Computeralgebra
GGT, Primeigenschaft, Faktorisierung

Hans-Gert Gräbe
Institut für Informatik, Universität Leipzig

17. Januar 1997

Inhaltsverzeichnis

1 Allgemeine Teilbarkeitstheorie

1.1 Teilbarkeit und Faktorisierung in Integritätsbereichen

Begriff des Nullteilers
Integritätsbereich gdw. Kürzungsregel gilt
R sei ein Integritätsbereich, d.h. ein kommutativer Ring mit 1 ohne Nullteiler, sowie $a, b, \ldots \in R$. R ist eine abelsche Gruppe bzgl. + und eine Halbgruppe mit 1 bzgl. *. Die invertierbaren Elemente

$$R^* := \{r \in R \;:\; \exists\, s \in R \;\; rs = sr = 1\}$$

bilden eine multiplikative Untergruppe.
Eigenschaft $a \in R^*,\ b|a \;\Rightarrow\; b \in R^*$
Teilbarkeit: $a|b \;:\Leftrightarrow\; \exists\, c \;:\; b = ac$
assoziierte Elemente: $a \sim b \;:\Leftrightarrow\; a|b,\ b|a$
a und b sind genau dann assoziiert, wenn es eine Einheit $u \in R^*$ gibt mit $a = ub$.
Teilbarkeitsaussagen kann man nur bis auf assoziierte Elemente treffen. Interessant sind deshalb Ringe R mit einer *Einheits-Normalform* $n : R \longrightarrow R$, die aus jeder Klasse assoziierter Elemente ein normalisiertes Element auswählt, d.h.

$$\forall a \in R \quad a = u(a) \cdot n(a), \quad u(a) \in R^*, n(a) \text{ normalisiert.}$$

Begriff der trivialen Teiler, irreduzible Elemente.
Beispiele: $\mathbf{Z}$, $R \longrightarrow R[x]$ mit

1. $deg(fg) = deg(f) + deg(g)$
2. $f|g \;\Rightarrow\; lc(f)|lc(g), abs(f)|abs(g)$
3. Einheiten: $(R[x])^* = R^*$.

Begriff der Teilerkette $\{r_n\}$ mit $\forall n\ r_{n+1}|r_n$.
In R gilt der Teilerkettensatz $:\Leftrightarrow\; \exists n_0\ \forall n > n_0\ r_n \sim r_0$ (TK)
Beispiele: in $\mathbf{Z}$ wegen $|.|$, in $K[x]$ wegen $deg(f) = deg(g) \;\Rightarrow\; f \sim g$.
Satz: Gilt in R der Teilerkettensatz, so auch in $R[x_1, \ldots, x_n]$.
Satz: Gilt in R der Teilerkettensatz, so kann man jedes Element $r \notin R^*$ als Produkt von endlich vielen irreduziblen Elementen

$$r = p_1 \cdot \ldots \cdot p_n$$

darstellen.
Beweis über alternative Formulierung von (TK)
$\forall M \subset R$ enthält bzgl. Teilbarkeit minimale Elemente, d.h.

$$\exists r \in M\ \forall s \in M\ s|r \;\Rightarrow\; s \sim r \qquad \text{(TM)}$$

1.1.1 Eindeutigkeit der Produktzerlegung

Zwei Produktzerlegungen $p_1 \cdot \ldots \cdot p_m = q_1 \cdot \ldots \cdot q_n$ heißen äquivalent $:\Leftrightarrow \ m = n$ und $\exists \pi \in S_n$ mit $\forall i \ p_i \sim q_{\pi(i)}$.

Bsp. $\mathbf{Z}$ $6 = 2 \cdot 3 = (-3) \cdot (-2)$

Es gibt Integritätsbereiche, in denen die Faktorzerlegung nicht eindeutig ist.

Aufgabe:
1. $R_n := \{a + b\sqrt{n} \ : \ a, b \in \mathbf{Z}\}$ ist ein Integritätsbereich.
2. In R_{-5} gilt $6 = 2 \cdot 3 = (1 + \sqrt{-5}) \cdot (1 - \sqrt{-5})$.
3. Diese vier Faktoren sind irreduzibel und paarweise nicht assoziiert.
4. In R gilt (TK).

Um die Eindeutigkeit zu erzwingen braucht man also eine schärfere Bedingung als irreduzbel.

Definition: Ein Element $r \in R - (0)$ heißt Primelement $:\Leftrightarrow \ p \notin R^*$ und $\forall \ a, b \in R: \ p|ab \ \Rightarrow \ p|a$ oder $p|b$.

Eigenschaften:
1. Primelemente sind irreduzibel
2. p Primelement und $s \sim p \ \Rightarrow \ s$ Primelement
3. Sind p, q Primelemente mit $p|q \ \Rightarrow \ p \sim q$
4. $p|a_1 \cdot \ldots \cdot a_n \ \Rightarrow \ \exists i \ p|a_i$

Satz über die Eindeutigkeit der Zerlegung in Primelemente:
Läßt sich r auf zwei Arten als Produkt von Primelementen

$$r = p_1 \cdot \ldots \cdot p_m = q_1 \cdot \ldots \cdot q_n$$

darstellen, so gilt $m = n$ und $\exists \pi \in S_n: \ \forall i \ p_i \sim q_{\pi(i)}$.

NB: Ist allgemein $r|s$, so ist die Zahl der Primelemente in einer Zerlegung von r nicht größer als die in einer Zerlegung von s.

1.1.2 Faktorielle Ringe

Definition: Ein Integritätsbereich R heißt faktoriell $:\Leftrightarrow$ jedes Element $r \notin R^*$ läßt sich als endliches Produkt von Primelementen schreiben.

Satz: Ein Integritätsbereich R ist genau dann faktoriell, wenn
1. die Teilerkettenbedingung in R gilt und
2. jedes irreduzible Element ein Primelement ist.

Dann gilt der Satz von der Existenz und Eindeutigkeit der Primfaktorzerlegung.

$\mathcal{P}$ Menge von Vertretern der Primelementklassen (normierte Primelemente). Dann hat jedes Element $r \notin R^*$ eine eindeutig bestimmte Darstellung

$$r = \epsilon \cdot \prod_{p \in \mathcal{P}} p^{\nu_p(r)}$$

mit $\epsilon \in R^*$ und $\nu_p(r) \in \mathbf{N}$ fast alle gleich 0. Damit definiert $n(r) = \prod_{p \in \mathcal{P}} p^{\nu_p(r)}$ eine Einheits-Normalform auf R.

$\nu_p(r)$ bezeichnet man als Ordnung von r an der Stelle p. Obige Zerlegung heißt dann auch *die normierte Primelementzerlegung* von r zum Repräsentantensystem $\mathcal{P}$.

1.1.3 Polynomringe über faktoriellen Ringen

Theorem (Gauß): Ist R ein faktorieller Ring, dann auch $R[x_1, \ldots, x_n]$.

Beweis vgl. [5, S. 45]

Folgerung: $\mathbf{Z}[x_1, \ldots, x_n]$ und $k[x_1, \ldots, x_n]$ für jeden Körper k sind faktorielle Ringe.

1.1.4 ggT und kgV

Definition:
$c := gcd(a,b) \;:\Leftrightarrow$ 1. $c|a,\ c|b$
2. $d|a,\ d|b \;\Rightarrow\; d|c$.
$c := lcm(a,b) \;:\Leftrightarrow$ 1. $a|c,\ b|c$
2. $a|d,\ b|d \;\Rightarrow\; c|d$.
Satz: 1. gcd und lcm sind eindeutig bestimmt bis auf eine Einheit.
2. $c = gcd(a,b) \;\Rightarrow\; \frac{a\,b}{c}$ ist ein $lcm(a,b)$.
Satz: Sei R ein faktorieller Ring und $\mathcal{P}$ ein Repräsentantensystem für die Klassen assoziierter Primelemente von R. Für $r,s \in R-(0)$ existieren dann ggT und kgV und es gilt

$$gcd(r,s) = \prod_{p\in\mathcal{P}} p^{min(\nu_p(r),\nu_p(s))}$$

$$lcm(r,s) = \prod_{p\in\mathcal{P}} p^{max(\nu_p(r),\nu_p(s))}$$

Folgerung: In den Ringen $\mathbf{Z}[x_1,\ldots,x_n]$ und $k[x_1,\ldots,x_n]$ für jeden Körper k gibt es ggT und kgV.
Es gibt allerdings auch Integritätsbereiche, in denen es nicht zu allen Elementpaaren einen ggT gibt.
Aufgabe: Zeige, daß in R_{-5} die beiden Elemente 6 und $2(1+\sqrt{-5})$ keinen ggT haben.

1.2 Euklidsche Ringe

Es gibt eine Klasse von Integritätsbereichen, in denen man den ggT effektiver als über eine vollständige Faktorzerlegung ausrechnen kann. Zu der gehören aber leider nicht mehr alle uns interessierenden Ringe.
Definition: Ein Integritätsbereich D heißt *Euklidscher Ring* wenn auf ihm eine Bewertung $\nu : D-(0) \longrightarrow \mathbf{N}$ mit folgenden Eigenschaften gegeben ist:
1. $\forall a,b \in D-(0)\ \nu(ab) \geq \nu(a)$
2. (Division mit Rest) $\forall a,b \in D-(0)$ mit $b \neq 0$ existieren $q,r \in D$ mit $a = qb+r$ und entweder $r = 0$ oder $\nu(r) < \nu(b)$.
Für konstruktive Zwecke muß also neben der konstruktiven Version der Ringoperationen und der Teilbarkeitsfunktion auf D noch eine effektive Funktion

$$DivRem : (D, D-(0)) \longrightarrow (D,D) \quad \text{via} \quad DivRem(a,b) = (q,r)$$

gegeben sein.
q bezeichnet man als den Quotienten und r als den Rest bei der Division von a durch b.
Beispiele: $\mathbf{Z}$ mit $\nu(a) = |a|$.
$k[x]$ für einen Körper k mit $\nu(a) = deg(a)$.
Das kann man effektiv implementieren, z.B. in REDUCE:
Univariate Polynome über $\mathbf{Q}$

```
procedure mylcof(f,x);
      if deg(f,x)=0 then f else lcof(f,x);

procedure divrem(f,g);
      begin scalar c,q;
      while (f neq 0) and ((d:=deg(f,x)-deg(g,x))≥0) do
      << c:=mylcof(f,x)/mylcof(g,x)*x^d;
```

```
            q:=q+c; f:=f-c*g; >>;
      return {q,f};
      end;
```

Beispiel Ganze Zahlen

```
procedure divrem(f,g);
      begin scalar r,q;
      q:=floor(f/g); r:=f-q*g;
      return {q,r};
      end;
```

Der Euklidsche Algorithmus

```
procedure Euklid(a,b:D):D
      while (b≠ 0) do
            (q,r):=DivRem(a,b);
            a:=b; b:=r;
      return a;
end;
```

Satz:
1. In einem Euklidschen Ring D terminiert `Euklid` und berechnet g=gcd(a,b).
2. $\exists u, v \in D: \; g = au + bv$
3. D ist ein Hauptidealring.
2. u, v können mit `ExtendedEuklid` berechnet werden:

```
procedure ExtendedEuklid(a,b:D):(D,D,D)
      (u1,v1):=(1,0);(u2,v2):=(0,1);
      while (b≠ 0) do
            (q,r):=DivRem(a,b); (u3,v3):=(u1-qu2,v1-qv2);
            a:=b; b:=r; (u1,v1):=(u2,v2); (u2,v2):=(u3,v3);
      return (a,u1,v1);
end;
```

