# A verified factorization algorithm for integer polynomials with polynomial complexity* 

Jose Divasón Sebastiaan Joosten René Thiemann<br>Akihisa Yamada

September 13, 2023


#### Abstract

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 The work is based on our previous Berlekamp-Zassenhaus development, where the exponential reconstruction phase has been replaced by the polynomial-time basis reduction algorithm.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


## Contents

1 Introduction 2
2 Factor bound 5
3 Executable dvdm operation 5
3.1 Uniqueness of division algorithm for polynomials . . . . . . . 6
3.2 Executable division operation modulo $m$ for polynomials . . . 7

4 The LLL factorization algorithm 8
5 Correctness of the LLL factorization algorithm 10
5.1 Basic facts about the auxiliary functions . . . . . . . . . . . . 10
5.2 Facts about Sylvester matrices and norms . . . . . . . . . . . 11

[^0]5.3 Proof of the key lemma 16.20 ..... 12
5.4 Properties of the computed lattice and its connection with Sylvester matrices ..... 12
5.5 Proving that factorization-lattice returns a basis of the lattice ..... 13
5.6 Being in the lattice is being a multiple modulo ..... 13
5.7 Soundness of the LLL factorization algorithm ..... 14
6 Calculating All Possible Sums of Sub-Multisets ..... 16
7 Implementation and soundness of a modified version of Al- gorithm 16.22 ..... 17
7.1 Previous lemmas obtained using local type definitions ..... 17
7.2 The modified version of Algorithm 16.22 ..... 17
7.3 Soundness proof ..... 20
7.3.1 Starting the proof ..... 20
7.3.2 Inner loop ..... 22
7.3.3 Outer loop ..... 25
7.3.4 Final statement ..... 26
8 Mistakes in the textbook Modern Computer Algebra (2nd edition) ..... 26
8.1 A real problem of Algorithm 16.22 ..... 27
8.2 Another potential problem of Algorithm 16.22 ..... 27
8.3 Verified wrong results ..... 28

## 1 Introduction

In order to factor an integer polynomial $f$, we may assume a modular factorization of $f$ into several monic factors $u_{i}: f \equiv \operatorname{lc}(f) \cdot \prod_{i} u_{i}$ 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 $h=g c d(f, g)$ 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 $0<\operatorname{degree}\left(f_{1}\right)<n$. 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 $\operatorname{degree}(g)<n$. The short vector requirement will provide an upper bound to satisfy the assumption $\|f\|^{\text {degree }(g)} \cdot\|g\|^{\text {degree }(f)}<m$.

$$
\begin{gather*}
\|g\| \leq 2^{(n-1) / 2} \cdot\left\|f_{1}\right\| \leq 2^{(n-1) / 2} \cdot 2^{n-1}\|f\|=2^{3(n-1) / 2}\|f\|  \tag{1}\\
\|f\|^{\text {degree }(g)} \cdot\|g\|^{\text {degree }(f)} \leq\|f\|^{n-1} \cdot\left(2^{3(n-1) / 2}\|f\|\right)^{n}=\|f\|^{2 n-1} \cdot 2^{3 n(n-1) / 2} \tag{2}
\end{gather*}
$$

Here, the first inequality in (1) is the short vector approximation $\left(f_{1} \in L_{u, k}\right)$. 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 $m=p^{l}>\|f\|^{2 n-1} \cdot 2^{3 n(n-1) / 2}$ then all preconditions of Lemma 1 are satisfied, and $h=g c d(f, g)$ 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 $i r$ reducible factors [2, Algorithm 16.22]. We initially implemented and tried to verify this pseudo-code algorithm (see files Factorization_Algorithm_16_22.thy and Modern_Computer_Algebra_Problem.thy). After some work, we had only one remaining goal to prove: the content of the polynomial $g$ corresponding to the short vector is not divisible by the chosen prime $p$. However, we were unable to figure out how to discharge this goal and then also started to search for inputs where the algorithm delivers wrong results. After a while we realized that Algorithm 16.22 indeed has a serious flaw as demonstrated in the upcoming example.

Example 1 Consider the square-free and content-free polynomial $f=(1+$ $x) \cdot\left(1+x+x^{3}\right)$. 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 $\bmod p$ 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 $(1,1,0,1),\left(p^{l}, 0,0,0\right),\left(0, p^{l}, 0,0\right)$, ( $0,0, p^{l}, 0$ ).
- 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$)$.

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

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 $g^{*} \bmod p$. Thus in step 9 we compute $S=T, h=1$, enter the then-branch and update $T=\emptyset$, $G=G \cup\left\{1+x+x^{3}\right\}, f^{*}=1, b=1$.
- Then in step 10 we update $G=\left\{1+x+x^{3}, 1\right\}$ and finally return that the factorization of $f$ is $\left(1+x+x^{3}\right) \cdot 1$.

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 \(L L L\) factorization \(f=g s\)
shows \(f=\) prod_list \(g s\)
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 the slightly modified version. It can be seen as an implementation of the pseudo-code factorization algorithm given by Lenstra, Lenstra, and Lovász [3].

## 2 Factor bound

This theory extends the work about factor bounds which was carried out in the Berlekamp-Zassenhaus development.
theory Factor-Bound-2
imports Berlekamp-Zassenhaus.Factor-Bound
LLL-Basis-Reduction.Norms
begin
lemma norm-1-bound-mignotte: norm1 $f \leq \mathcal{L}^{\wedge}($ degree $f) *$ mahler-measure $f$〈proof〉
lemma mahler-measure-l2norm: mahler-measure $f \leq \operatorname{sqrt}\left(\right.$ of-int $\left.\|f\|^{2}\right)$
$\langle p r o o f\rangle$
lemma sq-norm-factor-bound:
fixes $f h$ :: int poly
assumes dvd: $h$ dvd $f$ and $f 0: f \neq 0$
shows $\|h\|^{2} \leq 2^{\wedge}(2 *$ degree $h) *\|f\|^{2}$
$\langle p r o o f\rangle$
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<br>Berlekamp-Zassenhaus.Poly-Mod-Finite-Field<br>Berlekamp-Zassenhaus.Polynomial-Record-Based<br>Berlekamp-Zassenhaus.Hensel-Lifting<br>Subresultants.Subresultant<br>Perron-Frobenius.Cancel-Card-Constraint<br>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 $g 1: g=q * u+$ smult $m r$
and $q: q \neq 0$ and $m$-not0: $m \neq 0$
shows degree $q<j-d$
$\langle p r o o f\rangle$

