)

shows poly-mod. dvd m k u (poly-of-vec g)

proof −

let ?S = sylvester-mat-sub d (n − d) u [; k:]

define q where q ≡ poly-of-vec (vec-first (vec n c) (n − d))

define r where r ≡ poly-of-vec (vec-last (vec n c) d)

have ℓ = ?S T * v vec n c

apply (subst lincomb-list-as-mat-mult)

using d d-def

apply (force simp: factorization-lattice-def)

apply (fold transpose-mat-of-rows)

using d d-def by (simp add: factorization-lattice-as-sylvester)

also have poly-of-vec ... = q * u + smult k r

apply (subst sylvester-sub-poly)

using d-def d q-def r-def by auto

finally have ... = poly-of-vec g

unfolding lincomb-of-int-hom.hom-poly-of-vec by auto

then have poly-of-vec g = q * u + Polynomial.smult k r by auto

then have poly-mod.Mp k (poly-of-vec g) = poly-mod.Mp k (q * u + Polynomial.smult k r) by auto

also have ... = poly-mod.Mp k (q * u + poly-mod.Mp k (Polynomial.smult k r))

using poly-mod.plus-Mp(2) by auto

also have ... = poly-mod.Mp k (q * u)

using poly-mod.plus-Mp(2) unfolding poly-mod.Mp-smult-m-0 by simp

also have ... = poly-mod.Mp k (u * q) by (simp add: mult.commute)

finally show ?thesis unfolding poly-mod.dvdm-def by auto

qed

**lemma** dvd-modulo-to-lincomb:

fixes u :: int poly and d

defines d ≡ degree u

assumes d: d < n and dvd: poly-mod.dvdm k u (poly-of-vec g)

and k-not0: k ≠ 0 and monic-u: monic u and dim-g: dim-vec g = n
and \( \text{deg-u} \): degree \( u > 0 \)

shows \( \exists c. \text{lincomb-list} \ s \ (\text{factorization-lattice} \ u \ (n-d) \ k) = g \)

proof

interpret \( p: \text{poly-mod} \ k \)

have \( u\not\equiv0 \): \( u \not= 0 \) using \( \text{monic-u by auto} \)

hence \( n[\text{simp}]: 0 < n \) using \( d \) by auto

obtain \( q''r'' \) where \( g: \text{poly-of-vec} \ g = q''u + \text{smult} \ k \ r'' \\
using \( \text{p.ddm-imp-div-mod[OF ddef]} \) by auto

obtain \( q''r'' \) where \( r': r'' = q''u + r'' \) and \( \text{deg-r''}: \text{degree} \ r'' < \text{degree} \ u \\
using \( \text{monic-imp-ddm-int-poly-degree2[OF monic-u deg-u, of r']} \) by auto

have \( g1: \text{poly-of-vec} \ g = (q' + \text{smult} \ k \ q'')u + \text{smult} \ k \ r'' \\
unfolding \ g \ r'' \\
by (metis \ (\text{no-types, lifting}) \ \text{combine-common-factor} \ \text{mult-smult-left} \ \text{smult-add-right})

define \( q \) where \( q = (q' + \text{smult} \ k \ q'') \\
define \( r \) where \( r: r = r'' \\
have \( \text{degree-q-q}: \text{degree} \ q = 0 \lor \text{degree} \ ((q' + \text{smult} \ k \ q'') < n - d \\
proof (cases \ q = 0, \text{auto, rule degree-div-mod-smult}[OF - - - g1]) \ 
show \( \text{degree} \ (\text{poly-of-vec} \ g) < n \) by (rule \text{degree-poly-of-vec-less}, \text{auto simp add: dim-g})

show \( \text{degree} \ r'' < d \) using \( \text{deg-r'' unfolding d-def} \) .

assume \( q\not=0 \) thus q'' + \text{smult} \ k \ q'' \not= 0 \ unfolding \ q \).

show \( k \not= 0 \) by fact

show \( \text{degree} \ u = d \) using \( d-def \) by auto

qed

have \( g2: (\text{vec-of-poly-n} \ (q*u) \ n) + (\text{vec-of-poly-n} \ (\text{smult} \ k \ r) \ n) = g \\
proof

have \( g = \text{vec-of-poly-n} \ (\text{poly-of-vec} \ g) \ n \\
by (rule \text{vec-of-poly-n-poly-of-vec}[\text{symmetric}], \text{auto simp add: dim-g})

also have \( \ldots = \text{vec-of-poly-n} \ ((q' + \text{smult} \ k \ q'')u + \text{smult} \ k \ r'') \ n \\
using \( g1 \) by auto

also have \( \ldots = \text{vec-of-poly-n} \ (q * u + \text{smult} \ k \ r'') \ n \ unfolding \ q \ by auto \\
also have \( \ldots = \text{vec-of-poly-n} \ (q * u) \ n + \text{vec-of-poly-n} \ (\text{smult} \ k \ r'') \ n \\
by (rule \text{vec-of-poly-n-add})

finally show \( \text{thesis unfolding r by simp} \)

qed

let \( ?c = \lambda i. \text{if} \ i < n \ - \ d \ \text{then coeff} \ q \ (n - d - 1 - i) \ \text{else coeff} \ r \ (n - Suc \ i) \) 
let \( ?c1 = \lambda i. \ ?c \ i \cdot \text{factorization-lattice} \ u \ (n-d) \ k ! \ i \\
show \( \text{thesis} \\
proof (rule \text{exI}[of - ?c]) \\
let ?part1 = \text{map} \ (\lambda i. \text{vec-of-poly-n} \ (u * \text{monom} \ i \ i) \ n) \ [n-d>..0] \\
let ?part2 = \text{map} \ (\lambda i. \text{vec-of-poly-n} \ (\text{monom} \ k \ i \ n) \ [d>..0] \\
have \( \text{simp}[\text{simp}]: \text{dim-vec} \ \text{M-sumlist} \ (\text{map} \ ?c1 \ [0..<n-d]) = n \\
by (rule \text{dim-sumlist}, \text{auto simp add: dim-factorization-lattice d-def})

have \( \text{simp}[\text{simp}]: \text{dim-vec} \ \text{M-sumlist} \ (\text{map} \ ?c1 \ [n-d..<n]) = n \\
by (rule \text{dim-sumlist}, \text{insert d, auto simp add: dim-factorization-lattice d-def})

have \( \text{simp}[\text{simp}]: \text{factorization-lattice} \ u \ (n-d) \ k \ ! \ x \in \text{carrier-vec n if} \ x: x < n \ \text{for x} \\
using \( x \ \text{dim-factorization-lattice-element nth-factorization-lattice}[\text{of} \ u \ n-d] \)
\(d\)  
by (auto simp; d-def)  

have \([0..<n\text{ length} \text{ (factorization-lattice u \((n-d)\) k)] = [0..<n}\]

using \(d\) by (simp add: d-def less-imp-le-nat)
also have ... = \([0..<n-d] \&\& [n-d..<n]\]
by (rule upt-minus-eq-append, auto)

finally have list-rw: \([0..<n\text{ length} \text{ (factorization-lattice u \((n-d)\) k)] = [0..<n-n-d\] \&\& \([n-d..<n]\)

have \(q\&u\): poly-of-vec \((M.\text{sumlist} \text{ (map ?c1 \([0..<n-d]\))} = q\&u\)
proof -

have poly-of-vec \((M.\text{sumlist} \text{ (map ?c1 \([0..<n-d]\))} = \text{poly-of-vec} \(\bigoplus_{i=\{0..<n-d\}}\)).
\(\&c1\ i\)  
by (subst \(\text{suml}\text{ist-map}\text{-map-\text{a}f\text{fin}}\text{sum}\), auto)
also have ... = \(\text{poly-of-vec} \(\bigoplus_{i=\{0..<n-d\}}\ ?c1\ i\)\) by auto
also have ... = \(\text{sum} \(\lambda.\ \text{poly-of-vec} \(\ ?c1\ i\)\) (\(\text{set}\ \{0..<n-d\})\)
by (auto simp:poly-of-vec-finsum)
also have ... = \(\text{sum} \(\lambda.\ \text{poly-of-vec} \(\ ?c1\ i\)\) \{0..<n-d\} by auto
also have ... = \(q\&u\)
proof -

have \(\text{deg}: \text{degree} \ (u \ast \text{monom} 1 \ (n - \text{Suc} \ (d + i))) < n\) if \(i\): \(i < n - d\)
for \(i\)
proof -
let \(?m=\text{monom} \ (1:\text{int}) \ (n - \text{Suc} \ (d + i))\)
have \(\text{monom-not0}: \ ?m \neq 0\) using \(i\) by auto

have \(\text{deg-n}: \text{degree} \ ?m = n - \text{Suc} \ (d + i)\) by (rule degree-monom-eq, auto)

have \(\text{degree} \ (u \ast \ ?m) = d + (n - \text{Suc} \ (d + i))\)
using degree-mul-eq[OF \(\text{u-not0}\ \text{monom-not0}\) d-def \(\text{deg-m}\) by auto
also have ... < \(n\) using \(i\) by auto
finally show \(?\text{thesis}\).
qed

have lattice-rw: \(\text{factorization-lattice u \((n-d)\) k} \ ! \ i = \text{vec-of-poly-n} \ (u \ast \text{monom} 1 \ (n - \text{Suc} \ (d + i)))\ n\)
if \(i\): \(i < n - d\) for \(i\) apply (subst \(\text{nth-factorization-lattice}\) using \(i\) by (auto simp:d-def)

have \(\text{q-rw}: \ q = (\sum i = 0..<n - d. \ (\text{smult} \ (\text{coeff} \ q \ (n - \text{Suc} \ (d + i))))\ (\text{monom} 1 \ (n - \text{Suc} \ (d + i))))\)

proof (auto simp add: poly-eq-iff coeff-sum)
fix \(j\)
let \(?\text{m} = n - d - 1 - j\)
let \(\?f = \lambda x. \text{coeff} \ q \ (n - \text{Suc} \ (d + x)) \ast (\text{if} \ n - \text{Suc} \ (d + x) = j \text{ then 1} \text{ else 0})\)

have set-rw: \(\{0..<n-d\} = \text{insert} \ ?m \ \{0..<n-d\} - \{?m\}\) using \(d\) by auto

have sum0: \((\sum x \in \{0..<n-d\} - \{?m\}). \ ?f \ x = 0\) by (rule sum.neutral, auto)

have \((\sum x = 0..<n - d. \ ?f \ x) = (\sum x \in \text{insert} \ ?m \ \{0..<n-d\} - \{?m\})\.
\(?f \ x\)
using set-rw by presburger
also have ... = ?f ?m + (\sum x \in \{0..<n-d\} - \{?m\}. ?f x) by (rule sum.insert, auto)
also have ... = ?f ?m unfolding sum0 by auto
also have ... = coeff q j
proof (cases j < n - d)
case True
then show ?thesis by auto
next
case False
have j>degree q using degree-q q False d by auto
then show ?thesis using coeff-eq-0 by auto
qed
finally show coeff q j = (\sum i = 0..<n - d. coeff q (n - Suc (d + i)) * (if n - Suc (d + i) = j then 1 else 0)) ...
qed
have sum (\lambda i. poly-of-vec (?c1 i)) \{0..<n-d\}
= (\sum i = 0..<n - d. poly-of-vec (coeff q (n - Suc (d + i)) \cdot v, factorization-lattice u (n-d) k ! i))
by (rule sum.cong, auto)
also have ... = (\sum i = 0..<n - d. (poly-of-vec (coeff q (n - Suc (d + i)) \cdot v (vec-of-poly n (u * monom 1 (n - Suc (d + i)))))))
by (rule sum.cong, auto simp add: lattice-rw)
also have ... = (\sum i = 0..<n - d. smult (coeff q (n - Suc (d + i)))) (u * monom 1 (n - Suc (d + i))))
by (rule sum.cong, auto simp add: poly-of-vec-scalar-mult[OF deg])
also have ... = (\sum i = 0..<n - d. u*smult (coeff q (n - Suc (d + i)))) (monom 1 (n - Suc (d + i))))
by auto
also have ...
by (rule sum-distrib-left[ symmetric])
also have ... = u * q using q-rw by auto
also have ...
finally show ?thesis .
qed
finally show ?thesis .
qed
have qv: M.sumlist (map ?c1 [0..<n - d]) = vec-of-poly-n (q*u) n
proof -
have vec-of-poly-n (q*u) n = vec-of-poly-n (M.sumlist (map ?c1 [0..<n - d]))) n
using qv1 by auto
also have vec-of-poly-n (poly-of-vec (M.sumlist (map ?c1 [0..<n - d]))) n
= M.sumlist (map ?c1 [0..<n - d])
by (rule vec-of-poly-n-poly-of-vec, auto)
finally show ?thesis .
qed
have rm1: poly-of-vec (M.sumlist (map ?c1 [n-d..<n])) = smult k r
proof -
have poly-of-vec \((M.\text{sumlist } (\text{map } ?c1 \{n - d.. < n\})) = \text{poly-of-vec } (\bigoplus \forall i \in \{n - d.. < n\}. \) ?c1 i)\)

by (subst sumlist-map-as-finsum, auto)
also have ... = poly-of-vec \((\bigoplus \forall i \in \{n - d.. < n\}. \ ) ?c1 i\) by auto
also have ... = \(\sum \bigl(\lambda i. \text{poly-of-vec } (?c1 i)\) \{n - d.. < n\}\)
by (auto simp: poly-of-vec-finsum)
also have ... = smult \(k r\)
proof –
  have \(\text{deg}: \text{degree } (\text{monom } k \{n - \text{Suc } i\}) < n\) if \(i: n - d \leq i\) and \(i2: i < n\) for \(i\)
  using degree-monom-le i i2
  by (simp add: degree-monom-eq k-not0)

have lattice-rw: factorization-lattice u \((n - d) \cdot k \cdot i\) = vec-of-poly-n \((\text{monom } k \{n - \text{Suc } i\}) \{n - \text{Suc } i\}\))
proof (auto simp add: poly-eq-iff coeff-sum)
fix \(j\)
show \(\text{coeff } r \cdot j = (\sum i = n - d.. < n. \text{if } n - \text{Suc } i = j \text{ then } \text{coeff } r \cdot (n - \text{Suc } i)\) \{n - \text{Suc } i\}\) else \(0\)
proof (cases \(j < d\))
case True
have \(j\)-eq: \(n - \text{Suc } (n - 1 - j) = j\) using \(d\) True by auto
let \(?i = n - 1 - j\)
let \(?f = \lambda i. \text{if } n - \text{Suc } i = j \text{ then } \text{coeff } r \cdot (n - \text{Suc } i)\) else \(0\)
have sum0: \(\sum \text{ if } \{\{n - d.. < n\} - \{?i\}\} = 0\) by (rule sum.neutral, auto)
hence sum \(\text{if } \{n - d.. < n\} = \text{sum } \text{if } \{\text{insert } \{n - d.. < n\} - \{?i\}\}\)
by auto
also have ... = \(\text{if } \{n - d.. < n\} - \{?i\}\)
  by (rule sum.insert, auto)
also have ... = coeff \(r \cdot j\) unfolding sum0 j-eq by simp
finally show \(?\text{thesis}..\)
next
case False
hence \(\sum i = n - d.. < n. \text{if } n - \text{Suc } i = j \text{ then } \text{coeff } r \cdot (n - \text{Suc } i)\) else \(0\) = \(0\)
by (intro sum.neutral ballI, insert False, simp, linarith)
also have ... = coeff \(r \cdot j\)
  by (rule coeff-eq-0[symmetric], insert False deg-r" \(r\) d-def, auto)
finally show \(?\text{thesis}..\)
qed
qed
have \(\sum \bigl(\lambda i. \text{poly-of-vec } (?c1 i)\) \{n - d.. < n\}\)
= \((\sum i \in \{n - d.. < n\}. \text{poly-of-vec } \text{coeff } r \cdot (n - \text{Suc } i)\) \cdot \text{factorization-lattice} \)
The factorization lattice precisely characterises the polynomials of a certain degree which divide \( u \) modulo \( M \).

**lemma** factorization-lattice: fixes \( M \) assumes
deg-u: degree \( u \neq 0 \) and \( M; M \neq 0 \)
shows degree \( u \leq n \implies n \neq 0 \implies f \in \text{poly-of-vec · lattice-of (factorization-lattice}
\( u \ (n - \text{degree } u) \ M \) \implies
\[ \text{degree } f < n \land \text{poly-mod.dvdm } M \ u \ f \]
monic \( u \) \implies \text{degree } u < n \implies
\[ \text{degree } f < n \implies \text{poly-mod.dvdm } M \ u \ f \implies f \in \text{poly-of-vec ' lattice-of } (\text{factorization-lattice } u \ (n - \text{degree } u) \ M) \]

proof –
from \( \text{deg-u} \) have \( \text{deg-u}: \text{degree } u > 0 \) by auto
let \( ?L = \text{factorization-lattice } u \ (n - \text{degree } u) \ M \)

\{ 
assume \( \text{deg}: \text{degree } f < n \) and \( \text{dvd}: \text{poly-mod.dvdm } M \ u \ f \) and \( \text{mon}: \text{monic } u \)
and \( \text{deg-u-lt}: \text{degree } u < n \)
define \( f \ v \) where \( f \ v = \text{vec } n \ (\lambda \ i. \ (\text{coeff } f \ (n - \text{Suc } i))) \)

have \( f : f = \text{poly-of-vec } f \ v \) unfolding \( f \ v \text{-def} \) \( \text{poly-of-vec-def} \) \( \text{Let-def} \) using \( \text{deg} \)
by (auto intro!: \( \text{poly-eqI} \) \( \text{coeff-eq-0} \) simp: \( \text{coeff-sum} \))

have \( \text{dim-fv} : \text{dim-vec } f \ v = n \) unfolding \( f \ v \text{-def} \) by simp
from \( \text{dvdm-modulo-to-lincomb} \ [\ OF \ \text{deg-u-lt} - M \ \text{mon} - \text{deg-u} \ (1) , \ of \ f \ v , \ \text{folded } f , \ OF \ \text{dvd} \ \text{dim-fv}] \)

obtain \( c \) where \( g \ v : f \ v = \text{lincomb-list } c \ ?L \) by auto

have \( f \ v \in \text{lattice-of } ?L \) unfolding \( g \ v \text{lattice-is-span} \) by (auto simp: \( \text{in-span-listI} \))
thus \( f \in \text{poly-of-vec ' lattice-of } ?L \) unfolding \( f \) by auto
\}

moreover
\{ 
assume \( f \in \text{poly-of-vec ' lattice-of } ?L \) and \( \text{deg-u}: \text{degree } u \leq n \) and \( n: n \neq 0 \)
then obtain \( f \ v \) where \( f : f = \text{poly-of-vec } f \ v \) and \( f : f \ v \in \text{lattice-of } ?L \) by auto
from \( \text{in-span-listE} \ [\ OF \ f \ v \text{[unfolded lattice-is-span]}] \)

obtain \( c \) where \( f : f \ v = \text{lincomb-list } c \ ?L \) by auto
from \( \text{lincomb-to-dvdm-modulo} \ [\ OF \ - f \ v \text{[symmetric]}] \ \text{deg-u } f \)

have \( \text{dvdm: poly-mod.dvdm } M \ u \ f \) by auto

have set \( ?L \subseteq \text{carrier-vec } n \) unfolding \( \text{factorization-lattice-def} \) using \( \text{deg-u} \) by auto

hence \( f \ v \in \text{carrier-vec } n \) unfolding \( f \) by (metis \( \text{lincomb-list-carrier} \)

degree \( f < n \) unfolding \( f \) using \( \text{degree-poly-of-vec-less} \) [of \( f \ v \) \( n \)]
using \( n \) by auto

with \( \text{dvdm} \) show \( \text{degree } f < n \land \text{poly-mod.dvdm } M \ u \ f \) by auto
\}
qed
end

5.7 Soundness of the LLL factorization algorithm

lemma \( \text{LLL-short-polynomial}: \text{assumes} \ \text{deg-u-0}: \text{degree } u \neq 0 \) and \( \text{deg-le}: \text{degree } u \leq n \)
and \( \text{pl1}: \text{pl} > 1 \)
and \( \text{monic: mニック } u \)
shows \( \text{degree } (\text{LLL-short-polynomial } \text{pl } n \ u) < n \)
and \( \text{LLL-short-polynomial } \text{pl } n \ u \neq 0 \)
and \( \text{poly-mod.dvdm } \text{pl } n \ u \ (\text{LLL-short-polynomial } \text{pl } n \ u) \)
and \( \text{degree } u < n \implies f \neq 0 \implies \)
poly-mod dvd-in-deg \( poly-mod \) \( pl \) \( u \) \( f \) \( \Rightarrow \) degree \( f < n \) \( \Rightarrow \) \( \| LLL\text{-short-polynomial} \) \( pl \) \( n \) \( u \) \( \|^2 \leq 2^2 \) \( (n - 1) \) \( * \) \( \| f \|^2 \)