Man prüft leicht nach, daß für die Eingabeparameter (a_0, b_0) stets $a = u_1\, a_0 + v_1\, b_0$ sowie $b = u_2\, a_0 + v_2\, b_0$ gilt. Daraus folgt 2. und 1.
Beispiel:

```
1: euklid(2134134,581931);

2134134 = 3 * 581931 + 388341

581931 = 1 * 388341 + 193590

388341 = 2 * 193590 + 1161

193590 = 166 * 1161 + 864

1161 = 1 * 864 + 297
```

```
864 = 2 * 297 + 270

297 = 1 * 270 + 27

270 = 10 * 27 + 0

Der ggT ist also gleich

27

2: euklid(2134134,581931);

{27,2005,-7353}
```

Beispiel:
$f := 48x^4 - 84x^2 + 42x - 36$, $g := -4x^3 - 10x^2 + 44x - 30$ in $\mathbf{Q}[x]$:

```
1: on rational;

2: euklid(f,g);

48*(x**4 - 7/4*x**2 + 7/8*x - 3/4) =
     ( - 12*(x - 5/2)) *
        ( - 4*(x**3 + 5/2*x**2 - 11*x + 15/2))
          + (744*(x**2 - 273/124*x + 36/31))

 - 4*(x**3 + 5/2*x**2 - 11*x + 15/2) =
     ( - 1/186*(x + 583/124)) *
        (744*(x**2 - 273/124*x + 36/31))
          + (27833/3844*(x - 31368/27833))

744*(x**2 - 273/124*x + 36/31) =
     (2859936/27833*(x - 3708777/3451292)) *
        (27833/3844*(x - 31368/27833))
          + ( - 28701533520/774675889)

27833/3844*(x - 31368/27833) =
     ( - 21561554018537/110328694850880*
        (x - 31368/27833)) *
          ( - 28701533520/774675889) + (0)

Der ggT ist also gleich

 - 28701533520/774675889

3: extendedeuklid(f,g);

{ - 28701533520/774675889,
 - 15376/27833*(x**2 + 201901/55666*x - 191005/27833),
 - 184512/27833*(x**3 + 31368/27833*x**2
       - 3683/8564*x + 111323/222664)}
```

1.3 Univariate Polynome

Begriff des Quotientenkörpers K eines Integritätsbereichs R.

Gegenstand dieses Punktes ist es, für einen faktoriellen Ring R zu untersuchen, wie Teilbarkeitsfragen in $R[x]$, $K[x]$ und R zusammenhängen.

Betrachte $f = a_0 + a_1 x + \ldots + a_n x^n \in R[x]$.

$a := gcd(a_0, \ldots, a_n)$ nennt man den *Inhalt* $cont(f)$ von f,

$pp(f) := \frac{a_0}{a} + \frac{a_1}{a} x + \ldots + \frac{a_n}{a} x^n$ heißt *primitiver Teil* von f.

Einheits-Normalformen in R vorausgesetzt, was wir oBdA tun wollen, sind diese Zerlegungen eindeutig.

Gaußlemma: Über einem faktoriellen Ring R ist das Produkt zweier primitiver Polynome wieder primitiv.

Folgerungen:
1. $\forall\, f, g \in R[x]\ cont(fg) \sim cont(f) cont(g)$,
2. $\forall\, f, g \in R[x]\ pp(fg) \sim pp(f) pp(g)$,
3. $\forall\, f, g \in R[x]\ g|F \Rightarrow cont(g)|cont(f),\ pp(g)|pp(f)$,
4. $\forall\, f \in R[x], r \in R\ cont(rf) \sim r \cdot cont(f),\ pp(rf) \sim pp(f)$.

Analoge Definitionen hat man für $f \in K[x]$: Jedes solche Polynom kann man durch Bilden des Hauptnenners auf die Form $f = \frac{g}{r}$ mit $g \in R[x],\ r \in R$ bringen. Definiere dann

$$cont(f) = \frac{cont(g)}{r} \quad \text{und} \quad pp(f) = pp(g) \in R[x]$$

und zeige, daß diese von der Repräsentation unabhängig sind.

Irreduzibilität univariater Polynome

Satz (Eisensteinkriterium): Sei $f = a_0 + a_1 x + \ldots + a_n x^n \in R[x]$ vom Grad $n > 0$ ein primitives Polynom. Wenn ein Primelement $p \in R$ existiert mit

$$p|a_i\ (i = 0, \ldots, n-1), \quad p^2 \not| a_0,$$

so ist f irreduzibel in $R[x]$.

Bew. [5, S. 56]

Beispiele in $\mathbf{Z}[x]$: $X^n - r$, wenn $r \in \mathbf{Z}$ einen Primfaktor nur in erster Potenz enthält.

Aufgabe: Untersuchen Sie die folgenden Polynome in $\mathbf{Z}[x]$ auf Irreduzibilität:

$x^4 + 1, x^4 + x + 1, x^4 - 6x^2 + 5, x^4 + 6x^2 + 1$
$x^5 - x^2 + 1, x^4 + 2x^2 + x + 3,$
$x^5 - 10x^4 + 10x^3 - 80x^2 - 75x + 17$

Frage: Sind Polynome, die in $\mathbf{Z}[x]$ irreduzibel sind, auch in $\mathbf{Q}[x]$ irreduzibel?

Satz von Gauß über irreduzible Polynome: Ist $f \in R[x]$ ein irreduzibles Polynom vom Grad > 0, dann ist f auch in $K[x]$ irreduzibel.

Bew.: Sei $f = g \cdot h$ eine Zerlegung in $K[x]$. Stelle zuerst $g = \frac{a}{b} g_1$ und $h = \frac{c}{d} h_1$ mit primitiven Polynomen $g_1, h_1 \in R[x]$ und $a, b, c, d \in R, gcd(a, b) = gcd(c, d) = 1$ dar.

Folglich gilt $(ac)(g_1 h_1) = (bd) f$. Aus dem Gaußlemma folgt $ac \sim bd$, d.h. $f = u \cdot g_1 h_1$ für ein $u \in R^*$. f wäre also auch in $R[x]$ zerlegbar.

Analoge Frage wie oben für Polynome in $\mathbf{Q}[x, y] = (\mathbf{Q}[x])[y]$

$x - y, y^3 + x^2 + 2, y^4 + (x+1)^2 y^2 + x^2 - 1$

ggT-Berechnung für univariate Polynome

Satz: Die ggT-Berechnung von $f, g \in R[x]$ kann man auf die in R und in $K[x]$ zurückführen:

$$gcd_{R[x]}(f, g) = gcd_R(cont(f), cont(g)) \cdot pp(gcd_{K[x]}(f, g)).$$

Bew.: RHS=$r \cdot h$. Da auch $h = pp(gcd_{K[x]}(pp(f), pp(g)))$, folgt
$r|cont(f), cont(g),\ h|pp(f), pp(g) \ \Rightarrow\ RHS|f, g$. Andererseits für $c|f, g$ folgt
$cont(c)|cont(f), cont(g) \ \Rightarrow\ cont(c)|r$ und $pp(c)|pp(f), pp(g) \ \Rightarrow\ pp(c)|h$.

2 Rechnen mit ganzen Zahlen

2.1 Die Langzahlarithmetik und deren Komplexität

Zahldarstellung in der Basis β:

$$z = \pm \sum_{i=0}^{m} a_i \beta^i =: [\varepsilon, m; a_o, \ldots, a_m]$$

Die Zahl $l(z) := m + 1 = \left[\frac{\log z}{\log \beta}\right] + 1$ nennt man die *Wortlänge* von z.

Ein- und Ausgabe

Wir haben dafür drei Datentypen zur Verfügung:

ZAHL = Zahlen zur Basis β, für die man die volle Arithmetik beherrscht.

DIGIT = Elementare Bestandteile von ZAHL.

STRING (als Sequenz of CHAR) = E/A-Format von ZAHL in einem anderen Zahlsystem, zusammen mit zwei Operationen $val : CHAR \longrightarrow DIGIT$ und $symb : DIGIT \longrightarrow CHAR$

Die Umrechnung einer Zahl in einen String erfolgt durch fortgesetzte Division mit Rest. Beispiel: $21357 = [242513]_6$

```
procedure STRING-to-ZAHL(s=[a_m ... a_0]_c:STRING):ZAHL
      z:=val(a_m);
      for i:=m-1 downto 0 do z:=c*z+val(a_i);
      return z;
```

Die Umrechnung eines Strings in eine Zahl erfolgt nach dem Hornerschema. Beispiel: $[73125]_{12} = 150509$

```
procedure ZAHL-to-STRING(z:ZAHL):STRING
      i:=0;
      while z ≠ 0 do
            (q,r):=divmod(s,c);
            a_i:=r; z:=q; inc(i);
      return [a_m ... a_0]_c;
```

Benötigte Operationen (unter der Annahme $l(\beta) = l(c) = 1$):

$Dmult : (ZAHL, DIGIT) \longrightarrow ZAHL$ und

$Ddivmod : (ZAHL, DIGIT) \longrightarrow (ZAHL, DIGIT)$

Die benötigen ihrerseits die elementaren Operationen

$Emult : (DIGIT, DIGIT) \longrightarrow DoubleDIGIT$ und

$Edivmod : (DoubleDIGIT, DIGIT) \longrightarrow (DIGIT, DIGIT)$

Komplexität: $C(Dmult(z, d)) = C(Ddivmod(z, d)) = O(l(z)^2)$

Rechnen in fremden Positionssystemen

Addition und Subtraktion sind trivial.

Komplexität: C(add(a,b))=	worst case:	$\max(l(a), l(b)) + 1$
	best case:	$\min(l(a), l(b))$
	average case:	$\min(l(a), l(b)) + \frac{1}{2}$.

Multiplikation: Hierfür brauchen wir eine Multiplikationstabelle für das "kleine EinmalEins" im fremden Positionssystem. Beispiel $\beta = 7.$ $213\,461 * 61\,534 = 16\,606\,345\,064$

Beachte, daß man im Unterschied zum schriftlichen Rechnen in der Schule mit einem Akkumulator c arbeiten muß, um den Übertrag korrekt zu bearbeiten.

```
procedure mult(a,b:ZAHL):ZAHL
      for i:=0 to l(a)+l(b)-1 do c_i := 0;
      for i:=0 to l(a)-1 do
            r:=0;
            for i:=0 to l(b)-1 do
                  t:=a_i * b_j + c_{i+j} + r;
                  (r,c_{i+j}):=Edivmod(t,β);
            c_{i+l(b)}:=r; (evtl. verbliebener Übertrag)
      return c;
```

Komplexität: $C(mult(a,b)) = O(l(a) \cdot l(b)$

Karatsuba-Multiplikation

Idee: Sind a, b beides Zahlen der Länge $2l$, so zerlegen wir sie in

$$a = A_1 \cdot \beta^l + A_2, \quad b = B_1 \cdot \beta^l + B_2$$

und erhalten

$$a \cdot b = (A_1 B_1)\beta^{2l} + (A_1 B_2 + A_2 B_1)\beta^l + (A_2 B_2)$$

Die drei Koeffizienten kann man mit *drei* Multiplikationen l-stelliger Zahlen berechnen wegen

$$(A_1 B_2 + A_2 B_1) = (A_1 + B_1)(A_2 + B_2) - A_1 B_1 - A_2 B_2$$

Komplexität: $C_{Karatsuba}(mult(a,b)) = O(l(z)^{\frac{\log 3}{\log 2}})$

2.2 Division mit Rest und die Berechnung des ggT von ganzen Zahlen

Allgemeines Schema der Division mit Rest

```
procedure divmod(a,b:ZAHL):(ZAHL,ZAHL)
      q:=0;
      while a ≥ b do
            Errate die nächste Ziffer q_i des Quotienten
            q:=q+(q_i β^i);
            a:=a-(q_i β^i)*b;
            Evtl. notwendige Korrektur
      return (q,a);
```

Komplexität: Wenn korrektes Ziffernerraten des Quotienten in konstanter Zeit möglich ist, dann $l(q) \cdot l(b)$.