### 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\not=0
        and r1:r1 = 0 \vee degree r1< degree g
        and r2: r2 = 0\vee degree r2 < degree g
    shows q1 = q2 ^r1 = r2
\langleproof\rangle
lemma pdivmod-eq-pdivmod-monic:
    assumes g: monic g
    shows pdivmod fg= pdivmod-monic f g
<proof\rangle
context poly-mod
begin
definition pdivmod2 fg=( if Mpg=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) (Mph)
    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 FG)=0)
<proof\rangle
lemma of-int-poly-Mp-O[simp]:(of-int-poly (Mp a) = (0:: 'a mod-ring poly)) =
(Mpa=0)
    \langleproof\rangle
lemma uniqueness-algorithm-division-of-int-poly:
    assumes g0:Mpg\not=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+
    and R:R=0\vee degree }R<\mathrm{ degree }
    and Mp-f:Mpf=Mpg*q+r
    and r:r=0\vee degree r<degree (Mp g)
shows }Q=of-int-poly q\wedgeR=of-int-poly 
\langleproof\rangle
```

```
corollary uniqueness-algorithm-division-to-int-poly:
    assumes g0: Mp g\not=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+
    and R:R=0\vee degree }R<\mathrm{ degree }
    and Mp-f:Mpf=Mpg*q+r
    and }r:r=0\vee\mathrm{ degree }r<\mathrm{ degree (Mp g)
    shows Mpq= to-int-poly Q}\wedgeMpr=to-int-poly 
    <proof\rangle
lemma uniqueness-algorithm-division-Mp-Rel:
    assumes monic-Mpg: monic (Mpg)
        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 (Mpf) (Mpg) = (q,r)
    and QR: pseudo-divmod F G = (Q,R)
shows MP-Rel q Q ^MP-Rel r R
\langleproof\rangle
definition MP-Rel-Pair A B\equiv(let (a,b)=A;(c,d)=B in MP-Rel a c\wedge MP-Rel
b d)
lemma pdivmod2-rel[transfer-rule]:
    (MP-Rel ===> MP-Rel ===> MP-Rel-Pair) (pdivmod2) (pdivmod)
<proof\rangle
```


### 3.2 Executable division operation modulo $m$ for polynomials

lemma dvdm-iff-Mp-pdivmod2:
shows $g d v d m f=(M p($ snd $($ pdivmod2 $f g))=0)$
$\langle p r o o f\rangle$
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:
$g d v d m f=($ if prime $m$ then $M p($ snd $($ pdivmod2 $f g))=0$
else Code.abort (STR "dvdm error: $m$ is not a prime number") $(\lambda-. g d v d m f))$
$\langle p r o o f\rangle$
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<br>imports LLL-Basis-Reduction.LLL-Certification<br>Factor-Bound-2<br>Missing-Dvd-Int-Poly<br>Berlekamp-Zassenhaus.Berlekamp-Zassenhaus<br>begin<br>hide-const (open) up-ring.coeff up-ring.monom<br>Unique-Factorization.factors Divisibility.factors<br>Unique-Factorization.factor Divisibility.factor<br>Divisibility.prime

definition factorization-lattice where factorization-lattice u $k \mathrm{~m} \equiv$ map ( $\lambda$ i. vec-of-poly-n $(u *$ monom $1 i)($ degree $u+k))[k>. .0] @$ map $(\lambda i$. vec-of-poly-n (monom $m i)($ degree $u+k)$ ) [degree $u>. .0]$
fun min-degree-poly :: int poly $\Rightarrow$ int poly $\Rightarrow$ int poly
where min-degree-poly a $b=($ if degree $a \leq$ degree $b$ then $a$ else $b)$
fun choose- $u$ :: int poly list $\Rightarrow$ int poly
where choose-u [] = undefined
| choose-u $[g i]=g i$
| choose-u (gi \# gj \# gs) = min-degree-poly gi (choose-u (gj \# gs))

```
lemma factorization-lattice-code[code]: factorization-lattice u \(k m=\) (
    let \(n=\) degree \(u\) in
    map
    ( \(\lambda i\). vec-of-poly-n (monom-mult \(i u)(n+k))[k>. .0]\)
    @ map ( \(\lambda\) i. vec-of-poly-n (monom mi) \((n+k)\) ) [ \(n>. .0]\)
) \(\langle p r o o f\rangle\)
```

Optimization: directly try to minimize coefficients of polynomial $u$.

## definition LLL-short-polynomial where

LLL-short-polynomial pl $n u=$ poly-of-vec (short-vector-hybrid 2 (factorization-lattice (poly-mod.inv-Mp pl (poly-mod.Mp pl u)) (n - degree u) pl))
locale $L L L$-implementation $=$
fixes $p p l::$ int

```
begin
function LLL-many-reconstruction where
    LLL-many-reconstruction fus=(let
        d = degree f;
        d2 = d div 2;
        f2-opt = find-map-filter
            (\lambdau.gcd f(LLL-short-polynomial pl (Suc d2) u))
            (\lambda f2. let deg = degree f2 in deg > 0 ^ deg <d)
            (filter ( }\lambdau\mathrm{ . degree }u\leqd2)us
    in case f2-opt of None }=>[f
    | Some f2 = let f1 = f div f2;
                (us1,us2) = List.partition ( }\lambda\mathrm{ gi. poly-mod.dvdm p gi f1) us
                in LLL-many-reconstruction f1 us1 @ LLL-many-reconstruction f2 us2)
    <proof\rangle
termination
\langleproof\rangle
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 fg;
        deg = degree f2
        in if deg = 0\vee deg \geqd then [f]
            else let f1 = f div f2;
            (us1,us2) = List.partition ( }\lambda\mathrm{ gi. poly-mod.dvdm p gi f1) us
            in LLL-reconstruction f1 us1 @ LLL-reconstruction f2 us2)
    <proof\rangle
```


## termination

```
〈proof〉
end
declare LLL-implementation.LLL-reconstruction.simps[code]
declare LLL-implementation.LLL-many-reconstruction.simps[code]
definition \(L L L\)-factorization :: int poly \(\Rightarrow\) int poly list where
LLL-factorization \(f=(\) let
- find suitable prime
\(p=\) suitable-prime-bz \(f\);
- compute finite field factorization
\((-, f s)=\) finite-field-factorization-int \(p f\);
- determine exponent 1 and \(B\)
\(n=\) degree \(f\);
no \(=\|f\|^{2}\);
\(B=\operatorname{sqrt-int-ceiling}(2 \uparrow(5 *(n-1) *(n-1)) * n o \wedge(2 *(n-1))) ;\)
```