**proof**

```
interpret poly-mod-2 pl
  by (unfold-locales, insert pl1, auto)
from pl1 have pl0: \( pl \neq 0 \) by auto
let \(?d = \) degree \(?u\)
let \(?u = \) inv-Mp \(?u\)
from Mp-inv-Mp-id[of \(?u\)] have \(?iu = m \) \(?u\)
also have \( \ldots = m \) \(?u\) by simp
finally have \(?u = m \) \(?u\) by simp
have degu[simp]: degree \(?u = \) degree \(?u\) using monic by simp
have mon: monic \(?u\) using monic by (rule monic-Mp)
have degree \(?iu = \) degree \(?u\) unfolding inv-Mp-def
  by (rule degree-map-poly, unfold mon, insert mon pl1, auto simp: inv-Mp-def)
with degu have deg-iu: degree \(?iu = \) degree \(?u\) by simp
have mon-iu: monic \(?iu\) unfolding deg-iu unfolding inv-Mp-def Mp-def inv-Mp-def
M-def
  by (insert pl1, auto simp: coeff-map-poly monic)
let \(?L = \) factorization-lattice \(?iu\) \( (n - \) d) \( pl\)
let \(?sv = \) short-vector-hybrid \( 2 \) \(?L\)
from deg-u-0 deg-le have \( n \neq 0 \) by auto
from deg-u-0 have u0: \( u \neq 0 \) by auto
have id: \( LLL\text{-short-polynomial} \) \( pl \) \( n \) \( u \) \( = \) \( \) poly-of-vec \(?sv\)
  unfolding LLL-short-polynomial-def by blast
have id': \( \| sv \|^2 = \| LLL\text{-short-polynomial} \) \( pl \) \( n \) \( u \) \( \|^2 \) unfolding id by simp
interpret vec-module TYPE(int) n.
interpret L: \( LLL\text{-short-polynomial} \) \( pl \) \( n \) \( ?L \) \( 2 \).
from deg-le deg-iu have deg-iu-le: degree \(?iu \leq \) n by simp
have len: length \(?L = n\)
  unfolding factorization-lattice-def using deg-le deg-iu by auto
from deg-u-0 deg-iu have deg-iu0: degree \(?iu \neq 0 \) by auto
hence iu0: \(?iu \neq 0 \) by auto
from L.lin-indpt-list-factorization-lattice[OF refl deg-iu-le iu0 pl0]
have*: \( 4^3 \leq (2 :: rat) \) \( L \) ?.gs.lin-indpt-list (L.RAT ?L) by (auto simp: deg-iu)
interpret L: \( LLL\text{-with-assms} \) \( n \) \( ?L \) \( 2 \)
  by (unfold-locales, insert *, auto simp: deg-iu deg-le)
note short = \( LLLL\text{-short-vector-hybrid} \) [OF refl n, unfolded id' L.LL-def]
from short(2) have mem: \( LLL\text{-short-polynomial} \) \( pl \) \( n \) \( u \) \( \in \) \( \) poly-of-vec \(?sv\)
  unfolding \( ?L\)
  unfolding id by auto
note fact = L.factorization-lattice(1)[OF deg-iu0 pl0 deg-iu-le n, unfolded deg-iu, OF mem]
show degree \( (LLL\text{-short-polynomial} \) \( pl \) \( n \) \( u \) \( < \) \( n \) using fact by auto
from fact have \(?iu dvdn \) \( (LLL\text{-short-polynomial} \) \( pl \) \( n \) \( u \) \( ) \) by auto
  then obtain \( h \) where \( LLL\text{-short-polynomial} \) \( pl \) \( n \) \( u = m \) \(?iu \) * \( h \) unfolding dvdn-def by auto
also have \(?iu * h = m \) \( M \) \(?iu \) * \( h \) unfolding mult-Mp by simp
```
also have \( Mp \ ?iu \ast h = \text{m} u \ast h \) unfolding \( \text{iu-u} \) unfolding \( \text{mult-Mp} \) by simp
finally show \( u \ \text{dvd}m \ (\text{LLL-short-polynomial pl n u}) \) unfolding \( \text{dvdm-def} \) by auto
from \( \text{short-have sv1: ?sv \in \text{carrier-vec n by auto} \)
from \( \text{short-have ?sv \neq 0_v \ for j by auto} \)
thus \( \text{LLL-short-polynomial pl n u \neq 0} \) unfolding \( \text{id by simp} \)
assume \( \text{degw: degree u < n and dvd: u dvdm f} \)
and \( \text{degf: degree f < n and f0: f \neq 0} \)
from \( \text{dvd obtain h where f = \text{m} u \ast h} \) unfolding \( \text{dvdm-def by auto} \)
also have \( u \ast h = \text{m} Mp \ u \ast h \) unfolding \( \text{mult-Mp by simp} \)
also have \( Mp \ u \ast h = \text{m} Mp \ ?iu \ast h \) unfolding \( \text{iu-u by simp} \)
also have \( Mp \ ?iu \ast h = \text{m} ?iu \ast h \) unfolding \( \text{mult-Mp by simp} \)
finally have \( \text{dvd: ?iu dvdm f} \) unfolding \( \text{dvdm-def by auto} \)
from \( \text{degu deg-iu have deg-iun: degree ?iu < n by auto} \)
from \( \text{L.factorization-lattice(2)(OF deg-iu0 pl0 mon-iu deg-iun degf dvd)} \)
\( \text{have f \in poly-of-vec \ ' lattice-of ?L using deg-iu by auto} \)
then obtain \( \text{fv where f = poly-of-vec fv and fv: fv \in lattice-of ?L by auto} \)
\( \text{have norm: } \|fv\|^2 = \|f\|^2 \) unfolding \( f \) by simp
have \( \text{f00: f0 \neq 0_v n using f0 unfolding f by auto} \)
with \( \text{fv have fvL: fv \in lattice-of ?L - \{0_v n\} by auto} \)
from \( \text{short(3)(OF this, unfolded norm)} \)
\( \text{have rat-of-int } \|\text{LLL-short-polynomial pl n u}\|^2 \leq \text{rat-of-int } (2 \ast (n - 1) \ast \|f\|^2) \)
by simp
thus \( \|\text{LLL-short-polynomial pl n u}\|^2 \leq 2 \ast (n - 1) \ast \|f\|^2 \) by linarith
qed

context \( \text{LLL-implementation} \)
begin

lemma \( \text{LLL-reconstruction: assumes LLL-reconstruction f us = fs} \)
\( \text{and degree f \neq 0} \)
\( \text{and poly-mod.unique-factorization-m pl f (lead-coeff f, mset us)} \)
\( \text{and f dvd F} \)
\( \text{and } \bigwedge \text{ui. ui \in set us \implies poly-mod.Mp pl ui = ui} \)
\( \text{and F0: F \neq 0} \)
\( \text{and cop: coprime (lead-coeff F) p} \)
\( \text{and sf: poly-mod.square-free-m p F} \)
\( \text{and pl1: pl > 1} \)
\( \text{and plp: pl = p \'} \)
\( \text{and p: prime p} \)
\( \text{and large: } 2^\ast(5 \ast (\text{degree F - 1}) \ast (\text{degree F - 1})) \ast \|F\|^2 \ast (2 \ast (\text{degree F - 1})) < pl^2 \)
shows \( f = \text{prod-list fs} \land (\forall fi \in \text{set fs. irreducible}_{\ast} fi) \)

proof —
\( \text{interpret p: poly-mod-prime p by (standard, rule p)} \)
\( \text{interpret pl: poly-mod-2 pl by (standard, rule pl1)} \)
from \( \text{pl plp have l0: l \neq 0 by (cases l, auto)} \)
show \( \text{thesis using asssns(1-5)} \)
proof (induct f us arbitrary; fs rule: \( \text{LLL-reconstruction.induct} \))
case (1 f us fs)
define u where u = choose-u us
define g where g = LLL-short-polynomial pl (degree f) u
define k where k = \gcd f g
note res = 1(3)
note degf = 1(4)
note uf = 1(5)
note fF = 1(6)
note norm = 1(7)
note fact = to-fact[pl.unique-factorization-m-imp-factorization]
note fact = to-fact[OF af]

have mon-gs: ui \in set us \implies \monic ui for ui using norm fact
  unfolding pl.factorization-m-def by auto
from p.coprime-lead-coeff-factor[OF p.prime] fF cop
have cop: coprime (lead-coeff f) p unfolding ded-def by blast
have pl0: pl.Mp f \neq 0
  using fact pl.factorization-m-lead-coeff pl.unique-factorization-m-zero uf by fastforce
  have degree f = pl.degree-m f
    by (rule sym, rule poly-mod.degree-m-eq[\OF - pl.m1],
        insert cop p, simp add: \l 0 p.coprime-exp-mod plp)
  also have \ldots = sum-mset (image-mset pl.degree-m (mset us))
    unfolding pl.factorization-m-degree[\OF fact plf0] ..
  also have \ldots = sum-list (map pl.degree-m us)
    unfolding sum-mset-sum-list[symmetric] by auto
  also have \ldots = sum-list (map degree us)
    by (rule arg-cong[\OF map-cong, \OF refl], rule pl.monic-degree-m, insert mon-gs, auto)
  finally have degf-gs: degree f = sum-list (map degree us) by auto
  hence gs: us \neq [] using degf by (cases us, auto)
from choose-u-member[\OF gs] have u-gs: u \in set us unfolding u-def by auto
from fact u-gs have irreduced: pl.irreducible_d m u unfolding pl.factorization-m-def
  by auto
  hence deg-a: degree u \neq 0 unfolding pl.irreducible_d-m-def norm[\OF u-gs] by auto
  have deg-af: degree u \leq degree f unfolding degf-gs using split-list[\OF u-gs]
  by auto
  from mon-gs[\OF u-gs] have mon-u: \monic u and u0: u \neq 0 by auto
  have f0: f \neq 0 using degf by auto
  from norm have norm': image-mset pl.Mp (mset us) = mset us by (induct us, auto)
  have pl0: pl \neq 0 using plf by auto
note short-main = LLL-short-polynomial[\OF deg-u deg-uf pl1 mon-a]
  from short-main(1-2)[\folded g-def]
  have degree f < degree f unfolding k-def
    by (smt Suc-le Suc-less-eq degree-gcd1 gcd.commute le-imp-less-Suc le-trans)
  hence deg-fk: (degree k = 0 \lor degree f \leq degree k) = (degree k = 0) by auto
note res = \res[unfolded LLL-reconstruction.simps[af f us]] Let-def, folded u-def,
folded g-def, folded k-def, unfolded deg-fk

show \text{case}

proof (cases degree k = 0)
case True
  with res have \(fs: \text{fs} = [f]\) by auto
from sf \(f\) have \(sf: \text{p.square-free-m f}\)
  using \(\text{p.square-free-m-factor}(1)\)[of f] unfolding dvd-def by auto
have \(\text{irr}: \text{irreducible}_d f\)
proof (rule ccontr)
  assume \(\neg \text{irreducible}_d f\)
from \(\text{reducible}_d E[\text{OF this}]\) obtain \(f_1 f_2\) where
  \(f: f = f_1 \ast f_2\) and
  \(\text{deg12}: \text{degree } f_1 \neq 0 \land \text{degree } f_2 \neq 0\) degree \(f_1 < \text{degree } f\) degree \(f_2 < \text{degree } f\)
  by (simp, metis)
from \(\text{pl.unique-factorization-m-factor}[\text{OF p uf}[\text{unfolded } f], \text{folded } f, \text{OF cop sf} \text{ l0 plp}]\)
obtain \(us1 us2\) where
  \(uf12: \text{pl.unique-factorization-m f1 (lead-coeff f1, us1)}\)
  \(\text{pl.unique-factorization-m f2 (lead-coeff f2, us2)}\)
  and \(\text{gs}: \text{mset } us = us1 + us2\)
  and \(\text{norm12: image-mset \text{pl.Mp us2 = us2 image-mset pl.Mp us1 = us1}}\)
  unfolding \(\text{pl.Mf-def norm'}\) split by (auto simp: \text{pl.Mf-def})
note \(\text{norm-u} = \text{norm}[\text{OF u-gs}]\)
from \(\text{u-gs have u-gs': } u \in \# \text{ mset us by auto}\)
with \(\text{pl.factorization-m-mem-dedm}[\text{OF fact, of } u]\)
have \(\text{u-f: } \text{pl.dvdms u f by auto}\)
from \(\text{u-gs'[unfolded gs]}\) have \(u \in \# \text{ us1 } \lor u \in \# \text{ us2 by auto}\)
with \(\text{pl.factorization-m-mem-dedm}[\text{OF to-fact}[\text{OF uf12(1)]}, \text{of } u]\)
  \(\text{pl.factorization-m-mem-dedm}[\text{OF to-fact}(\text{OF uf12(2)]}, \text{of } u]\)
have \(\text{pl.dvdms u f1 } \lor \text{pl.dvdms u f2 unfolding norm12 norm-u by auto}\)
from this have \(\exists \ f_1 f_2. f = f_1 \ast f_2\) and
  degree \(f_1 \neq 0\) and degree \(f_2 \neq 0\) and degree \(f_1 < \text{degree } f\) and degree \(f_2 < \text{degree } f\)
  and \(\text{prod: } f = f_1 \ast f_2\)
  and \(\text{deg: degree } f_1 \neq 0\) degree \(f_2 \neq 0\) degree \(f_1 < \text{degree } f\) degree \(f_2 < \text{degree } f\)
  and \(\text{af1: } \text{pl.dvdms u f1 by auto}\)
from \(\text{pl.unique-factorization-m-factor}[\text{OF p uf}[\text{unfolded prod}, \text{folded prod}, \text{OF cop sf} \text{ l0 plp}]\)
obtain \(us1\) where \(\text{fact-f1: } \text{pl.unique-factorization-m f1 (lead-coeff f1, us1)}\)
by \textit{auto}

have \(plf1\): \(pl.Mp\ f1 \neq 0\)
\hspace{1em}
using \textit{to-fact(OF fact-f1)} \(pl.factorization-m-lead-coeff\)
\hspace{1em}
\(pl.unique-factorization-m-zero fact-f1\) by \textit{fastforce}

have degree \(u \leq\) degree \(f1\)
\hspace{1em}
by \(\text{rule pl.dedm-degree(OF mon-u uf1 plf1)}\)

with \(\text{deg have deg-af: degree } u <\) degree \(f\) by \textit{auto}

have \(plb\): \(pl \neq 0\) using \(pl.m1 plp\) by \textit{linarith}

let \(?n = degree f\)

let \(?n1 = degree f1\)

let \(?d = degree u\)

from \(\text{prod } F^F\) have \(f1F\): \(f1 \text{ ded } F\) unfolding \textit{dvd-def} by \textit{auto}

from \(\text{deg-uf have deg-uf\': } ?d \leq ?n\) by \textit{auto}

from \(\text{deg have f1-0: } f1 \neq 0\) by \textit{auto}

have \(ug\): \(pl.dedm\ u\ g\) using \(\text{short-main(3)}\) unfolding \(g\)-\textit{def}.

have \(?g\): \(?g \neq 0\) using \(\text{short-main(2)}\) unfolding \(g\)-\textit{def}.

let \(?N = degree F\)

from \(\text{F0 prod have f1F: } f1 \text{ ded } F\) unfolding \textit{dvd-def} by \textit{auto}

have \(\|g\|^2 \leq 2 \times (\?n - 1) \times \|f1\|^2\) unfolding \(g\)-\textit{def}
\hspace{1em}
by \(\text{rule short-main(4)}(\text{OF deg-uf - uf1}),\) insert \(\text{deg, auto}\)

also have \(\ldots \leq 2 \times (\?n - 1) \times (2 \times (2 \times \text{degree } f1) \times \|F\|^2)\)
\hspace{1em}
by \(\text{rule mult-left-mono}(\text{OF sq-norm-factor-bound(OF } f1F F0)],\) \textit{simp}\)

also have \(\ldots = 2 \times ((\?n - 1) + 2 \times \text{degree } f1) \times \|F\|^2\)
\hspace{1em}
unfolding \textit{power-add} by \textit{simp}\)

also have \(\ldots \leq 2 \times ((\?n - 1) + 2 \times (?n)) \times \|F\|^2\)
\hspace{1em}
by \(\text{rule mult-right-mono, insert deg(3), auto}\)

also have \(\ldots = 2 \times (3 \times (?n - 1)) \times \|F\|^2\) by \textit{simp}

finally have \(\text{ineq-2: } \|g\|^2 \leq 2 \times (3 \times (?n - 1)) \times \|F\|^2\).

from \(\text{power-mono(OF this, of } \?n1)\)
have \(\text{ineq-1: } \|g\|^2 \times \?n1 \leq (2 \times (3 \times (?n - 1)) \times \|F\|^2) \times \?n1\) by \textit{auto}

from \(\text{F0 have norm-F: } \|f\|^2 \geq 1\) using \(\text{sq-norm-poly-pos[of } F]\) by \textit{presburger}

from \(?g\) have \(\text{norm-g: } \|g\|^2 \geq 1\) using \(\text{sq-norm-poly-pos[of } g]\) by \textit{presburger}

from \(\text{F0 have norm-f: } \|f\|^2 \geq 1\) using \(\text{sq-norm-poly-pos[of } f]\) by \textit{presburger}

from \(f1-0\) have \(\text{norm-f1: } \|f1\|^2 \geq 1\) using \(\text{sq-norm-poly-pos[of } f1]\) by \textit{presburger}

from \(\text{power-mono(OF sq-norm-factor-bound(OF } f1F F0],\) of degree } g\)
have \(\text{ineq-2: } \|f1\|^2 \times \text{degree } g \leq (2 \times (2 \times \?n1) \times \|F\|^2) \times \text{degree } g\) by \textit{auto}

also have \(\ldots \leq (2 \times (2 \times \?n1) \times \|F\|^2) \times (?n - 1)\)
\hspace{1em}
by \(\text{rule pow-mono-exp, insert deg-gf normF, auto}\)

finally have \(\text{ineq-2: } \|f1\|^2 \times \text{degree } g \leq (2 \times (2 \times \?n1) \times \|F\|^2) \times (?n - 1)\)
by (rule mult-mono[OF power-both-mono[OF - [mult-mono]
  power-both-mono]], insert normF n1N nN, auto intro: power-both-mono
mult-mono)
also have \ldots = 2 ^ (2 * (?N - 1)) * (?N - 1) + 3 * (?N - 1) * (?N - 1))
  by simp
unfolding power-mult-distrib power-add power-mult by simp
also have 2 * (?N - 1) * (?N - 1) + 3 * (?N - 1) * (?N - 1) = 5 *
  (?N - 1) * (?N - 1) by simp
also have (?N - 1 + (?N - 1)) = 2 * (?N - 1) by simp
also have 2 * (?N - 1) * (?N - 1) * \|F\|^2 \cdot (2 * (?N - 1)) < pl^2 by (rule large)
finally have large: \|f1\|^2 \cdot degree g * \|g\|^2 \cdot degree f1 < pl^2 .
have deg-ug: degree u \leq degree g
proof (rule pl.dvdm-degree[OF mon-u ug], standard)
  assume pl.Mp g = 0
  from ary-cong[OF this, of \lambda p. coeff p (degree g)]
  have pl.M (coeff g (degree g)) = 0 by (auto simp: pl.Mp-def coeff-map-poly)
  from this[unfolded pl.M-def] obtain c where lg: lead-coeff g = pl * c by auto
with g0 have c0: c \neq 0 by auto
hence pl^2 \leq (lead-coeff g)^2 unfolding lg.abs-le-square-iff[symmetric]
  by (rule aux-abs-int)
also have \ldots \leq \|g\|^2 \cdot \|f1\|^2 \cdot degree f1
  by (rule pow-mono-exp, insert deg norm g, auto)
also have \ldots = 1 * \ldots by simp
also have \ldots \leq \|f1\|^2 \cdot degree g * \|g\|^2 \cdot degree f1
  by (rule mult-right-mono, insert norm f1, auto)
also have \ldots < pl^2 by (rule large)
finally show False by auto
qed
from deg deg-u deg-ug have degree f1 > 0 degree g > 0 by auto
from common-factor-via-short[OF this mon-u - uf1 ag large] deg-u pl.m1
  have 0 < degree (gcd f1 g) by auto
moreover from True[unfolded k-def] have degree (gcd f g) = 0 .
moreover have dvd: gcd f1 g dvd gcd f g using f0 unfolding prod by simp
ultimately show False using divides-degree[OF dvd] using f0 by simp
qed
show \{thesis unfolding fs using irr\} by auto
next
case False
define f1 where f1 = f div k
have f: f = f1 * k unfolding f1-def k-def by auto
with ary-cong[OF this, of degree] f0 have deg.f1k: degree f = degree f1 +
  degree k
  by (auto simp: degree-mult-eq)
from f f' have dvd: f1 dvd F k dvd F unfolding dvd-def by auto
obtain gs1 gs2 where part: List.partition (\lambda g. p.dvdm gi f1) us = (gs1,
gs2) by force

note IH = 1(1−2)[OF refl u-def g-def k-def refl, unfolded deg-fk, OF False
f1-def part[symmetric] refl]