Diskussion, daß man die Ziffer niemals mit konstantem Aufwand auf Anhieb genau erraten kann, und mit welchen Verfahren man gut fährt.

```
Evtl. notwendige Korrektur ==
      while a ≥ β^i * b do
            a:=a-b; q_i := q_i + 1;
```

gcd-Berechnung

Satz von Lamè, 1845:
Für $r \geq 1$ seien $a > b > 0$ zwei Zahlen, für die der Euklidsche Algorithmus genau r Schritte benötigt und a sei so klein wie möglich. Dann gilt

$$a = F_{r+2}, \quad b = F_{r+1},$$

wobei F_i die Fibonaccifolge ist.
Binetsche Formel: $F_i = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2}\right)^i - \left(\frac{1-\sqrt{5}}{2}\right)^i \right) \sim O(\alpha^i)$ mit $\alpha = \frac{1+\sqrt{5}}{2} \sim 1.6$.
Folgerung: Der Euklidsche Algorithmus benötigt für die Berechnung des gcd zweier Zahlen a, b mit $l(a), l(b) \leq l$ höchstens $l \frac{\log \beta}{\log \alpha}$ Divisionen mit Rest.
Im allgemeinen treten entweder viele, aber billige Divisionen oder wenige, aber teure Divisionen auf. Deshaln ist eien genauere Analyse der Komplexität der gcd-Berechnung notwendig.
Satz: $C(gcd(a, b)) = O(l(a)\, l(b))$.
Beweis: Der Euklidsche Algorithmus führt die Schritte

$$r_{i-1} = q_i\, r_i + r_{i+1} \qquad \text{mit } r_0 = a,\ r_1 = b \text{ und } i = 1, \ldots, m$$

aus. Die Gesamtkosten dieser m Divisionen mit Rest ist von der Größenordnung

$$C = \sum_{i=1}^{m} l(q_i) l(r_i) \leq l(r_1) \left(\sum_{i=1}^{m} l(q_i) \right),$$

wobei $l(q_i) \sim l(r_{i-1}) - l(r_i)$ gilt, also insgesamt

$$C \leq l(a) l(b).$$

Der binäre gcd-Algorithmus. Beispiel: gcd(2 134 134,581 931)=27
Durchschnittliche Kosten: $(l(a) + l(b))$ Subtraktionen von Zahlen der maximalen Längen $l(a)$ und $l(b)$, also (average) $l(b)$ Elementaradditionen.

3 Primzahl-Tests und Faktorisierung langer Zahlen

3.1 Zahlentheoretische Vorbereitungen

Begriffe Restklassenring $R = \mathbf{Z}/n\mathbf{Z}$, prime Restklassen als Gruppe der invertierbaren Elemente in R. Eulersche ϕ-Funktion.
Satz (Chinesischer Restklassensatz):
Seien $m_1, \ldots, m_n$ paarweise teilerfremde natürliche Zahlen und $m = m_1 \cdot \ldots \cdot m_n$ deren Produkt. Das System von Kongruenzen

$$\begin{aligned} x &\equiv x_1 \ (mod\ m_1) \\ &\cdots \\ x &\equiv x_n \ (mod\ m_n) \end{aligned}$$

hat für jede Wahl von $(x_1, \ldots, x_n)$ *genau* eine Restklasse $x\ (mod\ m)$ als Lösung. Anders formuliert, die natürliche Abbildung

$$P : \mathbf{Z}/m\mathbf{Z} \longrightarrow \mathbf{Z}/m_1\mathbf{Z} \times \ldots \times \mathbf{Z}/m_n\mathbf{Z}$$

via $(x\ (mod\ m)) \mapsto (x\ (mod\ m_1), \ldots, x\ (mod\ m_n))$ ist ein Isomorphismus.
Beweis der Injektivität ist trivial. Beweis der Surjektivität über eine Lagrange-Interpolation.

Aufgabe: Löse $x \equiv 2\ (mod\ 11),\ x \equiv 5\ (mod\ 13),\ x \equiv 1\ (mod\ 17),\ x \equiv 3\ (mod\ 19),\ x \equiv 7\ (mod\ 23)$.
Ordnung eines Elements in einer Gruppe.
Satz von Lagrange: Ist H eine Untergruppe von G, so ist $|H|$ ein Teiler von $|G|$.
Folgerung (Satz von Euler): $(a, m) = 1$. Dann gilt

$$a^{\phi(m)} \equiv 1\ (mod\ m)$$

P induziert eine Gruppenisomorphismus

$$P^* : (\mathbf{Z}/m\mathbf{Z})^* \longrightarrow (\mathbf{Z}/m_1\mathbf{Z})^* \times \ldots \times (\mathbf{Z}/m_n\mathbf{Z})^*$$

Folgerung: Ist $m = p_1^{a_1} \ldots p_k a_k$ die Primfaktorzerlegung von m, so gilt

$$\phi(m) = \phi(p_1^{a_1}) \ldots \phi(p_k a_k).$$

Für Primzahlen p hat man $\phi(p^a) = p^a - p^{a-1}$.
Zusammengenommen erhält man die Formel

$$\phi(m) = \prod_{p|m,\ p\ prim} \left(1 - \frac{1}{p}\right)$$

Auf der Basis des Isomorphismus P kann man auch den Satz von Euler verfeinern. Zeige dazu folgendes
Lemma:
Das Gruppenelement $x = (x_1, \ldots, x_n) \in (\mathbf{Z}/m_1\mathbf{Z})^* \times \ldots \times (\mathbf{Z}/m_n\mathbf{Z})^*$ hat die Ordnung

$$ord(x) = kgV(ord(x_1), \ldots, ord(x_n)).$$

Insbesondere folgt damit und aus dem Satz von Euler, daß bereits

$$ord(x)\,|\,kgV(\phi(m_1), \ldots, phi(m_n))$$

gilt.
Folgerung (Satz von Carmichael):
Ist $m = p_1^{a_1} \ldots p_k a_k$ die Primfaktorzerlegung von m und

$$\psi(m) := kgV(\phi(m_1), \ldots, phi(m_n)),$$

so gilt

$$a^{\psi(m)} \equiv 1\ (mod\ m).$$

Dieses Ergebnis ist zugleich ziemlich optimal. Wir bezeichnen für eine Gruppe G das Maximum $max(ord(g)\ :\ g \in G)$ als Exponente $exp(G)$ der Gruppe.
Aufgabe: Zeigen Sie

$$\forall a, b \in G \exists c \in G\ :\ ord(c) = kgV(ord(a), ord(b))$$

Folgern Sie daraus, daß

$$\forall a \in G\ :\ ord(a)\,|\,exp(G).$$

Satz (o. Bew.):
Für $p > 2$ ist $G = (\mathbf{Z}/p^n\mathbf{Z})^*$ zyklisch, d.h. $exp(G) = |G|$.

- $G = (\mathbf{Z}/2^n\mathbf{Z})^*$ ist für $a = 1, 2$ zyklisch.
- Für $a > 2$ gilt $exp(G) = \frac{|G|}{2}$.

Folgerung:
Für ungerades m ist $\psi(m)$ die Exponente der Gruppe $G = (\mathbf{Z}/m\mathbf{Z})^*$.

3.2 Primzahl-Testverfahren

1. Methode: Probedivision

Kosten: $O(r \cdot l(m)^2)$, wobei r der kleinste Teiler von der zu untersuchenden Zahl m ist. Im Falle einer Primzahl ist der Algorithmus also exponentiell in der Bitlänge $l(m)$ der Eingangsgröße.

2. Methode: Der Fermat-Test als notwendiges Kriterium

Ist für eine Zahl a mit $(a, m) = 1$

$$a^{m-1} \not\equiv 1 \ (mod\ m),$$

so kann m keine Primzahl sein.

Kosten: $O(l(m)^3)$ mit binärem Potenzieren.

Ist also ein polynomiales Verfahren, mit dem man viele zusammengesetzte Zahlen als solche erkennt.

Lemma:

Die Menge der Restklassen

$$P_m := \{a \in (\mathbf{Z}/m\mathbf{Z})^* \ : \ a^{m-1} \equiv 1 \ (mod\ m)\}$$

ist eine Untergruppe der Gruppe der primen Restklassen $\mathbf{Z}/m\mathbf{Z})^*$.

Las Vegas-Test auf der Basis des Fermat-Tests.

Carmichael-Zahlen.

Lemma:

Die ungerade Zahl m ist genau dann Carmichael-Zahl, wenn $\psi(m)|m-1$.

3. Methode: Der Solovay-Strassen-Test

Quadratische Reste. Das Jacobi-Symbol.

Satz:

Ist $m > 2$ eine Primzahl, so gilt

1. Es gibt genau $\frac{m-1}{2}$ quadratische Reste und $\frac{m-1}{2}$ quadratische Nichtreste.
2. $(\mathbf{Z}/m\mathbf{Z})^* \to \{+1, -1\}$ via $a \mapsto \left(\frac{a}{m}\right)$ ist ein Gruppenhomomorphismus.
3. $$a^{\frac{m-1}{2}} \equiv \left(\frac{a}{m}\right) \ (mod\ m).$$

Folgerung: Ist $m > 2$ eine Primzahl, so gilt stets

$$a^{\frac{m-1}{2}} \cdot \left(\frac{a}{m}\right) \equiv 1 \ (mod\ m).$$

Ist andererseits die ungerade Zahl m keine Primzahl, so findet sich stets eine prime Restklasse a, für die $a^{\frac{m-1}{2}} \left(\frac{a}{m}\right) \not\equiv 1 \ (mod\ m)$ gilt. (o.Bew.)

Zur Berechnung des Jacobi-Symbols.

Solovay-Strassen-Test

Implementierung dieser Verfahren in in AXIOM-XL

3.3 Ein harter Primzahltest

Die ungerade Zahl m ist genau dann eine Primzahl, wenn $(\mathbf{Z}/m\mathbf{Z})^*$ eine zyklische Gruppe der Ordnung $m-1$ ist, d.h. wenn es ein Element $a \in (\mathbf{Z}/m\mathbf{Z})^*$ gibt, so daß $ord(a) = m-1$ gilt.

Faktorisiere dazu $m - 1 \Rightarrow \{p_1, \ldots, p_m\}$.

Lemma:

$$\exists\, a \in (\mathbf{Z}/m\mathbf{Z})^* \ \forall i \ a^{\frac{m-1}{p_i}} \not\equiv 1 \ (mod\ m)$$

genau dann, wenn

$$\forall i \ \exists\, a_i \in (\mathbf{Z}/m\mathbf{Z})^* \ a_i^{\frac{m-1}{p_i}} \not\equiv 1 \ (mod\ m)$$

Zertifizierte Primzahlen.
AXIOM-Implementierung CertifyPrime(n:Integer)
Einige Klassen von Primzahlen: Mersennesche Primzahlen, Fermatsche Primzahlen.

