

5.3	Proof of the key lemma 16.20	21
5.4	Properties of the computed lattice and its connection with Sylvester matrices	22
5.5	Proving that <i>factorization-lattice</i> returns a basis of the lattice	23
5.6	Being in the lattice is being a multiple modulo	25
5.7	Soundness of the LLL factorization algorithm	31
6	Calculating All Possible Sums of Sub-Multisets	46
7	Implementation and soundness of a modified version of Al- gorithm 16.22	47
7.1	Previous lemmas obtained using local type definitions	47
7.2	The modified version of Algorithm 16.22	48
7.3	Soundness proof	50
7.3.1	Starting the proof	50
7.3.2	Inner loop	56
7.3.3	Outer loop	73
7.3.4	Final statement	77
8	Mistakes in the textbook Modern Computer Algebra (2nd edition)	78
8.1	A real problem of Algorithm 16.22	78
8.2	Another potential problem of Algorithm 16.22	79
8.3	Verified wrong results	80

1 Introduction

In order to factor an integer polynomial f , we may assume a *modular* factorization of f into several monic factors u_i : $f \equiv \text{lc}(f) \cdot \prod_i u_i$ 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 $\|f\|^{\text{degree}(g)} \cdot \|g\|^{\text{degree}(f)} < m$, then $h = \text{gcd}(f, g)$ 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 $0 < \text{degree}(f_1) < n$. 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 $\text{degree}(g) < n$. The short vector requirement will provide an upper bound to satisfy the assumption $\|f\|^{\text{degree}(g)} \cdot \|g\|^{\text{degree}(f)} < m$.

$$\|g\| \leq 2^{(n-1)/2} \cdot \|f_1\| \leq 2^{(n-1)/2} \cdot 2^{n-1} \|f\| = 2^{3(n-1)/2} \|f\| \quad (1)$$

$$\|f\|^{\text{degree}(g)} \cdot \|g\|^{\text{degree}(f)} \leq \|f\|^{n-1} \cdot (2^{3(n-1)/2} \|f\|)^n = \|f\|^{2n-1} \cdot 2^{3n(n-1)/2} \quad (2)$$

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 $h = \text{gcd}(f, g)$ 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 `FactorizationAlgorithm16.22.thy` and `ModernComputerAlgebraProblem.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  $\forall g_i \in set\ gs.$  irreducible  $g_i$ 

```

Finally, we also have been able to fix Algorithm 16.22 and provide a formal correctness proof of 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*: $\text{norm1 } f \leq 2^{(\text{degree } f)} * \text{mahler-measure } f$

proof (*cases* $f = 0$)

case $f0$: *False*

have cf : $\text{coeffs } f = \text{map } (\lambda i. \text{coeff } f i) [0 ..< \text{Suc}(\text{degree } f)]$ **unfolding** *coeffs-def*

using $f0$ **by** *auto*

have *real-of-int* ($\text{sum-list } (\text{map } \text{abs } (\text{coeffs } f))$)

$= (\sum i \leq \text{degree } f. \text{real-of-int } |\text{poly.coeff } f i|)$

unfolding *cf of-int-hom.hom-sum-list* **unfolding** *sum-list-sum-nth*

by (*rule sum.cong, force, auto simp: o-def nth-append*)

also have $\dots \leq (\sum i \leq \text{degree } f. \text{real } (\text{degree } f \text{ choose } i) * \text{mahler-measure } f)$

by (*rule sum-mono, rule Mignotte-bound*)

also have $\dots = \text{real } (\text{sum } (\lambda i. (\text{degree } f \text{ choose } i)) \{.. \text{degree } f\}) * \text{mahler-measure } f$

unfolding *sum-distrib-right[symmetric]* **by** *auto*

also have $\dots = 2^{(\text{degree } f)} * \text{mahler-measure } f$ **unfolding** *choose-row-sum* **by** *auto*

finally show *?thesis* **unfolding** *norm1-def* .

qed (*auto simp: mahler-measure-ge-0 norm1-def*)

lemma *mahler-measure-l2norm*: $\text{mahler-measure } f \leq \text{sqrt } (\text{of-int } \|f\|^2)$

using *Landau-inequality-mahler-measure[of f]* **unfolding** *sq-norm-poly-def*

by (*auto simp: power2-eq-square*)

lemma *sq-norm-factor-bound*:

fixes $f h :: \text{int poly}$

assumes *dvd*: $h \text{ dvd } f$ **and** $f0$: $f \neq 0$

shows $\|h\|^2 \leq 2^{(2 * \text{degree } h)} * \|f\|^2$

proof –

let $?r = \text{real-of-int}$

have $h21$: $?r \|h\|^2 \leq (?r (\text{norm1 } h))^2$ **using** *norm2-le-norm1-int[of h]*

by (*metis of-int-le-iff of-int-power*)

also have $\dots \leq (2^{(\text{degree } h)} * \text{mahler-measure } h)^2$

using *power-mono[OF norm-1-bound-mignotte[of h], of 2]*

by (*auto simp: norm1-ge-0*)

also have $\dots = 2^{(2 * \text{degree } h)} * (\text{mahler-measure } h)^2$

by (*simp add: power-even-eq power-mult-distrib*)

also have $\dots \leq 2^{(2 * \text{degree } h)} * (\text{mahler-measure } f)^2$

by (*rule mult-left-mono[OF power-mono], auto simp: mahler-measure-ge-0*

mahler-measure-dvd[OF f0 dvd])

```

also have ...  $\leq 2^{(2 * \text{degree } h) * ?r (\|f\|^2)}$ 
proof (rule mult-left-mono)
  have ?r ( $\|f\|^2$ )  $\geq 0$  by auto
  from real-sqrt-pow2[OF this]
  show (mahler-measure f)2  $\leq ?r (\|f\|^2)$ 
    using power-mono[OF mahler-measure-l2norm[of f], of 2]
    by (auto simp: mahler-measure-ge-0)
qed auto
also have ... = ?r (2(2*degree h) *  $\|f\|^2$ )
  by (simp add: ac-simps)
finally show  $\|h\|^2 \leq 2^{(2 * \text{degree } h) * \|f\|^2}$  unfolding of-int-le-iff .
qed

end

```

3 Executable dvdm 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::int poly*
assumes *g: degree g < j*
and *r: degree r < d*
and *u: degree u = d*
and *g1: g = q * u + smult m r*
and *q: q ≠ 0 and m-not0: m ≠ 0*

shows *degree q < j - d*

proof –

have *u-not0: u ≠ 0* **using** *u r* **by** auto
have *d-uq: d ≤ degree (u*q)* **using** *u degree-mult-right-le[OF q]* **by** auto
have *j: j > degree (q* u + smult m r)* **using** *g1 g* **by** auto
have *degree (smult m r) < d* **using** *degree-smult-eq m-not0 r* **by** auto
also have ... $\leq \text{degree } (u*q)$ **using** *d-uq* **by** auto
finally have *deg-mr-uq: degree (smult m r) < degree (q*u)*
by (simp add: mult commute)
have *j2: degree (q* u + smult m r) = degree (q*u)*
by (rule degree-add-eq-left[OF deg-mr-uq])

also have $\dots = \text{degree } q + \text{degree } u$
by (rule degree-mult-eq[OF q u-not0])
finally have $\text{degree } q = \text{degree } g - \text{degree } u$ **using** g1 **by** auto
thus ?thesis
using j j2 $\langle \text{degree } (q * u) = \text{degree } q + \text{degree } u \rangle u$
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: $r1 = 0 \vee \text{degree } r1 < \text{degree } g$
and r2: $r2 = 0 \vee \text{degree } r2 < \text{degree } g$
shows $q1 = q2 \wedge r1 = r2$
proof –
have $0 = g * q1 + r1 - (g * q2 + r2)$ **using** f1 f2 **by** auto
also have $\dots = g * (q1 - q2) + r1 - r2$
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 $\text{degree } (g * (q1 - q2)) \geq \text{degree } g$
by (simp add: degree-mult-right-le)
moreover have $\text{degree } (r2 - r1) < \text{degree } g$
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 $\text{pdivmod } f g = \text{pdivmod-monic } f g$
proof –
obtain q r **where** qr: $\text{pdivmod } f g = (q,r)$ **by** simp
obtain Q R **where** QR: $\text{pdivmod-monic } f g = (Q,R)$ **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: $r=0 \vee \text{degree } r < \text{degree } g$
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  $\vee$  degree R < degree g
  by (rule pdivmod-monic[OF g QR])
  have q=Q  $\wedge$  r=R by (rule uniqueness-algorithm-division-poly[OF f1 f2 g0 r R])
  thus ?thesis using qr QR by auto
qed

```

```

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

```

```

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  $\neq$  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  $\vee$  degree R < degree G
  and Mp-f: Mp f = Mp g * q + r
  and r: r = 0  $\vee$  degree r < degree (Mp g)
shows Q = of-int-poly q  $\wedge$  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' : $Mp\ g = to-int-poly\ G$ **unfolding** g
by (*simp add: Mp-f-representative*)
have f'' : $of-int-poly\ (Mp\ f) = F$
by (*metis (no-types, lifting) Dp-Mp-eq Mp-f-representative*
Mp-smult-m-0 add-cancel-left-right f map-poly-zero of-int-hom.map-poly-hom-add
to-int-mod-ring-hom.hom-zero to-int-mod-ring-hom.injectivity)
have g'' : $of-int-poly\ (Mp\ g) = G$
by (*metis (no-types, lifting) Dp-Mp-eq Mp-f-representative*
Mp-smult-m-0 add-cancel-left-right g map-poly-zero of-int-hom.map-poly-hom-add
to-int-mod-ring-hom.hom-zero to-int-mod-ring-hom.injectivity)
have $F = of-int-poly\ (Mp\ g * q + r)$ **using** $Mp-f\ f''$ **by** *auto*
also have $\dots = G * of-int-poly\ q + of-int-poly\ r$
by (*simp add: g'' of-int-poly-hom.hom-add of-int-poly-hom.hom-mult*)
finally show $F = G * of-int-poly\ q + of-int-poly\ r$.
show $of-int-poly\ r = 0 \vee degree\ (of-int-poly\ r :: 'a\ mod-ring\ poly) < degree\ G$
proof (*cases r = 0*)
case *True*
hence $of-int-poly\ r = 0$ **by** *auto*
then show *?thesis* **by** *auto*
next
case *False*
have $degree\ (of-int-poly\ r :: 'a\ mod-ring\ poly) \leq degree\ (r)$
by (*simp add: degree-map-poly-le*)
also have $\dots < degree\ (Mp\ g)$ **using** $r\ False$ **by** *auto*
also have $\dots = degree\ G$ **by** (*simp add: g^*)
finally show *?thesis* **by** *auto*
qed
show $G \neq 0$ **using** $g0$ **unfolding** $g''[symmetric]$ **by** *simp*
qed

corollary *uniqueness-algorithm-division-to-int-poly*:

assumes $g0$: $Mp\ g \neq 0$
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 : $R = 0 \vee degree\ R < degree\ G$
and $Mp-f$: $Mp\ f = Mp\ g * q + r$
and r : $r = 0 \vee degree\ r < degree\ (Mp\ g)$
shows $Mp\ q = to-int-poly\ Q \wedge Mp\ r = to-int-poly\ R$
using *uniqueness-algorithm-division-of-int-poly[OF assms]*
by (*auto simp add: Mp-f-representative*)

lemma *uniqueness-algorithm-division-Mp-Rel*:

assumes *monic-Mpg*: *monic* $(Mp\ g)$
and f : $(F :: 'a\ mod-ring\ poly) = of-int-poly\ f$
and g : $(G :: 'a\ mod-ring\ poly) = of-int-poly\ g$
and qr : *pseudo-divmod* $(Mp\ f)\ (Mp\ g) = (q, r)$

and $QR: \text{pseudo-divmod } F \ G = (Q,R)$
shows $MP\text{-Rel } q \ Q \wedge MP\text{-Rel } r \ R$
proof (*unfold* $MP\text{-Rel-def}$, *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)[$OF \text{ monic-Mpg}$], *simp add: pdivmod-monic-pseudo-divmod*
 $qr \ \text{monic-Mpg}$)
have $\text{monic-G}: \text{monic } G$ **using** monic-Mpg
using $Mp\text{-f-representative } g$ **by** *auto*
show $F = G * Q + R$
by (*rule pdivmod-monic*(1)[$OF \text{ monic-G}$], *simp add: pdivmod-monic-pseudo-divmod*
 $QR \ \text{monic-G}$)
show $Mp \ g \neq 0$ **using** monic-Mpg **by** *auto*
show $R = 0 \vee \text{degree } R < \text{degree } G$
by (*rule pdivmod-monic*(2)[$OF \ \text{monic-G}$],
auto simp add: pdivmod-monic-pseudo-divmod monic-G intro: QR)
show $r = 0 \vee \text{degree } r < \text{degree } (Mp \ g)$
by (*rule pdivmod-monic*(2)[$OF \ \text{monic-Mpg}$],
auto simp add: pdivmod-monic-pseudo-divmod monic-Mpg intro: qr)
qed

definition $MP\text{-Rel-Pair } A \ B \equiv (\text{let } (a,b) = A; (c,d) = B \text{ in } MP\text{-Rel } a \ c \wedge MP\text{-Rel } b \ d)$

lemma $\text{pdivmod2-rel[transfer-rule]}$:

$(MP\text{-Rel} \implies MP\text{-Rel} \implies MP\text{-Rel-Pair}) \ (\text{pdivmod2}) \ (\text{pdivmod})$

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

interpret $pm: \text{prime-field } m$

using m **unfolding** $\text{prime-field-def mod-ring-locale-def}$ **by** *auto*

have $p: \text{prime-field } TYPE('a) \ m$

using m **unfolding** $\text{prime-field-def mod-ring-locale-def}$ **by** *auto*

fix $f \ F \ g \ G \ a \ b$

assume $1[\text{transfer-rule}]: MP\text{-Rel } f \ F$

and $2[\text{transfer-rule}]: MP\text{-Rel } g \ G$

and $3: \text{pdivmod2 } f \ g = (a, b)$

have $MP\text{-Rel } a \ (F \ \text{div } G) \wedge MP\text{-Rel } b \ (F \ \text{mod } G)$

proof (*cases* $Mp \ g \neq 0$)

case $True$ **note** $Mp\text{-}g = True$

have $G: G \neq 0$ **using** $Mp\text{-}g \ 2$ **unfolding** $MP\text{-Rel-def}$ **by** *auto*

have $gG[\text{transfer-rule}]: pm.\text{mod-ring-rel } (\text{lead-coeff } (Mp \ g)) \ (\text{lead-coeff } G)$

using 2

unfolding $pm.\text{mod-ring-rel-def } MP\text{-Rel-def}$

by *auto*

have $[\text{transfer-rule}]: (pm.\text{mod-ring-rel} \implies pm.\text{mod-ring-rel}) \ (\text{inverse-p } m)$

inverse

by (*rule prime-field.mod-ring-inverse*[$OF \ p$])

hence $\text{rel-inverse-p}[\text{transfer-rule}]:$

$pm.\text{mod-ring-rel } (\text{inverse-p } m \ ((\text{lead-coeff } (Mp \ g)))) \ (\text{inverse } (\text{lead-coeff } G))$

using gG **unfolding** rel-fun-def **by** *auto*

```

let ?h= (Polynomial.smult (inverse-p m (lead-coeff (Mp g))) g)
define h where h: h = Polynomial.smult (inverse-p m (lead-coeff (Mp g)))
(Mp g)
define H where H: H = Polynomial.smult (inverse (lead-coeff G)) G
have hH': MP-Rel ?h H unfolding MP-Rel-def unfolding H
by (metis (mono-tags, hide-lams) 2 MP-Rel-def M-to-int-mod-ring Mp-f-representative

rel-inverse-p functional-relation left-total-MP-Rel of-int-hom.map-poly-hom-smult

pm.mod-ring-rel-def right-unique-MP-Rel to-int-mod-ring-hom.injectivity
to-int-mod-ring-of-int-M)
have Mp (Polynomial.smult (inverse-p m (lead-coeff (Mp g))) g)
= Mp (Polynomial.smult (inverse-p m (lead-coeff (Mp g))) (Mp g)) by simp
hence hH: MP-Rel h H using hH' h unfolding MP-Rel-def by auto
obtain q x where pseudo-fh: pseudo-divmod (Mp f) (Mp h) = (q, x) by (meson
surj-pair)
hence lc-G: (lead-coeff G) ≠ 0 using G by auto
have a: a = Polynomial.smult (inverse-p m ((lead-coeff (Mp g)))) q
using 3 pseudo-fh Mp-g
unfolding pdivmod2-def Let-def h by auto
have b: b = x using 3 pseudo-fh Mp-g
unfolding pdivmod2-def Let-def h by auto
have Mp-Rel-FH: MP-Rel q (F div H) ∧ MP-Rel x (F mod H)
proof (rule uniqueness-algorithm-division-Mp-Rel)
show monic (Mp h)
proof –
have aux: (inverse-p m (lead-coeff (Mp g))) = to-int-mod-ring (inverse
(lead-coeff G))
using rel-inverse-p unfolding pm.mod-ring-rel-def by auto
hence M (inverse-p m (M (poly.coeff g (degree (Mp g)))))
= to-int-mod-ring (inverse (lead-coeff G))
by (simp add: M-to-int-mod-ring Mp-coeff)
thus ?thesis unfolding h unfolding Mp-coeff by auto
(metis (no-types, lifting) 2 H MP-Rel-def Mp-coeff aux degree-smult-eq gG
hH'
inverse-zero-imp-zero lc-G left-inverse pm.mod-ring-rel-def to-int-mod-ring-hom.degree-map-poly-hom
to-int-mod-ring-hom.hom-one to-int-mod-ring-times)
qed
hence monic-H: monic H using hH H lc-G by auto
show f: F = of-int-poly f
using 1 unfolding MP-Rel-def
by (simp add: Mp-f-representative poly-eq-iff)
have pdivmod F H = pdivmod-monic F H
by (rule pdivmod-eq-pdivmod-monic[OF monic-H])
also have ... = pseudo-divmod F H
by (rule pdivmod-monic-pseudo-divmod[OF monic-H])
finally show pseudo-divmod F H = (F div H, F mod H) by simp
show H = of-int-poly h
by (meson MP-Rel-def Mp-f-representative hH right-unique-MP-Rel right-unique-def)

```

show $\text{pseudo-divmod } (Mp\ f)\ (Mp\ h) = (q, x)$ **by** (rule pseudo-fh)
qed
hence $Mp\text{-Rel-}F\text{-div-}H: MP\text{-Rel } q\ (F\ \text{div } H)$ **and** $Mp\text{-Rel-}F\text{-mod-}H: MP\text{-Rel } x$
 $(F\ \text{mod } H)$ **by** auto
have $F\ \text{div } H = \text{Polynomial.smult } (\text{lead-coeff } G)\ (F\ \text{div } G)$
unfolding H **using** $\text{div-smult-right}[OF\ \text{lc-}G]$ $\text{inverse-inverse-eq}$
by $(\text{metis div-smult-right inverse-zero})$
hence $F\text{-div-}G: (F\ \text{div } G) = \text{Polynomial.smult } (\text{inverse } (\text{lead-coeff } G))\ (F\ \text{div } H)$
using $\text{lc-}G$ **by** auto
have $MP\text{-Rel } a\ (F\ \text{div } G)$
proof –
have $\text{of-int-poly } (\text{Polynomial.smult } (\text{inverse-p } m\ ((\text{lead-coeff } (Mp\ g))))\ q)$
 $= \text{smult } (\text{inverse } (\text{lead-coeff } G))\ (F\ \text{div } H)$
by $(\text{metis } (\text{mono-tags})\ MP\text{-Rel-def } M\text{-to-int-mod-ring } Mp\text{-Rel-}F\text{-div-}H$
 $Mp\text{-f-representative}$
 $\text{of-int-hom.map-poly-hom-smult } pm.\text{mod-ring-rel-def rel-inverse-p right-unique-}MP\text{-Rel}$
 $\text{right-unique-def to-int-mod-ring-hom.injectivity to-int-mod-ring-of-int-}M)$
thus $?thesis$
using $Mp\text{-Rel-}F\text{-div-}H$
unfolding $MP\text{-Rel-def } a\ F\text{-div-}G\ Mp\text{-f-representative}$ **by** auto
qed
moreover **have** $MP\text{-Rel } b\ (F\ \text{mod } G)$
using $Mp\text{-Rel-}F\text{-mod-}H\ b\ H\ \text{inverse-zero-imp-zero } \text{lc-}G$
by $(\text{metis mod-smult-right})$
ultimately show $?thesis$ **by** auto
next
assume $Mp\text{-g-0}: \neg Mp\ g \neq 0$
hence $\text{pdivmod2 } f\ g = (0, f)$ **unfolding** pdivmod2-def **by** auto
hence $a: a = 0$ **and** $b: b = f$ **using** 3 **by** auto
have $G0: G = 0$ **using** $Mp\text{-g-0 } 2$ **unfolding** $MP\text{-Rel-def}$ **by** auto
have $MP\text{-Rel } a\ (F\ \text{div } G)$ **unfolding** $MP\text{-Rel-def } G0\ a$ **by** auto
moreover **have** $MP\text{-Rel } b\ (F\ \text{mod } G)$ **using** 1 **unfolding** $MP\text{-Rel-def } G0\ a\ b$
by auto
ultimately show $?thesis$ **by** simp
qed
thus $MP\text{-Rel } a\ (F\ \text{div } G)$ **and** $MP\text{-Rel } b\ (F\ \text{mod } G)$ **by** auto
qed

3.2 Executable division operation modulo m for polynomials

lemma $\text{dvdm-iff-}Mp\text{-pdivmod2}$:

shows $g\ \text{dvdm } f = (Mp\ (\text{snd } (\text{pdivmod2 } f\ g)) = 0)$

proof –

let $?F = (\text{of-int-poly } f)::'a\ \text{mod-ring } \text{poly}$

let $?G = (\text{of-int-poly } g)::'a\ \text{mod-ring } \text{poly}$

have $a[\text{transfer-rule}]: MP\text{-Rel } f\ ?F$

by $(\text{simp add: } MP\text{-Rel-def } Mp\text{-f-representative})$