  obtain fs1 where fs1: LLL-reconstruction f1 gs1 = fs1 by auto
  obtain fs2 where fs2: LLL-reconstruction k gs2 = fs2 by auto
  from False res[folded f1-def, unfolded part split fs1 fs2]
  have fs: fs = fs1 \& fs2 by auto
  from short-main(1)
  have deg-gf: degree g < degree f unfolding g-def by auto
  from short-main(2)
  have g0: g \neq 0 unfolding g-def by auto
  have deg-k: degree k \leq degree g unfolding k-def gcd.commute[of f g]
    by (rule degree-gcd1[OF g0])
  from deg-gf deg-k have deg-kf: degree k < degree f by auto
  with deg-f1k have deg-f1: degree f1 \neq 0 by auto
  have sf-f: p.square-free-m f using sf fF p.square-free-m-factor unfolding
dvd-def by blast
  from p.unique-factorization-m-factor-partition[OF l0 uf[unfolded plp] f cop
sf-f part]
  have uf: pl.unique-factorization-m f1 (lead-coeff f1, mset gs1)
   pl.unique-factorization-m k (lead-coeff k, mset gs2) by (auto simp: plp)
  have set us = set gs1 \cup set gs2 using part by auto
  with norm have norm-12: gi \in set gs1 \lor gi \in set gs2 \implies pl.Mp gi = gi
  for gi by auto
  note IH1 = IH(1)[OF fs1 deg-f1 uf(1) ded(1) norm-12]
  note IH2 = IH(2)[OF fs2 False uf(2) ded(2) norm-12]
  show \?thesis unfolding fs f using IH1 IH2 by auto
  qed
  qed

lemma LLL-many-reconstruction: assumes LLL-many-reconstruction f us = fs
  and degree f \neq 0
  and poly-mod.unique-factorization-m pl f (lead-coeff f, mset us)
  and f dvd F
  and \( \bigwedge_{ui.} ui. \in set us \implies poly-mod.Mp pl ui = ui \)
  and F0: F \neq 0
  and cop: coprime (lead-coeff F) p
  and sf: poly-mod.square-free-m p F
  and pl1: pl > 1
  and plp: pl = p\^l
  and p: prime p
  and large: \( 2^{<5 \ast \text{degree } F \text{ div } 2 \ast \text{(degree } F \text{ div } 2)} \ast \|F\|^2 \ast \text{(degree } F \text{ div } 2)} \) < pl\^2
  shows f = prod-list fs \land (\forall fi \in set fs. irreducible_d fi)
proof --
  interpret p: poly-mod-prime p by (standard, rule p)
  interpret pl: poly-mod-2 pl by (standard, rule pl1)
  from pl1 plp have ll: l \neq 0 by (cases l, auto)
show ?thesis using assms(1–5)
proof (induct f us arbitrary: fs rule: LLL-many-reconstruction.induct)
  case (1 f us fs)
  note res = 1(3)
  note degf = 1(4)
  note uf = 1(5)
  note FF = 1(6)
  note norm = 1(7)
  note to-fact = pl.unique-factorization-m-imp-factorization
  note fact = to-fact[OF uf]
  have mon-gs: ui ∈ set us ⇒ monic ui for ui using norm fact
    unfolding pl.factorization-m-def by auto
  from p.coprime-lead-coeff-factor[OF p.prime] fF cop
  have cop: coprime (lead-coeff f) p unfolding dvd-def by blast
  have plf0: pl.Mp f ≠ 0
    using fact pl.factorization-m-lead-coeff pl.factorization-m-zero uf by fastforce
  have degree f = pl.degree-m f
    by (rule sym, rule poly-mod.degree-m-eq[OF - pl.m1],
      insert cop p, simp add: l0 p.coprime-exp-mod plp)
  also have ... = sum-mset (image-mset pl.degree-m (mset us))
    unfolding pl.factorization-m-degree[OF fact plf0] ..
  also have ... = sum-list (map pl.degree-m us)
    unfolding sum-mset-sum-list[symmetric] by auto
  also have ... = sum-list (map degree us)
    by (rule arg-cong[OF map-cong, OF refl], rule pl.monic-degree-m, insert
      mon-gs, auto)
  finally have degf-gs: degree f = sum-list (map degree us) by auto
  hence gs: us ≠ [] using degf by (cases us, auto)
  from 1(4) have f0: f ≠ 0 and df0: degree f ≠ 0 by auto
  from norm have norm’: image-mset pl.Mp (mset us) = mset us by (induct
    us, auto)
  have pl0: pl ≠ 0 using pl1 by auto
let ?D2 = degree F div 2
let ?d2 = degree f div 2
define gg where gg = LLL-short-polynomial pl (Suc ?d2)
let ?us = filter (λu. degree u ≤ ?d2) us
  note res = res[unfolded LLL-many-reconstruction.simps[OF f us], unfolded
    Let-def, folded gg-def]
let ?f2-opt = find-map-filter (λu. gcd f (gg u))
  (λf2. 0 < degree f2 ∧ degree f2 < degree f) ?us
show ?case
proof (cases ?f2-opt)
  case (Some f2)
  from find-map-filter-Some[OF this]
  obtain g where deg-f2: degree f2 ≠ 0 degree f2 < degree f
    and dvd: f2 dvd f and gcd: f2 = gcd f g by auto
note res = res[unfolded Some option.simps]

define f1 where f1 = f div f2
have f: f = f1 * f2 unfolding f1-def using dvd by auto
with ary-cong[OF this, of degree] f0 have deg-sum: degree f = degree f1 +
degree f2
  by (auto simp: degree-mult-eq)
with deg-f2 have deg-f1: degree f1 ≠ 0 degree f < degree f by auto
from f f' have dvd: f1 dvd F f2 dvd F unfolding dvd-def by auto
obtain gs1 gs2 where part: List.partition (λg. p.dvdn gi f1) us = (gs1,
gs2) by force
note IH = 1(1-2)[OF refl refl refl, unfolded Let-def, folded gg-def, OF Some
f1-def part[ symmetric] refl]
  obtain fs1 where fs1: LLL-many-reconstruction f1 gs1 = fs1 by blast
  obtain fs2 where fs2: LLL-many-reconstruction f2 gs2 = fs2 by blast
from res[folded f1-def, unfolded part split fs1 fs2]
  have fs: fs = fs1 @ fs2 by auto
  have sf-f: p.square-free-m f using sf f F p.square-free-m-factor unfolding
dvd-def by blast
  from p.unique-factorization-m-factor-partition[OF l0 uf [unfolded plp] f cop
sf-f part]
  have uf: pl.unique-factorization-m f1 (lead-coeff f1, mset gs1)
    pl.unique-factorization-m f2 (lead-coeff f2, mset gs2) by (auto simp: plp)
  have set us = set gs1 ∪ set gs2 using part by auto
  with norm have norm-12: gi ∈ set gs1 ∨ gi ∈ set gs2 → pl.Mp gi = gi
for gi by auto
note IH1 = IH(1)[OF fs1 deg-f1(1) uf(1) dvd(1) norm-12]
note IH2 = IH(2)[OF fs2 deg-f2(1) uf(2) dvd(2) norm-12]
show ?thesis unfolding fs f using IH1 IH2 by auto
next
  case None
  from res[unfolded None option.simps] have fs-f: fs = [f] by simp
  from sf fs f' have sf: p.square-free-m f
    using p.square-free-m-factor(1)[of f] unfolding dvd-def by auto
  have irreducible f
proof (rule ccontr)
  assume ¬ irreducible f
  from reducible E[OF this] degf obtain f1 f2 where
    f: f = f1 * f2 and
degf12: degree f1 ≠ 0 degree f2 ≠ 0 degree f1 < degree f degree f2 < degree f
    by (simp, metis)
  from f0 have degree f = degree f1 + degree f2 unfolding f
    by (auto simp: degree-mult-eq)
  hence degree f1 ≤ degree f div 2 ∨ degree f2 ≤ degree f div 2 by auto
  then obtain f1 f2 where
    f: f = f1 * f2 and
degf12: degree f1 ≠ 0 degree f2 ≠ 0 degree f1 ≤ degree f div 2 degree f2 <
degree f
proof (standard, goal-cases)

  case 1
  from 1(1)[of f1 f2] 1(2) f deg12 show ?thesis by auto
  next
  case 2
  from 2(1)[of f2 f1] 2(2) f deg12 show ?thesis by auto
qed

from f0 f have f10: f1 ≠ 0 by auto
from sf f have sf1: p.square-free-m f1
  using p.square-free-m-factor(1)[of f1] by auto
from p.coprime-lead-coeff-factor[OF p.prime cop[unfolded f]]
have cop1: coprime (lead-coeff f1) p by auto
have deg12: pl.degree-m f1 = degree f1
  by (rule poly-mod.degreem-eq[OF - pl.m1],
  insert cop1 p, simp add: l0 p.coprime-exp-mod plp)
from pl.unique-factorization-m-factor[OF p uf[unfolded f], folded f, OF cop
sf l0 plp]
  obtain us1 us2 where
  uf12: pl.unique-factorization-m f1 (lead-coeff f1, us1)
  pl.unique-factorization-m f2 (lead-coeff f2, us2)
  and gs: mset us = us1 + us2
  and norm12: image-mset pl.Mp us2 = us2 image-mset pl.Mp us1 = us1
  unfolding pl.Mf-def norm' split by (auto simp: pl.Mf-def)
from gs have x ∈# us1 ⇒ x ∈# mset us for x by auto
hence sub1: x ∈# us1 ⇒ x ∈ set us for x by auto
from to-fact[sf uf12(1)]
have fact1: pl.unique-factorization-m f1 (lead-coeff f1, us1) .
have plf10: pl.Mp f1 ≠ 0
  using fact1 pl.unique-factorization-m-lead-coeff pl.unique-factorization-m-zero
uf12(1) by fastforce
  have degree f1 = pl.degree-m f1 using deg1 by simp
  also have .. = sum-mset (image-mset pl.degree-m us1)
  unfolding pl.unique-factorization-m-degree[OF fact1 plf10] ..
  also have .. = sum-mset (image-mset degree us1)
    by (rule arg-cong[of - sum-mset], rule image-mset-cong,
    rule pl.monic-degree-m, rule mon-gs, rule sub1)
  finally have deg1-sum: degree f1 = sum-mset (image-mset degree us1)
by auto
with deg12 have us1 ≠ {#} by auto
then obtain u us11 where us1: us1 = {#u#} + us11
  by (cases us1, auto)
hence u1: u ∈# us1 by auto
hence u: u ∈ set us by (rule sub1)
let ?g = gg u
from pl.factorization-m-mem-dvdm[OF fact1, of u] a1 have u-f1: pl.dvdm
  u f1 by auto
note norm-u = norm[OF OF u] from fact u have irred: pl.irreducible₄-m u unfolding pl.factorization-m-def by auto
hence \(\deg u\): \(\deg u \neq 0\) unfolding \(\texttt{pl.irreducible}_{d-m\text{-def}}\) \(\texttt{norm}[OF\ u]\) by \(\texttt{auto}\)

\(\begin{align*}
&\text{have } \deg u \leq \deg f1 \text{ unfolding } \texttt{degf1-sum}\text{ unfolding }\texttt{us1} \text{ by } \texttt{simp} \\
&\text{also have } \ldots \leq \deg f \div 2 \text{ by } \texttt{fact} \\
&\text{finally have } \deg uf: \deg u \leq \deg f \div 2.
\end{align*}\)

\(\begin{align*}
&\text{hence } \deg uf: \deg u \leq \text{Suc} (\deg f \div 2) \deg u < \text{Suc} (\deg f \div 2) \text{ by } \texttt{auto} \\
&\text{from } \texttt{mon-gs}[OF\ u] \text{ have } \texttt{mon-u}: \texttt{monic} u.
\end{align*}\)

\(\texttt{note short} = \texttt{LLL-short-polynomial}[OF\ \deg u\ \deg uf'(1)\ \texttt{pl1}\ mon-u,\ folded\ \gg\text{-def}]\)

\(\begin{align*}
&\text{from } \texttt{short}(1-3) \texttt{short(4)}[OF\ \deg uf'(2)] \\
&\text{have } \ldots < \deg f \text{ using } \texttt{degf by simp}
\end{align*}\)

\(\begin{align*}
&\text{finally have } \deg (\gcd f \ ?g) < \deg f \text{ by simp} \\
&\text{with } \texttt{find-map-filter-None}[OF\ None,\ simplified,\ rule-format,\ of\ u] \texttt{deg uf u} \\
&\text{have } \gcd uf: \deg (\gcd f \ ?g) = 0 \text{ by } (\texttt{auto simp: gcd.commute}) \\
&\text{have } \gcd f1 \ ?g \ dvd \ \gcd f \ ?(\gcd f1 \ ?g) \text{ unfolding } f1 \text{ by simp}
\end{align*}\)

\(\begin{align*}
&\text{from } \texttt{divides-degree}[OF\ this,\ unfolded\ \deg-gcd] \texttt{f0} \\
&\text{have } \deg-gcd1: \deg (\gcd f1 \ ?g) = 0 \text{ by } \texttt{auto} \\
&\text{from } \texttt{from F0}\ \texttt{have}\ \texttt{normF}: \|F\|^2 \geq 1 \text{ using } \texttt{sq-norm-poly-pos}[OF\ F]\ \texttt{by}\ \texttt{presburger} \\
&\text{have } \texttt{g0}: \?g \neq 0 \text{ using } \texttt{short}(2).
\end{align*}\)

\(\begin{align*}
&\text{from } \texttt{g0} \texttt{have } \texttt{normg}: \|?g\|^2 \geq 1 \text{ using } \texttt{sq-norm-poly-pos}[OF\ ?g]\ \texttt{by}\ \texttt{presburger} \\
&\text{from } \texttt{f10}\ \texttt{have}\ \texttt{normf1}: \|f1\|^2 \geq 1 \text{ using } \texttt{sq-norm-poly-pos}[OF\ f1]\ \texttt{by}\ \texttt{presburger}
\end{align*}\)

\(\texttt{from } \texttt{fF}\ \texttt{have } \texttt{fF}: f1 \ dvd\ F \texttt{ unfolding } \texttt{dvd-def}\ \texttt{by}\ \texttt{auto} \\
\texttt{have } \texttt{pl-gd0}: \texttt{pl} \geq 0 \texttt{ using } \texttt{pl.poly-mod-2-axioms poly-mod-2-def}\ \texttt{by}\ \texttt{auto} \\
\texttt{from } \texttt{fF}\ \texttt{have } \deg f \leq \deg F \texttt{ using } \texttt{f0}\ \texttt{f0}\ \texttt{by}\ (\texttt{metis dvd-imp-deg-degree-le}) \\
\texttt{hence } d2D2: \?d2 \leq \?D2\ \texttt{by}\ \texttt{simp} \\
\texttt{with } \texttt{deg12(3)}\ \texttt{have}\ df1-D2: \deg f1 \leq \?D2\ \texttt{by}\ \texttt{linarith} \\
\texttt{from } \texttt{short(1)} \texttt{d2D2}\ \texttt{have}\ dg-D2: \deg (\gg\ u) \leq \?D2\ \texttt{by}\ \texttt{linarith} \\
\texttt{have } \|f1\|^2 \ ? \deg (\gg\ u) \ * \ |\?g\ u|\|^2 \ ? \deg f1 \\
\texttt{\leq } \|f1\|^2 \ ? \?D2 \ * |\?g\ u|\|^2 \ ? \?D2 \\
\texttt{by } (\texttt{rule power-mono}[OF\ \texttt{pow-mono-exp}\ \texttt{pow-mono-emp}], \\
\texttt{insert}\ \texttt{normf1}\ \texttt{normg},\ \texttt{auto intro: df1-D2\ dg-D2}) \\
\texttt{also have } \ldots = (\|f1\|^2 \ * |\?g\ u|\|^2) \ ? \?D2 \\
\texttt{by } (\texttt{simp add: power-mult-distrib}) \\
\texttt{also have } \ldots \leq (\|f1\|^2 \ * (2 \ * \?D2 \ * \|f1\|^2)) \ ? \?D2 \\
\texttt{by } (\texttt{rule power-mono}[OF\ \texttt{mult-left-mono}[OF\ \texttt{order.trans}[OF\ short(4)][OF\ f10\ u-f1]])
\end{align*}\)

\(\begin{align*}
&\texttt{insert deg12 d2D2, auto intro!: mult-mono} \\
&\texttt{also have } \ldots = \|f1\|^2 \ - \(\?D2 + \?D2) \ * 2 ^城市\(\?D2 \ * \?D2) \\
&\texttt{unfolding power-add power-mult-distrib power-mult}\ \texttt{by}\ \texttt{simp} \\
&\texttt{also have } \ldots \leq (2 ^城市\((2 \ * \?D2) \ * \|F\|^2)) \ - \(\?D2 + \?D2) \ * 2 ^城市\(\?D2 \ * \?D2) \\
&\texttt{by}\ (\texttt{rule power-right-mono}[OF\ \texttt{order.trans}[OF\ power-mono][OF\ \texttt{sq-norm-factor-bound}[OF\ f1\ F0]])
\end{align*}\)

\(\texttt{auto intro!: power-mono\ mult-right-mono\ df1-D2}) \\
\texttt{also have } \ldots = 2 ^城市\((2 \ * \?D2) \ * \(\?D2 + \?D2) \ + \?D2 \ * \?D2) \ * \|F\|^2 \) \ -
\[(?D^2 + ?D^2)\]

unfolding power-mult-distrib power-mult power-add by simp
also have \(2 \ast (?D^2 + ?D^2) + ?D^2 \ast ?D^2 = 5 \ast ?D^2 \ast ?D^2\) by simp
finally have large:
\[
\|f1\|^2 \circ \text{degree} (gg u) \ast \|gg u\|^2 \circ \text{degree} f1 < pl \ast 2 \text{ using large by simp}
\]
proof (rule pl.dvdn-degree[OF mon-u short(3)], standard)
assume \(pl.Mp (?g) = 0\)
from arg-cong[OF this, of \(\lambda p.\) coeff p (degree ?g)]
have \(pl.M (\text{coeff} ?g (\text{degree} ?g)) = 0\) by (auto simp: pl.Mp-def coeff-map-poly)
from this[unfolded pl.M-def] obtain \(c\) where lg: \(\text{lead-coeff} ?g = pl \ast c\)
by auto
with \(g0\) have \(c0: c \neq 0\) by auto
hence \(pl \ast 2 \leq (\text{lead-coeff} ?g) \ast 2\) unfolding lg abs-le-square-iff[ symmetric]
by (rule aux- abs-int)
also have \(\ldots \leq \|\?g\|^2\) using coeff-le-sq-norm[of ?g] by auto
also have \(\ldots = \|\?g\|^2 \circ 1\) by simp
also have \(\ldots \leq \|\?g\|^2 \circ \text{degree} f1\)
by (rule pow-mono-exp, insert deg12 normf1, auto)
also have \(\ldots = 1 \ast \ldots\) by simp
also have \(\ldots \leq \|f1\|^2 \circ \text{degree} ?g \ast \|?g\|^2 \circ \text{degree} f1\)
by (rule mult-right-mono, insert normf1, auto)
also have \(\ldots < \|f1\|^2\) by (rule large)
finally show \(\text{False}\) by auto
qed
with \(\text{deg-u}\) have \(\text{deg-g}: 0 < \text{degree} (gg u)\) by auto
have \(pl\)-deg0: \(pl \geq 0\) using pl.poly-mod-2-axioms poly-mod-2-def by auto
from \(fF\) have \(\text{degree} f \leq \text{degree} F\) using F0F0 by (metis dvd-imp-degree-le)
hsence \(d2D2: ?d2 \leq ?D2\) by simp
with deg12(3) have df1-D2: \(\text{degree} f1 \leq ?D2\) by linarith
from short(1) \(d2D2\) have dg-D2: \(\text{degree} (gg u) \leq ?D2\) by linarith
have \(0 < \text{degree} f1 \ast 0 < \text{degree} u\) using deg12 deg-u by auto
from common-factor-via-short[of f1 gg u, OF this(1) deg-g mon-u this(2)]
\(u-f1\) short(3) - \(pl\)-deg0] deg-gcd1
have \(pl \ast 2 \leq \|f1\|^2 \circ \text{degree} (gg u) \ast \|gg u\|^2 \circ \text{degree} f1\) by linarith
also have \(\ldots < \|f1\|^2\) by (rule large)
finally show \(\text{False}\) by simp
qed
thus \(\text{thesis}\) using fs-f by simp
qed
qed
qed
end