```
        \(l=\) find-exponent \(p B\);
        - perform hensel lifting to lift factorization to \(\bmod p^{l}\)
        \(u s=\) hensel-lifting plffs;
        - reconstruct integer factors via LLL algorithm
        \(p l=p^{\wedge} l\)
    in LLL-implementation.LLL-reconstruction p plfus)
definition LLL-many-factorization \(::\) int poly \(\Rightarrow\) int poly list where
    LLL-many-factorization \(f=\) (let
        - find suitable prime
        \(p=\) suitable-prime-bz \(f\);
        - compute finite field factorization
        \((-, f s)=\) finite-field-factorization-int \(p f\);
        - determine exponent l and B
        \(n=\) degree \(f\);
        \(n o=\|f\|^{2}\);
        \(B=\operatorname{sqrt-int-ceiling}\left(\mathfrak{2}^{\wedge}(5 *(n \operatorname{div} 2) *(n \operatorname{div} 2)) * n o \wedge(2 *(n \operatorname{div} 2))\right) ;\)
        \(l=\) find-exponent \(p B\);
        - perform hensel lifting to lift factorization to \(\bmod p^{l}\)
        \(u s=\) hensel-lifting plffs;
        - reconstruct integer factors via LLL algorithm
        \(p l=p^{\wedge} l\)
    in LLL-implementation.LLL-many-reconstruction p plfus)
```

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 $n \equiv$ degree $u$
assumes $i<n+d$
shows factorization-lattice $u d m!i=$
vec-of-poly-n (if $i<d$ then $u *$ monom $1(d-S u c i)$ else monom $m(n+d-S u c$
i)) $(n+d)$

```
    <proof\rangle
lemma length-factorization-lattice[simp]:
    shows length (factorization-lattice u d m) = degree }u+
    <proof\rangle
lemma dim-factorization-lattice:
    assumes }x<\mathrm{ degree }u+
    shows dim-vec (factorization-lattice u d m!x)= degree u+d
    <proof\rangle
lemma dim-factorization-lattice-element:
    assumes }x\in\mathrm{ set (factorization-lattice }udm\mathrm{ ) shows dim-vec }x=\mathrm{ degree }u+
    <proof\rangle
lemma set-factorization-lattice-in-carrier[simp]: set (factorization-lattice u d m)
\subseteq \text { carrier-vec (degree u+d)}
    <proof>
lemma choose-u-Cons: choose-u (x#xs) =
    (if xs = [] then x else min-degree-poly x (choose-u xs))
    <proof\rangle
lemma choose-u-member: xs }\not=[]\Longrightarrow\mathrm{ choose-u xs }\in\mathrm{ set xs
    <proof>
declare choose-u.simps[simp del]
```


### 5.2 Facts about Sylvester matrices and norms

lemma (in LLL) lattice-is-span [simp]: lattice-of $x s=$ span-list $x s$ $\langle p r o o f\rangle$
lemma sq-norm-row-sylvester-mat1:
fixes $f g$ :: ' $a$ :: conjugatable-ring poly
assumes $i$ : $i<$ degree $g$
shows $\|($ row (sylvester-mat f $g$ ) $i)\left\|^{2}=\right\| f \|^{2}$
$\langle p r o o f\rangle$
lemma sq-norm-row-sylvester-mat2:
fixes $f g$ :: ' $a$ :: conjugatable-ring poly
assumes i1: degree $g \leq i$ and $i 2: i<$ degree $f+$ degree $g$
shows $\|$ row (sylvester-mat $f g$ ) $i\left\|^{2}=\right\| g \|^{2}$
〈proof〉
lemma Hadamard's-inequality-int:
fixes $A$ ::int mat
assumes $A: A \in$ carrier-mat $n n$

```
    shows }|\operatorname{det}A|\leqsqrt (of-int (prod-list (map sq-norm (rows A))))
<proof\rangle
lemma resultant-le-prod-sq-norm:
    fixes fg::int poly
    defines }n\equiv\mathrm{ degree f and k}\equiv\mathrm{ degree g
    shows |resultant f g| s sqrt (of-int (|f||}\mp@subsup{|}{}{~}k*|g|\mp@subsup{|}{}{2`}n)
<proof>
```


## 5．3 Proof of the key lemma 16.20

```
lemma common-factor-via-short:
```

lemma common-factor-via-short:
fixes fgu :: int poly
fixes fgu :: int poly
defines }n\equiv\mathrm{ degree f and k}\equiv\mathrm{ degree g
defines }n\equiv\mathrm{ degree f and k}\equiv\mathrm{ degree g
assumes n0: n>0 and k0:k>0
assumes n0: n>0 and k0:k>0
and monic: monic u and deg-u: degree }u>
and monic: monic u and deg-u: degree }u>
and uf: poly-mod.dvdm muf and ug: poly-mod.dvdm m ug
and uf: poly-mod.dvdm muf and ug: poly-mod.dvdm m ug
and short: |f\mp@subsup{|}{}{2^}k*|g\mp@subsup{|}{}{2^n}<\mp@subsup{m}{}{2}
and short: |f\mp@subsup{|}{}{2^}k*|g\mp@subsup{|}{}{2^n}<\mp@subsup{m}{}{2}
and m:m\geq0
and m:m\geq0
shows degree (gcd fg)>0
shows degree (gcd fg)>0
\langleproof\rangle

```
\langleproof\rangle
```


## 5．4 Properties of the computed lattice and its connection with Sylvester matrices

lemma factorization－lattice－as－sylvester：
fixes $p::$＇$a$ ：：semidom poly
assumes $d j: d \leq j$ 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〉
context inj－comm－semiring－hom begin
lemma map－poly－hom－mult－monom［hom－distribs］：
map－poly hom（ $p *$ monom a $n$ ）$=$ map－poly hom $p * \operatorname{monom}($ hom a）$n$〈proof〉
lemma hom－vec－of－poly－n［hom－distribs］：
map－vec hom（vec－of－poly－n p $n$ ）$=$ vec－of－poly－n（map－poly hom $p$ ）$n$ $\langle p r o o f\rangle$
lemma hom－factorization－lattice［hom－distribs］：
shows map（map－vec hom）（factorization－lattice $u k m$ ）factorization－lattice
（map－poly hom u）$k$（hom m）
$\langle p r o o f\rangle$
end

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