3.4 Faktorisierung langer Integer-Zahlen

1. Methode: Trialfactor

2. Methode: Pollards Rho-Methode und deren Verbesserung. Genaue Analyse der verwendeten Aussagen über Zufallszahlen.

Implementierung in AXIOM-XL

Ein komplexes Beispiel

4 Lineare Algebra

4.1 Klassische Verfahren bei Einheitskostenarithmetik

Klassische Fragestellungen: gegeben ist eine Matrix $M \in M_n(k)$ über einem Körper k. Gesucht sind Rang, Determinante, Dreiecksform, inverse Matrix, charakteristisches Polynom $p_M(x) := det(x \cdot E - M)$.

Betrachten wir die folgende Matrix:

Ihre Determinante kann man direkt über die Definition berechnen. Für ersteres wären $n!$ Produkte der Länge $n$auszurechnen, also $n! \cdot (n-1)$ Multiplikationen auszuführen. Nach der Stirlingschen Formel hat diese Anzahl die Größenordnung $O(e^{n \log(n)})$.

Wendet man die Leibniz-Regel an, d.h. berechnet die Determinante rekursiv durch Entwickeln nach einer (günstigen) Zeile oder Spalte, so erhält man ebenfalls eine rekursive Formel für die Zahl der dabei auszuführenden Multiplikationen. Bezeichnet $D(n)$ die Zahl der Multiplikationen, die dabei zur Berechnung einer n-reihigen Determinante auszuführen sind, so erhält man

$$D(n) = n(D_{n-1} + 1), \quad \text{mit } D(1) = 0$$

Die ersten Werte bekommt man zu

$$0, 2, 9, 40, 205, 1236, 8659, 69280$$

Aufgabe: Zeigen Sie, daß

$$D(n) = n! \left(\sum_{i=1}^{n-1} \frac{1}{i!} \right) \approx e \cdot n!$$

gilt.

Damit ist das Verfahren etwas besser als die Berechnung unmittelbar nach der Definition, aber immer noch exponentiell.

Determinanten kann man auch als Nebenprodukt der über den Gaußalgorithmus herstellbaren Dreiecksform einer Matrix berechnen. Für den Gaußalgorithmus auf einer $(m \times n)$-Matrix $(m \leq n)$ sind insgesamt $m-1$ Umformungsschritte auszuführen, wobei im Schritt k $m-k$ Zeilen und $n-k$ Spalten zu bearbeiten sind. Für jedes dieser Elemente sind 2 Multiplikationen/Divisionen auszuführen, also insgesamt

$$C_k := \sum_{k=1}^{m-1} 2(m-k)(n-k) = O(m^2 n)$$

Multiplikationsoperationen. Eine ähnliche Komplexität bekommt man für die Überführung in Diagonalform.

Aufgabe: Finden Sie den genauen Wert dieser Summe.

Damit kann man den Wert der Determinanten mit $O(n^3)$ Multiplikationen, also in polynomialer Zeit bestimmen. Der Gaußalgorithmus ist also in diesem Kontext anderen klassischen Verfahren haushoch überlegen, was man selbst an nicht zu großen selbstgewählten Beispielen nachvollziehen möge.

Ähnliche Komplexität erreicht man auf diesem Wege auch für die anderen Matrixoperationen: Multiplikation zweier n-reihiger Matrizen $C_{mult} = O(n^3)$, Rangbestimmung (als Nebenprodukt des Gaußalgorithmus) $C_{rank} = O(m^2 n)$, Bestimmung der inversen Matrix (mit einem erweiterten Diagonalform-Verfahren) $C_{inv} = O(n^3)$.

Selbst das charakteristische Polynom $p_M(x)$ kann man in polynomialer Zeit berechnen. Dazu bestimmt man $p_M(x)$ an $n+1$ Stellen $a_0, \ldots, a_n$ und ermittelt das gesuchte Polynom, von dem man ja weiß, daß es den Grad n hat, durch Interpolation. Insgesamt benötigt man für dieses Vorgehen $C_{charpoly} = O(n^4)$ Elementar-Multiplikationen.

4.2 Lineare Gleichungssysteme über Q

Beispiel:

Wir sehen an diesem Beispiel, daß beim Rechnen über **Q** die Größe der zu behandelnden Zahlen mit Fortschreiten des Algorithmus wächst und damit auch der Aufwand von Schritt zu Schritt zunimmt.

Schlimmstenfalls verdoppeln sich von Schritt zu Schritt die Längen der Matrixeinträge ($l(a/b) := max(l(a), l(b))$), so daß wir im Schritt k Einträge der Länge $2^{k-1} l$ zu bearbeiten haben, wobei l eine Schranke für die Länge der Einträge in der Matrix M ist. Analog obiger Formel erhalten wir für die Komplexität des Gaußalgorithmus in diesem Fall

$$C_{\mathbf{Q}} := \sum_{k=1}^{m-1} 2^{2k-1}(m-k)(n-k)l^2 = O(n \cdot 4^m),$$

d.h. exponentielles Wachstum in der Anzahl der Zeilen der Matrix.

Aufgabe: Finden Sie den genauen Wert dieser Summe.

4.3 Der Bareiss-Algorithmus

5 Polynomiale ggT-Berechnung

Die Berechnung polynomialer ggT ist eine der zentralen Aufgaben jedes Computeralgebrasystems. Hätte man keine solche Simplifikationsmöglichkeit zur Verfügung, so würde sich etwa bei der Addition rationaler Funktionen nach der Formel

$$\frac{a(x)}{b(x)} + \frac{c(x)}{d(x)} = \frac{a(x)d(x) + b(x)c(x)}{b(x)d(x)}$$

in jedem Schritt Zähler- und Nennergrad verdoppeln.

Wie wir in unserer allgemeinen Theorie gesehen hatten, kann man die ggT-Berechnung auch von multivariaten Polynomen immer auf den Euklidschen Algorithmus zurückführen. Es stellt sich allerdings heraus, daß der Euklidsche Algorithmus in seiner einfachen Form für die direkte Anwendung in komplexeren Aufgabestellungen nicht schnell genug ist. In den letzten 25 Jahren wurden eine Reihe von Verbesserungen gefunden, die im folgenden diskutiert werden sollen.

5.1 Der Euklidsche Algorithmus für Polynome

Für univariate Polynome über einem Körper k kann man den Euklidschen Algorithmus als eine Folge von Resten, die bei Division mit Rest entstehen, betrachten.

Beispiel: $k = \mathbf{Z}_{23}$. Wir betrachten die Polynome

$$f := x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5 \qquad g := 3x^6 + 5x^4 - 4x^2 - 9x + 21$$

und erhalten nacheinander

$$\begin{array}{l} 3x^6 + 5x^4 - 4x^2 - 9x + 21 \\ 2x^4 - 5x^2 - 8 \\ -x^2 - 9x + 2 \\ -8 + 10x \\ -4 \end{array}$$

Also sind f und g relativ prim im Polynomring $\mathbf{Z}_{23}[x]$.

Als Komplexitätsmaß können wir die Anzahl der Koeffizienten $l(f) := deg(f) + 1$ verwenden. Mit $deg(f), deg(g) < d$ erhalten wir genau wie bei den entsprechenden Algorithmen für ganze Zahlen

$C_{divrem} = l(g) \cdot (l(f) - l(g) + 1)$ für die Komplexität der Division mit Rest und

$C_{gcd} = O(d^2)$ für die Komplexität des Euklidschen Algorithmus.

Führt man dieselben Rechnungen im Ring $\mathbf{Q}[x]$ aus, so erhält man

$$\begin{array}{l} 3\,x^6 + 5\,x^4 - 4\,x^2 - 9\,x + 21 \\ -\frac{5\,x^4}{9} + \frac{x^2}{9} - \frac{1}{3} \\ -\frac{117\,x^2}{25} - 9\,x + \frac{441}{25} \\ -\frac{102500}{6591} + \frac{233150\,x}{19773} \\ -\frac{1288744821}{543589225} \end{array}$$

Wiederum erhalten wir als Ergebnis, daß f und g relativ prim sind, stellen aber ein unverhältnismäßiges Wachstum der an der Rechnung beteiligten Koeffizienten fest. In jedem Schritt wurden außerdem eine Reihe von ggT-Berechnungen im Bereich der ganzen Zahlen ausgeführt, denn in den Ergebnissen liegen stets gekürzte Brüche als Koeffizienten vor.

Schätzen wir den Rechenaufwand für diese ggT-Berechnungen für Polynome in $\mathbf{Q}[x]$ vom Grad $< d$ ab unter der Annahme, daß das Koeffizientenwachstum nicht zu schnell, genauer gesagt linear ist. Dann entsteht im Schritt k des Euklidschen Algorithmus ein Rest $r_k(x) \in \mathbf{Q}[x]$ vom Grad $< d - k$ mit rationalen Koeffizienten, deren (Maximum der Zähler- und Nenner-)Bitlänge etwa $k \cdot l$ ist, wenn l die Bitlänge der Koeffizienten des Ausgangspolynoms bezeichnet. Im Schritt k sind also $d - k$ ggT von ganzen Zahlen der Länge $k \cdot l$ zu berechnen. Als Gesamtaufwand erhalten wir

$$C(d,1) := \sum_{k=1}^{d} (d-k) * (kl)^2 = \left(\frac{d^4}{12} - \frac{d^2}{12}\right) l^2 = O(d^4 l^2)$$

Betrachten wir nun Polynome in mehreren Veränderlichen.
Beispiel: $R = \mathbf{Q}[x, y]$,

$$f := (y+1)\,x^3 + (y-1)\,x + y, \qquad g := (y+2)\,x^3 + (y+3)\,x^2 + 1$$

Der Euklidsche Algorithmus liefert in diesem Fall

$$(y+2)\,x^3 + (y+3)\,x^2 + 1$$

$$-\frac{y^2+4\,y+3}{y+2}x^2 - \frac{-y+2-y^2}{y+2}x - \frac{-y-y^2+1}{y+2}$$

$$\frac{-30\,y-16+11\,y^2+37\,y^3+19\,y^4+3\,y^5}{\left(y^2+4\,y+3\right)^2}x + \frac{4+14\,y+2\,y^5+13\,y^4+32\,y^2+31\,y^3}{\left(y^2+4\,y+3\right)^2}$$

$$-\frac{\left(y^2+4\,y+3\right)^2\left(y^6+16\,y^5+54\,y^4+49\,y^3-9\,y^2+4\,y+12\right)}{\left(3\,y^4+13\,y^3+11\,y^2-11\,y-8\right)\left(-30\,y-16+11\,y^2+37\,y^3+19\,y^4+3\,y^5\right)}$$

mit $C(d,1) = O(d^2)$. Untersuchen wir auch hier die Komplexität der Rechnungen unter plausiblen Annahmen. Wir betrachten dazu die ggT-Berechnung durch rekursive Anwendung des Euklidschen Algorithmus im Ring $R = k(x_1,\ldots,x_{n-1})[x_n]$ (über einem Körper k mit Einheitskostenarithmetik) auf Polynome vom Grad $< d$ in x_n. Bezeichne $C(d,n)$ die gesuchte Komplexität. Gibt $b(d,k,n)$ eine Schranke für den Grad der Polynome an, die als Zähler und Nenner der Koeffizienten aus $k(x_1,\ldots,x_{n-1})$ im k-ten Schritt auftreten und folgen denselben Überlegungen wie oben, so erhalten wir eine Rekursionsformel

$$C(d,n) := \sum_{k=1}^{d} (d-k) * C(b(d,k,n), n-1).$$

Eine sehr optimistische Annahme, die selbst in obigem Beispiel nicht erfüllt ist, geht davon aus, daß Zähler- und Nennergrad in jedem Schritt nur um einen konstanten Betrag, etwa 1, zunehmen. Setzen wir entsprechend $b(d,k,n) = d+k$, so erhalten wir

$$\begin{aligned}
C(d,2) &= \tfrac{11}{12}\,d^4 + \tfrac{d^3}{2} - \tfrac{5}{12}\,d^2\\
C(d,3) &= \tfrac{209}{120}\,d^6 + \tfrac{133}{120}\,d^5 - \tfrac{19}{12}\,d^4 + \ldots\\
C(d,4) &= \tfrac{51623}{6720}\,d^8 + \tfrac{323}{80}\,d^7 - \tfrac{399}{32}\,d^6 + \ldots
\end{aligned}$$

und allgemein $C(d,n) = O(d^{2n})$. Die Anzahl der Variablen geht also bereits hier exponentiell ein.