lemma \(\text{LLL-factorization}\):
assumes res: \(\text{LLL-factorization} f = gs\)
and sff: square-free f
and deg: degree f ≠ 0
shows f = prod-list gs ∧ (∀ g∈ set gs. irreducible g)
proof –
let ?lc = lead-coeff f
define p where p ≡ suitable-prime-bz f
obtain res gs where fff: finite-field-factorization-int p f = (c,gs) by force
let ?degs = map degree gs
note res = res[unfolded LLL-factorization-def Let-def, folded p-def, unfolded fff split, folded]
from suitable-prime-bz[OF sff refl]
have prime: prime p and cop: coprime ?lc p and sf: poly-mod.square-free-m p f
  unfolding p-def by auto
note res from prime interpret p: poly-mod-prime p by unfold-locales
define K where K = 2^((degree f - 1) * (degree f - 1)) * ∥f∥^2 * (2 * (degree f - 1))
define N where N = sqrt-int-ceiling K
have K0: K ≥ 0 unfolding K-def by fastforce
have N0: N ≥ 0 unfolding N-def sqrt-int-ceiling using K0
by (smt of-int-nonneg real-sqrt-ge-0-iff zero-le-ceiling)
define n where n = find-exponent p N
note res = res[folded n-def[unfolded N-def K-def]]
note n = find-exponent[OF p.m1, of N, folded n-def]
note bh = p.berlekamp-and-hensel-separated(1)[OF cop sf refl fff n(2)]
from deg have f0: f ≠ 0 by auto
from n p.m1 have pn1: p ^ n > 1 by auto
note res = res[folded bh(1)]
note * = p.berlekamp-hensel-unique[OF cop bh n(2)]
note ** = p.berlekamp-hensel-main[OF n(2) bh cop sf fff]
from res * **
have uf: poly-mod.unique-factorization-m (p ^ n) f (lead-coeff f, mset (berlekamp-hensel p n f))
  and norm: \( \bigwedge_{ui. ui ∈ set (berlekamp-hensel p n f)} \Rightarrow poly-mod.Mp (p ^ n)\)
  = ui = ui
  unfolding berlekamp-hensel-def fff split by auto
have K: K < (p ^ n)^2 using n sqrt-int-ceiling-bound[OF K0]
  by (smt N0 N-def n(1) power2-le-imp-le)
show ?thesis
  by (rule LLL-implementation.LLL-reconstruction[OF res deg uf dvd-refl norm f0 cop sf pn1]
    refl prime K[unfolded K-def]])
qed

lemma LLL-many-factorization:
  assumes res: LLL-many-factorization f = gs
  and sff: square-free f
  and deg: degree f ≠ 0
shows \( f = \text{prod-list } gs \land (\forall g \in \text{set } gs. \text{irreducible}_d g) \)

proof –

let \(?lc = \text{lead-coeff } f\)

define \( p \) where \( p \equiv \text{suitable-prime-bz } f \)

obtain \( c \) \( gs \) where \( \text{fff}: \text{finite-field-factorization-def } p \ f = (c, gs) \) by force

let \(?degs = \text{map degree } gs\)

note \( \text{res } gs \) where \( fff: \text{finite-field-factorization-def } p \ f = (c, gs) \) by force

let \(?degs = \text{map degree } gs\)

note \( \text{res } gs \) where \( fff: \text{finite-field-factorization-def } p \ f = (c, gs) \) by force

define \( K \) where \( K = 2^5 (\text{degree } f \text{ div } 2) \star (\text{degree } f \text{ div } 2) \star \|f\|^2 \star (2 \star (\text{degree } f \text{ div } 2)) \)

define \( N \) where \( N = \text{sqrt-int-ceiling } K \)

have \( \text{K0: } K \geq 0 \) unfolding \( \text{K-def} \) by fastforce

have \( \text{N0: } N \geq 0 \) unfolding \( \text{N-def sqrt-int-ceiling} \) using \( \text{K0} \)

by (smt of-int-nonneg real-sqrt-ge-0-iff zero-le-ceiling)

define \( n \) where \( n = \text{find-exponent } p \ N \)

note \( \text{res } n \) where \( \text{res } n \equiv \text{find-exponent}[\text{OF } p.\text{m1}, \text{of } N, \text{folded } n\text{-def}] \)

note \( \text{bh} = p.\text{berlekamp-and-hensel-separated}(1)[\text{OF } \text{cop } sf \ \text{refl } fff \ n(2)] \)

from \( \text{deg have f0: } f \neq 0 \) by auto

from \( n p.m1 \) have \( \text{pn1: } p \ ^\star n > 1 \) by auto

note \( \text{res } bh \) where \( \text{res } bh \equiv \text{folded } bh(1) \)

note \( \text{bh} = p.\text{berlekamp-and-hensel-separated}(1)[\text{OF } \text{cop } sf \ \text{refl } fff \ n(2)] \)

from \( \text{res } ** \) have \( \text{uf: } \text{poly-mod-unique-factorization-m } (p \ ^\star n) \ f \) (\text{lead-coeff } f, \ \text{mset } (\text{berlekamp-hensel } p n f))

and \( \text{norm } \forall u.i. \ u.i \in \text{set } (\text{berlekamp-hensel } p n f) \implies \text{poly-mod-Mp } (p \ ^\star n) \)

\( u.i = u.i \)

unfolding \( \text{berlekamp-hensel-def} \) \( \text{fff split by auto} \)

have \( \text{K: } K < (p \ ^\star n)^2 \) using \( n \) \( \text{sqrt-int-ceiling-bound}[\text{OF } \text{K0}] \)

by (smt \( \text{N0 N-def n(1) power2-le-imp-le} \)

show \( \text{thesis} \)

by (rule \( \text{LLL-implementation.LLL-many-reconstruction}[\text{OF } \text{res } \text{deg } u.i \ \text{dvd-refl} \ \text{norm } f0 \ \text{cop } sf \ \text{pn1} \)

\( \text{refl prime } K[\text{unfolded } K\text{-def}]] \)

qed

lift-definition one-lattice-LLL-factorization :: \text{int-poly-factorization-algorithm}

is \text{LLL-factorization using LLL-factorization by auto} \)

lift-definition many-lattice-LLL-factorization :: \text{int-poly-factorization-algorithm}

is \text{LLL-many-factorization using LLL-many-factorization by auto} \)
lemma LLL-factorization-primitive: assumes LLL-factorization f = fs
square-free f
\(\emptyset < \deg f\)
primitive f
shows f = prod-list fs \& (\forall fi \in set fs. irreducible fi \& \(\emptyset < \deg fi \& primitive fi\))
using assms(1)
by intro int-poly-factorization-algorithm-irreducible[of one-lattice-LLL-factorization,
OF - assms(2-)], transfer, auto

thm factorize-int-poly[of one-lattice-LLL-factorization]
thm factorize-int-poly[of many-lattice-LLL-factorization]
end

6 Calculating All Possible Sums of Sub-Multisets

theory Sub-Sums
imports
Main
HOL-Library.Multiset
begin

fun sub-mset-sums :: 'a :: comm_monoid_add list \Rightarrow 'a set where
sub-mset-sums [] = {0}
| sub-mset-sums (x # xs) = (let S = sub-mset-sums xs in S \cup \{ (+) x \} \cdot S)

lemma subset-add-mset: ys \subseteq # add-mset x zs \iff (ys \subseteq# zs \lor (\exists xs. xs \subseteq# zs \land ys = add-mset x xs))
(is \?l = \?r)
proof
have sub: ys \subseteq# zs \implies ys \subseteq# add-mset x zs
by (metis add-mset-remove-trivial diff-subset_eq_self subset_mset.dual_order.trans)
assume \?r
thus \?l using sub by auto
next
assume l: \?l
show \?r
proof (cases x \in# ys)
  case True
  define xs where xs = (ys - \{ # x \})
  from True have ys: ys = add-mset x xs unfolding xs-def by auto
  from \{unfolded ys\} have xs \subseteq# zs by auto
  thus \?r unfolding ys by auto
next
  case False
with \?l have ys \subseteq# zs by (simp add: subset_mset.le_iff_sup)
  thus \?thesis by auto
qed
qed

46
lemma sub-mset-sums[simp]: sub-mset-sums xs = sum-mset ' { ys. ys ⊆# mset xs }
proof (induct xs)
case (Cons x xs)
  have id: { ys. ys ⊆# mset (x # xs)} = {ys. ys ⊆# mset xs} ∪ {add-mset x ys | ys. ys ⊆# mset xs}
  unfolding mset.simps subset-add-mset by auto
  show ?case unfolding sub-mset-sums.simps Let-def Cons id image-Un
  by force
qed auto

7 Implementation and soundness of a modified version of Algorithm 16.22

Algorithm 16.22 is quite similar to the LLL factorization algorithm that was verified in the previous section. Its main difference is that it has an inner loop where each inner loop iteration has one invocation of the LLL basis reduction algorithm. Algorithm 16.22 of the textbook is therefore closer to the factorization algorithm as it is described by Lenstra, Lenstra, and Lovász [3], which also uses an inner loop.

The advantage of the inner loop is that it can find factors earlier, and then small lattices suffice where without the inner loop one invokes the basis reduction algorithm on a large lattice. The disadvantage of the inner loop is that if the input is irreducible, then one cannot find any factor early, so that all but the last iteration have been useless: only the last iteration will prove irreducibility.

We will describe the modifications w.r.t. the original Algorithm 16.22 of the textbook later in this theory.
shows \( \exists b \in \text{set } xs, \ a \text{ dvd } b \)

proof –
let \(?A\) = (of-int-poly \( a \))::'a mod-ring poly
let \(?XS\) = (map of-int-poly \( xs \))::'a mod-ring poly
let \(?XS1\) = (of-int-poly (prod-list \( xs \)))::'a mod-ring poly

have [transfer-rule]: MP-Rel \( a \) \(?A\)
  by (simp add: MP-Rel-def Mp-f-representative)
have [transfer-rule]: MP-Rel (prod-list \( xs \)) \(?XS\)
  by (simp add: MP-Rel-def Mp-f-representative)
have [transfer-rule]: list-all2 MP-Rel \( xs \) \(?XS1\)
  by (simp add: MP-Rel-def Mp-f-representative list-all2-conv-all-nth)

have \(?A \text{ dvd } ?XS1\) using dvd by transfer
have \( \exists b \in \text{set } ?XS, \ ?A \text{ dvd } b\)
  by (rule irreducible-dvd-prod-list, insert irr, transfer, auto simp add: \(?A\))

from this[untransferred] show \(?\text{thesis}\).

qed

end

lemma (in poly-mod-prime) irreducible-m-dvdm-prod-list:
  assumes irr: irreducible-m \( a \)
  and dvd: \( a \text{ dvdm } \) (prod-list \( xs \))
  shows \( \exists b \in \text{set } xs, \ a \text{ dvd } b\)
  by (rule poly-mod-prime-type irreducible-m-dvdm-prod-list-connect unfolded poly-mod-type-simps,

  internalize-sort 'a :: prime-card, \( \text{OF type-to-set, unfolded remove-duplicate-premise,}\n
  cancel-type-definition, \( \text{OF non-empty irr dvd}\))

7.2 The modified version of Algorithm 16.22

definition B2-LLL :: int poly \Rightarrow int where
  B2-LLL \( f \) = \( 2 \times (2 \times \text{degree } f) \times \| f \|^2 \)

hide-const (open) factors
hide-const (open) factors
hide-const (open) factor
hide-const (open) factor

context
  fixes \( p :: \text{int and } l :: \text{nat} \)
begin

context
  fixes \( gs :: \text{int poly list} \)
  and \( f :: \text{int poly} \)
  and \( u :: \text{int poly} \)
  and \( \text{Degs :: nat set} \)
begin

48
This is the critical inner loop.

In the textbook there is a bug, namely that the filter is applied to $g'$ and not to the primitive part of $g'$. (Problems occur if the content of $g'$ is divisible by $p$.) We have fixed this problem in the obvious way.

However, there also is a second problem, namely it is only guaranteed that $g'$ is divisible by $u$ modulo $p^j$. However, for soundness we need to know that then also the primitive part of $g'$ is divisible by $u$ modulo $p^j$. This is not necessary true, e.g., if $g' = p^j$, then the primitive part is 1 which is not divisible by $u$ modulo $p^j$. It is open, whether such a large $g'$ can actually occur. Therefore, the current fix is to manually test whether the leading coefficient of $g'$ is strictly smaller than $p^j$.

With these two modifications, Algorithm 16.22 will become sound as proven below.

\[ \text{definition \ LLL-reconstruction-inner \ j \equiv} \]
\[ \text{let \ j' = j - 1 in} \]
\[ \text{— optimization: check whether degree j' is possible} \]
\[ \text{if \ j' \notin \ Degs \ then \ None \ else} \]
\[ \text{— short vector computation} \]
\[ \text{let} \]
\[ \quad ll = (\text{let} \ n = \text{sqrt-int-ceiling} (\|f\|^2 \cdot (2 * j') \cdot 2 ^ (5 * j' * j'))); \]
\[ \quad ll' = \text{find-exponent} \ p \ n \ \text{in} \quad \text{if} \ ll' < l \ \text{then} \ ll' \ \text{else} \ l; \]
\[ \text{— optimization: dynamically adjust the modulus} \]
\[ \quad pl = p \cdot ll; \]
\[ \quad g' = \text{LLL-short-polynomial} \ pl \ j \ u \]
\[ \text{— fix: forbid multiples of p^j as short vector, unclear whether this is really required} \]
\[ \text{in} \quad \text{if abs (lead-coeff g')} \geq \ pl \ \text{then} \ None \ \text{else} \]
\[ \text{let ppg = primitive-part g'} \]
\[ \text{— slight deviation from textbook: we check divisibility instead of norm-inequality} \]
\[ \text{case div-int-pol \ f \ ppg \ of \ Some \ f' \Rightarrow} \]
\[ \text{— fix: consider modular factors of ppg and not of g'} \]
\[ \quad \text{Some \ (filter (λg. \ ¬ \ poly-mod.dedm \ p \ gi \ ppg) \ gs, \ lead-coeff f', f', ppg)} \]
\[ \quad | \ \text{None} \ \Rightarrow \ None \]

\[ \text{function \ LLL-reconstruction-inner-loop \ where} \]
\[ \text{LLL-reconstruction-inner-loop \ j =} \]
\[ \text{(if} \ j > \text{degree f \ then} \ ([],1,1,f) \]
\[ \text{else case LLL-reconstruction-inner \ j} \]
\[ \text{of \ Some \ tuple \ ⇒ \ tuple} \]
\[ \text{| \ None} \ \Rightarrow \ \text{LLL-reconstruction-inner-loop \ (j+1)} \]
\[ \text{by \ auto} \]
\[ \text{termination \ by \ (relation \ measure \ (λ \ j. \ Suc \ (degree \ f) - j), \ auto)} \]

end
partial-function (tailrec) LLL-reconstruction" where [code]:
LLL-reconstruction" gs b f factors =
(if gs = [] then factors
else
  let u = choose-u gs;
  d = degree u;
  gs' = remove1 u gs;
  degs = map degree gs';
  Degs = ((+ d) · sub-mset-sums degs);
  (gs', b', f', factor) = LLL-reconstruction-inner-loop gs f u Degs (d+1)
in LLL-reconstruction" gs' b' f' (factor#factors)
)

definition reconstruction-of-algorithm-16-22 gs f ≡
let G = [];
  b = lead-coeff f
in LLL-reconstruction" gs b f G
end

definition factorization-algorithm-16-22 :: int poly ⇒ int poly list where
factorization-algorithm-16-22 f = (let
  — find suitable prime
  p = suitable-prime-bz f;
  — compute finite field factorization
  (−, fs) = finite-field-factorization-int p f;
  — determine l and B
  n = degree f;
  — bound improved according to textbook, which uses \( no = (n + 1) \cdot (\max - \| f \|^2) \)
  no = \| f \|^2;
  — possible improvement: \( B = \sqrt{(2^{5+n}(n-1)} \cdot no^{2n-1} \), cf. \( LLL\)-factorization
  B = sqrt-int-ceiling (2 · (5 * n * n) * no · (2 * n));
  l = find-exponent p B;
  — perform hensel lifting to lift factorization to mod \( p^l \)
  vs = hensel-lifting p l f fs
  — reconstruct integer factors
in reconstruction-of-algorithm-16-22 p l vs f)

7.3 Soundness proof
7.3.1 Starting the proof

Key lemma to show that forbidding values of \( p^l \) or larger suffices to find correct factors.

lemma (in poly-mod-prime) Mp-smult-p-removal: poly-mod.Mp (p * p ^ k) (smult p f) = 0 ⇒ poly-mod.Mp (p ^ k) f = 0
  by (smt add.left-neutral m1 poly-mod.Dp-Mp-eq poly-mod.Mp-smult-m-0 sdiv-poly-smult smult-smult)
lemma (in poly-mod-prime) eq-m-smult-p-removal: poly-mod.eq-m (p * p ^ k) (smult p f) (smult p g) 
   ==> poly-mod.eq-m (p ^ k) f g using Mp-smult-p-removal[of k f - g] 
   by (metis add-diff-cancel-left' diff-add-cancel diff-self poly-mod.Mp-0 poly-mod.minus-Mp(2) 
       smult-diff-right)

lemma content-le-lead-coeff: abs (content (f :: int poly)) \leq abs (lead-coeff f) 
proof (cases f = 0)
  case False 
  from content-dvd-coeff[of f degree f] have abs (content f) dvd abs (lead-coeff f) 
  by auto 
moreover have abs (lead-coeff f) \neq 0 using False by auto 
ultimately show \(?\)thesis by (smt dvd-imp-le-int)
qed auto

lemma poly-mod-dvd-drop-smult: assumes u: monic u and p: prime p and c: c \neq 0 | c| < p ^ l 
   and dvd: poly-mod.dvdm (p ^ l) u (smult c f) 
shows poly-mod.dvdm p u f 
using c dvd 
proof (induct l arbitrary: c rule: less-induct)
  case (less l c)
interpret poly-mod-prime p by (unfold-locales, insert p, auto)
note c = less(2-3)
note dvd = less(4)
note IH = less(1)
show \(?\)case 
proof (cases p dvd c)
  case False 
  let ?i = inverse-mod c (p ^ l) 
  have gcd c p = 1 using p False 
  by (metis Primes.prime-int-iff gcd-ge-0-int semiring-gcd-class.gcd-dvd1 
      semiring-gcd-class.gcd-dvd2) 
  hence coprime c p by (metis dvd-refl gcd-dvd-1)
  from pl.inverse-mod-coprime-exp[OF refl p l0 this] 
  have id: pl.M (?i * c) = 1 .
  have pl.Mp (smult ?i (smult c f)) = pl.Mp (smult (pl.M (?i * c)) f) by simp 
  also have ... = pl.Mp f unfolding id by simp
  finally have pl.dvdm u f using pl.dvdm-smult[OF dvd, of ?i] unfolding 
     pl.dvdm-def by simp 
  thus u dvdm f using l0 pl-dvdm-imp-p-dvdm by blast
next

case True
  then obtain d where cpd: c = p * d unfolding dvd-def by auto
from cpd c have d0: d ≠ 0 by auto
note to-p = Mp-Mp-pow-is-Mp[OF l0 m1]
from dvd obtain v where eq: pl.eq-m (u * v) (smult p (smult d f))
  unfolding pl.dvdm-def cpd by auto
from arg-cong[OF this, of Mp, unfolded to-p]
have Mp (u * v) = 0 unfolding Mp-smult-m-0 .
with u have Mp v = 0
  by (metis Mp-0 add-eq-0-iff-both-eq-0 degree-0
degree-m-mult-eq monic-degree-0 monic-degree-m mult-cancel-right2)
from Mp-0-smult-sdiv-poly[OF this]
obtain w where v: v = smult p w by metis
with eq have eq: pl.eq-m (smult p (u * w)) (smult p (smult d f)) by simp
from l0 obtain ll where l = Suc ll by (cases l, auto)
hence pl: p `l = p * p `ll and ll: ll < l by auto
from c(2) have d-small: |d| < p `ll unfolding pl cpd abs-mult
  using mult-less-cancel-left-pos[of p d p `ll] m1 by auto
from eq-m-smult-p-removal[OF eq[unfolded pl]]
have poly-mod.eq-m (p `ll) (u * w) (smult d f) .
hence dvd: poly-mod.dvdm (p `ll) u (smult d f) unfolding poly-mod.dvdm-def
by metis
  show ?thesis by (rule IH[OF ll d0 d-small dvd])
qed
qed

context
  fixes p :: int
  and F :: int poly
  and N :: nat
  and l :: nat
  defines [simp]: N ≡ degree F
  assumes p: prime p
    and N0: N > 0
    and bound-l: 2 ^ N^2 * B2-LLL F ^ (2 * N) ≤ (p `l)^2
begin

private lemma F0: F≠0 using N0
  by fastforce

private lemma p1: p > 1 using p prime-gt-1-int by auto

interpretation p: poly-mod-prime p using p by unfold-locales

interpretation pl: poly-mod p `l.