```

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 = (if prime m then Mp (snd (pdivmod2 f g)) = 0
    else Code.abort (STR "dvdM error: m is not a prime number") (λ -. 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.coeff up-ring.monom

```

```

    Unique-Factorization.factors Divisibility.factors

```

```

    Unique-Factorization.factor Divisibility.factor

```

```

    Divisibility.prime

```

definition *factorization-lattice* **where** *factorization-lattice* $u\ k\ m \equiv$
 $\text{map } (\lambda i. \text{vec-of-poly-n } (u * \text{monom } 1\ i) (\text{degree } u + k)) [k >..0]$ @
 $\text{map } (\lambda i. \text{vec-of-poly-n } (\text{monom } m\ i) (\text{degree } u + k)) [\text{degree } u >..0]$

fun *min-degree-poly* :: $\text{int poly} \Rightarrow \text{int poly} \Rightarrow \text{int poly}$
where *min-degree-poly* $a\ b = (\text{if degree } a \leq \text{degree } b \text{ then } a \text{ else } b)$

fun *choose-u* :: $\text{int poly list} \Rightarrow \text{int poly}$
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* $u\ k\ m =$ (
 $\text{let } n = \text{degree } u \text{ in}$
 map
 $(\lambda i. \text{vec-of-poly-n } (\text{monom-mult } i\ u) (n+k)) [k >..0]$
@ $\text{map } (\lambda i. \text{vec-of-poly-n } (\text{monom } m\ i) (n+k)) [n >..0]$
) **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 $pl\ n\ u = \text{poly-of-vec } (\text{short-vector-hybrid } 2 (\text{factorization-lattice}$
 $(\text{poly-mod.inv-Mp } pl (\text{poly-mod.Mp } pl\ u)) (n - \text{degree } u) pl))$

locale *LLL-implementation* =
fixes $p\ pl :: \text{int}$
begin

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

termination

proof (*relation measure* $(\lambda (f,us). \text{degree } f)$, *goal-cases*)
case $(\exists f\ us\ d\ d2\ f2\text{-opt}\ f2\ f1\ \text{pair}\ us1\ us2)$

```

from find-map-filter-Some[OF 3(4)[unfolded 3(3) Let-def]] 3(1,5)
show ?case by auto
next
case (2 f us d d2 f2-opt f2 f1 pair us1 us2)
from find-map-filter-Some[OF 2(4)[unfolded 2(3) Let-def]] 2(1,5)
have f: f = f1 * f2 and f0: f ≠ 0
  and deg: degree f2 > 0 degree f2 < degree f by auto
have degree f = degree f1 + degree f2 using f0 unfolding f
  by (subst degree-mult-eq, auto)
with deg show ?case by auto
qed auto

```

function *LLL-reconstruction* **where**

```

LLL-reconstruction f us = (let
  d = degree f;
  u = choose-u us;
  g = LLL-short-polynomial pl d u;
  f2 = gcd f g;
  deg = degree f2
  in if deg = 0 ∨ deg ≥ d then [f]
  else let f1 = f div f2;
  (us1, us2) = List.partition (λ gi. poly-mod.dvdm p gi f1) us
  in LLL-reconstruction f1 us1 @ LLL-reconstruction f2 us2)
by pat-completeness auto

```

termination

```

proof (relation measure (λ (f,us). degree f), goal-cases)
case (2 f us d u g f2 deg f1 pair us1 us2)
hence f: f = f1 * f2 and f0: f ≠ 0 by auto
have deg: degree f = degree f1 + degree f2 using f0 unfolding f
  by (subst degree-mult-eq, auto)
from 2 have degree f2 > 0 degree f2 < degree f by auto
thus ?case using deg by auto
qed auto
end

```

```

declare LLL-implementation.LLL-reconstruction.simps[code]
declare LLL-implementation.LLL-many-reconstruction.simps[code]

```

definition *LLL-factorization* :: int poly ⇒ int poly list **where**

```

LLL-factorization f = (let
  — find suitable prime
  p = suitable-prime-bz f;
  — compute finite field factorization
  (-, fs) = finite-field-factorization-int p f;
  — determine exponent l and B
  n = degree f;
  no = ||f||2;
  B = sqrt-int-ceiling (2^(5 * (n - 1) * (n - 1)) * no^(2 * (n - 1)));

```

```

  l = find-exponent p B;
  — perform hensel lifting to lift factorization to mod  $p^l$ 
  us = hensel-lifting p l f fs;
  — reconstruct integer factors via LLL algorithm
  pl = pl
  in LLL-implementation.LLL-reconstruction p pl f us)

```

definition *LLL-many-factorization* :: int poly ⇒ int poly list **where**

```

  LLL-many-factorization f = (let
    — find suitable prime
    p = suitable-prime-bz f;
    — compute finite field factorization
    (·, fs) = finite-field-factorization-int p f;
    — determine exponent l and B
    n = degree f;
    no = ||f||2;
    B = sqrt-int-ceiling (2^(5 * (n div 2) * (n div 2)) * no^(2 * (n div 2)));
    l = find-exponent p B;
    — perform hensel lifting to lift factorization to mod  $p^l$ 
    us = hensel-lifting p l f fs;
    — reconstruct integer factors via LLL algorithm
    pl = pl
    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 *LLL-Factorization*

imports

LLL-Factorization-Impl

Berlekamp-Zassenhaus.Factorize-Int-Poly

begin

5.1 Basic facts about the auxiliary functions

hide-const (**open**) *module.smult*

lemma *nth-factorization-lattice*:

fixes *u* **and** *d*

defines $n \equiv \text{degree } u$

assumes $i < n + d$

shows *factorization-lattice* $u \ d \ m \ ! \ i =$

vec-of-poly-n (*if* $i < d$ *then* $u * \text{monom } 1 \ (d - \text{Suc } i)$ *else* $\text{monom } m \ (n+d - \text{Suc } i)$) $(n+d)$

using *assms*
by (*unfold factorization-lattice-def*, *auto simp: nth-append smult-monom Let-def not-less*)

lemma *length-factorization-lattice*[*simp*]:
shows $\text{length } (\text{factorization-lattice } u \ d \ m) = \text{degree } u + d$
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*: $\text{choose-u } (x \# \text{xs}) =$
(if xs = [] then x else min-degree-poly x (choose-u xs))
by (*cases xs, auto*)

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

declare *choose-u.simps*[*simp del*]

5.2 Facts about Sylvester matrices and norms

lemma (*in LLL*) *lattice-is-span* [*simp*]: $\text{lattice-of } \text{xs} = \text{span-list } \text{xs}$
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 $i < \text{degree } g$
shows $\|(\text{row } (\text{sylvester-mat } f \ g) \ i)\|^2 = \|f\|^2$

proof (*cases f = 0*)

case *True*

thus *?thesis*

by (*auto simp add: sylvester-mat-def row-def sq-norm-vec-def o-def interv-sum-list-conv-sum-set-nat i intro!: sum-list-zero*)

next

case *False* **note** $f = \text{False}$

let $?f = \lambda j. \text{if } i \leq j \wedge j - i \leq \text{degree } f \text{ then coeff } f \ (\text{degree } f + i - j) \text{ else } 0$

let $?h = \lambda j. j + i$

```

let ?row = vec (degree f + degree g) ?f
let ?g = λj. degree f - j
have image-g: ?g ‘ {0..<Suc (degree f)} = {0..<Suc (degree f)}
  by (auto simp add: image-def)
    (metis (no-types, hide-lams) Nat.add-diff-assoc add commute add-diff-cancel-left'

      atLeastLessThan-iff diff-Suc-Suc diff-Suc-less less-Suc-eq-le zero-le)
have bij-h: bij-betw ?h {0..<Suc (degree f)} {i..< Suc (degree f + i)}
  unfolding bij-betw-def image-def
  by (auto, metis atLeastLessThan-iff le-add-diff-inverse2
    less-diff-conv linorder-not-less not-less-eq zero-order (3))
have ||row (sylvestermat f g) i||2 = ||?row||2
  by (rule arg-cong[of - - sq-norm-vec], insert i,
    auto simp add: row-def sylvestermat-def sylvestermat-sub-def)
also have ... = sum-list (map (sq-norm ∘ ?f) [0..<degree f + degree g])
  unfolding sq-norm-vec-def by auto
also have ... = sum (sq-norm ∘ ?f) {0..<degree f + degree g}
  unfolding interv-sum-list-conv-sum-set-nat by auto
also have ... = sum (sq-norm ∘ ?f) {i..< Suc (degree f + i)}
  by (rule sum.mono-neutral-right, insert i, auto)
also have ... = sum ((sq-norm ∘ ?f) ∘ ?h) {0..<Suc (degree f)}
  by (unfold o-def, rule sum.reindex-bij-betw[symmetric, OF bij-h])
also have ... = sum (λj. sq-norm (coeff f (degree f - j))) {0..<Suc (degree f)}

  by (rule sum.cong, auto)
also have ... = sum ((λj. sq-norm (coeff f j)) ∘ ?g) {0..<Suc (degree f)}
  unfolding o-def ..
also have ... = sum (λj. sq-norm (coeff f j)) (?g ‘ {0..<Suc (degree f)})
  by (rule sum.reindex[symmetric], auto simp add: inj-on-def)
also have ... = sum (sq-norm ∘ coeff f) {0..<Suc (degree f)} unfolding image-g
by simp
also have ... = sum-list (map sq-norm (coeffs f))
  unfolding coeffs-def using f
  by (simp add: interv-sum-list-conv-sum-set-nat)
finally show ?thesis unfolding sq-norm-poly-def by auto
qed

```

lemma sq-norm-row-sylvestermat2:

```

fixes f g :: 'a :: conjugatable-ring poly
assumes i1: degree g ≤ i and i2: i < degree f + degree g
shows ||row (sylvestermat f g) i||2 = ||g||2
proof -
let ?f = λj. if i - degree g ≤ j ∧ j ≤ i then coeff g (i - j) else 0
let ?row = vec (degree f + degree g) ?f
let ?h = λj. j + i - degree g
let ?g = λj. degree g - j
have image-g: ?g ‘ {0..<Suc (degree g)} = {0..<Suc (degree g)}
  by (auto simp add: image-def)

```

(*metis atLeastLessThan-iff diff-diff-cancel diff-le-self less-Suc-eq-le zero-le*)
have $x: x - (i - \text{degree } g) \leq \text{degree } g$ **if** $x: x < \text{Suc } i$ **for** x **using** x **by** *auto*
have $\text{bij-h}: \text{bij-betw } ?h \{0..<\text{Suc } (\text{degree } g)\} \{i - \text{degree } g..<\text{Suc } i\}$
unfolding $\text{bij-betw-def inj-on-def}$ **using** $i1 i2$ **unfolding** image-def
by (*auto, metis (no-types) Nat.add-diff-assoc atLeastLessThan-iff x less-Suc-eq-le*)

less-eq-nat.simps(1) ordered-cancel-comm-monoid-diff-class.diff-add
have $\|\text{row } (\text{sylvester-mat } f g) i\|^2 = \|\text{?row}\|^2$
by (*rule arg-cong[of - - sq-norm-vec], insert i1 i2,*
auto simp add: row-def sylvester-mat-def sylvester-mat-sub-def)
also have $\dots = \text{sum-list } (\text{map } (\text{sq-norm } \circ ?f) [0..<\text{degree } f + \text{degree } g])$
unfolding sq-norm-vec-def **by** *auto*
also have $\dots = \text{sum } (\text{sq-norm } \circ ?f) \{0..<\text{degree } f + \text{degree } g\}$
unfolding $\text{interv-sum-list-conv-sum-set-nat}$ **by** *auto*
also have $\dots = \text{sum } (\text{sq-norm } \circ ?f) \{i - \text{degree } g..<\text{Suc } i\}$
by (*rule sum.mono-neutral-right, insert i2, auto*)
also have $\dots = \text{sum } ((\text{sq-norm } \circ ?f) \circ ?h) \{0..<\text{Suc } (\text{degree } g)\}$
by (*unfold o-def, rule sum.reindex-bij-betw[symmetric, OF bij-h]*)
also have $\dots = \text{sum } (\lambda j. \text{sq-norm } (\text{coeff } g (\text{degree } g - j))) \{0..<\text{Suc } (\text{degree } g)\}$

by (*rule sum.cong, insert i1, auto*)
also have $\dots = \text{sum } ((\lambda j. \text{sq-norm } (\text{coeff } g j)) \circ ?g) \{0..<\text{Suc } (\text{degree } g)\}$
unfolding $o\text{-def } ..$
also have $\dots = \text{sum } (\lambda j. \text{sq-norm } (\text{coeff } g j)) (?g \text{ ' } \{0..<\text{Suc } (\text{degree } g)\})$
by (*rule sum.reindex[symmetric], auto simp add: inj-on-def*)
also have $\dots = \text{sum } (\text{sq-norm } \circ \text{coeff } g) \{0..<\text{Suc } (\text{degree } g)\}$ **unfolding** image-g
by *simp*
also have $\dots = \text{sum-list } (\text{map } \text{sq-norm } (\text{coeffs } g))$
unfolding coeffs-def
by (*simp add: interv-sum-list-conv-sum-set-nat*)
finally show *?thesis* **unfolding** sq-norm-poly-def **by** *auto*
qed

lemma *Hadamard's-inequality-int:*
fixes $A::\text{int mat}$
assumes $A: A \in \text{carrier-mat } n n$
shows $|\det A| \leq \text{sqr}t (\text{of-int } (\text{prod-list } (\text{map } \text{sq-norm } (\text{rows } A))))$
proof –
let $?A = \text{map-mat } \text{real-of-int } A$
have $|\det A| = |\det ?A|$ **unfolding** $\text{of-int-hom.hom-det}$ **by** *simp*
also have $\dots \leq \text{sqr}t (\text{prod-list } (\text{map } \text{sq-norm } (\text{rows } ?A)))$
by (*rule Hadamard's-inequality[of ?A n], insert A, auto*)
also have $\dots = \text{sqr}t (\text{of-int } (\text{prod-list } (\text{map } \text{sq-norm } (\text{rows } A))))$ **unfolding**
 $\text{of-int-hom.hom-prod-list map-map}$
by (*rule arg-cong[of - - $\lambda x. \text{sqr}t (\text{prod-list } x)$], rule nth-equalityI, force,*
auto simp: sq-norm-of-int[symmetric] row-def intro!: arg-cong[of - - sq-norm-vec])
finally show *?thesis* .
qed

```

lemma resultant-le-prod-sq-norm:
  fixes  $f g::int$  poly
  defines  $n \equiv degree\ f$  and  $k \equiv degree\ g$ 
  shows  $|resultant\ f\ g| \leq sqrt\ (of-int\ (\|f\|^2 \wedge k * \|g\|^2 \wedge n))$ 
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 k-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  $\dots = (sq-norm \circ row\ (sylvester-mat\ f\ g))\ (0 + i)$ 
    using sq-norm-row-sylvester-mat1 ik k-def by force
  also have  $\dots = ?M\ !\ i$  by (rule nth-map-upt[symmetric], simp add: i-deg-g)
  finally show  $?M\ !\ i = replicate\ k\ \|f\|^2\ !\ i ..$ 
qed
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  $i < length\ ?M$  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  $\dots = (sq-norm \circ row\ (sylvester-mat\ f\ g))\ (degree\ g + i)$ 
    using i-n n-def
    by (simp add: sq-norm-row-sylvester-mat2)
  also have  $\dots = ?M\ !\ i$ 
    by (simp add: i-deg-f)
  finally show  $?M\ !\ i = replicate\ n\ \|g\|^2\ !\ i ..$ 
qed
qed
  have  $p1: prod-list\ (map\ ?f\ [0..<degree\ g]) = \|f\|^2 \wedge k$ 
    unfolding map-rw1 by (rule prod-list-replicate)
  have  $p2: prod-list\ (map\ ?f\ [degree\ g..<degree\ f + degree\ g]) = \|g\|^2 \wedge n$ 
    unfolding map-rw2 by (rule prod-list-replicate)
  have  $list-rw: [0..<degree\ f + degree\ g] = [0..<degree\ g] @ [degree\ g..<degree\ f + degree\ g]$ 
    by (metis add.commute upt-add-eq-append zero-le)
  have  $|resultant\ f\ g| = |det\ ?S|$  unfolding resultant-def ..
  also have  $\dots \leq sqrt\ (of-int\ (prod-list\ (map\ sq-norm\ (rows\ ?S))))$ 