```
context LLL
begin
sublocale idom-vec n TYPE(int)\langleproof\rangle
lemma upper-triangular-factorization-lattice:
    fixes u :: 'a :: semidom poly and d :: nat
    assumes d:d}\leqn\mathrm{ and du:d = degree u
    shows upper-triangular (mat-of-rows n (factorization-lattice u (n-d)k))
        (is upper-triangular ?M)
<proof>
lemma factorization-lattice-diag-nonzero:
    fixes }u::\mathrm{ ' }a\mathrm{ :: semidom poly and d
    assumes d:d=degree u
        and dn:d\leqn
        and u:u\not=0
        and m0:k\not=0
        and i:i<n
    shows (factorization-lattice u(n-d)k)!i$ i\not=0
<proof\rangle
corollary factorization-lattice-diag-nonzero-RAT: fixes d
    assumes d=degree u
        and d\leqn
        and }u\not=
        and }k\not=
        and i<n
    shows RAT (factorization-lattice u(n-d)k)!i$i\not=0
    \langleproof\rangle
sublocale gs: vec-space TYPE(rat) n\langleproof\rangle
lemma lin-indpt-list-factorization-lattice: fixes d
    assumes d:d=degree u and dn:d\leqn and u:u\not=0 and k:k\not=0
    shows gs.lin-indpt-list (RAT (factorization-lattice u (n-d) k)) (is gs.lin-indpt-list
(RAT ?vs))
\langleproof\rangle
```

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)

$$
\langle p r o o f\rangle
$$

```
abbreviation of-int-vec \equiv map-vec of-int
context LLL
begin
lemma lincomb-to-dvd-modulo:
    fixes ud
    defines }d\equiv\mathrm{ degree u
    assumes d:d\leqn
        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\rangle
lemma dvd-modulo-to-lincomb:
    fixes u :: int poly and d
    defines d}\equiv\mathrm{ degree u
    assumes d:d<n
        and dvd: poly-mod.dvdm k u(poly-of-vec g)
        and k-not0: k\not=0
    and monic-u: monic u
    and dim-g:dim-vec g=n
    and deg-u: degree }u>
    shows \existsc. lincomb-list c (factorization-lattice u(n-d)k)=g
<proof>
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 \(u \leq n \Longrightarrow n \neq 0 \Longrightarrow f \in\) poly-of-vec 'lattice-of (factorization-lattice
\(u(n-\) degree \(u) M) \Longrightarrow\)
degree \(f<n \wedge\) poly-mod.dvdm Muf
monic \(u \Longrightarrow\) degree \(u<n \Longrightarrow\)
degree \(f<n \Longrightarrow\) poly-mod.dvdm \(M u f \Longrightarrow f \in\) poly-of-vec'lattice-of (factorization-lattice
\(u(n-\) degree \(u) M)\)
\(\langle p r o o f\rangle\)
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 $p l 1: p l>1$
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 $u<n \Longrightarrow f \neq 0 \Longrightarrow$
poly-mod.dvdm pl $u f \Longrightarrow$ degree $f<n \Longrightarrow \| L L L$-short-polynomial pl $n u \|^{2} \leq$ $2^{\wedge}(n-1) *\|f\|^{2}$ $\langle p r o o f\rangle$
context LLL-implementation
begin
lemma LLL-reconstruction: assumes LLL-reconstruction $f u s=f s$
and degree $f \neq 0$
and poly-mod.unique-factorization-m pl $f$ (lead-coeff $f$, mset us)
and $f d v d F$
and $\bigwedge u i$. $u i \in$ set $u s \Longrightarrow$ poly-mod.Mp pl $u i=u i$
and $F 0: F \neq 0$
and cop: coprime (lead-coeff $F$ ) $p$
and sf: poly-mod.square-free-m p F
and $p l 1: p l>1$
and $p l p: p l=p^{\imath} l$
and $p$ : prime $p$
and large: $\mathfrak{2}^{\wedge}(5 *($ degree $F-1) *($ degree $F-1)) *\|F\|^{2}(2 *($ degree $F-$ 1)) $<p l^{2}$
shows $f=$ prod-list $f s \wedge\left(\forall f i \in\right.$ set fs. irreducible $\left._{d} f\right)$
$\langle p r o o f\rangle$
lemma LLL-many-reconstruction: assumes LLL-many-reconstruction $f u s=f s$ and degree $f \neq 0$
and poly-mod.unique-factorization-m pl $f$ (lead-coeff $f$, mset us)
and $f d v d F$
and $\bigwedge u i . u i \in$ set $u s \Longrightarrow$ poly-mod. $M p$ pl $u i=u i$
and $F 0: F \neq 0$
and cop: coprime (lead-coeff F) $p$
and sf: poly-mod.square-free-m p F
and $p l 1: p l>1$
and $p l p: p l=p^{\wedge} l$
and $p$ : prime $p$
and large: $\mathcal{Z}^{\wedge}(5 *($ degree $F$ div 2 $) *($ degree $F$ div 2$)) *\|F\|^{2}$ (2 $*($ degree $F$ div 2)) $<p l^{2}$
shows $f=$ prod-list $f s \wedge\left(\forall f i \in\right.$ set $f$ s. irreducible $\left._{d} f\right)$
$\langle p r o o f\rangle$
end
lemma LLL-factorization:
assumes res: LLL-factorization $f=g s$
and sff: square-free $f$
and deg: degree $f \neq 0$
shows $f=$ prod-list gs $\wedge\left(\forall g \in\right.$ set gs. irreducible $\left.{ }_{d} g\right)$
$\langle p r o o f\rangle$
lemma LLL-many-factorization:
assumes res: LLL-many-factorization $f=g s$
and sff: square-free $f$
and deg: degree $f \neq 0$
shows $f=$ prod-list gs $\wedge\left(\forall g \in\right.$ set gs. irreducible $\left.{ }_{d} g\right)$
$\langle p r o o f\rangle$
lift-definition one-lattice-LLL-factorization :: int-poly-factorization-algorithm is LLL-factorization 〈proof〉
lift-definition many-lattice-LLL-factorization :: int-poly-factorization-algorithm is LLL-many-factorization $\langle p r o o f\rangle$
lemma LLL-factorization-primitive: assumes LLL-factorization $f=f_{s}$
square-free $f$
$0<$ degree $f$
primitive $f$
shows $f=$ prod-list $f s \wedge(\forall f i \in$ set $f$ s. irreducible $f i \wedge 0<$ degree $f i \wedge$ primitive $f i)$ $\langle p r o o f\rangle$
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 | ' }a\mathrm{ set where
    sub-mset-sums [] ={0}
| sub-mset-sums (x # xs) = (let S = sub-mset-sums xs in S U( (+)x)'S)
lemma subset-add-mset: ys \subseteq# add-mset x zs \longleftrightarrow (ys\subseteq# zs \vee (\exists xs. xs\subseteq# zs
\ys = add-mset x xs))
    (is ?l = ?r)
<proof\rangle
lemma sub-mset-sums[simp]: sub-mset-sums xs = sum-mset'{ ys.ys \subseteq# mset xs
}
<proof\rangle
```

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\mathrm{ set xs. a dvdm b
<proof\rangle
end
```

lemma (in poly-mod-prime) irreducible-m-dvdm-prod-list:
assumes irr: irreducible-m a
and $d v d:$ a dvdm (prod-list xs)
shows $\exists b \in$ set xs. $a$ dvdm $b$
$\langle p r o o f\rangle$