lemma B2-2: 2 ≤ B2-LLL F

52
proof
  from F0 have \|F\|^2 \neq 0 by simp
  hence F1: \|F\|^2 \geq 1 using sq-norm-poly-pos[of F] F0 by linarith
  have (2 :: int) = 2 * 1 + 1 by simp
  also have \ldots \leq B2-LLL F unfolding B2-LLL-def
    by (intro mult-mono power-increasing F1, insert N0, auto)
  finally show 2 \leq B2-LLL F .
qed

lemma l-gt-0: l > 0
proof (cases l)
case 0
  have 1 * 2 \leq 2 \cdot N^2 \cdot B2-LLL F \cdot (2 * N)
  proof (rule mult-mono)
    have 2 * 1 \leq (2 :: int) * (2 \cdot (2 * N - 1)) by (rule mult-left-mono, auto)
    also have \ldots = 2 \cdot (2 * N) using N0 by (cases N, auto)
    also have \ldots \leq B2-LLL F \cdot (2 * N)
      by (rule power-mono[OF B2-2], force)
    finally show 2 \leq B2-LLL F \cdot (2 * N) by simp
  qed auto
  also have \ldots \leq 1 using bound-l[unfolded 0] by auto
  finally show \?thesis by auto
qed auto

lemma l0: l \neq 0 using l-gt-0 by auto

lemma pl-not0: p \cdot l \neq 0 using p1 l0 by auto

interpretation pl: poly-mod-2 p \cdot l
  by (standard, insert p1 l0, auto)

private lemmas pl-dvdm-imp-p-dvdm = p.pl-dvdm-imp-p-dvdm[OF l0]

  using Mp-Mp-pow-is-Mp[OF l0 p.m1] .

context
  fixes u :: int poly
  and d and f and n
  and gs :: int poly list
  and Degr :: nat set
  defines [simp]: d \equiv degree u
  assumes d0: d > 0
  and u: monic u
  and irreducible-u: p.irreducible-m u
  and u-f: p.dvdm u f
  and f-dvd-F: f dvd F
  and [simp]: n \equiv degree f
  and f-gs: pl.unique-factorization-m f (lead-coeff f, mset gs)

53
and cop: coprime \( \text{lead-coeff } f \) \( p \)
and sf: \( p \).square-free-m \( f \)
and sf-F: square-free \( f \)
and u-gs: \( u \in \text{set } gs \)
and norm-gs: \( \text{map } pl.Mp gs = gs \)
and Degs: \( \land \text{factor. factor dvd } f \implies p.dvdm u \text{ factor } \implies \text{degree factor } \in \text{Degs} \)

\[ \begin{align*}
\text{interpretation } & pl: \text{poly-mod-2 } p' l \text{ using } l0 p1 \text{ by (unfold-locales, auto)} \\
\text{private lemma } & f0: f \neq 0 \text{ using sf-F unfolding square-free-def by fastforce} \\
\text{private lemma } & Mpf0: pl.Mp f \neq 0 \\
& \text{by (metis p.square-free-m-def p-Mp-pl-Mp sf)} \\
\text{private lemma } & pMpf0: p.Mp f \neq 0 \text{ using } p.square-free-m-def sf \text{ by auto} \\
\text{private lemma } & d0: d \leq n \text{ using } p.dvdm-imp-degree-le[\text{OF } f-dvd-F F0] \text{ by auto} \\
\text{private lemma } & n0: n > 0 \text{ using } d0 d n \text{ by auto} \\
\text{private lemma } & B2-0[introl]: B2-LLL F > 0 \text{ using } B2-2 \text{ by auto} \\
\text{private lemma } & deg-u: \text{degree } u > 0 \text{ using } d0 d-def \text{ by auto} \\
\text{private lemma } & n-le-N: n \leq N \text{ by (simp add: dvd-imp-degree-le[OF } f-dvd-F F0\text{])} \\
\text{lemma } & dvdm-power: \text{assumes } g dvd f \\
& \text{shows } p.dvdm u g \iff pl.dvdm u g \\
\text{proof} \\
& \text{assume } pl.dvdm u g \\
& \text{thus } p.dvdm u g \text{ by (rule pl-dvdm-imp-p-dvdm)} \\
\text{next} \\
& \text{assume } dvdm: p.dvdm u g \\
& \text{from norm-gs have norm-gsp: } \land f. f \in \text{set } gs \implies pl.Mp f = f \text{ by (induct gs, auto)} \\
& \text{with } f-gs[unfolded pl.unique-factorization-m-alt-def pl.factorization-m-def split] \\
& \text{have } gs-irred-mon: \land f. f \in\# \text{ mset } gs \implies pl.irreducible_d-m f \land \text{monic } f \text{ by auto} \\
& \text{from norm-gs have norm-gs: image-mset pl.Mp (mset gs) = mset gs by (induct gs, auto)} \\
& \text{from } assms \text{ obtain } h \text{ where } f: f = g * h \text{ unfolding dvd-def by auto} \\
& \text{from } pl.unique-factorization-m-factor[OF } p.prime f-gs[unfolded f] \text{ - - l0 refl, folded } f, \\
& \text{OF cop sf, unfolded pl.Mf-def split] norm-gs} \\
& \text{obtain } hs fs \text{ where } uf: pl.unique-factorization-m h (\text{lead-coeff } h, hs) \\
& \text{pl.unique-factorization-m g (lead-coeff } g, fs) \\
& \text{and id: mset gs = fs + hs} \\
\end{align*} \]

54
and norm: image-mset pl.Mp fs = fs image-mset pl.Mp hs = hs by auto
from p.square-free-m-prod-imp-coprime-m[OF sf[unfolded f]] have cop-h-f: p.coprime-m g h by auto
show pl.dvdm u g
proof (cases u ∈# fs)
case True
  hence pl.Mp u ∈# image-mset pl.Mp fs by auto
from pl.factorization-m-mem-dvdm[OF pl.unique-factorization-m-imp-factorization[OF uf(2)] this]
  show ?thesis .
next
case False
  from u-gs have u ∈# mset gs by auto
  from this[unfolded id] False have u ∈# hs by auto
  hence pl.Mp u ∈# image-mset pl.Mp hs by auto
from pl.factorization-m-mem-dvdm[OF pl.unique-factorization-m-imp-factorization[OF uf(1)] this]
  have pl.dvdm u h by auto
from pl-dvdm-imp-p-dvdm[OF this]
  have p.dvdm u h by auto
from cop-h-f[unfolded p.coprime-m-def, rule-format, OF dvd this]
  have p.dvdm u 1 .
  from p.dvdm-imp-degree-le[OF this u - p.m1] have degree u = 0 by auto
  with deg-u show ?thesis by auto
qed
qed

private lemma uf: pl.dvdm u f using dvdm-power[OF dvd-refl] u-f by simp

lemma exists-reconstruction: ∃ h0. irreducibleₜ h0 ∧ p.dvdm u h0 ∧ h0 dvd f
proof –
  have deg-f: degree f > 0 using (n ≡ degree f) n0 by blast
  from berlekamp-zassenhaus-factorization-irreducibleₜ[OF refl sf-F deg-f]
  obtain fs where f-fs: f = prod-list fs
    and c: (∀ fi∈set fs. irreducibleₜ fi ∧ 0 < degree fi ) by blast
  have pl.dvdm u (prod-list fs) using uf-fs by simp
  hence p.dvdm u (prod-list fs) by (rule pl-dvdm-imp-p-dvdm)
  from this obtain h0 where h0: h0 ∈ set fs and dvdm-u-h0: p.dvdm u h0
    using p.irreducible-m-dvdm-prod-list[OF irred-u] by auto
  moreover have h0 dvd f by (unfold f-fs, rule prod-list-dvd[OF h0])
  moreover have irreducibleₜ h0 using c h0 by auto
  ultimately show ?thesis by blast
qed

lemma factor-ded-f-0: assumes factor dvd f
shows pl.Mp factor ≠ 0
proof –
  from assms obtain h where f: f = factor * h unfolding dvd-def ..
  from arg-cong[OF this, of pl.Mp] have 0 ≠ pl.Mp (pl.Mp factor * h)
using Mpf\(0\) by auto
thus \(\theta\)thesis by fastforce
qed

lemma degree-factor-ge-degree-u:
  assumes u-dvdm-factor: p.dvdm u factor
  and factor-dvd: factor dvd f
  shows degree u \(\le\) degree factor
proof
  from factor-dvd-f-0 [OF factor-dvd] have factor0: pl.Mp factor \(\ne\) 0 .
  from u-dvdm-factor [unfolded dvdm-power [OF factor-dvd] pl.dvdm-def] obtain v
where
  *: pl.Mp factor = pl.Mp (u * pl.Mp v) by auto
  with factor0 have v0: pl.Mp v \(\ne\) 0 by fastforce
  hence 0 \(\ne\) lead-coeff (pl.Mp v) by auto
  also have lead-coeff (pl.Mp v) = pl.M (lead-coeff (pl.Mp v))
    by (auto simp: pl.Mp-def coeff-map-poly)
  finally have **: lead-coeff (pl.Mp v) \(\ne\) \(\ast\) l \(*\) r for r by (auto simp: pl.M-def)
  from * have degree factor \(\ge\) pl.degree-m (u * pl.Mp v) using pl.degree-m-le [of factor] by auto
  also have pl.degree-m (u * pl.Mp v) = degree (u * pl.Mp v)
    by (rule pl.degree-m-eq, unfold lead-coeff-mult, insert u pl.m1 **, auto)
  also have \(\ldots\) = degree u + degree (pl.Mp v)
    by (rule degree-mult-eq, insert v0 u, auto)
  finally show \(\theta\)thesis by auto
qed

7.3.2 Inner loop

context
  fixes j': nat
  assumes dj': d \(\le\) j'
  and j'n: j' < n
  and deg: \(\forall\) factor. p.dvdm u factor \(\Rightarrow\) factor dvd f \(\Rightarrow\) degree factor \(\ge\) j'
begin

private abbreviation (input) j \equiv Suc j'

private lemma jn: j \(\le\) n using j'n by auto

private lemma factor-irreducible,1: assumes hf: h dvd f
  and pub: p.dvdm u h
  and degh: degree h > 0
  and degh-j: degree h \(\le\) j'
  shows irreducible,1 h
proof
  from dvdm-power [OF hf] pub have plah: pl.dvdm u h by simp
  note uf-partition = p.unique-factorization-m-factor-partition [OF l0]
  obtain gs1 gs2 where part: List.partition (\lambda g. p.dvdm g h) gs = (gs1, gs2)

by force

from part u-gs puh
have u-gs1: \(u \in \text{set gs1}\) unfolding p by auto
have gs1: \(gs1 = \text{filter } (\lambda gi. \ p.devdm gi h) \ gs\) using part by auto
obtain \(k\) where \(f = h \ast k\) using hf unfolding dvd-def by auto
from uf-partition[OF f-gs f cop sf part] have uf-h: pl.unique-factorization-m h (lead-coeff h, mset gs1) by auto

show \(?thesis\)
proof (intro irreducible\_I degh)
  fix \(q\), \(r\)
  assume deg-q: degree \(q > 0\) degree \(q <\) degree \(h\)
  and deg-r: degree \(r > 0\) degree \(r <\) degree \(h\)
  and h: \(h = q \ast r\)
  then have \(r \text{ dvd } h\) by auto
  with \(h \text{ dvd-trans}(OF - hf)\) have 1: \(q \text{ dvd } r \ast \text{ dvd } f\) by auto
  from cop[unfolded f] have cop: coprime (lead-coeff h) p
    using p.prime pl.coprime-lead-coeff-factor(1) by blast
  from sf[unfolded f] have sf: p.square-free-m h using p.square-free-m-factor
  by metis

  have norm-gs1: image-mset p.Mp (mset gs1) = mset gs1 using norm-gs unfolding gs1
  by (induct gs, auto)
  from pl.unique-factorization-m-factor[OF p uf-h[unfolded h], folded h, OF cop sf l0 refl]
  obtain fs gs where uf-q: pl.unique-factorization-m q (lead-coeff q, fs)
    and uf-r: pl.unique-factorization-m r (lead-coeff r, gs)
    and id: mset gs1 = fs + gs
    unfolding pl.Mf-def split using norm-gs1 by auto
  from degh degh-j degh-j deg-r have \(gj'<:\) degree \(q <\) \(j'\) and \(rj'<:\) degree \(r <\) \(j'\) by auto

  have intro: \(u \in \#\) \(\Rightarrow\) pl.Mp u \(\in\) image-mset pl.Mp r for r by auto
  note dvdI = pl.factorization-m-mem-dvd[pl.unique-factorization-m-imp-factorization intro]
  from u-gs1 id have u\(\in\)\# fs \(\lor\) u\(\in\)\# gs unfolding in-multiset-in-set[symmetric]
  by auto

  with dvdI[OF uf-q] dvdI[OF uf-r] have pl.devdm u q \(\lor\) pl.devdm u r by auto
  hence p.devdm u q \(\lor\) p.devdm u r using pl.devdm-imp-p.devdm by blast
  with 1 \(gj' \ast \text{ show False}\)

  by (elim disjE, auto dest!: deg)
qed

qed

private definition ll = (let \(n = \text{sqrt-int-ceiling} (\|f\|^2 \ast (2 \ast j') \ast 2 \ast (5 \ast j' \ast j'))\));
  \(ll' = \text{find-exponent } p \ n\) in if \(ll' < 1\) then \(ll'\) else \(l\)

lemma ll: \(ll \leq l\) unfolding ll-def Let-def by auto

lemma ll0: \(ll \neq 0\) using l0 find-exponent[OF p.m1]
unfolding ll-def Let-def by auto

lemma pll1: \( p^\prime \|l > 1 \) using ll0 p.m1 by auto

interpretation pll: poly-mod-2 p^\prime ll
  using ll0 p.m1 by (unfold-locales, auto)

lemma pll0: p^\prime ll \neq 0 using p by auto

lemma dvdm-l-ll: assumes pl dvdm a b 
  shows pll dvdm a b 
proof 
  have id: \( p^\prime l = p^\prime ll * p * (l - ll) \) using ll unfolding power-add[symmetric] by auto 
  from assms[unfolded pl dvdm-def] obtain c where eq: pl.eq-m b (a * c) by blast 
  from pl.Mp-shrink-modulus[OF eq[unfolded id]] p have pll.eq-m b (a * c) by auto 
  thus ?thesis unfolding pll dvdm-def .. 
qed

private definition g ≡ LLL-short-polynomial (p ll) j u

lemma deg-g-j: degree \( g < j \) 
  and go: \( g \neq 0 \) 
  and ug:pll dvdm u g 
  and short-g: \( h \neq 0 \) → pll dvdm u h → degree h ≤ j' → \( ||g||^2 \leq 2^{j'} \) 
proof (atomize(full), goal-cases)
  case 1 
  from deg-a have degu0: degree a ≠ 0 by auto 
  have ju: j ≥ degree a using d-def djj Suc-eq by blast 
  have ju': j > degree a using d-def djj' by auto 
  note short = LLL-short-polynomial[OF degu0 ju pll1 u, folded g-def] 
  from short(1–3) short(4)(OF ju') show ?case by auto 
qed

lemma LLL-reconstruction-inner-simps: LLL-reconstruction-inner p l gs f u Degs j 
  = (if j' ≠ Degs then None else if p ll ≤ |lead-coeff g| then None 
    else case div-int-poly f (primitive-part g) of None ⇒ None 
    | Some f' ⇒ Some (\{g=gs . p dvdm gi (primitive-part g)\}, lead-coeff f', j', primitive-part g) 
  )
proof 
  have Suc: Suc j' = 1 by simp 
  show ?thesis unfolding LLL-reconstruction-inner-def Suc Let-def ll-def\[unfolded Let-def, symmetric]\ 
    g-def[unfolded Let-def, symmetric] by simp 
qed

58
lemma LLL-reconstruction-inner-complete:
  assumes ret: LLL-reconstruction-inner p l gs u Degs j = None
  shows \( \forall f. p.dvd m u \Rightarrow f.dvd f \Rightarrow deg(f) \geq j \)
proof (rule contr)
  fix factor
  assume pu-factor: p.dvd m u factor
  and factor-f: factor dvd f
  and deg-factor2: \( j \leq deg(f) \)
with \( deg(f) \leq j \) have deg-factor-j [simp]: \( deg(f) = j \)
  and deg-factor-lt-j: \( deg(f) < j \)
  from Degs[of factor-f pu-factor] have Degs: \( (j' \notin Degs) = False \)
  by auto
  from dvd-power[of factor-f] pu-factor have u-factor: pl.dvd m u factor by auto
  from dvd-power[of factor-f] have ll-factor: pl-l-factor by auto
  have deg-factor: deg(f) > 0
    using d0 deg-factor-j dj' by linarith
  from f0 deg-factor divides-degree[of factor-f] have deg-f: \( deg(f) > 0 \)
    by auto
  from deg-factor have j': \( j' > 0 \)
    by simp
  from factor-f f0 have factor0: factor \( \neq 0 \)
    by auto
  from factor-f obtain f2 where f = factor * f2 unfolding dvd-def by auto
  from deg-u have deg-u0: degree u \( \neq 0 \)
    by auto
  from pu-factor u have u'-factor: degree u \( \leq j' \)
  unfolding deg-factor-j [symmetric]
  using d-def deg-factor-j dj' by blast
  hence u-j: degree u \( \leq j \)
  by auto
note LLL = LLL-short-polynomial[of factor0]
  unfolding LLL-reconstruction-inner-simps Degs if-False
  note LLL = LLL-short-polynomial[of factor0]
  unfolding LLL-reconstruction-inner-simps Degs if-False
  hence deg-g: \( deg(g) \leq j' \)
    by simp
  from LLL(2) have normf: \( \|f\|^2 \geq 1 \)
    using sq-norm-poly-pos[of g] by presburger
from f0 have normf: \( \|f\|^2 \geq 1 \)
  using sq-norm-poly-pos[of f] by presburger
from factor0 have normf1: \( \|factor\|^2 \geq 1 \)
  using sq-norm-poly-pos[of factor] by presburger
from F0 have normF: \( \|F\|^2 \geq 1 \)
  using sq-norm-poly-pos[of F] by presburger
  have factor-F: factor dvd F by (rule dvd-trans)
  have \( \|factor\|^2 \geq degree g * \|g\|^2 \geq j' \)
    by (rule mult-mono[OF power-increasing], insert normm normm1 deg-g, auto)
  have \( \|factor\|^2 \leq (\|factor\|^2 * (2 * j' * \|factor\|^2)) \)
    by (rule power-mono[OF mult-left-mono], insert LLL(4), auto)
  have \( \|factor\|^2 \leq (2 * j')^2 \geq (j' * j') \)
    unfolding power-mult-distrib power-mult power-add mult-2 by simp
finally have approx-part-1: \( \|factor\|^2 \leq degree g * \|g\|^2 \leq \|factor\|^2 \leq (2 * j')^2 \)
  \( (j' * j') \)
  \{ fix f :: int poly
  assume *: factor dvd f f \( \neq 0 \)
  note approx-part-1
  also have \( \|factor\|^2 \leq (2 * j')^2 \leq (j' * j') \leq (2 * j') \)
  
59
2 * (j' * j')

by (rule mult-right-mono[OF power-mono], insert sq-norm-factor-bound[OF *], auto)
also have ... = ∥f∥^2 * (2 * j') * 2 * (2*j' * 2*j' + j' * j')
unfolding power-mult-distrib power-add by (simp add: power-mult[symmetric])
also have 2*j' * 2*j' + j' * j' = 5 * j' * j' by simp
finally have ∥factor∥^2 * degree g * ∥g∥^2 * degree factor ≤ ∥f∥^2 * (2 * j') * 2
* (5 * j' * j')