```

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 [0..<degree g])
 * 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-via-short:

fixes 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 < m²
 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: [: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: [: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 [resultant f g = 0] (mod m) using res-0-mod unfolding cong-def by
 auto
 have |resultant f g| ≤ sqrt ((sq-norm-poly f) ^k * (sq-norm-poly g) ^n)

```

    unfolding k-def n-def
    by (rule resultant-le-prod-sq-norm)
  also have ... < m
  proof (rule real-less-lsqr)
    show 0 ≤ real-of-int (||f||2 ^ k * ||g||2 ^ n)
      by (simp add: sq-norm-poly-ge-0)
    show 0 ≤ real-of-int m using m by simp
    show real-of-int (||f||2 ^ k * ||g||2 ^ n) < (real-of-int m)2
      by (metis of-int-less-iff of-int-power short)
  qed
  finally show |resultant f g| < m using of-int-less-iff by blast
  qed
  have ¬ 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 :: semidom poly
  assumes dj: d ≤ j and d: degree p = d
  shows mat-of-rows j (factorization-lattice p (j-d) m) = sylvester-mat-sub d
    (j-d) p [:m:]
  proof (cases p=0)
    case True
      have 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 = degree-mult-eq[OF p0, of monom -, unfolded monom-eq-0-iff, OF
          one-neq-zero]
        from 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 [hom-distrib]:
  map-poly hom (p * monom a n) = map-poly hom p * monom (hom a) n
  by (auto intro!: poly-eqI simp: coeff-mult-monom hom-mult)

```

lemma *hom-vec-of-poly-n* [*hom-distrib*]:
 $map\text{-}vec\ hom\ (vec\text{-}of\text{-}poly\text{-}n\ p\ n) = vec\text{-}of\text{-}poly\text{-}n\ (map\text{-}poly\ hom\ p)\ n$
by (*auto simp: vec-index-of-poly-n*)

lemma *hom-factorization-lattice* [*hom-distrib*]:
shows $map\ (map\text{-}vec\ hom)\ (factorization\text{-}lattice\ u\ k\ m) = factorization\text{-}lattice$
 $(map\text{-}poly\ hom\ u)\ k\ (hom\ m)$
by (*auto intro!: arg-cong[of - - $\lambda p. vec\text{-}of\text{-}poly\text{-}n\ p$ -] 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 $d: d \leq n$ **and** $du: d = degree\ u$
shows $upper\text{-}triangular\ (mat\text{-}of\text{-}rows\ n\ (factorization\text{-}lattice\ u\ (n-d)\ k))$
 $(is\ upper\text{-}triangular\ ?M)$
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 $d\ du$ **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
qed

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

```

    and u: u ≠ 0
    and m0: k ≠ 0
    and i: i < n
  shows (factorization-lattice u (n-d) k) ! i $ i ≠ 0
proof -
  have 1: monom (1::'a) (n - Suc (degree u + i)) ≠ 0 using m0 by auto
  have 2: i < degree u + (n - d) using i d by auto
  let ?p = u * monom 1 (n - Suc (degree u + i))
  have 3: i < n - degree u ⇒ degree (?p) = n - Suc i
    using assms by (auto simp: degree-mult-eq[OF - 1] degree-monom-eq)
  show ?thesis
    apply (unfold nth-factorization-lattice[OF 2] vec-index-of-poly-n[OF 2])
    using assms leading-coeff-0-iff[of ?p]
    apply (cases i < n - degree u, auto simp: d 3 degree-monom-eq)
    done
qed

```

corollary *factorization-lattice-diag-nonzero-RAT: fixes d*

```

  assumes d=degree u
    and d ≤ n
    and u ≠ 0
    and k ≠ 0
    and i < n
  shows RAT (factorization-lattice u (n-d) k) ! i $ i ≠ 0
  using factorization-lattice-diag-nonzero[OF assms] assms
  by (auto simp: nth-factorization-lattice)

```

sublocale *gs: vec-space TYPE(rat) n.*

lemma *lin-indpt-list-factorization-lattice: fixes d*

```

  assumes d: d = degree u and dn: d ≤ n and u: u ≠ 0 and k: k ≠ 0
  shows gs.lin-indpt-list (RAT (factorization-lattice u (n-d) k)) (is gs.lin-indpt-list
(RAT ?vs))

```

proof -

```

  have 1: rows (mat-of-rows n (map (map-vec rat-of-int) ?vs)) = map (map-vec
rat-of-int) ?vs
    using dn d
  by (subst rows-mat-of-rows, auto dest!: subsetD[OF set-factorization-lattice-in-carrier])
  note 2 = factorization-lattice-diag-nonzero-RAT[OF d dn u k]
  show ?thesis
    apply (intro gs.upper-triangular-imp-lin-indpt-list[of mat-of-rows n (RAT ?vs),
unfolded 1])
    using assms 2 by (auto simp: diag-mat-def mat-of-rows-index hom-distrib
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* \equiv *map-vec of-int*

context *LLL*

begin

lemma *lincomb-to-dvd-modulo*:

fixes *u d*

defines *d* \equiv *degree u*

assumes *d*: $d \leq n$

and *lincomb*: *lincomb-list c (factorization-lattice u (n-d) k) = g (is ?l = ?r)*

shows *poly-mod.dvdm k u (poly-of-vec g)*

proof –

let *?S* = *syvester-mat-sub d (n - d) u [:k:]*

define *q* **where** *q* \equiv *poly-of-vec (vec-first (vec n c) (n - d))*

define *r* **where** *r* \equiv *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 syvester-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* \equiv *degree u*

assumes *d*: $d < n$

and *dvd*: *poly-mod.dvdm k u (poly-of-vec g)*

and *k-not0*: $k \neq 0$

and *monic-u*: *monic u*

and *dim-g*: *dim-vec g = n*

```

    and deg-u: degree u > 0
  shows  $\exists c. \text{lincomb-list } c \text{ (factorization-lattice } u \text{ (} n-d \text{) } k) = g$ 
proof -
  interpret  $p: \text{poly-mod } k$  .
  have u-not0:  $u \neq 0$  using monic-u by auto
  hence n[simp]:  $0 < n$  using d by auto
  obtain  $q' r'$  where  $g: \text{poly-of-vec } g = q' * u + \text{smult } k r'$ 
    using p.dvdm-imp-div-mod[OF dvd] by auto
  obtain  $q'' r''$  where  $r': r' = q'' * u + r''$  and deg-r'': degree  $r'' < \text{degree } u$ 
    using monic-imp-div-mod-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 (no-types, lifting) combine-common-factor mult-smult-left smult-add-right)
  define q where  $q: q = (q' + \text{smult } k q'')$ 
  define r where  $r: r = r''$ 
  have degree-q:  $q = 0 \vee \text{degree } (q' + \text{smult } k q'') < n - d$ 
  proof (cases  $q = 0, \text{auto}, \text{rule degree-div-mod-smult}[OF - - - g1]$ )
    show degree (poly-of-vec g) < n by (rule degree-poly-of-vec-less, auto simp add:
dim-g)
    show degree  $r'' < d$  using deg-r'' unfolding d-def .
    assume  $q \neq 0$  thus  $q' + \text{smult } k q'' \neq 0$  unfolding q .
    show  $k \neq 0$  by fact
    show 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 vec-of-poly-n-poly-of-vec[symmetric], auto simp add: dim-g)
  also have ... =  $\text{vec-of-poly-n } ((q' + \text{smult } k q'') * u + \text{smult } k r'') \ n$ 
    using g1 by auto
  also have ... =  $\text{vec-of-poly-n } (q * u + \text{smult } k r'') \ n$  unfolding q by auto
  also have ... =  $\text{vec-of-poly-n } (q * u) \ n + \text{vec-of-poly-n } (\text{smult } k r'') \ n$ 
    by (rule vec-of-poly-n-add)
  finally show ?thesis unfolding r by simp
qed
let ?c =  $\lambda i. \text{if } i < n - d \text{ then coeff } q \ (n - d - 1 - i) \text{ else coeff } r \ (n - \text{Suc } i)$ 
let ?c1 =  $\lambda i. ?c \ i \cdot_v \text{factorization-lattice } u \text{ (} n-d \text{) } k ! i$ 
show ?thesis
proof (rule exI[of - ?c])
  let ?part1 =  $\text{map } (\lambda i. \text{vec-of-poly-n } (u * \text{monom } 1 \ i) \ n) \ [n-d > .. 0]$ 
  let ?part2 =  $\text{map } (\lambda i. \text{vec-of-poly-n } (\text{monom } k \ i) \ n) \ [d > .. 0]$ 
  have [simp]:  $\text{dim-vec } (M.\text{sumlist } (\text{map } ?c1 \ [0 .. < n - d])) = n$ 
    by (rule dim-sumlist, auto simp add: dim-factorization-lattice d-def)
  have [simp]:  $\text{dim-vec } (M.\text{sumlist } (\text{map } ?c1 \ [n-d .. < n])) = n$ 
    by (rule dim-sumlist, insert d, auto simp add: dim-factorization-lattice d-def)
  have [simp]:  $\text{factorization-lattice } u \text{ (} n-d \text{) } k ! x \in \text{carrier-vec } n \text{ if } x: x < n \text{ for}$ 
x
    using x dim-factorization-lattice-element nth-factorization-lattice[of x u n-d]

```

d
 by (auto simp: d-def)
 have $[0..<\text{length } (\text{factorization-lattice } u \ (n-d) \ k)] = [0..<n]$
 using d by (simp add: d-def less-imp-le-nat)
 also have $\dots = [0..<n - d] \ @ \ [n-d..<n]$
 by (rule upt-minus-eq-append, auto)
 finally have $\text{list-rw}: [0..<\text{length } (\text{factorization-lattice } u \ (n-d) \ k)] = [0..<n - d] \ @ \ [n-d..<n]$.
 have $qu1: \text{poly-of-vec } (M.\text{sumlist } (\text{map } ?c1 \ [0..<n - d])) = q * u$
 proof -
 have $\text{poly-of-vec } (M.\text{sumlist } (\text{map } ?c1 \ [0..<n - d])) = \text{poly-of-vec } (\bigoplus_{v \in \{0..<n-d\}} ?c1 \ v)$
 by (subst sumlist-map-as-finsum, auto)
 also have $\dots = \text{poly-of-vec } (\bigoplus_{v \in \text{set } [0..<n-d]} ?c1 \ v)$ by auto
 also have $\dots = \text{sum } (\lambda i. \text{poly-of-vec } (?c1 \ i)) \ (\text{set } [0..<n-d])$
 by (auto simp: poly-of-vec-finsum)
 also have $\dots = \text{sum } (\lambda i. \text{poly-of-vec } (?c1 \ i)) \ \{0..<n-d\}$ by auto
 also have $\dots = q * u$
 proof -
 have $\text{deg}: \text{degree } (u * \text{monom } 1 \ (n - \text{Suc } (d + i))) < n$ 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 $\text{deg-m}: \text{degree } ?m = n - \text{Suc } (d + i)$ by (rule degree-monom-eq, auto)
 have $\text{degree } (u * ?m) = d + (n - \text{Suc } (d + i))$
 using degree-mult-eq[OF u-not0 monom-not0] d-def deg-m by auto
 also have $\dots < n$ using i by auto
 finally show ?thesis .
 qed
 have $\text{lattice-rw}: \text{factorization-lattice } u \ (n-d) \ k \ ! \ i = \text{vec-of-poly-n } (u * \text{monom } 1 \ (n - \text{Suc } (d + i))) \ n$
 if $i: i < n - d$ for i apply (subst nth-factorization-lattice) using i by (auto simp: d-def)
 have $q\text{-rw}: q = (\sum_{i = 0..<n - d}. (\text{smult } (\text{coeff } q \ (n - \text{Suc } (d + i)))) \ (\text{monom } 1 \ (n - \text{Suc } (d + i))))$
 proof (auto simp add: poly-eq-iff coeff-sum)
 fix j
 let $?m = n - d - 1 - j$
 let $?f = \lambda x. \text{coeff } q \ (n - \text{Suc } (d + x)) * (\text{if } n - \text{Suc } (d + x) = j \ \text{then } 1 \ \text{else } 0)$
 have $\text{set-rw}: \{0..<n-d\} = \text{insert } ?m \ (\{0..<n-d\} - \{?m\})$ using d by auto
 have $\text{sum0}: (\sum_{x \in \{0..<n-d\} - \{?m\}} ?f \ x) = 0$ 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 = ( $\sum i = 0..<n - d. \text{coeff } q (n - \text{Suc } (d + i))$ 
* (if n - Suc (d + i) = j then 1 else 0)) ..
    qed
    have sum ( $\lambda i. \text{poly-of-vec } (?c1 i) \{0..<n-d\}$ 
= ( $\sum i = 0..<n - d. \text{poly-of-vec } (\text{coeff } q (n - \text{Suc } (d + i))) \cdot_v \text{factorization-lattice}$ 
u (n-d) k ! i))
      by (rule sum.cong, auto)
    also have ... = ( $\sum i = 0..<n - d. (\text{poly-of-vec } (\text{coeff } q (n - \text{Suc } (d + i)))$ 
 $\cdot_v (\text{vec-of-poly-n } (u * \text{monom } 1 (n - \text{Suc } (d + i))) n))$ )
      by (rule sum.cong, auto simp add: lattice-rw)
    also have ... = ( $\sum i = 0..<n - d. \text{smult } (\text{coeff } q (n - \text{Suc } (d + i))) (u *$ 
monom 1 (n - Suc (d + i)))
      by (rule sum.cong, auto simp add: poly-of-vec-scalar-mult[OF deg])
    also have ... = ( $\sum i = 0..<n - d. u * (\text{smult } (\text{coeff } q (n - \text{Suc } (d + i)))$ 
(monom 1 (n - Suc (d + i))))
      by auto
    also have ... = u * ( $\sum i = 0..<n - d. (\text{smult } (\text{coeff } q (n - \text{Suc } (d + i)))$ 
(monom 1 (n - Suc (d + i))))
      by (rule sum-distrib-left[symmetric])
    also have ... = u * q using q-rw by auto
    also have ... = q * u by auto
    finally show ?thesis .
  qed
  finally show ?thesis .
qed
have qu: 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 (poly-of-vec (M.sumlist (map ?c1
[0..<n - d]))) n
    using qu1 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.sumlist (map ?c1 [n-d..<n])) = poly-of-vec ( $\bigoplus_{vi \in \{n-d..<n\}}$ ).
?c1 i)
  by (subst sumlist-map-as-finsum, auto)
also have ... = poly-of-vec ( $\bigoplus_{vi \in \text{set } [n-d..<n]}$ . ?c1 i) by auto
also have ... = sum ( $\lambda i.$  poly-of-vec (?c1 i)) {n-d..<n}
  by (auto simp: poly-of-vec-finsum)
also have ... = smult k r
proof -
  have deg: degree (monom k (n - Suc i)) < n if i: n-d ≤ 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) k ! i = vec-of-poly-n (monom
k (n - Suc i)) n
  if i: n - d ≤ i and i2: i < n for i
  using i2 i d d-def
  by (subst nth-factorization-lattice, auto)
  have r-rw: r = ( $\sum_{i \in \{n-d..<n\}}$ . (monom (coeff r (n - Suc i)) (n -
Suc i)))
  proof (auto simp add: poly-eq-iff coeff-sum)
  fix j
  show coeff r j = ( $\sum_{i = n - d..<n}$ . if n - Suc i = j then coeff r (n -
Suc i) else 0)
  proof (cases j < d)
  case True
  have j-eq: n - Suc (n - 1 - j) = j using d True by auto
  let ?i = n-1-j
  let ?f =  $\lambda i.$  if n - Suc i = j then coeff r (n - Suc i) else 0
  have sum0: sum ?f ({n-d..<n} - {?i}) = 0 by (rule sum.neutral,
auto)
  have {n-d..<n} = insert ?i ({n-d..<n} - {?i}) using True by auto
  hence sum ?f {n - d..<n} = sum ?f (insert ?i ({n-d..<n} - {?i}))
by auto
  also have ... = ?f ?i + sum ?f ({n-d..<n} - {?i})
  by (rule sum.insert, auto)
  also have ... = coeff r j unfolding sum0 j-eq by simp
  finally show ?thesis ..
  next
  case False
  hence ( $\sum_{i = n - d..<n}$ . if n - Suc i = j then coeff r (n - Suc i) else
0) = 0
  by (intro sum.neutral ballI, insert False, simp, linarith)
  also have ... = coeff r j
  by (rule coeff-eq-0[symmetric], insert False deg-r'' r d-def, auto)
  finally show ?thesis ..
  qed
qed
have sum ( $\lambda i.$  poly-of-vec (?c1 i)) {n-d..<n}
= ( $\sum_{i \in \{n-d..<n\}}$ . poly-of-vec (coeff r (n - Suc i)) ·v factorization-lattice

```

```

u (n-d) k ! i))
  by (rule sum.cong, auto)
  also have ... = (∑ i ∈ {n-d..<n}. (poly-of-vec (coeff r (n - Suc i)
    ·v vec-of-poly-n (monom k (n - Suc i)) n)))
  by (rule sum.cong, auto simp add: lattice-rw)
  also have ... = (∑ i ∈ {n-d..<n}. smult (coeff r (n - Suc i)) (monom k
(n - Suc i)))
  by (rule sum.cong, auto simp add: poly-of-vec-scalar-mult[OF deg])
  also have ... = (∑ i ∈ {n-d..<n}. smult k (monom (coeff r (n - Suc i))
(n - Suc i)))
  by (rule sum.cong, auto simp add: smult-monom smult-sum2)
  also have ... = smult k (∑ i ∈ {n-d..<n}. (monom (coeff r (n - Suc i))
(n - Suc i)))
  by (simp add: smult-sum2)
  also have ... = smult k r using r-rw by auto
  finally show ?thesis .
qed
finally show ?thesis .
qed
have rm: (M.sumlist (map ?c1 [n-d..<n])) = vec-of-poly-n (smult k r) n
proof -
  have vec-of-poly-n (smult k r) n
    = vec-of-poly-n (poly-of-vec (M.sumlist (map ?c1 [n-d..<n]))) n
  using rm1 by auto
  also have vec-of-poly-n (poly-of-vec (M.sumlist (map ?c1 [n-d..<n]))) n
    = M.sumlist (map ?c1 [n-d..<n])
  by (rule vec-of-poly-n-poly-of-vec, auto)
  finally show ?thesis ..
qed
have lincomb-list ?c (factorization-lattice u (n-d) k) = M.sumlist (map ?c1
([0..<n - d] @ [n-d..<n]))
  unfolding lincomb-list-def list-rw by auto
  also have ... = M.sumlist (map ?c1 [0..<n - d] @ map ?c1 [n-d..<n]) by
auto
  also have ... = M.sumlist (map ?c1 [0..<n - d]) + M.sumlist (map ?c1
[n-d..<n])
  using d by (auto simp add: d-def nth-factorization-lattice intro!: M.sumlist-append)
  also have ... = vec-of-poly-n (q*u) n + vec-of-poly-n (smult k r) n
  unfolding qu rm by auto
  also have ... = g using g2 by simp
  finally show lincomb-list ?c (factorization-lattice u (n-d) k) = g .
qed
qed