### 7.2 The modified version of Algorithm 16.22

definition $B 2-L L L::$ int poly $\Rightarrow$ int where B2-LLL $f=$ 2 $^{\wedge}(2 *$ 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 $g^{\prime}$ and not to the primitive part of $g^{\prime}$. (Problems occur if the content of $g^{\prime}$ 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 $g^{\prime}$ is divisible by $u$ modulo $p^{l}$. However, for soundness we need to know that then also the primitive part of $g^{\prime}$ is divisible by $u$ modulo $p^{l}$. This is not necessary true, e.g., if $g^{\prime}=p^{l}$, then the primitive part is 1 which is not divisible by $u$ modulo $p^{l}$. It is open, whether such a large $g^{\prime}$ can actually occur. Therefore, the current fix is to manually test whether the leading coefficient of $g^{\prime}$ is strictly smaller than $p^{l}$.
With these two modifications, Algorithm 16.22 will become sound as proven below.
definition $L L L$-reconstruction-inner $j \equiv$
let $j^{\prime}=j-1$ in

- optimization: check whether degree j ' is possible
if $j^{\prime} \notin$ Degs then None else
- short vector computation
let
$l l=\left(\right.$ let $n=$ sqrt-int-ceiling $\left(\|f\|^{2} \wedge\left(2 * j^{\prime}\right) * 2^{\wedge}\left(5 * j^{\prime} * j^{\prime}\right)\right)$;
$l l^{\prime}=$ find-exponent $p n$ in if $l l^{\prime}<l$ then $l l^{\prime}$ else $l$ );
- optimization: dynamically adjust the modulus
$p l=p^{\wedge} l l ;$
$g^{\prime}=L L L$-short-polynomial pl $j u$
- fix: forbid multiples of $p^{l}$ as short vector, unclear whether this is really required in if abs (lead-coeff $\left.g^{\prime}\right) \geq p l$ then None else
let $p p g=$ primitive-part $g^{\prime}$
in
- slight deviation from textbook: we check divisibility instead of norm-inequality case div-int-poly $f$ ppg of Some $f^{\prime} \Rightarrow$
- fix: consider modular factors of ppg and not of g' Some (filter ( $\lambda$ gi. $\neg$ poly-mod.dvdm p gi ppg) gs, lead-coeff $f^{\prime}, f^{\prime}, p p g$ )
| None $\Rightarrow$ None
function $L L L$-reconstruction-inner-loop where
LLL-reconstruction-inner-loop $j=$
(if $j>$ degree $f$ then ( []$, 1,1, f$ )
else case LLL-reconstruction-inner $j$
of Some tuple $\Rightarrow$ tuple
$\mid$ None $\Rightarrow$ LLL-reconstruction-inner-loop $(j+1))$
〈proof〉
termination $\langle p r o o f\rangle$
end

```
partial-function (tailrec) LLL-reconstruction" where [code]:
    LLL-reconstruction" gs bffactors \(=\)
    (if gs \(=[]\) then factors
        else
            let \(u=\) choose-u gs;
                    \(d=\) degree \(u\);
                    \(g s^{\prime}=\) remove1 \(u \mathrm{gs}\);
                    degs \(=\) map degree gs \(^{\prime} ;\)
                        Degs \(=((+) d)\) 'sub-mset-sums degs;
                            \(\left(g s^{\prime}, b^{\prime}, f^{\prime}\right.\), factor \()=L L L\)-reconstruction-inner-loop gs \(f u\) Degs \((d+1)\)
            in LLL-reconstruction" gs' \(b^{\prime} f^{\prime}\) (factor\#factors)
        )
definition reconstruction-of-algorithm-16-22 gs \(f \equiv\)
    let \(G=[]\);
            \(b=\) lead-coeff \(f\)
        in LLL-reconstruction" gs bf \(G\)
end
```

definition factorization-algorithm-16-22 :: int poly $\Rightarrow$ int poly list where
factorization-algorithm-16-22 $f=$ (let

- find suitable prime
$p=$ suitable-prime-bz $f$;
- compute finite field factorization
$\left(-, f_{s}\right)=$ finite-field-factorization-int $p f$;
- determine l and B
$n=$ degree $f$;
- bound improved according to textbook, which uses $n o=(n+1) *(\max -$ normf $)^{2}$
$n o=\|f\|^{2} ;$
- possible improvement: $B=\operatorname{sqrt}\left(2^{5 * n *(n-1)} * n o^{2 * n-1}\right.$, cf. LLL-factorization

```
    B=sqrt-int-ceiling (2^ (5*n*n)*no ^(2 * n));
    l = find-exponent p B;
    - perform hensel lifting to lift factorization to mod p
    vs = hensel-lifting plffs
    - reconstruct integer factors
in reconstruction-of-algorithm-16-22 pl 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 $\left(p * p^{\wedge} k\right)(s m u l t$ $p f)=0 \Longrightarrow$ poly－mod．Mp $\left(p^{\wedge} k\right) f=0$
$\langle p r o o f\rangle$
lemma（in poly－mod－prime）eq－m－smult－p－removal：poly－mod．eq－m $\left(p * p^{\wedge} k\right)$ （smult $p f$ ）（smult $p g$ ）

$$
\Longrightarrow \text { poly-mod.eq-m }\left(p^{\wedge} k\right) f g\langle p r o o f\rangle
$$