} note approx = this
note approx-1 = approx[OF factor-f f0]
note approx-2-part = approx[OF factor-F F0]
have large: ∥factor∥^2 * degree g * ∥g∥^2 * degree factor < (p ^ ll)^2
proof (cases ll = l)
  case False
  let ?n = ∥f∥^2 * (2 * j') * 2 * (5 * j' * j')
  have n: ?n ≥ 0 by auto
  let ?s = sqrt-int-ceiling ?n
  from False have ll = find-exponent p ?s unfolding ll-def Let-def by auto
  hence splll: ?s < p ^ ll using find-exponent(1)[OF p.m1] by auto
  have sqrt ?n ≥ 0 by auto
  hence sqrt: sqrt ?n > -1 by linarith
  have ns: ?n ≤ ?s^2 using sqrt-int-ceiling-bound[OF n] .
  also have ... < (p ^ ll)^2
    by (rule power-strict-mono[OF splll], insert sqrt, auto)
  finally show thesis using approx-1 by auto
next
  case True
  hence ll: p ^ ll = p ^ l by simp
  show thesis unfolding ll
proof (rule less-le-trans[OF le-less-trans[OF approx-2-part] bound-ll])
  have ∥F∥^2 * (2 * j') * 2 * (5 * j' * j')
    = 2 * (2 * j' * j' + 3 * j' * j') * ∥F∥^2 * (j' + j')
  unfolding mult-2 by simp
  also have ... < 2 * (N^2 + 4 * N*N) * ∥F∥^2 * (2 * N)
  proof (rule mult-less-le-imp-less[OF power-strict-increasing pow-mono-exp])
    show 1 ≤ ∥F∥^2 by (rule normF)
    have jN': j' < N and jN: j' ≤ N using jn divides-degree[OF jf dvd F]
  proof (rule normF)
    have j' + j' ≤ j' + j' using deg-g j'n by auto
    also have ... = 2 * j' by auto
    also have ... ≤ 2 * N using jN by auto
    finally show j' + j' ≤ 2 * N .
    show 0 < ∥F∥^2 * (j' + j')
      by (rule zero-less-power, insert normF, auto)
    have 2 * j' * j' + 3 * j' * j' ≤ 2 * j' * j' + 3 * j' * j' by auto
    also have ... = 5 * (j' * j') by auto
    also have ... < 5 * (N * N)
      by (rule mult-strict-left-mono[OF mult-strict-mono], insert jN', auto)
    also have ... = N^2 + 4 * N * N by (simp add: power2-eq-square)
finally show \( 2 \cdot j' + 3 \cdot j' < N^2 + 4 \cdot N \cdot N \).
qed auto
also have \( \ldots = 2 \cdot N^2 \cdot (2 \cdot N) \cdot \|F\|^2 \cdot (2 \cdot N) \)
unfolding \( \text{power-mult-distrib power-add} \) by \( \text{simp add: power-mult symmetric} \)
finally show \( \|F\|^2 \cdot (2 \cdot j') \cdot 2 \cdot (5 \cdot j' \cdot j') < 2 \cdot N^2 \cdot B2-LLL \) F \( \cdot (2 \cdot N) \)
unfolding \( \text{B2-LLL-def} \) by \( \text{simp} \)
qed
qed
have \((\text{lead-coeff } g)^2 < (p \cdot \text{lill})^2\)
proof \( \text{rule le-less-trans \{OF - large\}} \)
have \( I \cdot (\text{lead-coeff } g)^2 \cdot I \leq \|\text{factor}\|^2 \cdot \text{degree } g \cdot \|g\|^2 \cdot \text{degree factor} \)
by \( \text{rule mult-mono \{OF - order-trans \{OF power-mono pow-mono-exp\}, insert normg normf1 deg-f g0 coeff-le-sq-norm \{of g\} j'0, auto intro: pow-mono-one} \)
thus \( \|\text{lead-coeff } g\|^2 \leq \|\text{factor}\|^2 \cdot \text{degree } g \cdot \|g\|^2 \cdot \text{degree factor} \) by \( \text{simp} \)
qed
hence \((\text{lead-coeff } g)^2 < (p \cdot \text{lill})^2 \) by \( \text{simp} \)
hence \((\text{lead-coeff } g) < p \cdot \text{lill} \) using \( \text{p.m1 abs-le-square-iff \{of p \cdot \text{lill lead-coeff g\}} \) by \( \text{auto} \)
hence \((p \cdot \text{lill} \leq (\text{lead-coeff } g)) = \text{False} \) by \( \text{auto} \)
note \( \text{ref = rel\{unfolded this if-False\}} \)
have \( \text{deg-f}: \text{degree } f \geq 0 \) using \( \text{n0} \) by \( \text{auto} \)
have \( \text{deg-ug}: \text{degree } u \leq \text{degree } g \)
proof \( \text{rule pl.lvdm-degree \{OF u LLL(3)\}, standard} \)
assume \( \text{pll.Mp } g = 0 \)
from \( \text{arg-cong \{OF this, of } \lambda p. \text{coeff } p \text{ (degree } g)\} \)
have \( \text{pll.M } (\text{coeff } g \text{ (degree } g) ) = 0 \) by \( \text{auto simp: pll.Mp-def coeff-map-poly} \)
from \( \text{this\{unfolded pll.M-def\}} \) obtain \( c \) where \( \text{ly: lead-coeff } g = p \cdot \text{lill } \cdot c \) by \( \text{auto} \)
with \( \text{LLL(2)} \) have \( c0: c \neq 0 \) by \( \text{auto} \)
hence \((p \cdot \text{lill})^2 \leq (\text{lead-coeff } g)^2 \) unfolding \( \text{ly abs-le-square-iff \{symmetric\}} \)
by \( \text{rule aux-abs-int} \)
also have \( \ldots \leq \|g\|^2 \) using \( \text{coeff-le-sq-norm \{of g\}} \) by \( \text{auto} \)
also have \( \ldots = \|g\|^2 \) by \( \text{simp} \)
also have \( \ldots \leq \|g\|^2 \cdot \text{degree factor} \)
by \( \text{rule pow-mono-exp, insert \text{deg-f normj'0, auto}} \)
also have \( \ldots = 1 \cdot \ldots \) by \( \text{simp} \)
also have \( \ldots \leq \|\text{factor}\|^2 \cdot \text{degree } g \cdot \|g\|^2 \cdot \text{degree factor} \)
by \( \text{rule mult-right-mono, insert normf1, auto} \)
also have \( \ldots < (p \cdot \text{lill})^2 \) by \( \text{rule large} \)
finally show \( \text{False} \) by \( \text{auto} \)
qed
with \( \text{deg-u} \) have \( \text{deg-g}: \text{degree } g \geq 0 \) by \( \text{simp} \)
from \( j'0 \) have \( \text{deg-factor: degree factor } \geq 0 \) by \( \text{simp} \)
let \(?g = \text{gcd factor } g \)
from \( \text{common-factor-via-short \{OF deg-factor deg-g u deg-u pll-u-factor LLL(3) large\}} \) \( \text{pll.m1} \)
have \( \text{gcd: } 0 < \text{degree } ?g \) by \( \text{auto} \)
have \( \gcd\text{-factor}: \ ?g \ d \vdots \text{factor} \) by auto
from dvd-trans[\( \text{OF this factor-f} \)] have \( \gcd\text{-f}: \ ?g \ d \vdots f \).
from \( \deg\text{-g} \) have \( g0: \ ?g \neq 0 \) by auto
have \( \gcd\text{-g}: \ ?g \leq \deg g \) using \( g0 \) using divides-degree by blast
from \( \gcd\text{-g} \ LLL(1) \) have \( hj\): \ ?g \leq j' by auto
let \( ?pp = \text{primitive-part} \ g \)
from ret have \( \text{div-int-poly} f \ ?pp = \text{None} \) by (auto split: option.splits)
from \( \text{div-int-poly}[f \ ?pp, \text{unfolded this}] \ g0 \)
have \( \text{ppf}: \ ?pp \ d \vdots f \) unfolding dvd-def by (auto simp: ac-simps)
have \( \text{irr-f1}: \ \text{irreducible}_d \text{ factor} \)
\[ \text{by (rule factor-irreducible}_d \text{f}[\text{OF factor-f pu-factor deg-factor}, \text{simp}]) \]
from \( \gcd\text{-factor} \) obtain \( h \) where \( \text{factor}: \ ?g \ast h \) unfolding dvd-def by auto
from irreducible_d D(2)[\( \text{OF irr-f1, \ of} \ ?g \ h, \text{folded factor} \)] have \( \neg(\deg \ ?g < j') \wedge \deg h < j' \)
\[ \text{by auto} \]
moreover have \( j' = \deg \ ?g + \deg h \) using \( \text{factor0} \text{ arg-cong}[\text{OF factor, of degree}] \)
\[ \text{by (subst (asm) degree-mult-eq, insert j'0, auto)} \]
ultimately have \( \deg h = 0 \) using \( \gcd\text{ by linarith} \)
from \( \deg\text{-coeffs}[\text{OF this}] \) \( \text{factor factor0} \)
obtain \( c \) where \( h: \ ?h = \text{[c:]} \) and \( c: \ c \neq 0 \) by fastforce
from \( \text{arg-cong}[\text{OF factor, of degree}] \) have \( \text{id}: \ \deg \ ?g = \deg \text{ factor} \)
\[ \text{unfolding} h \text{ using} c \text{ by auto} \]
moreover have \( \deg \ ?g \leq \deg g \)
\[ \text{by (subst \( \gcd\text{.commute, rule degree-gcd1[OF g0]} \))} \]
ultimately have \( \deg g \geq \deg \text{ factor} \) by auto
with \( \text{id deg-factor2 deg-g-j} \) have \( \deg: \ ?g = \deg g \)
and \( \deg g = \deg \text{ factor by auto} \)
have \( ?g \ d \vdots g \) by auto
then obtain \( q \) where \( g: \ ?g \ast q \) unfolding dvd-def by auto
from \( \text{arg-cong}[\text{OF this, of degree}] \) \( \text{deg} \)
have \( \deg q = 0 \)
\[ \text{by (subst (asm) degree-mult-eq, insert g g0, force, force) simp} \]
from \( \deg\text{-coeffs}[\text{OF this}] \ g g0 \)
obtain \( d \) where \( p: \ ?p = \text{[d:]} \) and \( d: \ d \neq 0 \) by fastforce
from \( \text{arg-cong}[\text{OF factor, of}] \) \( \text{(*)} \) \( q \)
have \( q \ast \text{ factor} = h \ast g \)
\[ \text{by (subst g, auto simp: ac-simps)} \]
hence \( \smult d \text{ factor} = h \ast g \) unfolding \( p \ h \) by auto
hence \( g \ d \vdots \smult d \text{ factor by simp} \)
from \( \text{dvd-smult-int[OF d this]} \)
have \( \text{primitive-part} g \ d \vdots \text{ factor} \).
from \( \text{dvd-trans[OF this factor-f]} \) \( ppf \) show \( \text{False} \) by auto
qed

lemma \( \text{LLL-reconstruction-inner-sound} \):
assumes \( \text{ret: LLL-reconstruction-inner p l gs f u Deqs j = Some (gs',b',f',h)} \)
shows \( f = f' \ast h \) (is \( ?q1 \))
and irreducible,\( h \) (is \( \text{?g2} \))
and \( b' = \text{lead-coeff} f' \) (is \( \text{?g3} \))
and \( \text{pl.unique-factorization-m f'} \) (lead-coeff \( f' \), mset \( gs' \)) (is \( \text{?g4} \))
and \( \text{p.dvdm u} h \) (is \( \text{?g5} \))
and degree \( h = j' \) (is \( \text{?g6} \))
and length \( gs' < \) length \( gs \) (is \( \text{?g7} \))
and set \( gs' \subseteq \) set \( gs \) (is \( \text{?g8} \))
and \( gs' \neq [] \) (is \( \text{?g9} \))

proof

- let \( \text{?ppg} = \text{primitive-part g} \)

  note ret = ret[unfolded LLL-reconstruction-inner-simps]

  from ret have lc: \( \text{abs} \) (lead-coeff \( g \)) < \( p \)'ll by (auto split: if-splits)

  from ret obtain rest where rest: \( \text{div-int-poly} f \) (primitive-part \( g \)) = Some rest

  by (auto split: if-splits option.splits)

  from ret[unfolded this] div-int-then-rqp[OF this] lc

  have out [simp]: \( h = \text{?ppg} gs' = \text{filter} (\lambda gi. \neg p.\text{dvdm gi} \text{ ?ppg}) gs \)

  \( f' = \) rest \( b' = \) lead-coeff rest

  and f: \( f = \text{?ppg} \ast \) rest by (auto split: if-splits)

  with div-int-then-rqp[OF rest] show \( \text{?g1} \text{ ?g3} \) by auto

  from \( \text{?g1} \) \( \text{f0} \) have h0: \( h \neq 0 \) by auto

  let \( \text{?c} = \text{content} g \)

  from \( \text{g0} \) have ct0: \( ?c \neq 0 \) by auto

  have \( \{?c\} \leq \text{lead-coeff} g \) by (rule content-le-lead-coeff)

  also have \( \ldots < \) \( p \)'ll by fact

  finally have ct-pl: \( \{?c\} < p \)'ll .

  from ug have pl.dvdm u (smult ?c ?ppg) by simp

  from poly-mod-dvdm-drop-smult[OF u p ct0 ct-pl this]

  show puh: p.dvdm u h by simp

  with dvdm-power[OF h f]

  have uh: pl.dvdm u h by (auto simp: dvdm-def)

  from f have hf: h dvd f by (auto intro:dvdI)

  have degh: degree \( h > 0 \)

  by (metis d-def deg-deg-u puh dj' hf le-neq-implies-less not-less0 neq0-conv)

  show irr-h: \( \text{?g2} \)

  by (intro factor-irreducible,\( _1 \) deg hf puh, insert deg-g-j, simp)

  show deg-h: \( \text{?g6} \) using deg deg-g-j g-def hf le-less-Suc-eq puh degree-primitive-part

  by force

  show \( ?g7 \) unfolding out

  by (rule length-filter-less[of u], insert pl-dvdm-imp-p-dvdm[OF uh] u-gs, auto)

  show \( ?g8 \) by auto

  from f out have fh: \( f = h \ast f' \) and gs': \( gs' = [gi \leftarrow gs. \neg p.\text{dvdm gi} h] \) by auto

  note [simp def] = out

  let \( ?fs = \text{filter} (\lambda gi. p.\text{dvdm gi} h) gs \)

  have part: List.partition (\lambda gi. p.\text{dvdm gi} h) gs = (?fs, gs')

  unfolding gs' by (auto simp: o-def)

  from p.unique-factorization-m-factor-partition[OF l0 f-gs fh cop sf part]

  show uf: p.unique-factorization-m f' (lead-coeff f', mset \( gs' \)) by auto

  show \( ?g9 \)

  proof
assume gs′ = []
with pl.unique-factorization-m-imp-factorization[OF uf, unfolded pl.factorization-m-def]
have pl.Mp f′ = pl.Mp (smult (lead-coeff f') 1) by auto
from arg-cong[OF this, of degree] pl.degree-m-le[of smult (lead-coeff f') 1]
have pl.degree-m f′ = 0 by simp
also have pl.degree-m f′ = degree f'
proof (rule poly-mod.degree-m-eq[OF - pl.m1])
  have coprime (lead-coeff f') p
    by (rule p.coprime-lead-coeff-factor[OF p.prime cop[unfolded fh]])
  thus lead-coeff f' mod p ^ l ≠ 0 using l0 p.prime by fastforce
qed
finally have degf': degree f' = 0 by auto
from degree0-coeffs[OF this] f0 fh obtain c where f' = [:c:]
and c: c ≠ 0
and fch: f = smult c h
by auto
from irreducible_d h have irr-f: irreducible_d f
  using irreducible_d-smult-int[OF c, of h] unfolding fch by auto
have degree f = j' using hf irr-h deg-h
  using irr-f (n ≡ degree f): degh j'n
  by (metis add.right-neutral degf' degree-mult-eq f0 fh mult-not-zero)
  thus False using j'n by auto
qed
qed
end

interpretation LLL d .

lemma LLL-reconstruction-inner-None-upt-j':
  assumes i j: ∀ i∈{d+1..j}. LLL-reconstruction-inner p l gs f u Degs i = None
  and d j: d<j and j≤n
  shows (∀ factor. p.dvdm u factor =⇒ factor dvd f =⇒ degree factor ≥ j
using asms
proof (induct j)
  case (Suc j)
  show ?case
  proof (rule LLL-reconstruction-inner-complete)
    show (∀ factor2. p.dvdm u factor2 =⇒ factor2 dvd f =⇒ j ≤ degree factor2
    proof (cases d = j)
      case False
      show (∀ factor2. p.dvdm u factor2 =⇒ factor2 dvd f =⇒ j ≤ degree factor2
        by (rule Suc.hyps, insert Suc.prems False, auto)
      next
      case True
      then show (∀ factor2. p.dvdm u factor2 =⇒ factor2 dvd f =⇒ j ≤ degree factor2
        using degree-factor-ge-degree-u by auto
      qed
      qed (insert Suc.prems, auto)
    qed
    qed auto
corollary LLL-reconstruction-inner-None-upt-j:
assumes \( ij: \forall i \in \{d+1..j\} \), LLL-reconstruction-inner \( p \ l gs f u \) Degs \( i = \text{None} \)
and \( dj: d \leq j \) and \( jn: j \leq n \)
shows \( \forall \text{factor}. \ p.\dvdm u \text{factor} \implies \text{factor} \text{ded} f \implies \text{degree} \text{factor} \geq j \)
proof (cases \( d=j \))
  \begin{enumerate}
  \item case True
    \begin{enumerate}
    \item then show \( \forall \text{factor}. \ p.\dvdm u \text{factor} \implies \text{factor} \text{ded} f \implies d = j \implies j \leq \text{degree factor} \)
      using degree-factor-ge-degree-u by auto
    \end{enumerate}
  \end{enumerate}
next
  case False
  hence \( dj2: d < j \) using \( dj \) by auto
  then show \( \forall \text{factor}. \ p.\dvdm u \text{factor} \implies \text{factor} \text{ded} f \implies d \neq j \implies j \leq \text{degree factor} \)
  using LLL-reconstruction-inner-None-upt-j′[OF \( ij \) \( dj2 \) \( jn \)] by auto
qed

lemma LLL-reconstruction-inner-all-None-imp-irreducible:
assumes \( i: \forall i \in \{d+1..n\} \), LLL-reconstruction-inner \( p \ l gs f u \) Degs \( i = \text{None} \)
shows irreducible\( d \) \( f \)
proof –
  obtain factor
  where irreducible-factor: irreducible\( d \) factor
  and dvdp-u-factor: \( p.\dvdm u \text{factor} \) and \( \text{factor-dvd-f}: \text{factor} \text{ded} f \)
  using exists-reconstruction by blast
  have \( f0: f \neq 0 \) using \( n0 \) by auto
  have deg-factor1: degree \( u \) \( \leq \) degree factor
    by (rule degree-factor-ge-degree-u[OF dvdp-u-factor factor-dvd-f])
  hence factor-not0: factor \( \neq 0 \) using \( d0 \) by auto
  hence deg-factor2: degree factor \( \leq \) degree \( f \) using divides-degree[OF factor-dvd-f]
  \( f0 \) by auto
  let \( ?j = \text{degree factor} \)
  show \( \exists \text{thesis} \)
  proof (cases degree \( \text{factor} = \text{degree} \ f \))
  case True
    from factor-dvd-f obtain \( g \) where \( f\)-factor: \( f = \text{factor} \ast \text{g} \)
    unfolding dvd-def by auto
    from True[unfolded \( f\)-factor] \( f0[unfolded \( f\)-factor] \) have degree \( g = 0 \) \( g \neq 0 \)
    by (subst (asm) degree-mul-eq, auto)
    from degree0-coefs[OF this(1)] this(2) obtain \( c \) where \( g = [:c:] \) and \( c: c \neq 0 \)
    by auto
    with \( f\)-factor have \( fc: f = \text{smult} c \text{factor} \) by auto
    from irreducible-factor irreducible\( d \)-smult-int[OF \( c, \) of factor, folded \( fc \)]
    show \( \exists \text{thesis} \) by simp
  next
  case False
  hence Suc-j: Suc \( ?j \leq \text{degree} f \) using deg-factor2 by auto
  have Suc \( ?j \leq \text{degree factor} \)
proof (rule LLL-reconstruction-inner-None-upt-j[\{OF - - - dvd-u-factor factor-dvd-f\}])
  show \(d \leq \text{Suc} \ ?j\) using deg-factor1 by auto
  show \(\forall i \in \{d + 1..(\text{Suc} \ ?j)\}\. \text{LLL-reconstruction-inner} \ p l \ gs \ f \ u \ Degs \ i =\) None
    using Suc-j i by auto
  qed
  then show ?thesis by auto
  qed

lemma irreducible-imp-LLL-reconstruction-inner-all-None:
  assumes irr-f: irreducible \(d\) \(f\)
  shows \(\forall i \in \{d + 1..n\}. \text{LLL-reconstruction-inner} \ p l \ gs \ f \ u \ Degs \ i =\) None using Suc-j
  proof (rule ccontr)
    let \(\ ?LLL-inner = \lambda i. \text{LLL-reconstruction-inner} \ p l \ gs \ f \ u \ Degs \ i\)
    let \(\ ?G =\{j. \ ?LLL-inner \ j \neq \text{None}\}\)
    assume \(\neg (\forall i \in \{d + 1..n\}. \text{LLL-inner} \ i = \text{None})\)
    hence G-not-empty: \(?G \neq \{\}\\) by auto
    define \(j\) where \(j = \text{Min} \ ?G\)
    have j-in-G: \(j \in \ ?G\) by (unfold j-def, rule Min-in[\{OF - G-not-empty\}], simp)
    hence j: \(j \in \{d + 1..n\}\) and LLL-not-None: \(?LLL-inner \ j \neq \text{None}\) using j-in-G
      by auto
    have \(\forall i \in \{d + 1..<j\}. \ ?LLL-inner \ i = \text{None}\) by auto
    proof (rule ccontr)
      assume \(\neg (\forall i \in \{d + 1..<j\}. \ ?LLL-inner \ i = \text{None})\)
      from this obtain i where i: \(i \in \{d + 1..<j\}\) and LLL-i: \(?LLL-inner \ i \neq \text{None}\)
        by auto
      hence iG: \(i \in \ ?G\) using i by auto
      moreover have j<i using iG j-def by auto
      ultimately show False by linarith
    qed
    hence all-None: \(\forall i \in \{d + 1..j - 1\}. \ ?LLL-inner \ i = \text{None}\) by auto
    obtain gs\(^{\prime}\) b\(^{\prime}\) f\(^{\prime}\) factor where LLL-inner-eq: \(?LLL-inner \ j = \text{Some} (gs\(^{\prime}\), b\(^{\prime}\), f\(^{\prime}\), factor)\)
      using LLL-not-None by force
    have Suc-jl-eq: \(\text{Suc} \ (j - 1) = j\) using j d0 by auto
    have jn: \(j - 1 < n\) using j by auto
    have dj: \(d \leq j - 1\) using j d0 by auto
    have degree: \(\bigwedge\text{factor}. p.dvdm \ u \ factor \implies \text{factor} \ dvd \ f \implies j - 1 \leq \text{degree} \ factor\)
      by (rule LLL-reconstruction-inner-None-upt-j[\{OF all-None dj\}, insert jn, auto])
    have LLL-inner-Some: \(?LLL-inner \ (\text{Suc} \ (j - 1)) = \text{Some} (gs\(^{\prime}\), b\(^{\prime}\), f\(^{\prime}\), factor)\)
      using LLL-inner-eq Suc-jl-eq by auto
    have deg-factor: \(\text{degree} \ factor = j - 1\) and ff\(^{\prime}\): \(f = f\(^{\prime}\) * factor\) and irreducible-factor: \text{irreducible}\_d \ factor\)
  qed
using LLL-reconstruction-inner-sound[OF dj jn degree LLL-inner-None] by (metis+)