Eine weniger optimistische Annahme geht von linearem Wachstum der Grade aus, was noch immer bedeutet, daß sich aus Zähler und Nenner in späteren Schritten bedeutende Teile gegenüber einem nicht simplifizierenden Verfahren wegkürzen müßten. Setzen wir $b(d,k,n) = d*k$, so erhalten wir

$$\begin{aligned}
C(d,2) &= \tfrac{d^6}{12} - \tfrac{d^4}{12}\\
C(d,3) &= \tfrac{d^{14}}{672} - \tfrac{d^{12}}{144} + \tfrac{11}{1440}\,d^{10} + \tfrac{d^8}{504} - \tfrac{d^6}{240}\\
C(d,4) &= \tfrac{d^{30}}{161280} - \tfrac{d^{28}}{8064} + \tfrac{127}{116480}\,d^{26} + \ldots
\end{aligned}$$

und allgemein $C(d,n) = O(d^{c(n)})$ mit $c(n) = 2\cdot c(n-1)+2$, also $C(d,n) = O(d^{2^n})$. Selbst in diesem noch relativ guten Szenario geht die Anzahl der Variablen doppelt exponentiell ein.

Aufgabe: Finden Sie eine explizite Formel für $c(n)$.

Die letzten Überlegungen wurden der Einfachheit halber über einem Körper k mit Einheitskostenarithmetik angestellt. Verwenden wir dagegen ganzzahlige Koeffizienten, so sind die Ergebnisse noch zu modifizieren. Generell haben wir folgende Analogie der Komplexitätsberechnungen in $\mathbf{Z}$ und in $k[x]$: In beiden Fällen haben wir eine Längenfunktion $l(a)$ (die Bitlänge bzw. die Zahl der Terme in einem dichten Polynom), deren Verhalten bei der Multiplikation von a und b sich in beiden Fällen wie folgt beschreiben läßt:

Bitlänge des Ergebnisses: $O(l(a)+l(b))$,

Kosten für die Multiplikation: $O(l(a)\cdot l(b))$

Damit erhalten wir für Polynome mit ganzzahligen Koeffizienten Ergebnisse, die denen von Polynomen über einem Körper mit Einheitskostenarithmetik, aber einer zusätzlichen Variablen, gleichen. Das ist auch plausibel, wenn wir uns die Darstellung ganzer Zahlen in einem Positionssystem zur Basis β ins Gedächtnis rufen.

Beide Funktionen geben zugleich auch ein Maß für den zur Speicherung des entsprechenden Datums notwendigen Platz an. Im Polynomring $k[x_1, \ldots, x_n]$ hat ein (dichtes) Polynom f vom (Multi-)Grad $Deg(f) = (d_1, \ldots, d_n)$ insgesamt $O(d_1 \cdot \ldots \cdot d_n)$ Terme und benötigt damit $L(f) = O(d_1 \cdot \ldots \cdot d_n \cdot b)$ Einheiten Speicherplatz, wenn $b = b(f)$ die durchschnittliche Bitlänge ist, die für die Darstellung eines Koeffizienten von f erforderlich ist. Die Bitlänge der Koeffizienten tritt hier wie eine zusätzliche Variable in Erscheinung, was ein weiteres Argument für obige Heuristik gibt. $L(f)$ bezeichnen wir als die *Seminorm* des Polynoms f. Wegen $L(f \cdot g) = O(\prod(Deg(f) + Deg(g)) \cdot (b(f) + b(g)))$ gilt obige Linearität für dieses Funktional nicht mehr. Andererseits benötigen wir wieder $O(L(f)L(g))$ Elementarmultiplikationen zur Berechnung des Produkts der Polynome f und g.

5.2 Polynomiale Restsequenzen

Wir untersuchen die Berechnung von ggT im Polynomring $R[x]$ über einem Integritätsbereich R, in dem wir bereits wissen, wie ggT zu berechnen sind. Ist $K = Q(R)$ der Quotientenkörper von R, sohatten wir gesehen, daß

$$gcd_{R[x]}(f,g) = gcd_R(cont(f), cont(g)) \cdot primPart(gcd_{K[x]}(f,g))$$

gilt, wobei man $gcd_{K[x]}(f,g)$ etwa über den Euklidschen Algorithmus berechnen kann.

Um zu effektiveren Verfahren zu kommen, wollen wir deshalb zuerst versuchen, die Rechnungen nicht in $K[x]$, sondern in $R[x]$ auszuführen.

Seien dazu $f(x), g(x) \in R[x]$. Betrachten wir die Division mit Rest $divrem(f,g)$. In der klassischen Version über K entsteht als Nenner eine Potenz von $lc(g)$, weil in jedem Schleifendurchlauf durch diesen Leitkoeffizienten geteilt wird. Da die Schleife maximal $d := deg(f) - deg(g) + 1$ mal durchlaufen wird, können wir durch vorherige Multiplikation mit $lc(g)^d$ erreichen, daß keine Nenner entstehen. In praktischen Anwendungen kann dieser Exponent kleiner gewählt werden, wenn während eines Schleifendurchlaufs der Grad um mehr als 1 sinkt.

Definition: Ist R ein Integritätsbereich und $f(x), g(x) \in R[x], deg\ f \geq deg\ g$, so gibt es eindeutig bestimmte Polynome $q(x), r(x) \in R[x]$ mit $r = 0$ oder $deg\ r < deg\ g$ und

$$lc(g)^{deg(f)-deg(g)+1} f(x) = q(x) \cdot g(x) + r(x).$$

q nennt man den *Pseudoquotienten* $pquot(f,g)$ und r nennt man den *Pseudorest* $prem(f,g)$ der Polynome f und g.

Entsprechende REDUCE-Prozeduren haben das folgende Aussehen:
Eine Prozedur, die nur mit der wirklich notwendigen $lc(g)$-Potenz multipliziert:

```
algebraic procedure prem(f,g);
      begin
      while f neq 0 and ((d:=deg(f,x)-deg(g,x)) geq 0) do
            f:=lcof(g,x)*f-lcof(f,x)*x^d*g;
      return f;
      end;
```

Eine Prozedur entsprechend der Definition:

```
algebraic procedure prem(f,g);
```

```
begin
integer n; n:=deg(f,x)-deg(g,x)+1;
while f neq 0 and ((d:=deg(f,x)-deg(g,x)) geq 0) do
     << f:=lcof(g,x)*f-lcof(f,x)*x^d*g; n:=n-1>>;
return f*lcof(g,x)^n;
end;
```

Bemerkung: Über $K[x]$ sind der übliche Rest und der Pseudorest zueinander assoziierte Polynome. Also kann man bei der Berechnung von $gcd_{K[x]}(f,g)$ mit Hilfe des Euklidschen Algorithmus die Restbildung durch Pseudorestbildung ersetzen. Die Folge der dabei entstehenden Reste bezeichnet man als die *Euklidsche Polynomiale Restsequenz*. Hier einige Beispiele ($r_0 = f$, $r_1 = g$):

$$\begin{aligned} f &:= x^8 + x^6 - 3\cdot x^4 - 3\cdot x^3 + 8\cdot x^2 + 2\cdot x - 5, \\ g &:= 3\cdot x^6 + 5\cdot x^4 - 4\cdot x^2 - 9\cdot x + 21 \end{aligned}$$

$$\begin{aligned} r_2 &= -5\cdot x^4 + x^2 - 3 \\ r_3 &= 9\cdot(-13\cdot x^2 - 25\cdot x + 49) \\ r_4 &= 36450\cdot(-4663\cdot x + 6150) \\ r_5 &= -1712229591042652500 \end{aligned}$$

also $gcd(f,g) = 1$.

$$f = 3\cdot x^6 + x^3 + x + 5, \quad g = 5\cdot x^5 - 3\cdot x^4 + 1$$

$$\begin{aligned} r_2 &= 27\cdot x^4 + 25\cdot x^3 + 10\cdot x + 116 \\ r_3 &= 5150\cdot x^3 - 1350\cdot x^2 - 13600\cdot x + 24625 \\ r_4 &= 2114100000\cdot x^2 - 912161250\cdot x - 991440000 \\ r_5 &= -4830802248163570312500\cdot x + 1118872524568050000000 \\ r_6 &= 1922193696103275798684425644447126464843750000000000000 \end{aligned}$$

also wiederum $gcd(f,g) = 1$. Mit $R := Z[y]$ erhalten wir im obigen Beispiel

$$f := (y+1)\,x^3 + (y-1)\,x + y, \qquad g := (y+2)\,x^3 + (y+3)\,x^2 + 1$$

die Euklidsche Restsequenz

$$\begin{aligned} r_2 &= x^2\cdot(-y^2 - 4\cdot y - 3) + x\cdot(y^2 + y - 2) + y^2 + y - 1 \\ r_3 &= x\cdot(3\cdot y^5 + 19\cdot y^4 + 37\cdot y^3 + 11\cdot y^2 - 30\cdot y - 16) \\ &\quad +2\cdot y^5 + 13\cdot y^4 + 31\cdot y^3 + 32\cdot y^2 + 14\cdot y + 4 \\ r_4 &= -y^{12} - 28\cdot y^{11} - 304\cdot y^{10} - 1769\cdot y^9 - 6208\cdot y^8 - 13734\cdot y^7 - 19164\cdot y^6 \\ &\quad -16241\cdot y^5 - 7947\cdot y^4 - 3076\cdot y^3 - 2520\cdot y^2 - 1728\cdot y - 432, \end{aligned}$$

also ebenfalls $gcd(f,g) = 1$.

Das Ergebnis ist im Hinblick auf Vermeidung von Rechenaufwand auf den ersten Blick allerdings nicht sehr ermutigend, denn die entstehenden Koeffizienten sind deutlich größer als die, die bei der rationalen Version auftreten. Eine genauere Analyse der in den Restsequenzen auftretenden Koeffizienten zeigt wie bereits im Fall linearer Gleichungssysteme, daß sich diese gewöhnlich aus vielen Faktoren zusammensetzen. Es entsteht also die Frage, ob man Faktoren finden kann, die in allen Koeffizienten eines Restpolynoms gemeinsam vorkommen. Diese könnte man dann ausdividieren, bevor die Rechnung fortgesetzt wird.