```
lemma content-le-lead-coeff:abs (content (f :: int poly)) \leqabs (lead-coeff f)
```

〈proof〉
lemma poly-mod-dvd-drop-smult: assumes $u$ : monic $u$ and $p$ : prime $p$ and $c: c$
$\neq 0|c|<p^{\wedge} l$
and dvd: poly-mod.dvdm ( $\left.p^{\wedge} l\right) u(s m u l t ~ c f)$
shows poly-mod.dvdm puf
〈proof〉
context
fixes $p$ :: int
and $F::$ int poly
and $N::$ nat
and $l::$ nat
defines $[$ simp $]: N \equiv$ degree $F$
assumes $p$ : prime $p$
and NO: $N>0$
and bound-l: $\mathcal{2}^{\wedge} N^{2} * B 2-L L L F^{\wedge}(2 * N) \leq(p \wedge)^{2}$
begin
private lemma $F 0: F \neq 0\langle p r o o f\rangle$ lemma $p 1: p>1\langle p r o o f\rangle$
interpretation $p$ : poly-mod-prime $p\langle$ proof $\rangle$
interpretation $p l:$ poly-mod $p^{\wedge} l\langle p r o o f\rangle$
lemma B2-2: $2 \leq$ B2-LLL F
〈proof〉

```
lemma l-gt-0:l>0
<proof\rangle
lemma l0:l\not=0\langleproof\rangle
lemma pl-not0: p^ l = 0 \langleproof\rangle
interpretation pl: poly-mod-2 p^l
    \langleproof\rangle lemmas pl-dvdm-imp-p-dvdm = p.pl-dvdm-imp-p-dvdm[OF lO]
lemma p-Mp-pl-Mp[simp]: p.Mp (pl.Mp k) = p.Mp k
    <proof>
context
    fixes u :: int poly
        and d and f and n
        and gs :: int poly list
        and Degs :: nat set
    defines [simp]:d \equiv degree u
    assumes d0:d>0
        and u:monic u
        and irred-u: p.irreducible-m u
        and u-f:p.dvdm uf
        and f-dvd-F: f dvd F
        and [simp]: n== degree f
        and f-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\in set gs
        and norm-gs: map pl.Mp gs = gs
        and Degs: \ factor. factor dvd f \Longrightarrow p.dvdm u factor }\Longrightarrow\mathrm{ degree factor }
Degs
begin
interpretation pl: poly-mod-2 p^l \langleproof\rangle lemma f0: f = 0 \langleproof\rangle lemma
Mpf0: pl.Mp f = 0
    <proof\rangle lemma pMpf0: p.Mp f}\not=
    \langleproof\rangle lemma dn:d\leqn\langleproof\rangle lemma n0: n>0 \langleproof\rangle lemma B2-0[intro!]:
B2-LLL F>0\langleproof\rangle lemma deg-u: degree u>0\langleproof\rangle lemma n-le-N:n\leqN
<proof>
lemma dvdm-power: assumes g dvd f
    shows p.dvdm ug\longleftrightarrow pl.dvdm ug
<proof\rangle lemma uf:pl.dvdm uf \langleproof\rangle
lemma exists-reconstruction: \existsh0. irreducible }h0\wedgep.dvdm u h0 ^ h0 dvd
\langleproof\rangle
```

lemma factor－dvd－f－0：assumes factor $d v d f$
shows pl．Mp factor $\neq 0$
〈proof〉
lemma degree－factor－ge－degree－u：
assumes $u$－dvdm－factor：p．dvdm u factor
and factor－dvd：factor dvd $f$ shows degree $u \leq$ degree factor
〈proof〉

## 7．3．2 Inner loop

context
fixes $j^{\prime}::$ nat
assumes $d j^{\prime}: d \leq j^{\prime}$
and $j^{\prime} n: j^{\prime}<n$
and deg：$\bigwedge$ factor．p．dvdm u factor $\Longrightarrow$ factor $d v d f \Longrightarrow$ degree factor $\geq j^{\prime}$
begin
private abbreviation（input）$j \equiv S u c j^{\prime}$
private lemma $j n: j \leq n\langle p r o o f\rangle$ lemma factor－irreducible ${ }_{d} I$ ：assumes $h f: h$ dvd $f$
and puh：p．dvdm $u h$
and degh：degree $h>0$
and degh－j：degree $h \leq j^{\prime}$
shows irreducible $_{d} h$
$\langle$ proof $\rangle$ definition $l l=\left(\right.$ let $n=$ sqrt－int－ceiling $\left(\|f\|^{2} \wedge\left(2 * j^{\prime}\right) * \mathscr{Q}^{\wedge}\left(5 * j^{\prime} *\right.\right.$ $\left.j^{\prime}\right)$ ）；
$l l^{\prime}=$ find－exponent $p n$ in if $l l^{\prime}<l$ then $l l^{\prime}$ else $l$ ）
lemma $l l: l l \leq l\langle p r o o f\rangle$
lemma $l l 0: l l \neq 0\langle$ proof $\rangle$
lemma pll1：$p^{\wedge} l l>1\langle p r o o f\rangle$
interpretation pll：poly－mod－2 p ${ }^{\wedge} l l$
〈proof〉
lemma pll0：$p \wedge l l \neq 0\langle p r o o f\rangle$
lemma dvdm－l－ll：assumes pl．dvdm a b
shows pll．dvdm a b
$\langle p r o o f\rangle$ definition $g \equiv$ LLL－short－polynomial $\left(p^{\wedge} l l\right) j u$
lemma deg－$g$－$j$ ：degree $g<j$
and $g 0: g \neq 0$
and $u g: p l l . d v d m u g$
and short－g：$h \neq 0 \Longrightarrow$ pll．dvdm $u h \Longrightarrow$ degree $h \leq j^{\prime} \Longrightarrow\|g\|^{2} \leq \mathcal{Z}^{\wedge} j^{\prime} *$