```

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 $u \leq n \implies n \neq 0 \implies f \in \text{poly-of-vec } \textit{lattice-of (factorization-lattice}$

```

u (n - degree u) M)  $\implies$ 
  degree f < n  $\wedge$  poly-mod.dvdm M u f
  monic u  $\implies$  degree u < n  $\implies$ 
  degree f < n  $\implies$  poly-mod.dvdm M u f  $\implies$  f  $\in$  poly-of-vec 'lattice-of (factorization-lattice
u (n - degree u) M)
proof -
  from deg-u have deg-u: degree u > 0 by auto
  let ?L = factorization-lattice u (n - degree u) M
  {
    assume deg: degree f < n and dvd: poly-mod.dvdm M u f and mon: monic u
    and deg-u-lt: degree u < n
    define fv where fv = vec n ( $\lambda$  i. (coeff f (n - Suc i)))
    have f: f = poly-of-vec fv unfolding fv-def poly-of-vec-def Let-def using deg
    by (auto intro!: poly-eqI coeff-eq-0 simp: coeff-sum)
    have dim-fv: dim-vec fv = n unfolding fv-def by simp
    from dvd-modulo-to-lincomb[OF deg-u-lt - M mon - deg-u(1), of fv, folded f,
OF dvd dim-fv]
    obtain c where gv: fv = lincomb-list c ?L by auto
    have fv  $\in$  lattice-of ?L unfolding gv lattice-is-span by (auto simp: in-span-listI)
    thus f  $\in$  poly-of-vec 'lattice-of ?L unfolding f by auto
  }
  moreover
  {
    assume f  $\in$  poly-of-vec 'lattice-of ?L and deg-u: degree u  $\leq$  n and n: n  $\neq$  0
    then obtain fv where f: f = poly-of-vec fv and fv: fv  $\in$  lattice-of ?L by auto
    from in-span-listE[OF fv[unfolded lattice-is-span]]
    obtain c where fv: fv = lincomb-list c ?L by auto
    from lincomb-to-dvd-modulo[OF - fv[symmetric]] deg-u f
    have dvd: poly-mod.dvdm M u f by auto
    have set ?L  $\subseteq$  carrier-vec n unfolding factorization-lattice-def using deg-u
by auto
    hence fv  $\in$  carrier-vec n unfolding fv by (metis lincomb-list-carrier)
    hence degree f < n unfolding f using degree-poly-of-vec-less[of fv n] using
n by auto
    with dvd show degree f < n  $\wedge$  poly-mod.dvdm M u f by auto
  }
qed
end

```

5.7 Soundness of the LLL factorization algorithm

lemma *LLL-short-polynomial*: **assumes** deg-u-0: degree u \neq 0 **and** deg-le: degree u \leq n
and pl1: pl > 1
and monic: monic u
shows degree (LLL-short-polynomial pl n u) < n
and LLL-short-polynomial pl n u \neq 0
and poly-mod.dvdm pl u (LLL-short-polynomial pl n u)
and degree u < n \implies f \neq 0 \implies

$\text{poly-mod.dvdm } pl \ u \ f \implies \text{degree } f < n \implies \|LLL\text{-short-polynomial } pl \ n \ u\|^2 \leq 2^{(n-1)} * \|f\|^2$

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* = *Mp u*
let *?iu* = *inv-Mp ?u*
from *Mp-inv-Mp-id*[*of ?u*] **have** *?iu =m ?u* .
also have $\dots =m u$ **by** *simp*
finally have *iu-u*: *?iu =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-M-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-M-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*: *n* $\neq 0$ **by** *auto*
from *deg-u-0* **have** *u0*: *u* $\neq 0$ **by** *auto*
have *id*: *LLL-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 n 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 ***: $4/3 \leq (2 :: \text{rat}) \ L.\text{gs.lin-indpt-list } (L.\text{RAT } ?L)$ **by** (*auto simp: deg-iu*)
interpret *L*: *LLL-with-assms n n ?L 2*
by (*unfold-locales*, *insert **, *auto simp: deg-iu deg-le*)
note *short* = *L.short-vector-hybrid*[*OF refl n*, *unfolded id' L.L-def*]
from *short*(2) **have** *mem*: *LLL-short-polynomial pl n u* \in *poly-of-vec ' lattice-of*
?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-short-polynomial pl n u)* $< n$ **using** *fact* **by** *auto*
from *fact* **have** *?iu dvd m* (*LLL-short-polynomial pl n u*) **by** *auto*
then obtain *h* **where** *LLL-short-polynomial pl n u* = *m ?iu * h* **unfolding**
dvd m-def **by** *auto*
also have *?iu * h =m Mp ?iu * h* **unfolding** *mult-Mp* **by** *simp*

also have $Mp \text{ ?iu} * h = m \ u * h$ **unfolding** *iu-u* **unfolding** *mult-Mp* **by** *simp*
finally show $u \ \text{dvdm} \ (LLL\text{-short-polynomial} \ pl \ n \ u)$ **unfolding** *dvdm-def* **by**
auto
from short have $?sv \in \text{carrier-vec} \ n$ **by** *auto*
from short have $?sv \neq 0_v \ j$ **for** j **by** *auto*
thus $LLL\text{-short-polynomial} \ pl \ n \ u \neq 0$ **unfolding** *id* **by** *simp*
assume *degu*: $\text{degree} \ u < n$ **and** *dvd*: $u \ \text{dvdm} \ f$
and *degf*: $\text{degree} \ f < n$ **and** *f0*: $f \neq 0$
from *dvd* **obtain** h **where** $f = m \ u * h$ **unfolding** *dvdm-def* **by** *auto*
also have $u * h = m \ Mp \ u * h$ **unfolding** *mult-Mp* **by** *simp*
also have $Mp \ u * h = m \ Mp \ \text{?iu} * h$ **unfolding** *iu-u* **by** *simp*
also have $Mp \ \text{?iu} * h = m \ \text{?iu} * h$ **unfolding** *mult-Mp* **by** *simp*
finally have *dvd*: $\text{?iu} \ \text{dvdm} \ f$ **unfolding** *dvdm-def* **by** *auto*
from *degu* *deg-iu* **have** *deg-iun*: $\text{degree} \ \text{?iu} < n$ **by** *auto*
from *L.factorization-lattice(2)*[*OF deg-iu0 pl0 mon-iu deg-iun degf dvd*]
have $f \in \text{poly-of-vec} \ \text{'lattice-of} \ ?L$ **using** *deg-iu* **by** *auto*
then obtain fv **where** $f: f = \text{poly-of-vec} \ fv$ **and** $fv: fv \in \text{lattice-of} \ ?L$ **by** *auto*
have *norm*: $\|fv\|^2 = \|f\|^2$ **unfolding** *f* **by** *simp*
have *fv0*: $fv \neq 0_v \ n$ **using** *f0* **unfolding** *f* **by** *auto*
with fv **have** *fvL*: $fv \in \text{lattice-of} \ ?L - \{0_v \ n\}$ **by** *auto*
from *short(3)*[*OF this, unfolded norm*]
have *rat-of-int* $\|LLL\text{-short-polynomial} \ pl \ n \ u\|^2 \leq \text{rat-of-int} \ (2 \wedge (n - 1) * \|f\|^2)$
by *simp*
thus $\|LLL\text{-short-polynomial} \ pl \ n \ u\|^2 \leq 2 \wedge (n - 1) * \|f\|^2$ **by** *linarith*
qed

context *LLL-implementation*
begin

lemma *LLL-reconstruction*: **assumes** *LLL-reconstruction* $f \ us = fs$
and *degree* $f \neq 0$
and *poly-mod.unique-factorization-m* $pl \ f$ (*lead-coeff* f , *mset* us)
and $f \ \text{dvd} \ F$
and $\bigwedge \ ui. ui \in \text{set} \ us \implies \text{poly-mod.Mp} \ pl \ ui = ui$
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*: $pl = p \wedge l$
and p : *prime* p
and *large*: $2 \wedge (5 * (\text{degree} \ F - 1) * (\text{degree} \ F - 1)) * \|F\|^2 \wedge (2 * (\text{degree} \ F - 1)) < pl^2$
shows $f = \text{prod-list} \ fs \wedge (\forall \ fi \in \text{set} \ fs. \text{irreducible}_d \ fi)$
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** *l0*: $l \neq 0$ **by** (*cases* l , *auto*)
show *?thesis* **using** *assms(1-5)*
proof (*induct* $f \ us$ *arbitrary*: fs *rule*: *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 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.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: 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 choose-u-member[OF gs] have u-gs: u ∈ set us unfolding u-def by auto
from fact u-gs have irred: pl.irreduciblea-m u unfolding pl.factorization-m-def
by auto
hence deg-u: degree u ≠ 0 unfolding pl.irreduciblea-m-def norm[OF u-gs] by
auto
have deg-uf: degree u ≤ 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 ≠ 0 by auto
have f0: f ≠ 0 using degf 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
note short-main = LLL-short-polynomial[OF deg-u deg-uf pl1 mon-u]
from short-main(1-2)[folded g-def]
have degree k < degree f unfolding k-def
  by (smt Suc-leI Suc-less-eq degree-gcd1 gcd commute le-imp-less-Suc le-trans)

hence deg-fk: (degree k = 0 ∨ degree f ≤ degree k) = (degree k = 0) by auto
note res = res[unfolded LLL-reconstruction.simps[of f us] Let-def, folded u-def,

```

```

    folded g-def, folded k-def, unfolded deg-fk]
  show ?case
  proof (cases degree k = 0)
    case True
    with res have fs: fs = [f] by auto
    from sf fF have sf: p.square-free-m f
      using p.square-free-m-factor(1)[of f] unfolding dvd-def by auto
    have irr: irreduciblea f
    proof (rule ccontr)
      assume ¬ irreduciblea f
      from reducibleaE[OF this] degf obtain f1 f2 where
        f: f = f1 * f2 and
        deg12: degree f1 ≠ 0 degree f2 ≠ 0 degree f1 < degree f degree f2 < degree
f
      by (simp, metis)
    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)
    note norm-u = norm[OF u-gs]
    from u-gs have u-gs': u ∈# mset us by auto
    with pl.factorization-m-mem-dvdm[OF fact, of u]
    have u-f: pl.dvdm u f by auto
    from u-gs'[unfolded gs] have u ∈# us1 ∨ u ∈# us2 by auto
    with pl.factorization-m-mem-dvdm[OF to-fact[OF uf12(1)], of u]
      pl.factorization-m-mem-dvdm[OF to-fact[OF uf12(2)], of u]
    have pl.dvdm u f1 ∨ pl.dvdm u f2 unfolding norm12 norm-u by auto
    from this have ∃ f1 f2. f = f1 * f2 ∧
      degree f1 ≠ 0 ∧ degree f2 ≠ 0 ∧ degree f1 < degree f ∧ degree f2 < degree
f ∧
      pl.dvdm u f1
    proof
      assume pl.dvdm u f1 thus ?thesis using f deg12 by auto
    next
      from f have f: f = f2 * f1 by auto
      assume pl.dvdm u f2 thus ?thesis using f deg12 by auto
    qed
    then obtain f1 f2 where prod: f = f1 * f2
      and deg: degree f1 ≠ 0 degree f2 ≠ 0 degree f1 < degree f degree f2 <
degree f
      and uf1: pl.dvdm u f1 by auto
    from pl.unique-factorization-m-factor[OF p uf[unfolded prod], folded prod,
OF cop sf l0 plp]
    obtain us1 where fact-f1: pl.unique-factorization-m f1 (lead-coeff f1, us1)

```

by auto
have $plf1: pl.Mp\ f1 \neq 0$
using $to\text{-}fact[OF\ fact\text{-}f1]$ $pl.factorization\text{-}m\text{-}lead\text{-}coeff$
 $pl.unique\text{-}factorization\text{-}m\text{-}zero\ fact\text{-}f1$ **by** $fastforce$
have $degree\ u \leq degree\ f1$
by $(rule\ pl.dvdm\text{-}degree[OF\ mon\text{-}u\ uf1\ plf1])$
with deg **have** $deg\text{-}uf: degree\ u < degree\ f$ **by auto**
have $pl0: pl \neq 0$ **using** $pl.m1\ plp$ **by** $linarith$
let $?n = degree\ f$
let $?n1 = degree\ f1$
let $?d = degree\ u$
from $prod\ fF$ **have** $f1F: f1\ dvd\ F$ **unfolding** $dvd\text{-}def$ **by auto**
from $deg\text{-}uf$ **have** $deg\text{-}uf': ?d \leq ?n$ **by auto**
from deg **have** $f1\text{-}0: f1 \neq 0$ **by auto**
have $ug: pl.dvdm\ u\ g$ **using** $short\text{-}main(3)$ **unfolding** $g\text{-}def$.
have $g0: g \neq 0$ **using** $short\text{-}main(2)$ **unfolding** $g\text{-}def$.
have $deg\text{-}gf: degree\ g < degree\ f$ **using** $short\text{-}main(1)$ **unfolding** $g\text{-}def$.
let $?N = degree\ F$
from $fF\ prod$ **have** $f1F: f1\ dvd\ F$ **unfolding** $dvd\text{-}def$ **by auto**
have $\|g\|^2 \leq 2^{(?n - 1)} * \|f1\|^2$ **unfolding** $g\text{-}def$
by $(rule\ short\text{-}main(4)[OF\ deg\text{-}uf' - uf1],\ insert\ deg,\ auto)$
also have $\dots \leq 2^{(?n - 1)} * (2^{(2 * degree\ f1)} * \|F\|^2)$
by $(rule\ mult\text{-}left\text{-}mono[OF\ sq\text{-}norm\text{-}factor\text{-}bound[OF\ f1F\ F0]],\ simp)$
also have $\dots = 2^{((?n - 1) + 2 * degree\ f1)} * \|F\|^2$
unfolding $power\text{-}add$ **by** $simp$
also have $\dots \leq 2^{((?n - 1) + 2 * (?n - 1))} * \|F\|^2$
by $(rule\ mult\text{-}right\text{-}mono,\ insert\ deg(3),\ auto)$
also have $\dots = 2^{(3 * (?n - 1))} * \|F\|^2$ **by** $simp$
finally have $ineq\text{-}g: \|g\|^2 \leq 2^{(3 * (?n - 1))} * \|F\|^2$.
from $power\text{-}mono[OF\ this,\ of\ ?n1]$
have $ineq1: \|g\|^2^{?n1} \leq (2^{(3 * (?n - 1))} * \|F\|^2)^{?n1}$ **by auto**
from $F0$ **have** $normF: \|F\|^2 \geq 1$ **using** $sq\text{-}norm\text{-}poly\text{-}pos[of\ F]$ **by** $presburger$
from $g0$ **have** $normg: \|g\|^2 \geq 1$ **using** $sq\text{-}norm\text{-}poly\text{-}pos[of\ g]$ **by** $presburger$
from $f0$ **have** $normf: \|f\|^2 \geq 1$ **using** $sq\text{-}norm\text{-}poly\text{-}pos[of\ f]$ **by** $presburger$
from $f1\text{-}0$ **have** $normf1: \|f1\|^2 \geq 1$ **using** $sq\text{-}norm\text{-}poly\text{-}pos[of\ f1]$ **by**
 $presburger$
from $power\text{-}mono[OF\ sq\text{-}norm\text{-}factor\text{-}bound[OF\ f1F\ F0],\ of\ degree\ g]$
have $ineq2: \|f1\|^2^{degree\ g} \leq (2^{(2 * ?n1)} * \|F\|^2)^{degree\ g}$ **by auto**
also have $\dots \leq (2^{(2 * ?n1)} * \|F\|^2)^{(?n - 1)}$
by $(rule\ pow\text{-}mono\text{-}exp,\ insert\ deg\text{-}gf\ normF,\ auto)$
finally have $ineq2: \|f1\|^2^{degree\ g} \leq (2^{(2 * ?n1)} * \|F\|^2)^{(?n - 1)}$.

have $nN: ?n \leq ?N$ **using** $fF\ F0$ **by** $(metis\ dvd\text{-}imp\text{-}degree\text{-}le)$
from $deg\ nN$ **have** $n1N: ?n1 \leq ?N - 1$ **by auto**
have $\|f1\|^2^{degree\ g} * \|g\|^2^{?n1} \leq$
 $(2^{(2 * ?n1)} * \|F\|^2)^{(?n - 1)} * (2^{(3 * (?n - 1))} * \|F\|^2)^{?n1}$
by $(rule\ mult\text{-}mono[OF\ ineq2\ ineq1],\ force+)$
also have $\dots \leq (2^{(2 * (?N - 1))} * \|F\|^2)^{(?N - 1)} *$
 $(2^{(3 * (?N - 1))} * \|F\|^2)^{(?N - 1)}$

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 ... = $2 \wedge (2 * (?N - 1) * (?N - 1) + 3 * (?N - 1) * (?N - 1))$

$* (\|F\|^2) \wedge ((?N - 1) + (?N - 1))$

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 $2 \wedge (5 * (?N - 1) * (?N - 1)) * \|F\|^2 \wedge (2 * (?N - 1)) < pl \wedge 2$

by (rule large)

finally have large: $\|f1\|^2 \wedge degree\ g * \|g\|^2 \wedge degree\ f1 < pl^2$.

have deg-ug: $degree\ u \leq degree\ g$

proof (rule pl.dvdm-degree[OF mon-u ug], standard)

assume pl.Mp $g = 0$

from arg-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 $pl \wedge 2 \leq (lead-coeff\ g) \wedge 2$ unfolding lg abs-le-square-iff[symmetric]

by (rule aux-abs-int)

also have ... $\leq \|g\|^2 \wedge 1$ using coeff-le-sq-norm[of g] by auto

also have ... $\leq \|g\|^2 \wedge degree\ f1$

by (rule pow-mono-exp, insert deg normg, auto)

also have ... = $1 * \dots$ by simp

also have ... $\leq \|f1\|^2 \wedge degree\ g * \|g\|^2 \wedge degree\ f1$

by (rule mult-right-mono, insert normf1, auto)

also have ... $< pl^2$ 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 ug 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 arg-cong[OF this, of degree] f0 have deg-f1k: $degree\ f = degree\ f1 +$

$degree\ k$

by (auto simp: degree-mult-eq)

from f fF have dvd: $f1\ dvd\ F\ k\ dvd\ F$ unfolding dvd-def by auto

obtain gs1 gs2 where part: $List.partition\ (\lambda gi. p.dvdm\ gi\ f1)\ us = (gs1,$

$gs2$) **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 $deg-kg: degree\ k \leq degree\ g$ **unfolding** $k-def\ gcd.commute[of\ f\ g]$
by $(rule\ degree-gcd1[OF\ g0])$
from $deg-gf\ deg-kg$ **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** $norm-12: gi \in set\ gs1 \vee gi \in set\ gs2 \implies pl.Mp\ gi = gi$
for gi **by auto**
note $IH1 = IH(1)[OF\ fs1\ deg-f1\ uf(1)\ dvd(1)\ norm-12]$
note $IH2 = IH(2)[OF\ fs2\ False\ uf(2)\ dvd(2)\ norm-12]$
show $?thesis$ **unfolding** $fs\ f$ **using** $IH1\ IH2$ **by auto**
qed
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 $plp: pl = p \wedge l$
and $p: prime\ p$
and $large: 2^{(5 * (degree\ F\ div\ 2) * (degree\ F\ div\ 2)) * \|F\|^2} \wedge (2 * (degree\ F\ div\ 2)) < pl^2$
shows $f = prod-list\ fs \wedge (\forall fi \in set\ fs. irreducible_d\ fi)$
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** $l0: l \neq 0$ **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.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: 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 arg-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 f1 < degree f by auto
from f fF have dvd: f1 dvd F f2 dvd F unfolding dvd-def by auto
obtain gs1 gs2 where part: List.partition (λgi. p.dvdm 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 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 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 fF have sf: p.square-free-m f
using p.square-free-m-factor(1)[of f] unfolding dvd-def by auto
have irreducibled f
proof (rule ccontr)
assume ¬ irreducibled f
from reducibledE[OF this] degf obtain f1 f2 where
f: f = f1 * f2 and
deg12: 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
deg12: 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[unfolding f]]
have cop1: coprime (lead-coeff f1) p by auto
have deg-m1: pl.degree-m f1 = degree f1
  by (rule poly-mod.degree-m-eq[OF - pl.m1],
    insert cop1 p, simp add: l0 p.coprime-exp-mod plp)
from pl.unique-factorization-m-factor[OF p uf[unfolding 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[OF uf12(1)]
have fact1: pl.factorization-m f1 (lead-coeff f1, us1) .
have plf10: pl.Mp f1 ≠ 0
  using fact1 pl.factorization-m-lead-coeff pl.unique-factorization-m-zero
uf12(1) by fastforce
have degree f1 = pl.degree-m f1 using deg-m1 by simp
also have ... = sum-mset (image-mset pl.degree-m us1)
  unfolding pl.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 degf1-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] u1 have u-f1: pl.dvdm
u f1 by auto
  note norm-u = norm[OF u]
from fact u have irred: pl.irreducible_d-m u unfolding pl.factorization-m-def
by auto