Definition: Seien $f(x), g(x) \in R[x]$ zwei Polynome mit $deg\ f \geq deg\ g$. Als *Polynomiale Restsequenz* (PRS) bezeichnet man eine Folge von Polynomen

$$R_0 = f,\ R_1 = g,\ R_2,\ R_3,\ \ldots,\ R_k \neq 0$$

mit

(1) $deg\ R_1 > deg\ R_2 >\ \ldots,\ > deg\ R_k$.

(2) $\alpha_i \cdot R_{i-1}(x) = Q_i(x) \cdot R_i(x) + \beta_i \cdot R_{i+1}(x)$
mit geeigneten $\alpha_i, \beta_i \in R$ und $Q_i(x) \in R[x]$.

(3) $prem(R_{k-1}, R_k) = 0$

Wie oben gilt dann stets

$$primPart(R_k) = primPart(gcd_{K[x]}(f, g)).$$

Sei weiter $\delta_i := deg(R_{i-1}) - deg(R_i)$. Dann nimmt man gewöhnlich $\alpha_i = lc(R_i)^{\delta_i+1}$
Zur Komplexität dieser Berechnungen: Sei $a \in R = k[x_1, \ldots, x_{n-1}]$ und $Deg(a) = (d_1, \ldots, d_{n-1})$ die oben eingeführte lineare "Längen"-funktion. Für die in Komplexitätsabschätzungen auftretende Seminorm gilt dann $L(a) = \prod Deg(a)$. Haben die Ausgangspolynome f, g Koeffizienten der Länge $l = (d_1, \ldots, d_{n-1})$, so folgt

$$l_{i+1} = l(\alpha_i) + l_{i-1} - l(\beta_i) = (\delta_i + 1)l_i + l_{i-1} - l(\beta_i)$$

mit $l_0 = l_1 = l, l_2 = (\delta_1 + 1)l$, wenn $l_i := l(lc(R_i))$ die Länge des Leitkoeffizienten des Rests R_i bezeichnet.

1) $\alpha_i = lc(R_i)^{\delta_i+1}, \beta_i = 1$ liefert die *Euklidsche PRS*. Für den allgemeinen Fall $\delta_i = 1$ (d.h. der Grad in der Restsequenz fällt in jedem Schritt um genau 1; solche PRS heißen *normal*) ergibt sich deren Koeffizientenwachstum damit aus der Rekursion

$$l_{i+1} = 2l_i + l_{i-1} \quad \text{mit} \quad l_1 = l, l_2 = 2l$$

Wir erhalten

$$[l_i, i = 1..10] = [1, 2, 5, 12, 29, 70, 169, 408, 985, 2378] \cdot l$$

und allgemein $l_i = 2^{O(i)} \cdot l$, also exponentielles Koeffizientenwachstum.
Aufgabe: Finden Sie das genaue Wachstum $l_i = O(?)$

2) Man könnte auch in jedem Schritt $\beta_i = cont(R_{i+1})$ ausdividieren. Das wäre der größtmögliche Faktor. Für obige Beispiele erhalten wir die folgenden Restsequenzen

1)

$$\begin{aligned}
r_2 &= -5 \cdot x^4 + x^2 - 3 \\
r_3 &= -13 \cdot x^2 - 25 \cdot x + 49 \\
r_4 &= -4663 \cdot x + 6150 \\
r_5 &= 1
\end{aligned}$$

2)

$$\begin{aligned}
r_2 &= 27 \cdot x^4 + 25 \cdot x^3 + 10 \cdot x + 116 \\
r_3 &= 206 \cdot x^3 - 54 \cdot x^2 - 544 \cdot x + 985 \\
r_4 &= 2320 \cdot x^2 - 1001 \cdot x - 1088 \\
r_5 &= -109673 \cdot x + 254016 \\
r_6 &= 1
\end{aligned}$$

3)

$$\begin{aligned}
r_2 &= x^2 \cdot (-y^2 - 4 \cdot y - 3) + x \cdot (y^2 + y - 2) + y^2 + y - 1 \\
r_3 &= x \cdot (3 \cdot y^4 + 13 \cdot y^3 + 11 \cdot y^2 - 11 \cdot y - 8) \\
&\quad +2 \cdot y^4 + 9 \cdot y^3 + 13 \cdot y^2 + 6 \cdot y + 2 \\
r_4 &= 1
\end{aligned}$$

Für den Rechenaufwand gilt allerdings das oben Gesagte, d.h. bei selbst weitgehenden Annahmen über ein langsames Koeffizientenwachstum ist die Laufzeit dieses Verfahrens doppelt exponentiell in der Anzahl der Variablen.

Es entsteht deshalb die Frage, ob man ähnlich wie beim Bareiss-Algorithmus Faktoren, die man in früheren Schritten hineingesteckt hat, in späteren Schritten wieder herausdividieren kann. Dies ist in der Tat möglich:

Satz (Silvester 1853, Collins 1967):

$$\alpha_i = lc(R_i)^{\delta_i+1}, \qquad \beta_1 = 1, \beta_i = \alpha_{i-1} \text{ für } i > 1$$

liefert eine PRS, die *reduzierte Polynomiale Rest-Sequenz.*

Für einen Beweis dieses Satzes sei auf [2, ch. 7] bzw. [6] verwiesen. Von seiner Evidenz kann man sich an Hand der folgenden REDUCE-Implementierung überzeugen:

```
algebraic procedure redPRS(f,g);
     begin scalar r,b; b:=1;
     while not (g = 0) do
          << r:=prem(f,g)/b; write ''r='',r;
          b:=lcof(g,x)^(deg(f,x)-deg(g,x)+1);
          f:=g; g:=r;>>;
     return f; end;
```

Aufgabe: Überzeugen Sie sich von der Richtigkeit des Satzes an Hand selbstgewählter Beispiele.

Für unsere drei Beispiele erhalten wir jedenfalls

1)

$$\begin{aligned}
r_2 &= 3 \cdot (-5 \cdot x^4 + x^2 - 3) \\
r_3 &= 45 \cdot (13 \cdot x^2 + 25 \cdot x - 49) \\
r_4 &= 4050 \cdot (-4663 \cdot x + 6150) \\
r_5 &= 527933700
\end{aligned}$$

2)

$$\begin{aligned}
r_2 &= 27 \cdot x^4 + 25 \cdot x^3 + 10 \cdot x + 116 \\
r_3 &= 206 \cdot x^3 - 54 \cdot x^2 - 544 \cdot x + 985 \\
r_4 &= 2 \cdot (2320 \cdot x^2 - 1001 \cdot x - 1088) \\
r_5 &= 2 \cdot (-109673 \cdot x + 254016) \\
r_6 &= 40399367
\end{aligned}$$

3)

$$\begin{aligned}
r_2 &= x^2 \cdot (-y^2 - 4 \cdot y - 3) + x \cdot (y^2 + y - 2) + y^2 + y - 1 \\
r_3 &= x \cdot (3 \cdot y^4 + 13 \cdot y^3 + 11 \cdot y^2 - 11 \cdot y - 8) + 2 \cdot y^4 + 9 \cdot y^3 + 13 \cdot y^2 + 6 \cdot y + 2 \\
r_4 &= -y^6 - 16 \cdot y^5 - 54 \cdot y^4 - 49 \cdot y^3 + 9 \cdot y^2 - 4 \cdot y - 12
\end{aligned}$$

Für normale Restsequenzen, d.h. wenn $\delta_i = 1$ für alle $i > 1$ gilt erhalten wir für das Koeffizientenwachstum

$$l_{i+1} = 2l_i + l_{i-1} - 2l_{i-1} = 2l_i - l_{i-1} \quad \text{mit} \quad l_1 = l, l_2 = 2l.$$

Aufgabe: Zeigen Sie, daß in diesem Fall $l_i = i\,l$ gilt, d.h. das Koeffizientenwachstum linear ist.

Liegt dagegen eine Restsequenz mit größeren Gradsprüngen vor, so kann das Koeffizientenwachstum noch immer exponentiell sein.

Aufgabe: Leiten Sie für den Fall, daß alle $\delta_i = 2$ sind, eine Formel für l_i her und überzeugen Sie sich, daß diese exponentiell in i ist.

Ein noch besseres Verhalten legt die *Subresultanten-PRS* an den Tag:

Satz (Collins 1967, Brown 1971): $\alpha_i = lc(R_i)^{\delta_i+1}$ und

$$h_1 = 1, \quad h_i = lc(R)^{\delta_{i-1}} \cdot h_{i-1}^{1-\delta_{i-1}} \quad (i > 1),$$

$$\beta_1 = 1, \quad \beta_i = lc(R_{i-1}) \cdot h_i^{\delta_i} \quad (i > 1)$$

definiert eine Polynomiale Restsequenz, die *Subresultanten-PRS.*

===> Beweis wieder durch Evidenz an Beispielen.

Bemerkung: Für normale Restsequenzen fällt diese PRS mit der reduzierten PRS zusammen. Es stellt sich allerdings heraus, daß hier für beliebige Gradsprünge lineares Koeffizientenwachstum zu verzeichnen ist.

Aufgabe: Zeigen Sie, daß beim Subresultantenalgorithmus das Koeffizientenwachstum immer linear ist, indem Sie aus den Rekursionsbeziehungen $l_i = (\delta_1 + \ldots + \delta_{i-1} + 1)\, l$ und $l(h_i) = l_i - l$ herleiten.

Als Maß für die Komplexität haben wir in unseren Betrachtungen bisher stets die Länge der entstehenden Koeffizienten verwendet. Untersuchen wir deshalb zum Schluß noch die tatsächliche Komplexität der Berechnung einer PRS unter der Annahme, daß $\delta_i = 1$ und $l_i = i\, l$ für alle $i > 1$ gilt, also $deg(R_i) = d - i$ mit $d = deg(f) - 1 = deg(g)$. Mit $l = (d_1, \ldots, d_{n-1})$ und $L(l) = \prod_{i=1}^{n-1} d_i$ erhalten wir für die Komplexität der Berechnung von R_{i+1} aus

$$\alpha_i \cdot R_{i-1} = Q_i \cdot R_i + \beta_i \cdot R_{i+1}$$

den Ausdruck

$$\begin{aligned} s_i &= (d-i+2)L(l_{i-1})L(2l_i) + 2(d-i+1)L(l_i + l_{i-1})L(l_i) + (d-i)L(2l_{i-1})L(l_{i+1}) \\ &= (d-i+2)L((i-1)l)L(2il) + 2(d-i+1)L((2i-1)l)L(il) + \\ &\qquad (d-i)L(2(i-1)l)L((i+1)l) \end{aligned}$$

und schließlich wegen $L(i\, l) = i^{n-1} L(l)$

$$\begin{aligned} s_i = ((d-i+2)(2(i-1)i)^{n-1} + 2(d-i+1)((2i-1)i)^{n-1} + \\ (d-i)(2(i^2-1))^{n-1})L(l)^2 \end{aligned}$$

Summieren wir den Aufwand der einzelnen Schritte, und berücksichtigen, daß die Anzahl der Terme $L(f)$ im Polynom $f \in k[x_1, \ldots, x_n]$ von der Ordnung $L(f) = O(d \cdot L(l)) = O(d_1 \cdot \ldots \cdot d_n)$ ist, so erhalten wir

$$\sum_{i=1}^{d-1} s_i = O(d^{2n-2} L(f)^2).$$

Satz: Sei $R[x_n]$ der univariate Polynomring über $R = k[x_1, \ldots, x_{n-1}]$ und einem Grundkörper k mit Einheitskostenarithmetik und $f, g \in R[x_n]$ mit durch $(d_1, \ldots, d_n)$ beschränkten Gradvektoren $Deg(f), Deg(g)$. Zur Berechnung der Subresultanten-PRS werden dann $O((d_1 \cdots d_n)^2 (d_n)^{2n-2})$ Elementarmultiplikationen benötigt.