have degree \( f' = n - (j - 1) \) using arg-cong[OF \( ff' \), of degree]
  by (subth (asm) degree-mult-eq, insert \( f0 \) \( ff' \) deg-factor, auto)
also have \( \ldots < n \) using irreducible-factor jn unfolding irreducible_d-deq deg-factor
by auto

finally have degree \( f' \): degree \( f' < degree f \) by auto
from \( ff' \) have factor-dvd-f: factor dvd \( f \) by auto
have \( \neg \) irreducible\_d \( f \)
  by (rule reducible_d, rule exI[\( of - f \)], rule exI[\( of - factor \)],
    intro conjI \( ff' \), insert deg-factor jn deg-f', auto)
thus False using irr-f by contradiction
qed

lemma LLL-reconstruction-inner-all-None:
  assumes \( i \): \( \forall i \in \{d+1..n\} \). LLL-reconstruction-inner \( p \ l gs f u \) Degs \( i = \) None
  and \( dj \): \( d < j \)
shows LLL-reconstruction-inner-loop \( p \ l gs f u \) Degs \( j = ([], 1, 1, f) \)
using dj
proof (induct \( j \) rule: LLL-reconstruction-inner-loop.induct[of \( f \) \( p \) \( l \) \( gs \) \( f u \) Degs])
  case \( 1 \ j \)
  let ?innerl = LLL-reconstruction-inner-loop \( p \ l gs f u \) Degs
  let ?inner = LLL-reconstruction-inner \( p \ l gs f u \) Degs
  note hyp = 1.hyps
  note dj = 1.prems(1)
  show ?case
  proof (cases \( j \leq n \))
    case \( True \)
    note jn = True
    have step: ?inner \( j = \) None
      by (cases \( d = j \), insert \( i \) jn \( dj \), auto)
    have ?inner \( j = \) ?innerl \( (j+1) \)
      using jn step by auto
    also have \( \ldots = ([], 1, 1, f) \)
      by (rule hyp[of - step], insert jn \( dj \), auto simp add: jn \( dj \))
    finally show ?thesis .
  qed auto
  qed

corollary irreducible-imp-LLL-reconstruction-inner-loop-f:
  assumes \( irr-f \): irreducible\_d \( f \) and \( dj \): \( d < j \)
shows LLL-reconstruction-inner-loop \( p \ l gs f u \) Degs \( j = ([], 1, 1, f) \)
  using irreducible-imp-LLL-reconstruction-inner-all-None[OF \( irr-f \)]
  using LLL-reconstruction-inner-all-None[OF - \( dj \)] by auto

lemma exists-index-LLL-reconstruction-inner-None:
  assumes inner-loop: LLL-reconstruction-inner-loop \( p \ l gs f u \) Degs \( j = (gs', b', f' \) factor)
  and \( i \): \( \forall i \in \{d+1..<j\} \). LLL-reconstruction-inner \( p \ l gs f u \) Degs \( i = \) None
  and \( dj \): \( d < j \) and \( jn \): \( j \leq n \) and \( f \): \( \neg \) irreducible\_d \( f \)
shows \( \exists j', j \leq j' \wedge j' \leq n \wedge d < j' \)
∧ (LLL-reconstruction-inner p l gs f u Degs j' = Some (gs', b', f', factor))
∧ (∀ i∈{d+1..<j'}). LLL-reconstruction-inner p l gs f u Degs i = None)
using inner-loop i dj jn
proof (induct j rule: LLL-reconstruction-inner-loop.induct[of f p l gs u Degs])
case (j)
let ?innerl = LLL-reconstruction-inner-loop p l gs f u Degs
let ?inner = LLL-reconstruction-inner p l gs f u Degs
note hyp = 1.hyps
note 1 = 1.prems(1)
note 2 = 1.prems(2)
note dj = 1.prems(3)
note jn = 1.prems(4)
show ?case
proof (cases ?inner j = None)
case True
proof (cases j=n)
case True note j-eq-n = True
show ?thesis
proof (cases ?inner n = None)
case True
have i2: ∀ i∈{d + 1..n}. ?inner i = None
  using 2 j-eq-n True by auto
have irreducible d f
  by(rule LLL-reconstruction-inner-all-None-imp-irreducible[OF i2])
thus ?thesis using f by simp
next
case False
have ?inner n = Some (gs', b', f', factor)
  using False 1 j-eq-n by auto
moreover have ∀ i∈{d + 1..<n}. ?inner i = None
  using 2 j-eq-n by simp
moreover have d < n using 1 2 jn j-eq-n
  using False d n nat-less-le
  using d-def dj by auto
ultimately show ?thesis using j-eq-n by fastforce
qed
next
case False
have ∃ j'≥ j + 1. j' ≤ n ∧ d < j' ∧
  ?inner j' = Some (gs', b', f', factor) ∧
  (∀ i∈{d + 1..<j'}. ?inner i = None)
proof (rule hyp)
  show ¬ degree f < j using jn by auto
  show ?inner j = None using True by auto
  show ?innerl (j + 1) = (gs', b', f', factor)
    using 1 True jn by auto
  show ∀ i∈{d + 1..<j + 1}. ?inner i = None
    by (metis 2 One-nat-def True add.comm-neutral add-Suc-right atLeastLessThan-iff

68
from this obtain \( j' \) where \( a1: j' \geq j + 1 \) and \( a2: j' \leq n \) and \( a3: d < j' \) and \( a4: \ ?inner j' = \text{Some} (g's', b', f', \text{factor}) \) and \( a5: (\forall i \in \{d + 1..<j\}). \ ?inner i = \text{None} \) by auto
moreover have \( j' \geq j \) using \( a1 \) by auto
ultimately show \( \\text{thesis} \) by fastforce
qed

next case False
have 1: \( \ ?inner j = \text{Some} (g's', b', f', \text{factor}) \) using False 1 \( jn \) by auto
moreover have 2: \( (\forall i \in \{d + 1..<j\}). \ ?inner i = \text{None} \) by (rule 2)
moreover have 3: \( j \leq n \) using \( jn \) by auto
moreover have 4: \( d < j \) using 2 False \( dj \) \( jn \) using \( \text{le-neg-implies-less} \) by fastforce
ultimately show \( \\text{thesis} \) by auto
qed

lemma \( \text{unique-factorization-m-1: pl.unique-factorization-m 1 (1, \{\#\})} \)
proof (intro \( \text{pl.unique-factorization-mI} \))
  fix \( d \) \( gs \)
  assume \( pl: \text{pl.factorization-m 1 (d,gs)} \)
  from \( \text{pl.factorization-m-degree[OF this]} \) have \( \text{deg0: } \bigwedge g. g \in \# gs \implies \text{pl.degree-m} \)
  \( g = 0 \) by auto
  \{
    assume \( gs \neq \{\#\} \)
    then obtain \( g \) \( hs \) where \( gs = \{\# g \#\} + hs \) by (cases \( gs \), auto)
    with \( pl \) have (*: \text{pl.irreducible}_{d-m} (pl.Mp g) \text{monic} (pl.Mp g) \) by (auto simp: \( \text{pl.factorization-m-def} \))
    with \( \text{deg0[of g, unfolded gs]} \) have False by (auto simp: \( \text{pl.irreducible}_{d-m-def} \))
  }
  hence \( gs = \{\#\} \) by auto
  with \( \text{pl show pl.Mf (d, gs) = pl.Mf (1, \{\#\}) by (cases d = 0,} \)
  \( \text{auto simp: \text{pl.factorization-m-def pl.Mf-def pl.Mp-def}} \)
qed (auto simp: \( \text{pl.factorization-m-def} \))

lemma \( \text{LLL-reconstruction-inner-loop-j-le-n:} \)
  assumes ret: \( \text{LLL-reconstruction-inner-loop p l gs f u Degs j = (g's',b'f',\text{factor})} \)
  and \( ij: \forall i \in \{d+1..<j\}. \text{LLL-reconstruction-inner p l gs f u Degs i = None} \)
  and \( n: n = \text{degree f} \)
  and \( jn: j \leq n \)
  and \( dj: d < j \)

69
shows $f = f' \ast \text{factor (is } ?g1\text{)}$
and \text{irreducible, factor (is } ?g2\text{)}
and $b' = \text{lead-coeff } f' (\text{is } ?g3\text{)}$
and $\text{pl.unique-factorization-m } f' (b', \text{mset } gs') (\text{is } ?g4\text{)}$
and $p.\text{dvdm } u \text{ factor (is } ?g5\text{)}$
and $gs \neq [] \rightarrow \text{length } gs' < \text{length } gs (\text{is } ?g6\text{)}$
and $\text{factor dvd } f (\text{is } ?g7\text{)}$
and $f' \text{ dvd } f (\text{is } ?g8\text{)}$
and $\text{set } gs' \subseteq \text{set } gs (\text{is } ?g9\text{)}$
and $gs' = [] \rightarrow f' = 1 (\text{is } ?g10\text{)}$

using $\text{ret } ij jn dj$
proof (atomize(full), induct $j$)
case 0
then show ?case using deg-u by auto
next
case (Suc $j$)
let $?innerl = \text{LLL-reconstruction-inner-loop } p l gs f u \text{Degs}$
let $?inner = \text{LLL-reconstruction-inner } p l gs f u \text{Degs}$
have $ij: \forall i \in \{d+1..j\}. \ ?inner i = None$
  using Suc.prems by auto
have $dj: d \leq j$ using Suc.prems by auto
have $jn: j<n$ using Suc.prems by auto
have $\text{deg: } Suc j \leq \text{degree } f$ using Suc.prems by auto
have $c: \land \text{factor. } p.\text{dvdm } u \text{ factor } \implies \text{factor dvd } f \implies j \leq \text{degree factor}$
  by (rule LLL-reconstruction-inner-None-upt-j[OF $ij dj$, insert $n jn$, auto])
have $1: ?innerl (Suc j) = (gs', b', f', \text{factor})$
  using Suc.prems by auto
show ?case
proof (cases $?inner (Suc j) = None$)
case False
have $\text{LLL-rw: } ?inner (Suc j) = \text{Some (gs', b', f', factor)}$
  using False deg Suc.prems by auto
  show ?thesis using LLL-reconstruction-inner-sound[OF $dj jn c \text{LLL-rw}$] by fastforce
next
case True note Suc-j-None = True
show ?thesis
proof (cases $d = j$)
case False
  have $nj: j \leq \text{degree } f$ using Suc.prems False by auto
  moreover have $dj2: d < j$ using Suc.prems False by auto
  ultimately show ?thesis using Suc.prems Suc.hyps by fastforce
next
case True note d-eq-j = True
show ?thesis
proof (cases irreducible, $d$ $f$)
case True
  have $\text{pl-Mp-1: } \text{pl.Mp } 1 = 1$ by auto
  have $d-Suc-j: d < Suc j$ using Suc.prems by auto
have \( ?inner \) (Suc \( j \)) = (\[1\], \( 1 \), \( f \))
by (rule irreducible-imp-LLL-reconstruction-inner-loop]\( \langleOF\) True d Suc-j\( \rangle\))
hence result-eq: (\[1\], \( 1 \), \( f \)) = (\( gs' \), \( b' \), \( f' \), \( \text{factor} \)) using Suc.prems by auto
moreover have thesis1: \( p \cdot \text{dvd} m \) \( u \cdot \text{factor} \) using \( u \cdot \text{result-eq} \) by auto
moreover have thesis2: \( f' = \text{pl}\cdot\text{Mp} (\text{Polynomial}\cdot\text{smult} \ b' \ (\text{prod}-\text{list} \ gs' ))\) using result-eq \( \text{pl-Mp-1} \) by auto
ultimately show \( ?\text{thesis} \) using \( \text{True} \) by (auto simp: unique-factorization-m-1)
next
case \( \text{False} \) note irreducible-f = \( \text{False} \)
have \( \exists j'. \text{Suc} \ j \leq j' \land j' \leq n \land d < j' \land (?inner j' = \text{Some} \ (\ gs' \), \( b' \), \( f' \), \( \text{factor} \)) \)
\( \land (\forall i \in \{d+1..<\text{Suc} \ j\}. \ ?\text{inner} \ i \ = \ \text{None} \)
proof (rule exists-index-LLL-reconstruction-inner-Some[\langleOF\ -\ -\ -\ \text{False}\rangle])

show ?innerl (Suc \( j \)) = (\( gs' \), \( b' \), \( f' \), \( \text{factor} \))
using Suc.prems by auto
show \( \forall i \in \{d+1..\langle\text{Suc} \ j\\} \cdot ?\text{inner} \ i \ = \ \text{None} \) using Suc.prems by auto
show \( \text{Suc} \ j \leq n \) using \( jn \) by auto
show \( d < \text{Suc} \ j \) using Suc.prems by auto
qed
from this obtain \( a \) where \( da \): \( d < a \) and \( an \): \( a \leq n \) and \( ja \): \( j \leq a \) and \( a1 \): \( ?\text{inner} \ a \ = \ \text{Some} \ (\ gs' \), \( b' \), \( f' \), \( \text{factor} \)) and \( a2 \): \( \forall i \in \{d+1..<a\}. \ ?\text{inner} \ i \ = \ \text{None} \) by auto
define \( j' \) where \( j'[\text{simp}]: j' \equiv a - 1 \)
have \( dj': d \leq j' \) using \( da \) by auto
have \( j': j' \neq 0 \) using \( dj' \) do by auto
hence \( j'n: j' < n \) using \( an \) by auto
have \( \text{LLL} \): \( ?\text{inner} \ (\text{Suc} \ j') \ = \ \text{Some} \ (\ gs' \), \( b' \), \( f' \), \( \text{factor} \)) using a1 j' by auto
have prev-None: \( \forall i \in \{d+1..j'\}. \ ?\text{inner} \ i \ = \ \text{None} \) using a2 j' by auto
have Suc-rw: \( \text{Suc} \ (j' - 1) = j' \) using \( j' \) by auto
have c: \( \forall \text{factor}. \ p\cdot\text{dvd} m \ u \text{ factor} \implies \text{factor} \ dvd f \implies \text{Suc} \ (j' - 1) \leq \text{degree factor} \)
by (rule LLL-reconstruction-inner-None-upt-j, insert \( dj' \) Suc-rw \( j'n \) prev-None, auto)
hence c2: \( \forall \text{factor}. \ p\cdot\text{dvd} m \ u \text{ factor} \implies \text{factor} \ dvd f \implies j' \leq \text{degree factor} \)
using \( j' \) by force
show \( ?\text{thesis} \) using LLL-reconstruction-inner-sound[\langleOF \ d'j' \ j'n \ c2 \ LLL\rangle] by fastforce
qed
qed
qed

lemma LLL-reconstruction-inner-loop-j-ge-n:
assumes ret: LLL-reconstruction-inner-loop p l gs f u Degs j = (\( gs' \), \( b' \), \( f' \), \( \text{factor} \))
and ij: \( \forall i \in \{d+1..n\}. \ LLL\text{-reconstruction-inner} \ p \ l \ gs \ f \ u \ Degs \ i \ = \ \text{None} \)

71
and \(dj: d < j\)
and \(jn: j > n\)
shows \(f = f' \cdot \text{factor}(is \ ?g1)\)
and irreducible\(_d\) factor \((is \ ?g2)\)
and \(b' = \text{lead-coeff} f'(is \ ?g3)\)
and \(\text{pl.unique-factorization-m} f'(b', \text{mset gs'}) (is \ ?g4)\)
and \(\text{p.dvdum u factor}(is \ ?g5)\)
and \(gs \neq [] \rightarrow \text{length gs'} < \text{length gs}(is \ ?g6)\)
and factor \(\text{dvd} f(is \ ?g7)\)
and \(f' \text{ dvd} f(is \ ?g8)\)
and set gs' \(\subseteq\) set gs \((is \ ?g9)\)
and \(f' = 1(is \ ?g10)\)
proof

- have \(\text{LLL-reconstruction-inner-loop p l gs f u Degs j} = ([],1,1,f)\) using \(jn\) by auto
- hence \(gs' = []\) and \(b' = 1\) and \(f': f' = 1\) and factor: \(\text{factor} = f\) using \(ret\) by auto
  - have irreducible\(_d\) \(f\)
    - by \(\text{(rule LLL-reconstruction-inner-all-None-imp-irreducible[OF ij])}\)
  - thus \(?g1 \ ?g2 \ ?g3 \ ?g4 \ ?g5 \ ?g6 \ ?g7 \ ?g8 \ ?g9 \ ?g10\) using \(f'\) factor \(b'\) \(gs'\) \(u-f\)
    - by \(\text{(auto simp: unique-factorization-m-1)}\)

qed

lemma \(\text{LLL-reconstruction-inner-loop}:\)
assumes \(\text{ret: LLL-reconstruction-inner-loop p l gs f u Degs j} = (gs',b',f',\text{factor})\)
and \(ij: \forall i \in \{d + 1 . . . j\}. \text{LLL-reconstruction-inner p l gs f u Degs i} = \text{None}\)
and \(n: n = \text{degree f}\)
and \(dj: d < j\)
shows \(f = f' \cdot \text{factor}(is \ ?g1)\)
and irreducible\(_d\) factor \((is \ ?g2)\)
and \(b' = \text{lead-coeff} f'(is \ ?g3)\)
and \(\text{pl.unique-factorization-m} f'(b', \text{mset gs'}) (is \ ?g4)\)
and \(\text{p.dvdum u factor}(is \ ?g5)\)
and \(gs \neq [] \rightarrow \text{length gs'} < \text{length gs}(is \ ?g6)\)
and factor \(\text{dvd} f(is \ ?g7)\)
and \(f' \text{ dvd} f(is \ ?g8)\)
and set gs' \(\subseteq\) set gs \((is \ ?g9)\)
and \(gs' = [] \rightarrow f' = 1(is \ ?g10)\)
proof \(\text{(atomize(full),(cases j>n; intro conjI))}\)

- case \(\text{True}\)
  - have \(ij2: \forall i \in \{d + 1 . . n\}. \text{LLL-reconstruction-inner p l gs f u Degs i} = \text{None}\)
    - using \(ij\) \(\text{True}\) by auto
  - show \(?g1 \ ?g2 \ ?g3 \ ?g4 \ ?g5 \ ?g6 \ ?g7 \ ?g8 \ ?g9 \ ?g10\)
    - using \(\text{LLL-reconstruction-inner-loop-j-ge-n}[OF ret ij2 dj True]\) by blast+