```

hence $\text{deg-}u$: $\text{degree } u \neq 0$ **unfolding** $\text{pl.irreducible}_a\text{-m-def norm}[OF\ u]$ **by**
auto
have $\text{degree } u \leq \text{degree } f1$ **unfolding** degf1-sum **unfolding** $us1$ **by** *simp*
also have $\dots \leq \text{degree } f \text{ div } 2$ **by** *fact*
finally have $\text{deg-}uf$: $\text{degree } u \leq \text{degree } f \text{ div } 2$.
hence $\text{deg-}uf'$: $\text{degree } u \leq \text{Suc } (\text{degree } f \text{ div } 2)$ $\text{degree } u < \text{Suc } (\text{degree } f \text{ div } 2)$
2) by *auto*
from $\text{mon-gs}[OF\ u]$ **have** $\text{mon-}u$: *monic* u .

note $\text{short} = \text{LLL-short-polynomial}[OF\ \text{deg-}u\ \text{deg-}uf'(1)\ \text{pl1}\ \text{mon-}u, \text{folded } gg\text{-def}]$
note $\text{short} = \text{short}(1-3)\ \text{short}(4)[OF\ \text{deg-}uf'(2)]$
from $\text{short}(1,2)\ \text{deg12}(1,3)\ f10$ **have** $\text{degree } (\text{gcd } f\ ?g) \leq \text{degree } f \text{ div } 2$
by (*metis Suc-leI Suc-le-mono degree-gcd1 gcd commute le-trans*)
also have $\dots < \text{degree } f$ **using** degf **by** *simp*
finally have $\text{degree } (\text{gcd } f\ ?g) < \text{degree } f$ **by** *simp*
with $\text{find-map-filter-None}[OF\ \text{None}, \text{simplified}, \text{rule-format}, \text{of } u]\ \text{deg-}uf\ u$
have deg-gcd : $\text{degree } (\text{gcd } f\ (?g)) = 0$ **by** (*auto simp: gcd commute*)
have $\text{gcd } f1\ (?g)\ \text{dvd } \text{gcd } f\ (?g)$ **using** $f0$ **unfolding** f **by** *simp*
from $\text{divides-degree}[OF\ \text{this}, \text{unfolded } \text{deg-gcd}]\ f0$
have deg-gcd1 : $\text{degree } (\text{gcd } f1\ (?g)) = 0$ **by** *auto*
from $F0$ **have** $\text{norm}F$: $\|F\|^2 \geq 1$ **using** $\text{sq-norm-poly-pos}[of\ F]$ **by** *presburger*
have $g0$: $?g \neq 0$ **using** $\text{short}(2)$.
from $g0$ **have** $\text{norm}g$: $\|?g\|^2 \geq 1$ **using** $\text{sq-norm-poly-pos}[of\ ?g]$ **by** *presburger*
from $f10$ **have** $\text{norm}f1$: $\|f1\|^2 \geq 1$ **using** $\text{sq-norm-poly-pos}[of\ f1]$ **by**
presburger
from $fF\ f$ **have** $f1F$: $f1\ \text{dvd } F$ **unfolding** dvd-def **by** *auto*
have $\text{pl-ge}0$: $\text{pl} \geq 0$ **using** $\text{pl.poly-mod-2-axioms poly-mod-2-def}$ **by** *auto*
from fF **have** $\text{degree } f \leq \text{degree } F$ **using** $F0\ f0$ **by** (*metis dvd-imp-degree-le*)
hence $d2D2$: $?d2 \leq ?D2$ **by** *simp*
with $\text{deg12}(3)$ **have** df1-D2 : $\text{degree } f1 \leq ?D2$ **by** *linarith*
from $\text{short}(1)\ d2D2$ **have** dg-D2 : $\text{degree } (gg\ u) \leq ?D2$ **by** *linarith*
have $\|f1\|^2 \wedge \text{degree } (gg\ u) * \|gg\ u\|^2 \wedge \text{degree } f1$
 $\leq \|f1\|^2 \wedge ?D2 * \|gg\ u\|^2 \wedge ?D2$
by (*rule mult-mono[OF pow-mono-exp pow-mono-exp]*,
insert normf1 normg, auto intro: df1-D2 dg-D2)
also have $\dots = (\|f1\|^2 * \|gg\ u\|^2) \wedge ?D2$
by (*simp add: power-mult-distrib*)
also have $\dots \leq (\|f1\|^2 * (2^{?D2} * \|f1\|^2)) \wedge ?D2$
by (*rule power-mono[OF mult-left-mono[OF order.trans[OF short(4)[OF f10 u-f1]]]]*,
insert deg12 d2D2, auto intro!: mult-mono)
also have $\dots = \|f1\|^2 \wedge (?D2 + ?D2) * 2^{(?D2 * ?D2)}$
unfolding $\text{power-add power-mult-distrib power-mult}$ **by** *simp*
also have $\dots \leq (2 \wedge (2 * ?D2) * \|F\|^2) \wedge (?D2 + ?D2) * 2^{(?D2 * ?D2)}$
by (*rule mult-right-mono[OF order.trans[OF power-mono[OF sq-norm-factor-bound[OF f1F F0]]]]*,
auto intro!: power-mono mult-right-mono df1-D2)
also have $\dots = 2 \wedge (2 * ?D2 * (?D2 + ?D2)) + ?D2 * ?D2 * \|F\|^2 \wedge$

$(?D2 + ?D2)$
unfolding *power-mult-distrib power-mult power-add* **by** *simp*
also have $2 * ?D2 * (?D2 + ?D2) + ?D2 * ?D2 = 5 * ?D2 * ?D2$ **by**
simp
also have $?D2 + ?D2 = 2 * ?D2$ **by** *simp*
finally have *large*:
 $\|f1\|^2 \wedge \text{degree } (gg\ u) * \|gg\ u\|^2 \wedge \text{degree } f1 < pl^2$ **using** *large* **by** *simp*
have $\text{degree } u \leq \text{degree } (?g)$
proof (*rule pl.dvdm-degree[OF mon-u short(3)], standard*)
assume $pl.Mp\ (?g) = 0$
from *arg-cong[OF this, of $\lambda p. \text{coeff } p\ (\text{degree } ?g)$]*
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** $lg: \text{lead-coeff } ?g = pl * c$
by *auto*
with $g0$ **have** $c0: c \neq 0$ **by** *auto*
hence $pl^2 \leq (\text{lead-coeff } ?g)^2$ **unfolding** lg *abs-le-square-iff[symmetric]*
by (*rule aux-abs-int*)
also have $\dots \leq \|?g\|^2$ **using** *coeff-le-sq-norm[of ?g]* **by** *auto*
also have $\dots = \|?g\|^2 \wedge 1$ **by** *simp*
also have $\dots \leq \|?g\|^2 \wedge \text{degree } f1$
by (*rule pow-mono-exp, insert deg12 normg, auto*)
also have $\dots = 1 * \dots$ **by** *simp*
also have $\dots \leq \|f1\|^2 \wedge \text{degree } ?g * \|?g\|^2 \wedge \text{degree } f1$
by (*rule mult-right-mono, insert normf1, auto*)
also have $\dots < pl^2$ **by** (*rule large*)
finally show *False* **by** *auto*
qed
with $deg-u$ **have** $deg-g: 0 < \text{degree } (gg\ u)$ **by** *auto*
have $pl-ge0: pl \geq 0$ **using** *pl.poly-mod-2-axioms poly-mod-2-def* **by** *auto*
from fF **have** $\text{degree } f \leq \text{degree } F$ **using** $F0\ f0$ **by** (*metis dvd-imp-degree-le*)
hence $d2D2: ?d2 \leq ?D2$ **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 $0 < \text{degree } f1\ 0 < \text{degree } u$ **using** $deg12\ deg-u$ **by** *auto*
from *common-factor-via-short[of f1 gg u, OF this(1) deg-g mon-u this(2)*
 $u-f1\ short(3) - pl-ge0]$ $deg-gcd1$
have $pl^2 \leq \|f1\|^2 \wedge \text{degree } (gg\ u) * \|gg\ u\|^2 \wedge \text{degree } f1$ **by** *linarith*
also have $\dots < pl^2$ **by** (*rule large*)
finally show *False* **by** *simp*
qed
thus *?thesis* **using** $fs-f$ **by** *simp*
qed
qed
qed
end

lemma *LLL-factorization*:

assumes $res: \text{LLL-factorization } f = gs$

and *sff*: *square-free* *f*
and *deg*: *degree* *f* $\neq 0$
shows $f = \text{prod-list } gs \wedge (\forall g \in \text{set } gs. \text{irreducible}_d g)$
proof –
let *?lc* = *lead-coeff* *f*
define *p* **where** $p \equiv \text{suitable-prime-bz } f$
obtain *c* *gs* **where** *fff*: *finite-field-factorization-int* $p f = (c, gs)$ **by** *force*
let *?degs* = *map degree* *gs*
note *res* = *res*[*unfolded 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** $K = 2^{(5 * (\text{degree } f - 1) * (\text{degree } f - 1)) * \|f\|^2 \wedge (2 * (\text{degree } f - 1))}$
define *N* **where** $N = \text{sqrt-int-ceiling } K$
have *K0*: $K \geq 0$ **unfolding** *K-def* **by** *fastforce*
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** $n = \text{find-exponent } p N$
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*: $f \neq 0$ **by** *auto*
from *n p.m1* **have** *pn1*: $p \wedge n > 1$ **by** *auto*
note *res* = *res*[*folded bh(1)*]
note $*$ = *p.berlekamp-hensel-unique*[*OF cop sf bh n(2)*]
note $**$ = *p.berlekamp-hensel-main*[*OF n(2) bh cop sf fff*]
from *res * ***
have *uf*: *poly-mod.unique-factorization-m* ($p \wedge n$) *f* (*lead-coeff* *f*, *mset* (*berlekamp-hensel* $p n f$))
and *norm*: $\bigwedge ui. ui \in \text{set } (\text{berlekamp-hensel } p n f) \implies \text{poly-mod.Mp } (p \wedge n)$
ui = *ui*
unfolding *berlekamp-hensel-def* *fff split* **by** *auto*
have *K*: $K < (p \wedge n)^2$ **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* $\neq 0$

```

shows  $f = \text{prod-list } gs \wedge (\forall g \in \text{set } gs. \text{irreducible}_d g)$ 
proof –
  let ?lc = lead-coeff  $f$ 
  define  $p$  where  $p \equiv \text{suitable-prime-bz } f$ 
  obtain  $c \text{ } gs$  where  $\text{fff}: \text{finite-field-factorization-int } p \text{ } f = (c, gs)$  by force
  let ?degs = map degree  $gs$ 
  note  $res = \text{res}[\text{unfolded } \text{LLL-many-factorization-def } \text{Let-def}, \text{folded } p\text{-def},$ 
     $\text{unfolded } \text{fff split}, \text{folded}]$ 
  from suitable-prime-bz [OF sff refl]
  have  $\text{prime}: \text{prime } p$  and  $\text{cop}: \text{coprime } ?lc \text{ } p$  and  $\text{sf}: \text{poly-mod.square-free-m } p$ 
 $f$ 
    unfolding  $p\text{-def}$  by auto
  note  $res$ 
  from  $\text{prime}$  interpret  $p: \text{poly-mod-prime } p$  by unfold-locales
  define  $K$  where  $K = 2^{(5 * (\text{degree } f \text{ div } 2) * (\text{degree } f \text{ div } 2)) * \|f\|^2 ^{(2 * (\text{degree } f \text{ div } 2))}$ 
  define  $N$  where  $N = \text{sqrt-int-ceiling } K$ 
  have  $K0: K \geq 0$  unfolding  $K\text{-def}$  by fastforce
  have  $N0: N \geq 0$  unfolding  $N\text{-def}$  sqrt-int-ceiling using  $K0$ 
    by (smt of-int-nonneg real-sqrt-ge-0-iff zero-le-ceiling)
  define  $n$  where  $n = \text{find-exponent } p \text{ } N$ 
  note  $res = \text{res}[\text{folded } n\text{-def}[\text{unfolded } N\text{-def } K\text{-def}]]$ 
  note  $n = \text{find-exponent}[OF p.m1, \text{of } N, \text{folded } n\text{-def}]$ 
  note  $\text{bh} = p.\text{berlekamp-and-hensel-separated}(1)[OF \text{cop sf refl fff } n(2)]$ 
  from  $\text{deg}$  have  $f0: f \neq 0$  by auto
  from  $n \text{ } p.m1$  have  $\text{pn1}: p ^ n > 1$  by auto
  note  $res = \text{res}[\text{folded } \text{bh}(1)]$ 
  note  $*$  =  $p.\text{berlekamp-hensel-unique}[OF \text{cop sf bh } n(2)]$ 
  note  $**$  =  $p.\text{berlekamp-hensel-main}[OF n(2) \text{bh cop sf fff}]$ 
  from  $res * **$ 
  have  $\text{uf}: \text{poly-mod.unique-factorization-m } (p ^ n) \text{ } f (\text{lead-coeff } f, \text{mset } (\text{berlekamp-hensel } p \text{ } n \text{ } f))$ 
    and  $\text{norm}: \bigwedge ui. ui \in \text{set } (\text{berlekamp-hensel } p \text{ } n \text{ } f) \implies \text{poly-mod.Mp } (p ^ n)$ 
 $ui = ui$ 
    unfolding berlekamp-hensel-def fff split by auto
  have  $K: K < (p ^ n)^2$  using  $n \text{ } \text{sqrt-int-ceiling-bound}[OF K0]$ 
    by (smt N0 N-def n(1) power2-le-imp-le)
  show ?thesis
    by (rule LLL-implementation.LLL-many-reconstruction [OF res deg uf dvd-refl]
      norm f0 cop sf pn1
      refl prime K [unfolded K-def])
qed

```

lift-definition *one-lattice-LLL-factorization* :: *int-poly-factorization-algorithm*
 is *LLL-factorization* **using** *LLL-factorization* **by** *auto*

lift-definition *many-lattice-LLL-factorization* :: *int-poly-factorization-algorithm*
 is *LLL-many-factorization* **using** *LLL-many-factorization* **by** *auto*

```

lemma LLL-factorization-primitive: assumes LLL-factorization  $f = fs$ 
  square-free  $f$ 
   $0 < \text{degree } f$ 
  primitive  $f$ 
shows  $f = \text{prod-list } fs \wedge (\forall fi \in \text{set } fs. \text{irreducible } fi \wedge 0 < \text{degree } fi \wedge \text{primitive } fi)$ 
  using assms(1)
  by (intro int-poly-factorization-algorithm-irreducible[of one-lattice-LLL-factorization,
     $OF - \text{assms}(2-)$ ], 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  $S = \text{sub-mset-sums } xs$  in  $S \cup ( (+ ) x ` S)$ )

lemma subset-add-mset:  $ys \subseteq\# \text{add-mset } x \text{ } zs \iff (ys \subseteq\# zs \vee (\exists xs. xs \subseteq\# zs \wedge ys = \text{add-mset } x \text{ } xs))$ 
  (is ?l = ?r)
proof
  have sub:  $ys \subseteq\# zs \implies ys \subseteq\# \text{add-mset } x \text{ } zs$ 
  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  $ys: ys = \text{add-mset } x \text{ } xs$  unfolding xs-def by auto
    from  $l[\text{unfolded } ys]$  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

```

```

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

end

```

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.

```

theory Factorization-Algorithm-16-22
  imports
    LLL-Factorization
    Sub-Sums
begin

```

7.1 Previous lemmas obtained using local type definitions

```

context poly-mod-prime-type
begin

```

```

lemma irreducible-m-dvdm-prod-list-connect:
  assumes irr: irreducible-m a
  and dvd: a dvdm (prod-list xs)

```

```

shows  $\exists b \in \text{set } xs. a \text{ dvdm } b$ 
proof –
  let ?A=(of-int-poly a)::'a mod-ring poly
  let ?XS=(map of-int-poly xs)::'a mod-ring poly list
  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) ?XS1
    by (simp add: MP-Rel-def Mp-f-representative)
  have [transfer-rule]: list-all2 MP-Rel xs ?XS
    by (simp add: MP-Rel-def Mp-f-representative list-all2-conv-all-nth)
  have A: ?A 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 ?thesis .
qed

```

end

lemma (in poly-mod-prime) irreducible-m-dvdm-prod-list:

assumes irr: irreducible-m a

and dvd: a dvdm (prod-list xs)

shows $\exists b \in \text{set } xs. a \text{ dvdm } b$

by (rule poly-mod-prime-type.irreducible-m-dvdm-prod-list-connect[unfolded poly-mod-type-simps,

internalize-sort 'a :: prime-card, OF type-to-set, unfolded remove-duplicate-premise,

cancel-type-definition, 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 = $2^{\wedge} (2 * \text{degree } f) * \|f\|^2$

hide-const (open) factors

hide-const (open) factors

hide-const (open) factor

hide-const (open) factor

context

fixes p :: int **and** l :: nat

begin

context

fixes gs :: int poly list

and f :: int poly

and u :: int poly

and Degs :: nat set

begin

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^l . However, for soundness we need to know that then also the primitive part of g' is divisible by u modulo p^l . This is not necessary true, e.g., if $g' = p^l$, then the primitive part is 1 which is not divisible by u modulo p^l . 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^l .