Aufgabe: Zeigen Sie, daß die entsprechende Komplexität für $b(f), b(g) \leq b$ über $R = \mathbf{Z}$ von der Größenordnung $O(b^2 d^4)$ und über $R = \mathbf{Z}[x_1, \ldots, x_{n-1}]$ von der Größenordnung

$O((b \cdot d_1 \cdots d_n)^2 (d_n)^{2n})$ ist. Dies bestätigt die oben beschriebene Heuristik, daß die Berücksichtigung der Koeffizientenlänge der Hinzunahme einer zusätzlichen Variablen entspricht, ein weiteres Mal.

Dies ist allerdings noch nicht der Gesamtaufwand für die Berechnung des ggT, da nach der Formel

$$gcd_{R[x]}(f,g) = gcd_R(cont(f), cont(g)) \cdot primPart(gcd_{K[x]}(f,g))$$

noch die Berechnung sowohl des Inhalts-ggT als auch des primitiven Teils aussteht. Dies führt auf die Berechnung von maximal $3d_n$ ggT's in $n-1$ Variablen. Bezeichnet $C_{gcd}(d_1,\ldots,d_n)$ die Komplexität der ggT-Berechnung von Polynomen $f,g \in k[x_1,\ldots,x_n]$, deren Grad $Deg(f), Deg(g)$ durch $\mathbf{d} = (d_1,\ldots,d_n)$ beschränkt ist, so erhalten wir für den Gesamtaufwand der ggT-Berechnung

$$C_{gcd}(d_1,\ldots,d_n) \le 3d_n \cdot C_{gcd}(d_1,\ldots,d_{n-1}) + O((d_1 \cdots d_n)^2 (d_n)^{2n-2})$$

Aufgabe: Zeigen Sie, daß $C_{gcd}(d_1,\ldots,d_n) = O(max_i\{(d_1 \cdots d_n)^2 (d_i)^{2i-2}\})$.

Satz: Der Gesamtaufwand für die ggT-Berechnung von $f,g \in k[x_1,\ldots,x_n]$ über einem Körper k mit Einheitskostenarithmetik mit durch $(d_1,\ldots,d_n)$ beschränkten Gradvektoren $Deg(f), Deg(g)$ über die Subresultanten-PRS ist von der Größenordnung

$$O(max_i\{(d_1 \cdots d_n)^2 (d_i)^{2i-2}\}).$$

Insbesondere bestätigt diese Schranke, daß man die Variablen so anordnen sollte, daß $d_1 \ge d_2 \ge \ldots \ge d_n$ gilt.

5.3 Modulare ggT-Berechnung

Die bisherigen Verfahren zur ggT-Berechnung zeichnen sich durch außerordentliches Wachstum der intermediären Daten aus. Dieser für viele symbolische Rechnungen typische Effekt läßt sich manchmal dadurch umgehen, daß man die entsprechenden symbolischen Rechnungen statt mit Unbestimmten mit konkreten Parameterwerten ausführt und aus diesem Ergebnis z.B. durch Interpolationstechniken versucht, das entsprechende korrekte symbolische Ergebnis zu gewinnen.

Bei der Berechnung größter gemeinsamer Teiler trat Koeffizientenwachstum besonders hinderlich für Polynome mit ganzzahligen Koeffizienten bzw. für multivariate Polynome auf. Entsprechende Rechnungen wollen wir deshalb im ersten Fall mit modularen Koeffizienten, d.h. modulo einer Primzahl, und im zweiten Fall nach Substitution konkreter Werte für $x_1,\ldots,x_{n-1}$ ausführen.

Betrachten wir zunächst einige Beispiele:

1) Das folgende Beispiel führte bei der Berechnung polynomialer Restsequenzen über $\mathbf{Z}$ zu langen Koeffizienten:

$$f = 3 \cdot x^6 + x^3 + x + 5, \quad g = 5 \cdot x^5 - 3 \cdot x^4 + 1$$

Berechnet man dieselben Sequenzen über modularen Bereichen, so erhält man

mod 2:

$$\begin{aligned} r_0 &= f = x^6 + x^3 + x + 1, \\ r_1 &= g = x^5 + x^4 + 1, \\ r_2 &= x^4 + x, \\ r_3 &= 1 \end{aligned}$$

mod 3:

$$\begin{aligned} r_0 &= f = x^3 + x + 2, \\ r_1 &= g = 2 \cdot x^5 + 1, \\ r_2 &= x^3 + x + 2, \\ r_3 &= x^2 + x + 1, \\ r_4 &= 1 \end{aligned}$$

mod 7:

$$\begin{aligned}
&f = 3 \cdot x^6 + x^3 + x + 5,\\
&g = 5 \cdot x^5 + 4 \cdot x^4 + 1,\\
&r_2 = x^4 + 3 \cdot x^3 + 4 \cdot x + 3,\\
&r_3 = x^3 + 3 \cdot x^2 + 3 \cdot x + 4,\\
&r_4 = x^2 + 6,\\
&r_5 = x,\\
&r_6 = 1
\end{aligned}$$

Stets ist das Ergebnis dasselbe wie über den ganzen Zahlen, wenn auch unterschiedlich lange Restsequenzen zu bilden waren: f und g sind auch im Ring $\mathbf{Z}_p[x]$ zueinander teilerfremd.

2)

$$\begin{aligned}
&f := x^8 + x^6 - 3 \cdot x^4 - 3 \cdot x^3 + 8 \cdot x^2 + 2 \cdot x - 5,\\
&g := 3 \cdot x^6 + 5 \cdot x^4 - 4 \cdot x^2 - 9 \cdot x + 21
\end{aligned}$$

Wir erhalten folgende größten gemeinsamen Teiler für die verschiedenen modularen Bereiche:

$$\begin{array}{lcl}
\text{gcd mod } 2 & : & x^2 + x + 1,\\
\text{gcd mod } 3 & : & 1,\\
\text{gcd mod } 5 & : & 1,\\
\text{gcd mod } 7 & : & x + 3,\\
\text{gcd mod } 11 & : & 1,\\
\text{gcd mod } 13 & : & 1
\end{array}$$

3) Untersuchen wir nun ein Beispiel, das einen nichttrivialen gemeinsamen Teiler $5x + 1$ im Ring $\mathbf{Z}[x]$ hat:

$$\begin{aligned}
&f = 5 \cdot x^9 + x^8 + 5 \cdot x^7 + x^6 - 15 \cdot x^5 - 18 \cdot x^4 + 37 \cdot x^3 + 18 \cdot x^2 - 23 \cdot x - 5,\\
&g = 15 \cdot x^7 + 3 \cdot x^6 + 25 \cdot x^5 + 5 \cdot x^4 - 20 \cdot x^3 - 49 \cdot x^2 + 96 \cdot x + 21
\end{aligned}$$

Wir erhalten in diesem Fall über den verschiedenen modularen Bereichen

$$\begin{array}{lcl}
\text{gcd mod } 2 & : & x^3 + 1,\\
\text{gcd mod } 3 & : & x + 2,\\
\text{gcd mod } 5 & : & 1,\\
\text{gcd mod } 7 & : & x^2 + 6 \cdot x + 2,\\
\text{gcd mod } 11 & : & x + 9,\\
\text{gcd mod } 13 & : & x + 8
\end{array}$$

d.h. ein meist wenigstens vom Grad her korrektes Ergebnis.

Wir wollen nun zunächst die zu untersuchende Situation allgemein mathematisch beschreiben. Seien R und R' zwei Integritätsbereiche, die durch einen Homomorphismus $\phi : R \to R'$ verbunden sind. Ein solcher Homomorphismus induziert durch seine Wirkung auf den Koeffizienten einen Ringhomomorphismus $\phi : R[x] \to R'[x]$, den wir mit demselben Symbol bezeichnen wollen.

Satz: Sei $\phi : R[x] \to R'[x]$ wie eben beschrieben und $f(x), g(x) \in R[x]$. Ist $\phi(lc(gcd(f,g))) \neq 0$, so gilt

$$deg\ gcd(\phi(f), \phi(g)) \geq deg\ gcd(f, g).$$

In der Voraussetzung kommt der zu berechnende $gcd(f,g)$ vor. Berücksichtigt man $lc(gcd(f,g)) \,|\, gcd(lc(f), lc(g))$,so kann man diese Bedingung durch die schärfere, aber überprüfbare Bedingung $\phi(gcd(lc(f), lc(g))) \neq 0$ oder gar $\phi(lc(f) \cdot lc(g)) \neq 0$ ersetzen. Beispiel 3) zeigt, daß im Fall $\phi(gcd(lc(f), lc(g))) = 0$ der Grad des ggT in der Tat kleiner als erwartet sein kann. Immer gibt es nur endlich viele Primzahlen, für die man den Satz deshalb nicht anwenden kann und diese Primzahlen kann man aus den Eingabegrößen sofort direkt ablesen.

Folgerung: Sind unter obigen Voraussetzungen $\phi(f)$ und $\phi(g)$ teilerfremd, so auch f und g.

Das erlaubt es uns in vielen Fällen, schnell zu beweisen, daß zwei Polynome zueinander teilerfremd sind. Betrachten wir etwa die beiden Polynome

$$f(x,t) = \sum_{i=0}^{7} (t+i) \cdot x^i,$$
$$g(x,t) = \sum_{i=0}^{6} (t-i^2) \cdot x^i$$

im bivariaten Polynomring $\mathbf{Z}[t,x] = \mathbf{Z}[t][x]$ und den Ringhomomorphismus $\phi_{t=1} : \mathbf{Z}[t] \to \mathbf{Z}$, der durch die Zuordnung $t \mapsto 1$ induziert wird. Wir erhalten

$$\phi(f) = \sum_{i=0}^{7} (i+1) \cdot x^i = 8 \cdot x^7 + 7 \cdot x^6 + 6 \cdot x^5 + 5 \cdot x^4 + 4 \cdot x^3 + 3 \cdot x^2 + 2 \cdot x + 1,$$
$$\phi(g) = \sum_{i=0}^{6} (1-i^2) \cdot x^i = -35 \cdot x^6 - 24 \cdot x^5 - 15 \cdot x^4 - 8 \cdot x^3 - 3 \cdot x^2 + 1$$

Da die Voraussetzung des Satzes wiederum erfüllt ist, können wir aus der Teilerfremdheit von $\phi(f), \phi(g)$ auf die Teilerfremdheit der beiden Ausgangspolynome schließen.