```
|h|}\mp@subsup{}{}{2
\langleproof\rangle
lemma LLL-reconstruction-inner-simps: LLL-reconstruction-inner p l gs f u Degs
j
    =(if j' & Degs then None else if p``ll\leq llead-coeff g| then None
    else case div-int-poly f (primitive-part g) of None }=>\mathrm{ None
            | Some f' }=>\mathrm{ Some ([giזgs . ᄀ p.dvdm gi (primitive-part g)], lead-coeff f}\mp@subsup{f}{}{\prime}
f
<proof\rangle
lemma LLL-reconstruction-inner-complete:
    assumes ret:LLL-reconstruction-inner plgs fu Degs j=None
    shows \factor. p.dvdm u factor }\Longrightarrow\mathrm{ factor dvd f # degree factor }\geq
<proof\rangle
lemma LLL-reconstruction-inner-sound:
    assumes ret:LLL-reconstruction-inner plgs fu Degs j=Some (gs', b', f',h)
    shows f=\mp@subsup{f}{}{\prime}*h (is ?g1)
    and irreducible e}h\mathrm{ (is ?g2)
    and }\mp@subsup{b}{}{\prime}=lead-coeff f' (is ?g3
    and pl.unique-factorization-m f' (lead-coeff f', mset gs') (is ?g4)
    and p.dvdm uh (is ?g5)
    and degree h=\mp@subsup{j}{}{\prime}(is ?g6)
    and length gs ' < length gs (is ?g7)
    and set gs '\subseteq set gs (is ?g8)
    and gs'\not=[] (is ?g9)
<proof\rangle
end
interpretation LLL d \langleproof\rangle
lemma LLL-reconstruction-inner-None-upt-j':
    assumes ij:\foralli\in{d+1..j}.LLL-reconstruction-inner plgs f u Degs i=None
        and dj:d<j and j\leqn
    shows \factor. p.dvdm u factor }\Longrightarrow\mathrm{ factor dvd f # degree factor }\geq
    <proof>
corollary LLL-reconstruction-inner-None-upt-j:
    assumes ij:\foralli\in{d+1..j}. LLL-reconstruction-inner plgs f u Degs i=None
        and dj:d\leqj and jn: j\leqn
    shows \factor. p.dvdm u factor }\Longrightarrow\mathrm{ factor dvd f # degree factor }\geq
<proof\rangle
lemma LLL-reconstruction-inner-all-None-imp-irreducible:
assumes \(i: \forall i \in\{d+1 . . n\}\). LLL-reconstruction-inner plgs \(f u\) Degs \(i=\) None shows irreducible \(_{d} f\)
\(\langle p r o o f\rangle\)
```

```
lemma irreducible-imp-LLL-reconstruction-inner-all-None:
    assumes irr-f: irreducible d}
    shows }\foralli\in{d+1..n}.LLL-reconstruction-inner plgs fu Degs i=Non
<proof>
lemma LLL-reconstruction-inner-all-None:
    assumes i:\foralli\in{d+1..n}.LLL-reconstruction-inner plgs fu Degs i=None
    and dj: d<j
shows LLL-reconstruction-inner-loop p l gs f u Degs j=([],1,1,f)
    <proof\rangle
corollary irreducible-imp-LLL-reconstruction-inner-loop-f:
    assumes irr-f: irreducible }\mp@subsup{|}{d}{}f\mathrm{ and dj: d<j
shows LLL-reconstruction-inner-loop p l gs f u Degs j=([],1,1,f)
    <proof\rangle
lemma exists-index-LLL-reconstruction-inner-Some:
    assumes inner-loop:LLL-reconstruction-inner-loop plgs fu Degs j=(gs',\mp@subsup{b}{}{\prime},\mp@subsup{f}{}{\prime},factor)
        and i:\foralli\in{d+1..<j}.LLL-reconstruction-inner p l gs f u Degs i=None
        and dj:d<j and jn: j\leqn and f:\neg \mp@subsup{\mathrm{ irreducible }}{d}{}f
    shows }\exists\mp@subsup{j}{}{\prime}.j\leq\mp@subsup{j}{}{\prime}\wedge\mp@subsup{j}{}{\prime}\leqn\wedged<\mp@subsup{j}{}{\prime
        \wedge(LLL-reconstruction-inner plgs f u Degs j' = Some (gs', b', f', factor))
        \wedge(\foralli\in{d+1..<j'}.LLL-reconstruction-inner plgs f u Degs i=None)
    <proof\rangle
```

lemma unique-factorization-m-1: pl.unique-factorization-m 1 (1, \{\#\})
$\langle p r o o f\rangle$
lemma LLL-reconstruction-inner-loop-j-le-n:
assumes ret: LLL-reconstruction-inner-loop plgs f u Degs $j=\left(g s^{\prime}, b^{\prime}, f^{\prime}\right.$, factor $)$
and $i j: \forall i \in\{d+1 . .<j\}$. LLL-reconstruction-inner plgs f u Degs $i=$ None
and $n: n=$ degree $f$
and $j n: j \leq n$
and $d j: d<j$
shows $f=f^{\prime} *$ factor (is ?g1)
and irreducible ${ }_{d}$ factor (is ? ${ }^{2} 2$ )
and $b^{\prime}=$ lead-coeff $f^{\prime}$ (is ? $g 3$ )
and pl.unique-factorization-m $f^{\prime}\left(b^{\prime}\right.$, mset $\left.g s^{\prime}\right)($ is ? $g 4)$
and $p . d v d m$ u factor (is ? $g 5$ )
and $g s \neq[] \longrightarrow$ length $g s^{\prime}<$ length $g s$ (is ? $g 6$ )
and factor dvd $f$ (is ? $g^{7}$ )
and $f^{\prime} d v d f$ (is ? $g 8$ )
and set $g s^{\prime} \subseteq$ set gs (is ?g9)
and $g s^{\prime}=[] \longrightarrow f^{\prime}=1($ is ? $g 10)$
$\langle$ proof $\rangle$
lemma LLL-reconstruction-inner-loop-j-ge-n:
assumes ret: LLL-reconstruction-inner-loop plgs $f u$ Degs $j=\left(g s^{\prime}, b^{\prime}, f^{\prime}, f a c t o r\right)$