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

definition *LLL-reconstruction-inner* $j \equiv$

let $j' = j - 1$ in

— optimization: check whether degree j' is possible

if $j' \notin \text{Degs}$ then None else

— short vector computation

let

$ll = (\text{let } n = \text{sqrt-int-ceiling } (\|f\|^2 \wedge (2 * j') * 2 \wedge (5 * j' * j'));$

$ll' = \text{find-exponent } p \ n \ \text{in if } ll' < l \ \text{then } ll' \ \text{else } l);$

— optimization: dynamically adjust the modulus

$pl = p \wedge ll;$

$g' = \text{LLL-short-polynomial } pl \ j \ u$

— fix: forbid multiples of p^l as short vector, unclear whether this is really required
in if $\text{abs } (\text{lead-coeff } g') \geq pl$ then None else

let $ppg = \text{primitive-part } g'$

in

— slight deviation from textbook: we check divisibility instead of norm-inequality
case $\text{div-int-poly } f \ ppg \ \text{of Some } f' \Rightarrow$

— fix: consider modular factors of ppg and not of g'

$\text{Some } (\text{filter } (\lambda gi. \neg \text{poly-mod.dvdm } p \ gi \ ppg) \ gs, \ \text{lead-coeff } f', \ f', \ ppg)$

| None \Rightarrow None

function *LLL-reconstruction-inner-loop* **where**

LLL-reconstruction-inner-loop $j =$

(if $j > \text{degree } f$ then $([], 1, 1, f)$

else case *LLL-reconstruction-inner* j

of Some tuple \Rightarrow tuple

| None \Rightarrow *LLL-reconstruction-inner-loop* $(j+1)$)

by *auto*

termination by (*relation measure* $(\lambda j. \text{Suc } (\text{degree } f) - j), \ \text{auto})$

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 $no = (n + 1) * (max - norm f)^2$
 no = $\|f\|^2$;
 — *possible* improvement: $B = sqrt(2^{5*n*(n-1)} * no^{2*n-1})$, cf. *LLL-factorization*
 B = *sqrt-int-ceiling* ($2 \wedge (5 * n * n) * no \wedge (2 * n)$);
 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* ($p * p \wedge k$) (*smult* *p f*) = 0 ⇒ *poly-mod.Mp* ($p \wedge k$) *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* ($p * p \wedge k$)
(smult p f) (smult p g)
 \implies *poly-mod.eq-m* ($p \wedge k$) *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)) ≤ 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) ≠ 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*
 $\neq 0 \mid c \mid p \wedge l$
and *dvd*: *poly-mod.dvdm* ($p \wedge l$) *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 l = 0*)
case *True*
thus *?thesis* **using** *c dvd* **by** *auto*
next
case *l0: False*
interpret *pl: poly-mod-2 p \wedge l* **by** (*unfold-locales, insert m1 l0, auto*)
show *?thesis*
proof (*cases p dvd c*)
case *False*
let *?i = inverse-mod c (p \wedge 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 $\dots = pl.Mp f$ **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
  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

```

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 $\dots \leq B2\text{-LLL } F$ **unfolding** *B2-LLL-def*
by (*intro mult-mono power-increasing F1, insert N0, auto*)
finally show $2 \leq B2\text{-LLL } F$.
qed

lemma *l-gt-0*: $l > 0$

proof (*cases l*)
case 0
have $1 * 2 \leq 2^N * B2\text{-LLL } F \wedge (2 * N)$
proof (*rule mult-mono*)
have $2 * 1 \leq (2 :: int) * (2^{2*N - 1})$ **by** (*rule mult-left-mono, auto*)
also have $\dots = 2^{2 * N}$ **using** $N0$ **by** (*cases N, auto*)
also have $\dots \leq B2\text{-LLL } F \wedge (2 * N)$
by (*rule power-mono[OF B2-2], force*)
finally show $2 \leq B2\text{-LLL } F \wedge (2 * N)$ **by** *simp*
qed *auto*
also have $\dots \leq 1$ **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*: $p^l \neq 0$ **using** *p1 l0* **by** *auto*

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

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

lemma *p-Mp-pl-Mp[simp]*: $p.Mp (pl.Mp k) = p.Mp k$
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 $Degs :: nat$ *set*
defines [*simp*]: $d \equiv \text{degree } u$
assumes $d0: d > 0$
and u : *monic u*
and $irred\text{-}u$: *p.irreducible-m u*
and $u\text{-}f$: *p.dvdm u f*
and $f\text{-}dvd\text{-}F$: *f dvd F*
and [*simp*]: $n == \text{degree } f$
and $f\text{-}gs$: *pl.unique-factorization-m f (lead-coeff f, mset gs)*

```

    and cop: coprime (lead-coeff f) p
    and sf: p.square-free-m f
    and sf-F: square-free f
    and u-gs: u ∈ set gs
    and norm-gs: map pl.Mp gs = gs
    and Degr:  $\bigwedge$  factor. factor dvd f  $\implies$  p.dvdm u factor  $\implies$  degree factor ∈
  Degr
begin
interpretation pl: poly-mod-2 p ^ l using l0 p1 by (unfold-locales, auto)

private lemma f0: f ≠ 0 using sf-F unfolding square-free-def by fastforce

private lemma Mp0: pl.Mp f ≠ 0
  by (metis p.square-free-m-def p-Mp-pl-Mp sf)

private lemma pMp0: p.Mp f ≠ 0
  using p.square-free-m-def sf by auto

private lemma dn: d ≤ n using p.dvdm-imp-degree-le[OF u-f u pMp0 p1] by
  auto

private lemma n0: n > 0 using d0 dn by auto

private lemma B2-0[intro!]: B2-LLL F > 0 using B2-2 by auto
private lemma deg-u: degree u > 0 using d0 d-def by auto

private lemma n-le-N: n ≤ N by (simp add: dvd-imp-degree-le[OF f-dvd-F F0])

lemma dvdm-power: assumes g dvd f
  shows p.dvdm u g  $\longleftrightarrow$  pl.dvdm u g
proof
  assume pl.dvdm u g
  thus p.dvdm u g by (rule pl-dvdm-imp-p-dvdm)
next
  assume dvd: p.dvdm u g
  from norm-gs have norm-gsp:  $\bigwedge$  f. f ∈ set gs  $\implies$  pl.Mp f = f by (induct gs,
  auto)
  with f-gs[unfolded pl.unique-factorization-m-alt-def pl.factorization-m-def split]
  have gs-irred-mon:  $\bigwedge$  f. f ∈ # mset gs  $\implies$  pl.irreduciblea-m f  $\wedge$  monic f by
  auto
  from norm-gs have norm-gs: image-mset pl.Mp (mset gs) = mset gs by (induct
  gs, auto)
  from assms obtain h where f: f = g * h unfolding dvd-def by auto
  from pl.unique-factorization-m-factor[OF p.prime f-gs[unfolded f] - - l0 refl,
  folded f,
  OF cop sf, unfolded pl.Mf-def split] norm-gs
  obtain hs fs where uf: pl.unique-factorization-m h (lead-coeff h, hs)
  pl.unique-factorization-m g (lead-coeff g, fs)
  and id: mset gs = fs + hs

```

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*: $\exists h0. \text{irreducible}_d h0 \wedge p.dvdm u h0 \wedge h0 \text{ dvd } f$
proof –

have *deg-f*: *degree f > 0* **using** $\langle n \equiv \text{degree } f \rangle n0$ **by** *blast*
from *berlekamp-zassenhaus-factorization-irreducible_d*[*OF refl sf-F deg-f*]
obtain *fs* **where** *f-fs*: *f = prod-list fs*
 and *c*: $(\forall fi \in \text{set } fs. \text{irreducible}_d fi \wedge 0 < \text{degree } fi)$ **by** *blast*
have *pl.dvdm u (prod-list fs)* **using** *uf f-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** *irreducible_d h0* **using** *c h0* **by** *auto*
ultimately show *?thesis* **by** *blast*
qed

lemma *factor-dvd-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** $0 \neq pl.Mp (pl.Mp \text{ factor } * h)$

using *Mpf0* **by** *auto*
thus *?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** $\text{degree } u \leq \text{degree } \text{factor}$

proof –

from *factor-dvd-f-0*[*OF factor-dvd*] **have** *factor0*: *pl.Mp factor $\neq 0$* .
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 $\neq 0$* **by** *fastforce*
hence $0 \neq \text{lead-coeff } (\text{pl.Mp } v)$ **by** *auto*
also have $\text{lead-coeff } (\text{pl.Mp } v) = \text{pl.M } (\text{lead-coeff } (\text{pl.Mp } v))$
by (*auto simp: pl.Mp-def coeff-map-poly*)
finally have $*$: $\text{lead-coeff } (\text{pl.Mp } v) \neq p \wedge l * r$ **for** *r* **by** (*auto simp: pl.M-def*)

from $*$ **have** $\text{degree } \text{factor} \geq \text{pl.degree-m } (u * \text{pl.Mp } v)$ **using** *pl.degree-m-le*[*of factor*] **by** *auto*

also have $\text{pl.degree-m } (u * \text{pl.Mp } v) = \text{degree } (u * \text{pl.Mp } v)$
by (*rule pl.degree-m-eq, unfold lead-coeff-mult, insert u pl.m1 **, auto*)
also have $\dots = \text{degree } u + \text{degree } (\text{pl.Mp } v)$
by (*rule degree-mult-eq, insert v0 u, auto*)

finally show *?thesis* **by** *auto*

qed

7.3.2 Inner loop

context

fixes *j'* :: *nat*
assumes *dj'*: $d \leq j'$
and *j'n*: $j' < n$
and *deg*: $\bigwedge \text{factor. } p.dvdm \text{ } u \text{ } \text{factor} \implies \text{factor } dvd \text{ } f \implies \text{degree } \text{factor} \geq j'$

begin

private abbreviation (*input*) *j* \equiv *Suc j'*

private lemma *jn*: $j \leq n$ **using** *j'n* **by** *auto*

private lemma *factor-irreducible_dI*: **assumes** *hf*: *h dvd f*

and *puh*: *p.dvdm u h*
and *degh*: $\text{degree } h > 0$
and *degh-j*: $\text{degree } h \leq j'$

shows *irreducible_d h*

proof –

from *dvdm-power*[*OF hf*] *puh* **have** *pluh*: *pl.dvdm u h* **by** *simp*
note *uf-partition* = *p.unique-factorization-m-factor-partition*[*OF l0*]
obtain *gs1 gs2* **where** *part*: *List.partition* ($\lambda gi. p.dvdm gi h$) *gs* = (*gs1*, *gs2*)


```

by force
  from part u-gs puh
  have u-gs1:  $u \in \text{set } gs1$  unfolding p by auto
  have gs1:  $gs1 = \text{filter } (\lambda gi. p.\text{dvd}m \ gi \ h) \ gs$  using part by auto
  obtain k where  $f: f = h * k$  using hf unfolding dvd-def by auto
  from uf-partition[OF f-gs f cop sf part]
  have uf-h:  $pl.\text{unique-factorization-}m \ h \ (\text{lead-coeff } h, \ mset \ gs1)$  by auto
  show ?thesis
  proof (intro irreducibleaI degh)
    fix q r
    assume deg-q:  $\text{degree } q > 0 \ \text{degree } q < \text{degree } h$ 
      and deg-r:  $\text{degree } r > 0 \ \text{degree } r < \text{degree } h$ 
      and h:  $h = q * r$ 
    then have r dvd h by auto
    with h dvd-trans[OF - hf] have 1:  $q \ \text{dvd} \ f \ r \ \text{dvd} \ f$  by auto
    from cop[unfolded f] have cop:  $\text{coprime } (\text{lead-coeff } h) \ p$ 
      using p.prime pl.coprime-lead-coeff-factor(1) by blast
    from sf[unfolded f] have sf:  $p.\text{square-free-}m \ h$  using p.square-free-m-factor
  by metis
    have norm-gs1:  $\text{image-mset } pl.Mp \ (mset \ gs1) = mset \ gs1$  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:  $pl.\text{unique-factorization-}m \ q \ (\text{lead-coeff } q, \ fs)$ 
      and uf-r:  $pl.\text{unique-factorization-}m \ r \ (\text{lead-coeff } r, \ gs)$ 
      and id:  $mset \ gs1 = fs + gs$ 
    unfolding pl.Mf-def split using norm-gs1 by auto
  from degh degh-j deg-q deg-r have qj':  $\text{degree } q < j'$  and rj':  $\text{degree } r < j'$  by
  auto
    have intro:  $u \in \# \ r \implies pl.Mp \ u \in \# \ \text{image-mset } pl.Mp \ r$  for r by auto
  note dvdI =  $pl.\text{factorization-}m\text{-mem-}dvd\ m \ [OF \ pl.\text{unique-factorization-}m\text{-imp-factorization}$ 
  intro]
  from u-gs1 id have  $u \in \# \ fs \vee u \in \# \ gs$  unfolding in-multiset-in-set[symmetric]
  by auto
    with dvdI[OF uf-q] dvdI[OF uf-r] have  $pl.\text{dvd}m \ u \ q \vee pl.\text{dvd}m \ u \ r$  by auto
    hence  $p.\text{dvd}m \ u \ q \vee p.\text{dvd}m \ u \ r$  using pl.dvd-m-imp-p-dvd-m by blast
    with 1 qj' rj' show False
      by (elim disjE, auto dest!: deg)
  qed
  qed

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

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*: $p^{ll} > 1$ using *ll0 p.m1* by *auto*

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

lemma *pll0*: $p^{ll} \neq 0$ using *p* by *auto*

lemma *dvdm-l-ll*: **assumes** *pl.dvdm a b*
shows *pll.dvdm a b*

proof –
have *id*: $p^{ll} = p^{ll} * p^{l - ll}$ 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 *pll.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* \equiv *LLL-short-polynomial (p^{ll}) j u*

lemma *deg-g-j*: *degree g < j*
and *g0*: $g \neq 0$
and *ug* : *pll.dvdm u g*
and *short-g*: $h \neq 0 \implies pll.dvdm u h \implies degree h \leq j' \implies \|g\|^2 \leq 2^{j'} * \|h\|^2$

proof (*atomize(full), goal-cases*)
case 1
from *deg-u* **have** *deg-u0*: *degree u \neq 0* by *auto*
have *ju*: $j \geq degree u$ using *d-def dj' le-Suc-eq* by *blast*
have *ju'*: $j > degree u$ using *d-def dj'* by *auto*
note *short* = *LLL-short-polynomial[OF deg-u0 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 Degr j*
= (*if j' \notin Degr then None else if $p^{ll} \leq |lead-coeff g|$ then None else case *div-int-poly f (primitive-part g)* of *None \Rightarrow None* | *Some f' \Rightarrow Some ([gi \leftarrow gs . \neg p.dvdm gi (primitive-part g)], lead-coeff f', f', primitive-part g)*)*

proof –
have *Suc*: *Suc j' - 1 = j'* 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

lemma *LLL-reconstruction-inner-complete*:

assumes *ret*: *LLL-reconstruction-inner* *p l gs f u Degr* *j = None*

shows $\bigwedge \text{factor}. p.\text{dvd} m\ u\ \text{factor} \implies \text{factor}\ \text{dvd}\ f \implies \text{degree}\ \text{factor} \geq j$

proof (*rule ccontr*)

fix *factor*

assume *pu-factor*: *p.dvd* *m* *u* *factor*

and *factor-f*: *factor* *dvd* *f*

and *deg-factor2*: $\neg j \leq \text{degree}\ \text{factor}$

with *deg*[*OF this*(1,2)] **have** *deg-factor-j* [*simp*]: *degree* *factor* = *j'* **and** *deg-factor-lt-j*: *degree* *factor* < *j* **by** *auto*

from *Degr*[*OF factor-f pu-factor*] **have** *Degr*: (*j' ∉ Degr*) = *False* **by** *auto*

from *dvd**m*-*power*[*OF factor-f*] *pu-factor* **have** *u-factor*: *pl.dvd* *m* *u* *factor* **by** *auto*

from *dvd**m*-*l*-*ll*[*OF u-factor*] **have** *pll-u-factor*: *pll.dvd* *m* *u* *factor* **by** *auto*

have *deg-factor*: *degree* *factor* > 0

using *d0 deg-factor-j dj'* **by** *linarith*

from *f0 deg-factor divides-degree*[*OF factor-f*] **have** *deg-f*: *degree* *f* > 0 **by** *auto*

from *deg-factor* **have** *j'0*: *j' > 0* **by** *simp*

from *factor-f f0* **have** *factor0*: *factor* $\neq 0$ **by** *auto*

from *factor-f* **obtain** *f2* **where** *f*: *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-j'*: *degree* *u* $\leq j'$ **unfolding** *deg-factor-j*[*symmetric*]

using *d-def deg-factor-j dj'* **by** *blast*

hence *u-j*: *degree* *u* $\leq j$ *degree* *u* < *j* **by** *auto*

note *LLL* = *LLL-short-polynomial*[*OF deg-u0 u-j(1) pll1 u, folded g-def*]

note *ret* = *ret*[*unfolded LLL-reconstruction-inner-simps Degr if-False*]

note *LLL* = *LLL(1-3) LLL(4)*[*OF u-j(2) factor0 pll-u-factor deg-factor-lt-j*]

hence *deg-g*: *degree* *g* $\leq j'$ **by** *simp*

from *LLL(2)* **have** *normg*: $\|g\|^2 \geq 1$ **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*: $\|\text{factor}\|^2 \geq 1$ **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*

from *factor-f* (*f dvd F*) **have** *factor-F*: *factor* *dvd* *F* **by** (*rule dvd-trans*)

have $\|\text{factor}\|^2 \wedge \text{degree}\ g * \|g\|^2 \wedge \text{degree}\ \text{factor} \leq \|\text{factor}\|^2 \wedge j' * \|g\|^2 \wedge j'$

by (*rule mult-mono*[*OF power-increasing*], *insert normg normf1 deg-g, auto*)

also **have** $\dots = (\|\text{factor}\|^2 * \|g\|^2) \wedge j'$ **by** (*simp add: power-mult-distrib*)

also **have** $\dots \leq (\|\text{factor}\|^2 * (2 \wedge j' * \|\text{factor}\|^2)) \wedge j'$

by (*rule power-mono*[*OF mult-left-mono*], *insert LLL(4), auto*)

also **have** $\dots = \|\text{factor}\|^2 \wedge (2 * j') * 2 \wedge (j' * j')$

unfolding *power-mult-distrib power-mult power-add mult-2* **by** *simp*

finally **have** *approx-part-1*: $\|\text{factor}\|^2 \wedge \text{degree}\ g * \|g\|^2 \wedge \text{degree}\ \text{factor} \leq \|\text{factor}\|^2 \wedge (2 * j') * 2 \wedge (j' * j')$.