- next
  - case \(\text{False}\)
  - hence \(jn: j \leq n\) by simp
  - show \(?g1 \ ?g2 \ ?g3 \ ?g4 \ ?g5 \ ?g6 \ ?g7 \ ?g8 \ ?g9 \ ?g10\)
    - using \(\text{LLL-reconstruction-inner-loop-j-le-n}[OF ret ij n jn dj]\) by blast+
7.3.3 Outer loop

lemma \textit{LLL-reconstruction}''
\begin{align*}
\text{assumes } & 1: \text{LLL-reconstruction}'' p l g s \ b f \ G = G' \\
\text{and } & \text{irreducible-G: } \bigwedge \text{factor} \ \text{factor} \in \text{set} \ G \implies \text{irreducible}_{d \text{ factor}} \\
\text{and } & 3: F = f * \text{prod-list} \ G \\
\text{and } & 4: \text{pl.unique-factorization-m f} (\text{lead-coeff f}, \text{mset} \ g s) \\
\text{and } & 5: gs \not= [] \\
\text{and } & 6: \bigwedge gi \ \text{gi} \in \text{set} \ gs \implies \text{pl.Mp} \ gi = gi \\
\text{and } & 7: \bigwedge gi \ \text{gi} \in \text{set} \ gs \implies \text{p.irreducible}_{d-m} \ gi \\
\text{and } & 8: \text{p.square-free-m f} \\
\text{and } & 9: \text{coprime (lead-coeff f) p} \\
\text{and } & \text{sf-F: square-free F} \\
\text{shows } & (\forall g \in \text{set} \ G'). \text{irreducible}_{d \ g} \ \land \ F = \text{prod-list} \ G' \\
\text{using } & 1 \text{ irreducible-G 3 4 5 6 7 8 9} \\
\text{proof } & (\text{induction} \ gs \ \text{arbitrary: } b \ f \ G \ G' \ \text{rule: length-induct}) \\
\text{case } & (1 gs) \\
\text{note } & LLL-f' = 1.\text{prems}(1) \\
\text{note } & \text{irreducible-G} = 1.\text{prems}(2) \\
\text{note } & F-f-G = 1.\text{prems} (3) \\
\text{note } & f-gs-factor = 1.\text{prems} (4) \\
\text{note } & gs-not-empty = 1.\text{prems} (5) \\
\text{note } & \text{norm} = 1.\text{prems}(6) \\
\text{note } & \text{irred-p} = 1.\text{prems}(7) \\
\text{note } & sf = 1.\text{prems}(8) \\
\text{note } & \text{cop} = 1.\text{prems}(9) \\
\text{obtain } & u \text{ where choose-u-result: choose-u gs = u by auto} \\
\text{from } & \text{choose-u-member[OF gs-not-empty, unfolded choose-u-result]} \\
\text{have } & u-gs: u \in \text{set} \ gs \ \text{by auto} \\
\text{define } & d n \text{ where } [\text{simp}]: d = \text{degree} u n = \text{degree} f \\
\text{hence } & n-def: n = \text{degree} f n \equiv \text{degree} f \ \text{by auto} \\
\text{define } & gs''/\text{where} \ gs'' = \text{remove1} u g s \\
\text{define } & \text{degs where degs = map degree gs''} \\
\text{define } & \text{Degs where Degs = (+) d \cdot sub-mset-sums degs} \\
\text{obtain } & gs' b' h \text{ factor where inner-loop-result:} \\
& \text{LLL-reconstruction-inner-loop} p l g s f u \text{ Degs} (d+1) = (gs',b',h,\text{factor}) \\
& \text{by (metis prod-cases4)} \\
\text{have } & a1: \\
& \text{LLL-reconstruction-inner-loop} p l g s f u \text{ Degs} (d+1) = (gs', b', h, \text{factor}) \\
& \text{using inner-loop-result by auto} \\
\text{have } & a2: \\
& \forall i\in\{\text{degree} u + 1..< (d+1)\}. \text{LLL-reconstruction-inner} p l g s f u \text{ Degs} i = \text{None} \\
& \text{by auto} \\
\text{have } & \text{LLL-reconstruction}'' p l g s \ b f \ G = \text{LLL-reconstruction}'' p l g s' b' h (\text{factor} \ # G) \\
& \text{unfolding LLL-reconstruction}''.\text{simps[of p l gs] using gs-not-empty}
unfolding Let-def using choose-u-result inner-loop-result unfolding Degs-def
degs-def gs"-def by auto
hence LLL-eq: LLL-reconstruction" p l gs' b' h (factor # G) = G' using LLL-f'
by auto
from pl.unique-factorization-m-imp-factorization[OF f-gs-factor,
unfolded pl.factorization-m-def] norm
have f-gs: pl.eq-m f (small (lead-coeff f) (prod-mset (mset gs))) and
  mon: g ∈ set gs ⇒ monic g and irred: g ∈ set gs ⇒ pl.irreducible u-m g for
g by auto
{ from split-list[OF u-gs] obtain gs1 gs2 where gs: gs = gs1 @ u # gs2 by auto
  from f-gs[unfolded gs] have pl.dedv u f unfolding pl.dedv-def
  by (intro exI[of - smallt (lead-coeff f) (prod-mset (mset (gs1 @ gs2)))], auto)
} note pl-uf = this
hence p-uf: p.dedv u f by (rule pl.dedv-imp-p-dedv)
have monic-u: monic u using mon[OF u-gs] .
have irred-u: p.irreducible-m u using irred-p[OF u-gs] by auto
have degree-m-u: p.degree-m u = degree u using monic-u by simp
have degree-u[simp]: 0 < degree u
  using irred-u by (fold degree-m-u, auto simp add: p.irreducible-degree)
have deg-u-d: degree u < d + 1 by auto
from F-F-G have f-dedv-F: f dedv F by auto
from square-free-factor[OF f-dedv-F sf-F] have sf-f: square-free f .
from norm have norm-map: map pl.Mp gs = gs by (induct gs, auto)
{ fix factor
  assume factor-f: factor dvd f and u-factor: p.dedv u factor
  from factor-f obtain h where f: f = factor * h unfolding dvd-def by auto
  obtain gs1 gs2 where part: List.partition (λgi. p.dedv gi factor) gs = (gs1 ,
gs2) by force
  from p.unique-factorization-m-factor-partition[OF l0 f-gs-factor f cop sf part]
  have factor: pl.unique-factorization-m-factor (lead-coeff factor, mset gs1) by auto
  from u-factor part u-gs have u-gs1: u ∈ set gs1 by auto
define gs1' where gs1' = remove1 u gs1
from remove1-mset[OF u-gs1, folded gs1'-def]
have gs1: mset gs1 = add-mset u (mset gs1') by auto
from remove1-mset[OF u-gs, folded gs''-def]
have gs: mset gs = add-mset u (mset gs'') by auto
from part have filter: gs1 = [gs'@ gs . p.dedv gi factor] by auto
have mset gs1 ⊆# mset gs unfolding filter mset-filter by simp
hence sub: mset gs1' ⊆# mset gs'' unfolding gs gs1 by auto
from p.coprime-lead-coeff-factor[OF :prime p] cop[unfolded f]
have cop': coprime (lead-coeff factor) p by auto
have p-factor0: p.Mp factor ≠ 0
  by (metis f p.Mp-0 p.square-free-m-def poly-mod.square-free-m-factor(1) sf)
have pl-factor0: pl.Mp factor ≠ 0 using p-factor0 l0
  by (metis p.Mp-0 p-Mp-pl-Mp)
from pl.factorization-m-degree[OF pl.unique-factorization-m-imp-factorization[OF factor]] pl-factor0
have pl.degree-m factor = sum-mset (image-mset pl.degree-m (mset gs1)) .
also have image-mset pl.degree-m (mset gs1) = image-mset degree (mset gs1)
by (rule image-mset-cong, rule pl.monic-degree-m[OF mon], insert part, auto)
also have pl.degree-m factor = degree factor
by (rule pl.degree-m-eq[OF p.coprime-exp-mod[OF cop' l0] pl.m1])
finally have degree factor = d + sum-mset (image-mset degree (mset gs1'))
unfolding gs1 by auto
moreover have sum-mset (image-mset degree (mset gs1')) ∈ sub-mset-sums
degs unfolding degs-def
sub-mset-sums mset-map
by (intro imageI CollectI image-mset-subseteq-mono[OF sub])
ultimately have degree factor ∈ Degrees unfolding Degrees-def by auto
} note Degrees = this
have length-less: length gs' < length gs
and irreducible-factor: irreducible_d factor
and h-dvd-f: h dvd f
and f-h-factor: f = h * factor
and h-eq: pl.unique-factorization-m h (b', mset gs')
and gs'-gs: set gs' ⊆ set gs
and b': b' = lead-coeff h
and h1: gs' = [] → h = 1
using LLL-reconstruction-inner-loop[OF degree-u monic-u irred-u p-uf f-dvd-F
n-def(2)
] f-gs-factor cop sf sf-f u-gs norm-map Degrees
a1 a2 n-def(1)) deg-u-d gs-not-empty by metis+
have F-h-factor-G: F = h * prod-list (factor # G)
using F-h-G f-h-factor by auto
hence h-dvd-F: h dvd F using f-dvd-F dvd-trans by auto
have irreducible-factor-G: Λ x. x ∈ set (factor # G) → irreducible_d x
using irreducible-factor irreducible-G by auto
from p.coprime-lead-coeff-factor[OF (prime p) cop|unfolded f-h-factor]]
have cop': coprime (lead-coeff h) p by auto
have le': lead-coeff (prod-list (lead-coeff h) (prod-list gs')) = lead-coeff h
by (insert gs'-gs, auto intro!: monic-prod-list intro: mon)
have le: lead-coeff (pl.Mp (prod-list (lead-coeff h) (prod-list gs'))) = pl.M (lead-coeff h)
proof (subst pl.degree-m-eq-lead-coeff[OF pl.degree-m-eq[OF - pl.m1]]; unfold le')
  show lead-coeff h mod p 'l ≠ 0 using p.coprime-exp-mod[OF cop' l0] by auto
qed auto
have ah: pl.unique-factorization-m h (lead-coeff h, mset gs') using h-eq unfolding b' .
from p.square-free-m-factor[OF sf|unfolded f-h-factor]] have sf': p.square-free-m
h by auto
show ?case
proof (cases gs' ≠ [])
case gs'-not-empty: True
75
show ?thesis
  by (rule 1.IH\{rule-format, OF length-less LLL-eq irreducible-factor-G F-h-factor-G

  uh gs' -not-empty norm irre-d p sf' cop', insert gs' - gs, auto)

next
  case False
  have pl-ge0: \( p\cdot l > 0 \) using p1 by auto
  have G'-eq: \( G' = \text{factor} \# G \) using LLL-eq False using LLL-reconstruction''.simps
    by auto
  have condition1: \((\forall a \in \text{set} G'. \text{irreducible}_a a)\) using irreducible-factor-G G'-eq
    by auto
  have h-eq2: \( \text{pl}.Mp h = \text{pl}.Mp \[b^2\] \) using h-eq False
    unfolding \( \text{pl.unique-factorization-m-alt-def \ pl.factorization-m-def \ by auto} \)
  have Mp-const-rw[simp]: \( \text{pl}.Mp \[b^2\] = \[b' \mod p \cdot l\] \) using \( \text{pl}.Mp-const-poly \)
    by blast
  have condition2: \( F = \text{prod-list} G' \) using h1 False f-h-factor G'-eq F-h-factor-G
    by auto
  show ?thesis using condition1 condition2 by auto
qed

context
  fixes \( gs :: \text{int \ poly list} \)
  assumes gs-hen: berlekamp-hensel p l F = gs
    and cop: coprime (lead-coeff F) p
    and sf: poly-mod.square-free-m p F
    and sf-F: square-free F
begin

lemma gs-not-empty: \( gs \neq [] \)
proof (rule ccontr, simp)
  assume gs: \( gs = [] \)
  obtain c fs where c-fs: finite-field-factorization-int \( p \ F = (c, fs) \) by force
  have sort (map degree fs) = sort (map degree gs)
    by (rule p.berlekamp-hensel-main(2)[OF - gs-hen cop sf c-fs], simp add: l0)
  hence fs-empty: \( fs = [] \) using gs by (cases fs, auto)
  hence fs: \( \text{mset} \ fs = \{\#\} \) by auto
  have p.unique-factorization-m F (c, mset fs) and c: \( c \in \{0..<p\} \)
    using p.finite-field-factorization-int[OF sf c-fs] by auto
  hence p.factorization-m F (c, mset fs)
    using p.unique-factorization-m-imp-factorization by auto
  hence eq-m-F: \( p\cdot eq-m \ F \ [c::] \) unfolding fs p.factorization-m-def by auto
  hence \( \theta = p\cdot \text{degree} - m \ F \) by (simp add: p.Mp-const-poly)
  also have \( \ldots = \text{degree} \ F \) by (rule p.degree-m-eq[OF - p1], insert cop p1, auto)
  finally have \( \text{degree} \ F = \theta \ .. \)
  thus False using \( \text{N0} \) by simp
qed

lemma reconstruction-of-algorithm-16-22:
assumes 1: reconstruction-of-algorithm-16-22 p l gs F = G
shows (\forall g \in \text{set } G. \text{irreducible}_d g) \land F = \text{prod-list } G
proof
  note * = p.berlekamp-hensel-unique[of cop sf gs-hen l0]
  obtain e fs where finite-field-factorization-int p F = (e, fs) by force
  from p.berlekamp-hensel-main[of l0 gs-hen cop sf this]
  show ?thesis
    using 1 unfolding reconstruction-of-algorithm-16-22-def Let-def
      by (intro LLL-reconstruction' [OF - - - - gs-not-empty], insert * sf sf-F cop, auto)
qed
end

7.3.4 Final statement

lemma factorization-algorithm-16-22:
  assumes res: factorization-algorithm-16-22 f = G
  and sff: square-free f
  and deg: degree f > 0
  shows (\forall g \in \text{set } G. \text{irreducible}_d g) \land f = \text{prod-list } G
proof
  let ?lc = lead-coeff f
  define p where p = suitable-prime-bz f
  obtain c gs where fff: finite-field-factorization-int p f = (c, gs) by force
  let ?degs = map degree gs
  note res = res[unfolded factorization-algorithm-16-22-def Let-def, folded p-def,
    unfolded fff split, folded]
  from suitable-prime-bz[of sff refl]
  have prime: prime p and cop: coprime ?lc p and sf: poly-mod.square-free-m p f
    unfolding p-def by auto
  note res
  from prime interpret poly-mod-prime p by unfold-locales
  define K where K = 2 ^ (5 * degree f * degree f) * ||f||^2 ^ (2 * degree f)
  define N where N = sqrt-int-ceiling K
  have K0: K \geq 0 unfolding K-def by auto
  have N0: N \geq 0 unfolding N-def sqrt-int-ceiling using K0
    by (smt of-int-nonneg real-sqrt-ge-0-iff zero-le-ceiling)
  define n where n = find-exponent p N
  note res = res[folded n-def[unfolded N-def K-def]]
  note n = find-exponent[of m1, of N, folded n-def]
  note bh = berlekamp-and-hensel-separated[of cop sf refl fff n(2)]
  note res = res[folded bh(1)]
  show ?thesis
  proof (rule reconstruction-of-algorithm-16-22[of prime deg - refl cop sf sff res])
    from n(1) have N \leq p ^ n by simp
    hence *: N ^ 2 \leq (p ^ n) ^ 2
  qed
by (intro power-mono N0, auto)
show $2 \cdot (\text{degree } f)^2 \cdot \text{B2-LLL } f \cdot (2 \cdot \text{degree } f) \leq (p \cdot n)^2$
proof (rule order.trans[OF -])
  have $2 \cdot (\text{degree } f)^2 \cdot \text{B2-LLL } f \cdot (2 \cdot \text{degree } f) = K$
  unfolding K-def B2-LLL-def by (simp add: ac-simps
    power-mult-distrib power2-eq-square power-add[symmetric])
  also have $\ldots \leq N^2$ unfolding N-def by (rule sqrt-int-ceiling-bound[OF K0])
  finally show $2 \cdot (\text{degree } f)^2 \cdot \text{B2-LLL } f \cdot (2 \cdot \text{degree } f) \leq N^2$.
qed
qed

lift-definition increasing-lattices-LLL-factorization :: int-poly-factorization-algorithm
is factorization-algorithm-16-22 using factorization-algorithm-16-22 by auto

thm factorize-int-poly[of increasing-lattices-LLL-factorization]
end

8 Mistakes in the textbook Modern Computer Algebra (2nd edition)

theory Modern-Computer-Algebra-Problem
imports Factorization-Algorithm-16-22
begin

fun max-degree-poly :: int poly ⇒ int poly ⇒ int poly
where max-degree-poly a b = (if degree a ≥ degree b then a else b)

fun choose-u :: int poly list ⇒ int poly
where choose-u [] = undefined
| choose-u [gi] = gi
| choose-u (gi # gj # gs) = max-degree-poly gi (choose-u (gj # gs))

8.1 A real problem of Algorithm 16.22

Bogus example for Modern Computer Algebra (2nd edition), Algorithm 16.22, step 9: After having detected the factor $[1, 1, 0, 1:]$, the remaining polynomial $f^*$ will be 1, and the remaining list of modular factors will be empty.

lemma let f = [1,1]: * [1,1,0,1:];
  p = suitable-prime-bz f;
  b = lead-coeff f;
  A = linf-norm-poly f; n = degree f; B = sqrt-int-ceiling (n+1) * 2^n * A;
  Bnd = 2^(n+2 div 2) * B^(2*n); l = log-ceiling p Bnd;
  (-, fs) = finite-field-factorization-int p f;
  gs = hensel-lifting p l f fs;
\[ u = \text{choose-u } gs; \]
\[ d = \text{degree } u; \]
\[ g\text{-star} = [2,2,0,2 :: \text{int}]; \]
\[ (gs',hs') = \text{List.partition} (\lambda gi. \text{poly-mod.dedm p gi g\text{-star}}) gs; \]
\[ h\text{-star} = \text{smult b } (\text{prod-list hs'}); \]
\[ f\text{-star} = \text{primitive-part h\text{-star}} \]
\[ \text{in } (hs' = [] \land f\text{-star} = 1) \text{ by eval} \]

8.2 Another potential problem of Algorithm 16.22

Suppose that \( g^* \) is \( p^l \). (It is is not yet clear whether lattices exists where this \( g^* \) is short enough). Then \( pp(g^*) = 1 \) is detected as \textit{irreducible} factor and the algorithm stops.

\textbf{definition} \textit{input-poly} = [1,0,0,0,1,1,0,0,1,0,1,0,1 :: \text{int}] 

For \textit{input-poly} the factorization will result in a lattice where each initial basis element has a Euclidean norm of at least \( p^l \) (since the input polynomial \( u \) has a norm larger than \( p^l \)). So, just from the norm of the basis one cannot infer that the lattice contains small vectors.

\textbf{lemma} let \( f = \text{input-poly}; \)
\[ p = \text{suitable-prime-bz } f; \]
\[ b = \text{lead-coeff } f; \]
\[ A = \text{linf-norm-poly } f; n = \text{degree } f; B = \text{sqrt-int-ceiling } (n+1) * 2^n * A; \]
\[ \text{Bnd} = 2^\lceil (n^2 + 2) + B \rceil \cdot 2^n; l = \text{log-ceiling } p \cdot \text{Bnd}; \]
\[ (\cdot, fs) = \text{finite-field-factorization-int } p f; \]
\[ gs = \text{hensel-lifting p } f fs; \]
\[ a = \text{choose-a } gs; \]
\[ pl = p^l; \]
\[ pl2 = pl \div 2; \]
\[ a' = \text{poly-mod.inv-Mp2 } pl pl2 \text{ (poly-mod.Mp } pl \text{ (smult b u))} \]
\[ \text{in } \text{sqrt-int-floor } (\text{sq-norm } a') > pl \text{ by eval} \]

The following calculation will show that the norm of \( g^* \) is not that much shorter than \( p^l \) which is an indication that it is not obvious that in general \( p^l \) cannot be chosen as short polynomial.

\textbf{definition} \textit{compute-norms} = (let \( f = \text{input-poly}; \)
\[ p = \text{suitable-prime-bz } f; \]
\[ b = \text{lead-coeff } f; \]
\[ A = \text{linf-norm-poly } f; n = \text{degree } f; B = \text{sqrt-int-ceiling } (n+1) * 2^n * A; \]
\[ \text{Bnd} = 2^\lceil (n^2 + 2) + B \rceil \cdot 2^n; l = \text{log-ceiling } p \cdot \text{Bnd}; \]
\[ (\cdot, fs) = \text{finite-field-factorization-int } p f; \]
\[ gs = \text{hensel-lifting p } f fs; \]
\[ a = \text{choose-a } gs; \]
\[ pl = p^l; \]
\[ pl2 = pl \div 2; \]
\[ a' = \text{poly-mod.inv-Mp2 } pl pl2 \text{ (poly-mod.Mp } pl \text{ (smult b u))}; \]
\[ d = \text{degree } u; \]

79
\[ \begin{align*}
&\text{pl} = p \cdot l; \\
&L = \text{factorization-lattice } u' \ 1 \ pl; \\
&g\text{-star} = \text{short-vector } 2 \ L \\
&\text{in} \bigg( \\
\quad \left( p' \text{-l:} \right. \quad \text{"@ show pl @ shows-nl [] @} \\
\quad \left. \text{"norm u:} \quad \text{"@ show } (\sqrt{\text{int-floor}} \ (\text{sq-norm } u')) \text{ @ shows-nl [] @} \\
\quad \left. \text{"norm g\text{-star:} } \quad \text{"@ show } (\sqrt{\text{int-floor}} \ (\text{sq-norm } \text{vec g\text{-star}})) \text{ @ shows-nl [] @} \\
\quad \text{shows-nl [] @} \bigg) \\
\end{align*} \]

**export-code compute-norms in Haskell**

- \( p' \approx 6.61056 \cdot 10^{122} \), namely 66105596879024598951915308032771039828404682964281219284648374405791236311345825189210439715284847591212025023358304256
- \( \text{norm } u: \approx 6.67555 \cdot 10^{122} \), namely 66755505893812790838614155970749040661775649285326930673512573918235231876978270147733994030499205729993307341153235059302
- \( \text{norm } g\text{-star:} \approx 5.02568 \cdot 10^{110} \), namely 5025678718888937892581075993975033899734873138630

### 8.3 Verified wrong results

An equality in example 16.24 of the textbook which is not valid.

**lemma** let \( g2 = [-984,1]; \)

\( g3 = [-72,1]; \)

\( g4 = [-6828,1]; \)

\( \text{rhs} = [-1728,-840,-420,6]; \)

\( \text{in} \sim poly\text{-mod.eq-m } (5^6) \ (\text{smult } 6 \ (g2*g3*g4)) \ (\text{rhs}) \ \text{by eval} \)

**end**

### References