```
    and ij: \foralli\in{d+1..n}. LLL-reconstruction-inner pl gs f u Degs i=None
    and dj:d<j
    and jn: j>n
shows f=\mp@subsup{f}{}{\prime}*\mathrm{ factor (is ?g1)}
    and irreducibled factor (is ?g2)
    and }\mp@subsup{b}{}{\prime}=lead-coeff f f' (is ?g3
    and pl.unique-factorization-m f}\mp@subsup{f}{}{\prime}(\mp@subsup{b}{}{\prime},\mathrm{ mset gs') (is ?g4)
    and p.dvdm u factor (is ?g5)
    and gs \not=[]\longrightarrow length gs ' < length gs (is ?g6)
    and factor dvd f (is ?g7)
    and f}\mp@subsup{f}{}{\prime}dvdf(\mathrm{ is ?g8)
    and set gs'\subseteq set gs (is ?g9)
    and f}\mp@subsup{f}{}{\prime}=1(\mathrm{ is ?g10)
<proof\rangle
lemma LLL-reconstruction-inner-loop:
    assumes ret:LLL-reconstruction-inner-loop plgs f u Degs j = (gs',}\mp@subsup{b}{}{\prime},\mp@subsup{f}{}{\prime},factor
    and ij:\foralli\in{d+1..<j}.LLL-reconstruction-inner plgs f u Degs i=None
    and n: n= degree f
    and dj:d<j
shows f=\mp@subsup{f}{}{\prime}* factor (is ?g1)
    and irreducible e factor (is ?g2)
    and }\mp@subsup{b}{}{\prime}=lead-coeff f' (is ?g3
    and pl.unique-factorization-m f' ( }\mp@subsup{b}{}{\prime}\mathrm{ , mset gs') (is ?g4)
    and p.dvdm u factor (is ?g5)
    and gs \not=[]\longrightarrow length gs ' < length gs (is ?g6)
    and factor dvd f (is ?g7)
    and f}\mp@subsup{f}{}{\prime}dvdf(\mathrm{ is ?g8)
    and set gs '\subseteq set gs (is ?g9)
    and g\mp@subsup{s}{}{\prime}=[]\longrightarrow\mp@subsup{f}{}{\prime}=1(\mathrm{ is ?g10)}
<proof\rangle
end
```


### 7.3.3 Outer loop

lemma LLL-reconstruction ${ }^{\prime \prime}$ :
assumes 1: LLL-reconstruction" plgs bf $G=G^{\prime}$
and irreducible- $G: \bigwedge$ factor. factor $\in$ set $G \Longrightarrow$ irreducible $_{d}$ factor
and 3: $F=f *$ prod-list $G$
and 4: pl.unique-factorization-m $f$ (lead-coeff $f$, mset gs)
and 5: $g s \neq[]$
and $6: \bigwedge g i . g i \in$ set $g s \Longrightarrow p l . M p g i=g i$
and 7: $\bigwedge$ gi. gi $\in$ set $g s \Longrightarrow$ p.irreducible $e_{d}-m g i$
and 8: p.square-free-m $f$
and 9: coprime (lead-coeff f) $p$
and $s f$ - $F$ : square-free $F$
shows $\left(\forall g \in\right.$ set $G^{\prime}$. irreducible $\left._{d} g\right) \wedge F=$ prod-list $G^{\prime}$
$\langle p r o o f\rangle$

```
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 }\not=[
<proof\rangle
lemma reconstruction-of-algorithm-16-22:
    assumes 1:reconstruction-of-algorithm-16-22 plgs F=G
    shows ( }\forallg\in\mathrm{ set }G\mathrm{ . irreducible d g) ^F= prod-list }
<proof\rangle
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 $\left(\forall g \in\right.$ set $G$. irreducible $\left.{ }_{d} g\right) \wedge f=$ prod-list $G$
$\langle p r o o f\rangle$
lift-definition increasing-lattices-LLL-factorization :: int-poly-factorization-algorithm is factorization-algorithm-16-22 〈proof〉
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 }=>\mathrm{ int poly }=>\mathrm{ int poly
    where max-degree-poly a b = (if degree a d degree b then a else b)
fun choose-u :: int poly list }=>\mathrm{ 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 \(f\);
    \(A=\operatorname{linf}\)-norm-poly \(f ; n=\) degree \(f ; B=\) sqrt-int-ceiling \((n+1) * 2 \wedge n * A\);
    Bnd \(=\) 2^( \(^{\wedge}\) ^2 div 2) \(* B^{\wedge}(2 * n) ; l=\log\)-ceiling \(p\) Bnd;
    \((-, f s)=\) finite-field-factorization-int \(p f\);
    gs \(=\) hensel-lifting plffs;
    \(u=\) choose-u gs;
    \(d=\) degree \(u\);
    g-star \(=[: 2,2,0,2\) :: int :];
    \(\left(g s^{\prime}, h s^{\prime}\right)=\) List.partition ( \(\lambda\) gi. poly-mod.dvdm p gi g-star) gs;
    \(h\)-star \(=\) smult \(b\left(\right.\) prod-list hs \(\left.{ }^{\prime}\right)\);
    \(f\)-star \(=\) primitive-part h-star
    in \(\left(h s^{\prime}=[] \wedge f\right.\)-star \(\left.=1\right)\langle\) proof \(\rangle\)
```


### 8.2 Another potential problem of Algorithm 16.22

Suppose that $g^{*}$ is $p^{l}$. (It is is not yet clear whether lattices exists where this $g^{*}$ is short enough). Then $p p\left(g^{*}\right)=1$ 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 \(f\);
    \(A=\operatorname{linf}\)-norm-poly \(f ; n=\) degree \(f ; B=\operatorname{sqrt-int-ceiling~}(n+1) * 2 \uparrow n * A\);
    Bnd \(=2\) ^( \(n\) へ2 div 2) \(* B^{\wedge}(2 * n) ; l=\) log-ceiling \(p\) Bnd;
    \(\left(-, f_{s}\right)=\) finite-field-factorization-int \(p f\);
    \(g s=\) hensel-lifting plffs;
    \(u=\) choose-u gs;
    \(p l=p^{\wedge} l ;\)
    \(p l 2=p l d i v 2 ;\)
    \(u^{\prime}=\) poly-mod.inv-Mp2 pl pl2 (poly-mod.Mp pl (smult bu))
    in sqrt-int-floor (sq-norm \(u^{\prime}\) ) >pl \(\langle p r o o f\rangle\)
```

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 \((n+1) * 2 \wedge n * A\);
    Bnd \(=\) 2^( \(^{\wedge}\) ^2 div 2) \(* B^{\wedge}(2 * n) ; l=\log\)-ceiling \(p\) Bnd;
    \(\left(-, f_{s}\right)=\) finite-field-factorization-int \(p f\);
    gs \(=\) hensel-lifting p lffs;
    \(u=\) choose-u gs;
    \(p l=p^{\wedge} l ;\)
    \(p l 2=p l\) div \(2 ;\)
    \(u^{\prime}=\) poly-mod.inv-Mp2 pl pl2 (poly-mod.Mp pl (smult bu));
    \(d=\) degree \(u\);
    \(p l=p^{\wedge} l\);
    \(L=\) factorization-lattice \(u^{\prime} 1\) 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 6610559687902485989519153080327710398284046829642812192846
- norm $u: \approx 6.67555 \cdot 10^{122}$, namely 667555058938127908386141559707490406617756492853269306
- norm g-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 ᄀ poly-mod.eq-m (5^6) (smult 6 (g2*g3*g4)) (rhs) \langleproof\rangle
end
```


## References

[1] J. Divasón, S. J. C. Joosten, R. Thiemann, and A. Yamada. A formalization of the Berlekamp-Zassenhaus factorization algorithm. In $C P P$ 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.


[^0]:    *Supported by FWF (Austrian Science Fund) project Y757. Jose Divasón is partially funded by the Spanish project MTM2017-88804-P.