{

fix *f* :: *int* *poly*

assume *: *factor* *dvd* *f* *f* $\neq 0$

note *approx-part-1*

also **have** $\|\text{factor}\|^2 \wedge (2 * j') * 2 \wedge (j' * j') \leq (2 \wedge (2 * j')) * \|f\|^2 \wedge (2 * j') *$

$2^{j' * j'}$
by (rule *mult-right-mono*[*OF power-mono*], insert *sq-norm-factor-bound*[*OF **], *auto*)
also have ... = $\|f\|^2 \wedge (2 * j') * 2 \wedge (2 * j' * 2 * j' + j' * j')$
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 $\|factor\|^2 \wedge degree\ g * \|g\|^2 \wedge degree\ factor \leq \|f\|^2 \wedge (2 * j') * 2 \wedge (5 * j' * j')$.
} *note approx = this*
note *approx-1 = approx*[*OF factor-f f0*]
note *approx-2-part = approx*[*OF factor-F F0*]
have *large*: $\|factor\|^2 \wedge degree\ g * \|g\|^2 \wedge degree\ factor < (p \wedge l)^2$
proof (*cases ll = l*)
 case *False*
 let *?n* = $\|f\|^2 \wedge (2 * j') * 2 \wedge (5 * j' * j')$
 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 *spl*: *?s* $< p \wedge ll$ **using** *find-exponent(1)*[*OF p.m1*] **by** *auto*
 have *sqr* *?n* ≥ 0 **by** *auto*
 hence *sqr*: *sqr ?n* > -1 **by** *linarith*
 have *ns*: *?n* $\leq ?s \wedge 2$ **using** *sqrt-int-ceiling-bound*[*OF n*].
 also have ... $< (p \wedge ll) \wedge 2$
 by (*rule power-strict-mono*[*OF spl*], insert *sqr*, *auto*)
 finally show *?thesis* **using** *approx-1* **by** *auto*
 next
 case *True*
 hence *ll*: $p \wedge ll = p \wedge l$ **by** *simp*
 show *?thesis* **unfolding** *ll*
 proof (*rule less-le-trans*[*OF le-less-trans*[*OF approx-2-part*] *bound-l*])
 have $\|F\|^2 \wedge (2 * j') * 2 \wedge (5 * j' * j')$
 = $2 \wedge (2 * j' * j' + 3 * j' * j') * \|F\|^2 \wedge (j' + j')$
 unfolding *mult-2* **by** *simp*
 also have ... $< 2 \wedge (N^2 + 4 * N * N) * \|F\|^2 \wedge (2 * N)$
 proof (*rule mult-less-le-imp-less*[*OF power-strict-increasing pow-mono-exp*])
 show $1 \leq \|F\|^2$ **by** (*rule normF*)
 have *jN'*: *j' < N* **and** *jN*: *j' ≤ N* **using** *jn divides-degree*[*OF (f dvd F)*]
 F0 **by** *auto*
 have *j' + j' ≤ j' + j'* **using** *deg-g j'n* **by** *auto*
 also have ... = $2 * j'$ **by** *auto*
 also have ... $\leq 2 * N$ **using** *jN* **by** *auto*
 finally show *j' + j' ≤ 2 * N*.
 show $0 < \|F\|^2 \wedge (j' + j')$
 by (*rule zero-less-power*, insert *normF*, *auto*)
 have $2 * j' * j' + 3 * j' * j' \leq 2 * j' * j' + 3 * j' * j'$ **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 $2 * j' * j' + 3 * j' * j' < N^2 + 4 * N * N$.
qed *auto*
also have $\dots = 2 \wedge N^2 * (2 \wedge (2 * N) * \|F\|^2) \wedge (2 * N)$
unfolding *power-mult-distrib power-add* **by** (*simp add: power-mult[symmetric]*)
finally show $\|F\|^2 \wedge (2 * j') * 2 \wedge (5 * j' * j') < 2 \wedge N^2 * B2-LLL F \wedge (2$
 $* N)$
unfolding *B2-LLL-def* **by** *simp*
qed
qed
have $(|lead-coeff g|) \wedge 2 < (p \wedge ll) \wedge 2$
proof (*rule le-less-trans[OF - large]*)
have $1 * (|lead-coeff g|^2) \wedge 1 \leq \|factor\|^2 \wedge degree g * \|g\|^2 \wedge degree factor$
by (*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 $|lead-coeff g|^2 \leq \|factor\|^2 \wedge degree g * \|g\|^2 \wedge degree factor$ **by** *simp*
qed
hence $(lead-coeff g) \wedge 2 < (p \wedge ll) \wedge 2$ **by** *simp*
hence $|lead-coeff g| < p \wedge ll$ **using** *p.m1 abs-le-square-iff[of p \wedge ll lead-coeff g]* **by**
auto
hence $(p \wedge ll \leq |lead-coeff g|) = False$ **by** *auto*
note *ret = ret[unfolded this if-False]*
have *deg-f: degree f > 0* **using** *n0* **by** *auto*
have *deg-ug: degree u ≤ degree g*
proof (*rule pll.dvdm-degree[OF u LLL(3)], standard*)
assume *pll.Mp g = 0*
from *arg-cong[OF this, of λ p. coeff p (degree g)]*
have *pll.M (coeff g (degree g)) = 0* **by** (*auto simp: pll.Mp-def coeff-map-poly*)
from *this[unfolded pll.M-def]* **obtain** *c* **where** *lg: lead-coeff g = p \wedge ll * c* **by**
auto
with *LLL(2)* **have** *c0: c ≠ 0* **by** *auto*
hence $(p \wedge ll) \wedge 2 \leq (lead-coeff g) \wedge 2$ **unfolding** *lg abs-le-square-iff[symmetric]*
by (*rule aux-abs-int*)
also have $\dots \leq \|g\|^2$ **using** *coeff-le-sq-norm[of g]* **by** *auto*
also have $\dots = \|g\|^2 \wedge 1$ **by** *simp*
also have $\dots \leq \|g\|^2 \wedge degree factor$
by (*rule pow-mono-exp, insert deg-f normg j'0, auto*)
also have $\dots = 1 * \dots$ **by** *simp*
also have $\dots \leq \|factor\|^2 \wedge degree g * \|g\|^2 \wedge degree factor$
by (*rule mult-right-mono, insert normf1, auto*)
also have $\dots < (p \wedge ll)^2$ **by** (*rule large*)
finally show *False* **by** *auto*
qed
with *deg-u* **have** *deg-g: degree g > 0* **by** *simp*
from *j'0* **have** *deg-factor: degree factor > 0* **by** *simp*
let *?g = gcd factor g*
from *common-factor-via-short[OF deg-factor deg-g u deg-u pll-u-factor LLL(3)*
large] pll.m1
have *gcd: 0 < degree ?g* **by** *auto*

```

have gcd-factor: ?g dvd factor by auto
from dvd-trans[OF this factor-f] have gcd-f: ?g dvd f .
from deg-g have g0: g ≠ 0 by auto
have gcd-g: degree ?g ≤ degree g using g0 using divides-degree by blast
from gcd-g LLL(1) have hj': degree ?g ≤ j' by auto
let ?pp = primitive-part g
from ret have div-int-poly f ?pp = None by (auto split: option.splits)
from div-int-poly[of f ?pp, unfolded this] g0
have ppf: ¬ ?pp dvd f unfolding dvd-def by (auto simp: ac-simps)
have irr-f1: irreduciblea factor
  by (rule factor-irreducibleaI[OF factor-f pu-factor deg-factor], simp)
from gcd-factor obtain h where factor: factor = ?g * h unfolding dvd-def by
auto
from irreducibleaD(2)[OF irr-f1, of ?g h, folded factor] have ¬ (degree ?g < j'
∧ degree h < j')
  by auto
moreover have j' = degree ?g + degree h using factor0 arg-cong[OF factor, of
degree]
  by (subst (asm) degree-mult-eq, insert j'0, auto)
ultimately have degree h = 0 using gcd by linarith
from degree0-coeffs[OF this] factor factor0
obtain c where h: h = [:c:] and c: c ≠ 0 by fastforce
from arg-cong[OF factor, of degree] have id: degree ?g = degree factor
  unfolding h using c by auto
moreover have degree ?g ≤ degree g
  by (subst gcd.commute, rule degree-gcd1[OF g0])
ultimately have degree g ≥ degree factor by auto
with id deg-factor2 deg-g-j have deg: degree ?g = degree g
  and degree g = degree factor by auto
have ?g dvd g by auto
then obtain q where g: g = ?g * q unfolding dvd-def by auto
from arg-cong[OF this, of degree] deg
have degree q = 0
  by (subst (asm) degree-mult-eq, insert g g0, force, force) simp
from degree0-coeffs[OF this] g g0
obtain d where p: q = [:d:] and d: d ≠ 0 by fastforce
from arg-cong[OF factor, of (*) q]
have q * factor = h * g
  by (subst g, auto simp: ac-simps)
hence smult d factor = h * g unfolding p h by auto
hence g dvd smult d factor by simp
from dvd-smult-int[OF d this]
have primitive-part g dvd factor .
from dvd-trans[OF this factor-f] ppf show False by auto
qed

```

lemma LLL-reconstruction-inner-sound:

```

assumes ret: LLL-reconstruction-inner p l gs f u Degs j = Some (gs',b',f',h)
shows f = f' * h (is ?g1)

```

and *irreducible_a h* (**is** ?g2)
and $b' = \text{lead-coeff } f'$ (**is** ?g3)
and *pl.unique-factorization-m* f' (*lead-coeff* f' , *mset* gs') (**is** ?g4)
and *p.dvdm* u h (**is** ?g5)
and *degree* $h = j'$ (**is** ?g6)
and *length* $gs' < \text{length } gs$ (**is** ?g7)
and *set* $gs' \subseteq \text{set } gs$ (**is** ?g8)
and $gs' \neq []$ (**is** ?g9)

proof –

let ?ppg = *primitive-part* g
note *ret* = *ret[unfolded LLL-reconstruction-inner-simps]*
from *ret* **have** *lc*: *abs* (*lead-coeff* g) $< p^{\wedge}ll$ **by** (*auto split: if-splits*)
from *ret* **obtain** *rest* **where** *rest*: *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*]: $h = ?ppg \text{ } gs' = \text{filter } (\lambda \text{ } gi. \neg \text{p.dvdm } gi \text{ } ?ppg) \text{ } gs$
 $f' = \text{rest } b' = \text{lead-coeff } \text{rest}$
and $f: f = ?ppg * \text{rest}$ **by** (*auto split: if-splits*)
with *div-int-then-rqp*[*OF rest*] **show** ?g1 ?g3 **by** *auto*
from (?g1) *f0* **have** *h0*: $h \neq 0$ **by** *auto*
let ?c = *content* g
from *g0* **have** *ct0*: $?c \neq 0$ **by** *auto*
have $|?c| \leq |\text{lead-coeff } g|$ **by** (*rule content-le-lead-coeff*)
also **have** $\dots < p^{\wedge}ll$ **by** *fact*
finally **have** *ct-pl*: $|?c| < p^{\wedge}ll$.
from *ug* **have** *pll.dvdm* u (*smult* ?c ?ppg) **by** *simp*
from *poly-mod-dvd-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: dvd-def*)
from f **have** *hf*: $h \text{ dvd } f$ **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*: ?g2
by (*intro factor-irreducible_{aI} degh hf puh, insert deg-g-j, simp*)
show *deg-h*: ?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*: $f = h * f'$ **and** $gs': gs' = [gi \leftarrow gs. \neg \text{p.dvdm } gi \text{ } h]$ **by** *auto*
note [*simp del*] = *out*
let ?fs = *filter* ($\lambda gi. \text{p.dvdm } gi \text{ } h$) gs
have *part*: *List.partition* ($\lambda gi. \text{p.dvdm } gi \text{ } h$) $gs = (?fs, gs')$
unfolding gs' **by** (*auto simp: o-def*)
from *p.unique-factorization-m-factor-partition*[*OF l0 f-gs fh cop sf part*]
show *uf*: *pl.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 irreducibled h have irr-f: irreducibled f
  using irreducibled-smult-int[OF c, of h] unfolding fch by auto
have degree f = j' using hf irr-h deg-h
  using irr-f  $\langle n \equiv degree f \rangle$  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 ij:  $\forall i \in \{d+1..j\}. LLL-reconstruction-inner p l gs f u Degs i = None$ 
  and dj:  $d < j$  and  $j \leq n$ 
shows  $\bigwedge factor. p.dvdm u factor \implies factor \text{ dvd } f \implies degree factor \geq j$ 
using assms
proof (induct j)
case (Suc j)
show ?case
proof (rule LLL-reconstruction-inner-complete)
  show  $\bigwedge factor2. p.dvdm u factor2 \implies factor2 \text{ dvd } f \implies j \leq degree factor2$ 
  proof (cases d = j)
    case False
      show  $\bigwedge factor2. p.dvdm u factor2 \implies factor2 \text{ dvd } f \implies j \leq degree factor2$ 
      by (rule Suc.hyps, insert Suc.premys False, auto)
    next
      case True
      then show  $\bigwedge factor2. p.dvdm u factor2 \implies factor2 \text{ dvd } f \implies j \leq degree$ 
factor2
      using degree-factor-ge-degree-u by auto
  qed
qed (insert Suc.premys, auto)
qed auto

```


corollary *LLL-reconstruction-inner-None-upt-j*:
assumes $ij: \forall i \in \{d+1..j\}. \text{LLL-reconstruction-inner } p \text{ l gs } f \text{ u } \text{Degs } i = \text{None}$
and $dj: d \leq j$ **and** $jn: j \leq n$
shows $\bigwedge \text{factor}. p.\text{dvd} m \text{ u } \text{factor} \implies \text{factor } \text{dvd} f \implies \text{degree } \text{factor} \geq j$
proof (*cases* $d=j$)
case *True*
then show $\bigwedge \text{factor}. p.\text{dvd} m \text{ u } \text{factor} \implies \text{factor } \text{dvd} f \implies d = j \implies j \leq \text{degree}$
factor
using *degree-factor-ge-degree-u* **by** *auto*
next
case *False*
hence $dj2: d < j$ **using** dj **by** *auto*
then show $\bigwedge \text{factor}. p.\text{dvd} m \text{ u } \text{factor} \implies \text{factor } \text{dvd} f \implies d \neq j \implies j \leq \text{degree}$
factor
using *LLL-reconstruction-inner-None-upt-j* [*OF* $ij \ dj2 \ jn$] **by** *auto*
qed

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

```

proof (rule LLL-reconstruction-inner-None-upt-j[OF - - - dvdp-u-factor factor-dvd-f])
  show  $d \leq \text{Suc } ?j$  using deg-factor1 by auto
  show  $\forall i \in \{d + 1..(\text{Suc } ?j)\}. \text{LLL-reconstruction-inner } p \ l \ g \ s \ f \ u \ \text{Degs } i =$ 
None
    using Suc-j i by auto
    show  $\text{Suc } ?j \leq n$  using Suc-j by simp
  qed
  then show ?thesis by auto
qed
qed

lemma irreducible-imp-LLL-reconstruction-inner-all-None:
  assumes irr-f: irreducibled f
  shows  $\forall i \in \{d+1..n\}. \text{LLL-reconstruction-inner } p \ l \ g \ s \ f \ u \ \text{Degs } i = \text{None}$ 
proof (rule ccontr)
  let ?LLL-inner =  $\lambda i. \text{LLL-reconstruction-inner } p \ l \ g \ s \ f \ u \ \text{Degs } i$ 
  let ?G =  $\{j. j \in \{d + 1..n\} \wedge ?\text{LLL-inner } j \neq \text{None}\}$ 
  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  $\forall i \in \{d+1..<j\}. ?\text{LLL-inner } i = \text{None}$ 
  proof (rule ccontr)
    assume  $\neg (\forall i \in \{d + 1..<j\}. ?\text{LLL-inner } i = \text{None})$ 
    from this obtain i where  $i: i \in \{d + 1..<j\}$  and LLL-i: ?LLL-inner i  $\neq$ 
None by auto
    hence iG:  $i \in ?G$  using j-def G-not-empty by auto
    have  $i < j$  using i by auto
    moreover have  $j \leq i$  using iG j-def by auto
    ultimately show False by linarith
  qed
  hence all-None:  $\forall i \in \{d+1..j-1\}. ?\text{LLL-inner } i = \text{None}$  by auto
  obtain gs' b' f' factor where LLL-inner-eq: ?LLL-inner j = Some (gs', b', f',
factor)
    using LLL-not-None by force
  have Suc-j1-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:  $\bigwedge \text{factor}. p.\text{dvd} \ u \ \text{factor} \implies \text{factor } \text{dvd } f \implies j - 1 \leq \text{degree}$ 
factor
    by (rule LLL-reconstruction-inner-None-upt-j[OF all-None dj], insert jn, auto)

  have LLL-inner-Some: ?LLL-inner (Suc (j - 1)) = Some (gs', b', f', factor)
    using LLL-inner-eq Suc-j1-eq by auto
  have deg-factor: degree factor = j - 1
    and ff':  $f = f' * \text{factor}$ 
    and irreducible-factor: irreducibled factor

```

```

    using LLL-reconstruction-inner-sound[OF dj jn degree LLL-inner-Some] by
(metis+)
    have degree f' = n - (j - 1) using arg-cong[OF ff', of degree]
      by (subst (asm) degree-mult-eq, insert f0 ff' deg-factor, auto)
    also have ... < n using irreducible-factor jn unfolding irreducibled-def deg-factor
by auto
    finally have deg-f': degree f' < degree f by auto
    from ff' have factor-dvd-f: factor dvd f by auto
    have ¬ irreducibled f
      by (rule reducibledI, 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: ∀ i ∈ {d+1..n}. LLL-reconstruction-inner p l gs f u Degr i = None
    and dj: d < j
shows LLL-reconstruction-inner-loop p l gs f u Degr j = ([], 1, 1, f)
    using dj
proof (induct j rule: LLL-reconstruction-inner-loop.induct[of f p l gs u Degr])
  case (1 j)
  let ?innerl = LLL-reconstruction-inner-loop p l gs f u Degr
  let ?inner = LLL-reconstruction-inner p l gs f u Degr
  note hyp = 1.hyps
  note dj = 1.prem1
  show ?case
  proof (cases j ≤ n)
    case True note jn = True
    have step: ?inner j = None
      by (cases d=j, insert i jn dj, auto)
    have ?innerl j = ?innerl (j+1)
      using jn step by auto
    also have ... = ([], 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: irreducibled f and dj: d < j
shows LLL-reconstruction-inner-loop p l gs f u Degr 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-Some*:

```

    assumes inner-loop: LLL-reconstruction-inner-loop p l gs f u Degr j = (gs', b', f', factor)
    and i: ∀ i ∈ {d+1..<j}. LLL-reconstruction-inner p l gs f u Degr i = None
    and dj: d < j and jn: j ≤ n and f: ¬ irreducibled f
shows ∃ j'. j ≤ j' ∧ j' ≤ n ∧ d < j'

```

```

    ∧ (LLL-reconstruction-inner p l gs f u Degr j' = Some (gs', b', f', factor))
    ∧ (∀ i ∈ {d+1..<j'}. LLL-reconstruction-inner p l gs f u Degr i = None)
  using inner-loop i dj jn
proof (induct j rule: LLL-reconstruction-inner-loop.induct[of f p l gs u Degr])
  case (1 j)
  let ?innerl = LLL-reconstruction-inner-loop p l gs f u Degr
  let ?inner = LLL-reconstruction-inner p l gs f u Degr
  note hyp = 1.hyps
  note 1 = 1.prem1(1)
  note 2 = 1.prem1(2)
  note dj = 1.prem1(3)
  note jn = 1.prem1(4)
  show ?case
proof (cases ?inner j = None)
  case True
  show ?thesis
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 irreducibled 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 dn 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

```

```

      le-neq-implies-less less-Suc-eq-le)
    show  $d < j + 1$  using  $dj$  by auto
    show  $j + 1 \leq n$  using  $jn$  False by auto
  qed
  from this obtain  $j'$  where  $a1: j' \geq j + 1$  and  $a2: j' \leq n$  and  $a3: d < j'$ 
    and  $a4: ?inner\ j' = \text{Some}\ (gs', b', f', \text{factor})$ 
    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  $?thesis$  by fastforce
  qed
next
  case False
  have 1:  $?inner\ j = \text{Some}\ (gs', b', f', \text{factor})$ 
    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 le-neq-implies-less by fastforce
  ultimately show  $?thesis$  by auto
  qed
qed

```

```

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

```

```

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

```

```

shows  $f = f' * \text{factor}$  (is ?g1)
  and  $\text{irreducible}_d \text{ factor}$  (is ?g2)
  and  $b' = \text{lead-coeff } f'$  (is ?g3)
  and  $\text{pl.unique-factorization-m } f' (b', \text{mset } gs')$  (is ?g4)
  and  $p.\text{dvdm } u \text{ factor}$  (is ?g5)
  and  $gs \neq [] \rightarrow \text{length } gs' < \text{length } gs$  (is ?g6)
  and  $\text{factor } \text{dvd } f$  (is ?g7)
  and  $f' \text{ dvd } f$  (is ?g8)
  and  $\text{set } gs' \subseteq \text{set } gs$  (is ?g9)
  and  $gs' = [] \rightarrow f' = 1$  (is ?g10)
using  $\text{ret } ij \text{ jn } dj$ 
proof (atomize(full), induct j)
  case 0
  then show ?case using deg-u by auto
next
  case (Suc j)
  let ?innerl = LLL-reconstruction-inner-loop p l gs f u Degs
  let ?inner = LLL-reconstruction-inner p l gs f u Degs
  have  $ij: \forall i \in \{d+1..j\}. ?inner \ i = \text{None}$ 
    using Suc.prem by auto
  have  $dj: d \leq j$  using Suc.prem by auto
  have  $jn: j < n$  using Suc.prem by auto
  have  $\text{deg}: \text{Suc } j \leq \text{degree } f$  using Suc.prem by auto
  have  $c: \bigwedge \text{factor}. p.\text{dvdm } u \text{ factor} \implies \text{factor } \text{dvd } f \implies j \leq \text{degree } \text{factor}$ 
    by (rule LLL-reconstruction-inner-None-upt-j[OF ij dj], insert n jn, auto)
  have  $1: ?innerl \ (\text{Suc } j) = (gs', b', f', \text{factor})$ 
    using Suc.prem by auto
  show ?case
  proof (cases ?inner (Suc j) = None)
    case False
    have  $\text{LLL-rw}: ?inner \ (\text{Suc } j) = \text{Some } (gs', b', f', \text{factor})$ 
      using False deg Suc.prem by auto
    show ?thesis using LLL-reconstruction-inner-sound[OF dj jn c 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.prem False by auto
    moreover have  $dj2: d < j$  using Suc.prem False by auto
    ultimately show ?thesis using Suc.prem Suc.hyps by fastforce
  next
  case True note d-eq-j = True
  show ?thesis
  proof (cases  $\text{irreducible}_d f$ )
    case True
    have  $\text{pl-Mp-1}: \text{pl.Mp } 1 = 1$  by auto
    have  $d\text{-Suc-}j: d < \text{Suc } j$  using Suc.prem by auto

```

```

have ?innerl (Suc j) = ([],1,1,f)
by (rule irreducible-imp-LLL-reconstruction-inner-loop-f[OF True d-Suc-j])
hence result-eq: ([],1,1,f) = (gs', b', f', factor) using Suc.prem by auto
moreover have thesis1: p.dvdm u factor using u-f result-eq by auto
moreover have thesis2: f' = pl.Mp (Polynomial.smult b' (prod-list gs'))
using result-eq pl-Mp-1 by auto
ultimately show ?thesis using True by (auto simp: unique-factorization-m-1)
next
case False note irreducible-f = False
have  $\exists j'. \text{Suc } j \leq j' \wedge j' \leq n \wedge d < j'$ 
 $\wedge (\text{?inner } j' = \text{Some } (gs', b', f', \text{factor}))$ 
 $\wedge (\forall i \in \{d+1..<j'\}. \text{?inner } i = \text{None})$ 
proof (rule exists-index-LLL-reconstruction-inner-Some[OF - - - - False])

  show ?innerl (Suc j) = (gs', b', f', factor)
  using Suc.prem by auto
  show  $\forall i \in \{d + 1..<\text{Suc } j\}. \text{?inner } i = \text{None}$ 
  using Suc.prem by auto
  show  $\text{Suc } j \leq n$  using jn by auto
  show  $d < \text{Suc } j$  using Suc.prem by auto
qed
from this obtain a where da:  $d < a$  and an:  $a \leq n$  and ja:  $j \leq a$ 
and a1: ?inner a = Some (gs', b', f', factor)
and a2:  $\forall i \in \{d+1..<a\}. \text{?inner } i = \text{None}$  by auto
define j' where j'[simp]:  $j' \equiv a - 1$ 
have dj':  $d \leq j'$  using da by auto
have j':  $j' \neq 0$  using dj' d0 by auto
hence j'n:  $j' < n$  using an by auto
have LLL: ?inner (Suc j') = Some (gs', b', f', 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:  $\bigwedge \text{factor}. p.\text{dvdm } u \text{ factor} \implies \text{factor } \text{dvd } f \implies \text{Suc } (j' - 1) \leq$ 
degree factor
by (rule LLL-reconstruction-inner-None-upt-j, insert dj' Suc-rw j'n
prev-None, auto)
hence c2:  $\bigwedge \text{factor}. p.\text{dvdm } u \text{ factor} \implies \text{factor } \text{dvd } f \implies j' \leq \text{degree factor}$ 
using j' by force
show ?thesis using LLL-reconstruction-inner-sound[OF dj' j'n c2 LLL] by
fastforce
qed
qed
qed
qed
lemma LLL-reconstruction-inner-loop-j-ge-n:
assumes ret: LLL-reconstruction-inner-loop p l gs f u Degrs j = (gs',b',f',factor)
and ij:  $\forall i \in \{d+1..n\}. \text{LLL-reconstruction-inner } p \text{ l } gs \text{ f } u \text{ Degrs } i = \text{None}$ 

```

and $dj: d < j$
and $jn: j > n$
shows $f = f' * \text{factor}$ (**is** ?g1)
and $\text{irreducible}_d \text{ factor}$ (**is** ?g2)
and $b' = \text{lead-coeff } f'$ (**is** ?g3)
and $\text{pl.unique-factorization-m } f' (b', \text{mset } gs')$ (**is** ?g4)
and $\text{p.dvdm } u \text{ factor}$ (**is** ?g5)
and $gs \neq [] \longrightarrow \text{length } gs' < \text{length } gs$ (**is** ?g6)
and $\text{factor } \text{dvd } f$ (**is** ?g7)
and $f' \text{ dvd } f$ (**is** ?g8)
and $\text{set } gs' \subseteq \text{set } gs$ (**is** ?g9)
and $f' = 1$ (**is** ?g10)
proof –
have $\text{LLL-reconstruction-inner-loop } p \ l \ gs \ f \ u \ \text{Degs } j = ([], 1, 1, f)$ **using** jn **by**
auto
hence $gs': gs' = []$ **and** $b': b' = 1$ **and** $f': f' = 1$ **and** $\text{factor}: \text{factor} = f$ **using** *ret*
by *auto*
have $\text{irreducible}_d f$
by (*rule LLL-reconstruction-inner-all-None-imp-irreducible[OF ij]*)
thus ?g1 ?g2 ?g3 ?g4 ?g5 ?g6 ?g7 ?g8 ?g9 ?g10 **using** $f' \text{ factor } b' \ gs' \ u \text{-f}$
by (*auto simp: unique-factorization-m-1*)
qed

lemma *LLL-reconstruction-inner-loop*:

assumes *ret: LLL-reconstruction-inner-loop* $p \ l \ gs \ f \ u \ \text{Degs } j = (gs', b', f', \text{factor})$
and $ij: \forall i \in \{d+1..<j\}. \text{LLL-reconstruction-inner } p \ l \ gs \ f \ u \ \text{Degs } i = \text{None}$
and $n: n = \text{degree } f$
and $dj: d < j$
shows $f = f' * \text{factor}$ (**is** ?g1)
and $\text{irreducible}_d \text{ factor}$ (**is** ?g2)
and $b' = \text{lead-coeff } f'$ (**is** ?g3)
and $\text{pl.unique-factorization-m } f' (b', \text{mset } gs')$ (**is** ?g4)
and $\text{p.dvdm } u \text{ factor}$ (**is** ?g5)
and $gs \neq [] \longrightarrow \text{length } gs' < \text{length } gs$ (**is** ?g6)
and $\text{factor } \text{dvd } f$ (**is** ?g7)
and $f' \text{ dvd } f$ (**is** ?g8)
and $\text{set } gs' \subseteq \text{set } gs$ (**is** ?g9)
and $gs' = [] \longrightarrow f' = 1$ (**is** ?g10)
proof (*atomize(full), (cases j > n; intro conjI)*)
case *True*
have $ij2: \forall i \in \{d + 1..n\}. \text{LLL-reconstruction-inner } p \ l \ gs \ f \ u \ \text{Degs } i = \text{None}$
using ij *True* **by** *auto*
show ?g1 ?g2 ?g3 ?g4 ?g5 ?g6 ?g7 ?g8 ?g9 ?g10
using *LLL-reconstruction-inner-loop-j-ge-n*[*OF ret ij2 dj True*] **by** *blast+*
next
case *False*
hence $jn: j \leq n$ **by** *simp*
show ?g1 ?g2 ?g3 ?g4 ?g5 ?g6 ?g7 ?g8 ?g9 ?g10
using *LLL-reconstruction-inner-loop-j-le-n*[*OF ret ij n jn dj*] **by** *blast+*

qed
end

7.3.3 Outer loop

lemma *LLL-reconstruction''*:

assumes 1: *LLL-reconstruction''* p l gs b f G = G'
and *irreducible-G*: $\bigwedge \text{factor. factor} \in \text{set } G \implies \text{irreducible}_d \text{ factor}$
and 3: $F = f * \text{prod-list } G$
and 4: *pl.unique-factorization-m* f (lead-coeff f, mset gs)
and 5: $gs \neq []$
and 6: $\bigwedge gi. gi \in \text{set } gs \implies \text{pl.Mp } gi = gi$
and 7: $\bigwedge gi. gi \in \text{set } gs \implies \text{p.irreducible}_d\text{-m } gi$
and 8: *p.square-free-m* f
and 9: *coprime* (lead-coeff f) p
and *sf-F*: *square-free* F
shows $(\forall g \in \text{set } G'. \text{irreducible}_d g) \wedge F = \text{prod-list } G'$
using 1 *irreducible-G* 3 4 5 6 7 8 9
proof (*induction* gs *arbitrary*: b f G G' *rule*: *length-induct*)
case (1 gs)
note *LLL-f'* = 1.premis(1)
note *irreducible-G* = 1.premis(2)
note *F-f-G* = 1.premis(3)
note *f-gs-factor* = 1.premis(4)
note *gs-not-empty* = 1.premis(5)
note *norm* = 1.premis(6)
note *irred-p* = 1.premis(7)
note *sf* = 1.premis(8)
note *cop* = 1.premis(9)
obtain u **where** *choose-u-result*: *choose-u* gs = u **by** *auto*
from *choose-u-member*[OF *gs-not-empty*, *unfolded choose-u-result*]
have u-gs: u \in set gs **by** *auto*
define d n **where** [*simp*]: d = degree u n = degree f
hence n-def: n = degree f n \equiv degree f **by** *auto*
define gs'' **where** gs'' = remove1 u gs
define degs **where** degs = map degree gs''
define Degs **where** Degs = (+) d ' *sub-mset-sums* degs
obtain gs' b' h factor **where** *inner-loop-result*:
LLL-reconstruction-inner-loop p l gs f u Degs (d+1) = (gs', b', h, factor)
by (*metis prod-cases4*)
have a1:
LLL-reconstruction-inner-loop p l gs f u Degs (d+1) = (gs', b', h, factor)
using *inner-loop-result* **by** *auto*
have a2:
 $\forall i \in \{\text{degree } u + 1 .. (d+1)\}. \text{LLL-reconstruction-inner } p \text{ l } gs \text{ f } u \text{ Degs } i = \text{None}$
by *auto*
have *LLL-reconstruction''* p l gs b f G = *LLL-reconstruction''* p l gs' b' h (factor # G)
unfolding *LLL-reconstruction''*.*simps*[of p l gs] **using** *gs-not-empty*

```

unfolding Let-def using choose-u-result inner-loop-result unfolding Degs-def
deg-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 (smult (lead-coeff f) (prod-mset (mset gs))) and
mon: g ∈ set gs ⇒ monic g and irred: g ∈ set gs ⇒ pl.irreducibled-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.dvdm u f unfolding pl.dvdm-def
by (intro exI[of - smult (lead-coeff f) (prod-mset (mset (gs1 @ gs2))]), auto)
} note pl-uf = this
hence p-uf: p.dvdm u f by (rule pl-dvdm-imp-p-dvdm)
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-dvd-F: f dvd F by auto
from square-free-factor[OF f-dvd-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.dvdm 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.dvdm 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 = [gi ← gs . p.dvdm 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 ∈ Degs unfolding Degs-def by auto
} note Degs = this
have length-less: length gs' < length gs
  and irreducible-factor: irreducibled 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 Degs
  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-f-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: ∧ x. x ∈ set (factor # G) ⇒ irreducibled 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 lc': lead-coeff (smult (lead-coeff h) (prod-list gs')) = lead-coeff h
  by (insert gs'-gs, auto intro!: monic-prod-list intro: mon)
have lc: lead-coeff (pl.Mp (smult (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
lc')
    show lead-coeff h mod p ^ l ≠ 0 using p.coprime-exp-mod[OF cop' l0] by auto
  qed auto
  have uh: pl.unique-factorization-m h (lead-coeff h, mset gs') using h-eq unfold-
ing 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

```

```

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 irred-p sf' cop ], insert gs'-gs, auto)
next
  case False
  have pl-ge0:  $p^l > 0$  using p1 by auto
  have G'-eq:  $G' = \text{factor} \# G$  using LLL-eq False using LLL-reconstruction''.simps
by auto
  have condition1:  $(\forall a \in \text{set } G'. \text{irreducible}_d a)$  using irreducible-factor-G G'-eq
by auto
  have h-eq2:  $pl.Mp h = pl.Mp [:b':]$  using h-eq False
  unfolding pl.unique-factorization-m-alt-def pl.factorization-m-def by auto
  have Mp-const-rw[simp]:  $pl.Mp [:b':] = [:b' \bmod p^l:]$  using 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
qed

```

```

context
  fixes gs :: 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:  $mset fs = \{\#\}$  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:  $p.eq-m F [:c:]$  unfolding fs p.factorization-m-def by auto
  hence  $0 = p.degree-m F$  by (simp add: p.Mp-const-poly)
  also have ... = degree F by (rule p.degree-m-eq[OF - p1], insert cop p1, auto)
  finally have degree F = 0 ..
  thus False using N0 by simp
qed

```

```

lemma reconstruction-of-algorithm-16-22:

```

```

assumes 1: reconstruction-of-algorithm-16-22 p l gs F = G
shows ( $\forall g \in \text{set } G. \text{irreducible}_d g$ )  $\wedge$  F = prod-list G
proof -
  note * = p.berlekamp-hensel-unique[OF cop sf gs-hen l0]
  obtain c fs where finite-field-factorization-int p F = (c, 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
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 ( $\forall g \in \text{set } G. \text{irreducible}_d g$ )  $\wedge$  f = prod-list G
proof -
  let ?lc = lead-coeff f
  define p where p  $\equiv$  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 *: N2  $\leq$  (p^n)2

```

```

    by (intro power-mono N0, auto)
  show  $2^{\text{degree } f^2} * B2\text{-LLL } f^{\text{degree } f} \leq (p^{\text{degree } f})^2$ 
  proof (rule order.trans[OF - *])
    have  $2^{\text{degree } f^2} * B2\text{-LLL } f^{\text{degree } f} = K$ 
      unfolding K-def B2-LLL-def by (simp add: ac-simps
        power-mult-distrib power2-eq-square power-mult[symmetric] power-add[symmetric])
    also have  $\dots \leq N^2$  unfolding N-def by (rule sqrt-int-ceiling-bound[OF K0])
    finally show  $2^{\text{degree } f^2} * B2\text{-LLL } f^{\text{degree } f} \leq N^2$  .
  qed
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  $\Rightarrow$  int poly  $\Rightarrow$  int poly
  where max-degree-poly a b = (if degree a  $\geq$  degree b then a else b)

```

```

fun choose-u :: int poly list  $\Rightarrow$  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 = choose-u gs;
d = degree u;
g-star = [:2,2,0,2 :: int :];
(gs',hs') = List.partition ( $\lambda gi. \text{poly-mod.dvdm } p \text{ } gi \text{ } g\text{-star}$ ) gs;
h-star = smult b (prod-list hs');
f-star = primitive-part h-star
in (hs' = []  $\wedge$  f-star = 1) by eval

```

8.2 Another potential problem of Algorithm 16.22

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

definition *input-poly* = [: 1,0,0,0,1,1,0,0,1,0,1,0,1 :: int :]

For *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.

lemma *let f = input-poly;*
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 = choose-u gs;
pl = p^l;
pl2 = pl div 2;
u' = poly-mod.inv-Mp2 pl pl2 (poly-mod.Mp pl (smult b u))
*in sqrt-int-floor (sq-norm u') > pl **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.

definition *compute-norms = (let f = input-poly;*
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 = choose-u gs;
pl = p^l;
pl2 = pl div 2;
u' = poly-mod.inv-Mp2 pl pl2 (poly-mod.Mp pl (smult b u));
d = degree u;

```

pl = p ^ l;
L = factorization-lattice u' 1 pl;
g-star = short-vector 2 L
in (
  "p ^ l:      " @ show pl @ shows-nl [] @
  "norm u:    " @ show (sqrt-int-floor (sq-norm-poly u ^)) @ shows-nl [] @
  "norm g-star: " @ show (sqrt-int-floor (sq-norm-vec g-star)) @ shows-nl [] @
shows-nl []
))

```

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

- $p^l \approx 6.61056 \cdot 10^{122}$, namely 66105596879024859895191530803277103982840468296428121928464
- $\text{norm } u \approx 6.67555 \cdot 10^{122}$, namely 667555058938127908386141559707490406617756492853269306
- $\text{norm } g\text{-star} \approx 5.02568 \cdot 10^{110}$, namely 50256787188889378925810759939795033899734873138630

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:];
        rhs = [-1728,-840,-420,6:];
in ¬ poly-mod.eq-m (5 ^ 6) (smult 6 (g2*g3*g4)) (rhs) by eval

```

end

References

- [1] J. Divasón, S. J. C. Joosten, R. Thiemann, and A. Yamada. A formalization of the Berlekamp–Zassenhaus factorization algorithm. In *CPP 2017*, pages 17–29. ACM, 2017.
- [2] J. v. z. Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2nd edition, 2003.
- [3] A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.