Beweis des Satzes: Sei $h = gcd(f,g)$, d.h. $f = h \cdot f_1$ und $g = h \cdot g_1$. Dann gilt auch $\phi(f) = \phi(h) \cdot \phi(f_1)$ und $\phi(g) = \phi(h) \cdot \phi(g_1)$. Also ist $\phi(h)$ ein gemeinsamer Teiler von $\phi(f)$ und $\phi(g)$ und somit

$$deg\ gcd(\phi(f), \phi(g)) \geq deg\ \phi(h) = deg\ h$$

wegen der Bedingung an $lc(h)$. □

In unseren Beispielen hatten wir andererseits gesehen, daß in der Tat $deg\ gcd(\phi(f), \phi(g)) > deg\ gcd(f,g)$ eintreten kann. Mehr noch ist in obigen Beispielen keine direkte Verbindung zwischen den Eingangsdaten und dem Eintreten eines solchen Gradsprungs zu erkennen. Morphismen ϕ mit dieser Eigenschaft nennen wir *unglücklich*.

5.3.1 Der Übergang von ganzzahligen zu modularen Koeffizienten

Zur Berechnung des ggT von Polynomen mit ganzzahligen Koeffizienten betrachten wir den natürlichen Homomorphismus $\phi : \mathbf{Z} \longrightarrow \mathbf{Z}_p$ des Übergangs zu Restklassen modulo einer Primzahl p. Damit werden die Rechnungen auf solche mit Polynomen mit modularen Koeffizienten zurückgeführt. Da letzterer Koeffizientenbereich ein Körper (mit Einheitskostenarithmetik) ist, kann man den Euklidschen Algorithmus direkt anwenden. Für $f \in \mathbf{Z}[x]$ wollen wir dabei $\phi_p(f)$ kurz durch f_p bezeichnen.

Satz: In dieser Situation gibt es nur endlich viele unglückliche Primzahlen.

Beweis: Betrachten wir eine PRS zur gcd-Berechnung über $\mathbf{Z}[x]$. Wenn $p \in \mathbf{Z}$ keinen der Leitkoeffizienten $lc(R_i)$ teilt, dann liefert diese Sequenz eine Restfolge auch über $\mathbf{Z}_p[x]$. Damit ist in diesem Fall $deg\ gcd(\phi(f), \phi(g)) = deg\ gcd(f,g)$. □

Verwendet man Primzahlen p etwa in der Wortgröße des entsprechenden Computers, so ist die Wahrscheinlichkeit, daß man auf eine solche Primzahl trifft, außerordentlich gering.

Man kann nun versuchen, den gcd über $\mathbf{Z}$ aus den Werten des gcd über $\mathbf{Z}_p$ für verschiedene p zu rekonstruieren.

Beispiel:

1) Betrachten wir die beiden Polynome

$$f := x^4 + 25x^3 + 145x^2 - 171x - 360, \quad g := x^5 + 14x^4 + 15x^3 - x^2 - 14x - 15$$

und berechnen für $p \in \{2,3,5,7,11,13,17\}$ nacheinander $h_p := gcd(f,g)$. Für die ersten beiden Primzahlen erhalten wir $h_2 = x^3 + x^2 + x + 1$ und $h_3 = x^3 + x^2 + x$, so daß die Vermutung nahe liegt, daß dies die Reduktionen *mod* 2 bzw. *mod* 3 des ganzzahligen ggT sind, dieser also Grad 3 hat. Da f und g beide monisch sind, hat dieser ggT ebenfalls den Leitkoeffizienten 1, so daß wir mit dem chinesischen Restklassensatz versuchen können, die beiden Polynome zu liften, indem wir die jeweils betragsmäßig kleinsten ganzen Zahlen mit den entsprechenden Resten finden. Dies wäre eines der Polynome $gcd = x^3 + x^2 + x \pm 3$, die beide aber weder Teiler von f noch Teiler von g sind. Also ist die Näherung entweder noch nicht gut genug oder alle bisherigen Primzahlen sind unglücklich. Jedenfalls können wir bereits jetzt schließen, daß $deg\ gcd(f, g) \leq 3$ gilt.

Für $p = 5$ erhalten wir $h_5 = x^4 + 4x$, d.h. es handelt sich auf alle Fälle um eine unglückliche Primzahl, so daß wir mit dem gefundenen Ergebnis nichts anfangen können.

Für $p = 7$ erhalten wir $h_7 = x^2+1$. Damit gilt $deg\ gcd(f, g) \leq 2$ und auch $p = 2, 3$ waren unglückliche Primzahlen und die entsprechenden Ergebnisse irrelevant für das Liften. Für die weiteren Primzahlen erhalten wir entsprechend $h_{11} = x^2+3x+4, h_{13} = x^3+5x^2+6x+8$ (womit auch $p = 13$ unglücklich ist) und $h_{17} = x^2 + 14x + 15$. Wir vermuten nunmehr $deg\ gcd(f, g) = 2$ und erhalten nach Liften der Ergebnisse für $p = 7, 11, 17$ als Kandidaten $gcd = x^2 + 14x + 15$. Probedivision zeigt, daß das tatsächlich ein gemeinsamer Teiler von f und g ist. Da andererseits der Grad des ggT nicht größer als 2 sein kann, haben wir das korrekte Ergebnis, wenigstens in $\mathbf{Q}[x]$, gefunden.

Aufgabe: Untersuchen Sie dieselbe Problemstellung für die Polynome

$$\begin{aligned} f &:= 2x^6 - 13x^5 + 20x^4 + 12x^3 - 20x^2 - 15x - 18; \\ g &:= 2x^6 + x^5 - 14x^4 - 11x^3 + 22x^2 + 28x + 8; \end{aligned}$$

Hauptproblem beim Liften von modularen Kandidaten gleichen Grades zu einem ganzzahligen ggT-Kandidaten ist die Tatsache, daß der ggT nur bis auf assoziierte Elemente eindeutig bestimmt ist und die entsprechende Normierung über $\mathbf{Z}$ und über $\mathbf{Z}_p$ auf unterschiedliche Weise erfolgt. Während letztere üblicherweise auf den Leitkoeffizienten 1 normiert werden, kann ein solcher Leitkoeffizient über $\mathbf{Z}$ nicht garantiert werden. Dort ist nur $lc(gcd(f, g))|l := gcd(lc(f), lc(g))$ bekannt. Wir berechnen deshalb nicht den normierten (primitiven) $gcd(f, g)$, sondern einen solchen mit Leitkoeffizient l. Dieser ist in $\mathbf{Q}[x]$ zu dem gesuchten assoziiert. Da wir den Inhalt des ggT sowieso getrennt berechnen müssen, entsteht dadurch auch kein wesentlicher Zusatzaufwand. Die vorgenommene Normierung ermöglicht nun das Liften der entsprechenden Polynome.

Insgesamt erhalten wir den folgenden Algorithmus:

```
modularGCD(f,g);
      local
            M = kumulierter Modul
            p = aktuelle Primzahl
            (h,M) = Lift der modularen Ergebnisse
            d = Grad des gcd
            c = gcd(lc(f),lc(g))
      c := gcd(lc(f),lc(g)), M:=1
      d := deg f + deg g + 1 (* garantiert zu groß *)
      repeat
            Wähle Primzahl p > c,  (p,M) = 1
            h_p := c · gcd(f_p, g_p)
            if (deg h_p < d) then (* alle bisherigen p unlucky *)
                  d:=deg h_p
                  M:=p
                  (h,p) := Lift(h_p,p) mit lc(h) = c
            else if deg h_p > d then continue (* p unlucky *)
            else
                  (h,p · M) := Lift((h,M),(h_p,p)) mit lc(h) = c
                  if h|f, h|g then return
                        gcd(cont(f),cont(g)) · primPart(h)
                  M:=p · M
      forever
```

Aufgabe: Führen Sie die entsprechenden Berechnungen für $f = 12x^3 + 6x^2 + 14x + 7$, $g = 30x^3 + 15x^2 + 2x + 1$ aus.

Zur Komplexität dieses Algorithmus: Ist b eine Schranke für die Bitlänge der Koeffizienten von f, g, so sind in der Subresultanten-PRS die Bitlängen alle auftretenden Koeffizienten der Ordnung $O(d\,b)$, wobei d eine Schranke für den Grad von f und g ist.
NB: Beachte, daß selbst im ggT längere Koeffizienten als in den Ausgangspolynomen auftreten können.
Beispiel (von Davenport und Trager): $f := (x+1)^2 \cdot (x-1) = x^3 + x^2 - x - 1$ und $g := (x+1)^2 \cdot (x^2 - x + 1) = x^4 + x^3 + x + 1$.
Da eine Zahl der Bitlänge n höchstens $O(n)$ Primfaktoren haben kann, ist $O(d\,b)$ zugleich eine Schranke für die Anzahl der unglücklichen Primzahlen, die im Laufe des Algorithmus besucht werden können. Höchstens weitere $O(d\,b)$ Schleifendurchläufe sind notwendig, um zum korrekten Ergebnis zu liften. In jedem der insgesamt $O(d\,b)$ Durchläufe sind dann

1. $\phi(f)$ und $\phi(g)$ zu berechnen (Kosten $O(d\,b)$, da je d Koeffizienten modular zu reduzieren sind),
2. der modulare ggT zu bestimmen (Kosten $O(d^2)$) und
3. die Koeffizienten des Ergebnisses zu liften (Kosten $O(d^2\,b^2)$, da d mal das CRT auf Zahlen der Länge $O(d\,b)$ und $O(1)$ bzgl. derselben Moduln anzuwenden ist).

Die Gesamtkosten dieses Verfahrens sind also maximal von der Ordnung $O(d^3 b^3)$.

Dieselben Überlegungen, die wir oben für univariate Polynome ausgearbeitet haben, gelten auch für multivariate Polynome. ϕ_p induziert entsprechend einen Homomorphismus

$$\phi_p : \ S := \mathbf{Z}[x_1, \ldots, x_n] \longrightarrow S_p := \mathbf{Z}_p[x_1, \ldots, x_n].$$

Wir fixieren dazu auf dem Monoid der Terme $T(X) = \{X^A := x_1^{a_1} \cdot \ldots \cdot x_n^{a_n}\}$ eine (noethersche) Termordnung $<$ und definieren Leitkoeffizient $lc(f) \in \mathbf{Z}$ und Leitterm $lt(f) \in T(X)$ für $f \in S$ wie gewohnt. Auch in diesem Fall gilt

$$gcd_{\mathbf{Z}[x_1,\ldots,x_n]}(f,g) = gcd_{\mathbf{Z}}(cont(f), cont(g)) \cdot primPart(gcd_{\mathbf{Q}[x_1,\ldots,x_n]}(f,g)),$$

wobei *cont* und *primPart* hier multivariat zu verstehen sind.
Sind $f = c_1\, X^{A_1} + < kleinere\; Terme >$ und $g = c_2\, X^{A_2} + < kleinere\; Terme >$ zwei Polynome aus S, so gilt wiederum

Satz: Wenn $p \not| c := gcd(c_1, c_2)$, so ist $lt(gcd(f_p, g_p)) \geq lt(gcd(f,g))$ bzgl. der (fixierten) Termordnung und die beiden sind nur für endlich viele Primzahlen verschieden.

Aufgabe: Beweisen Sie den ersten Teil ($lt(gcd(f_p, g_p)) \geq lt(gcd(f,g))$) dieses Satzes.

Die folgende modifizierte Version des oben beschriebenen Algorithmus erlaubt dann die ggT-Berechnung in S, wenn man einen Algorithmus für die ggT-Berechnung in S_p zur Verfügung hat.

```
modularMGCD(f,g);
    Input: f,g ∈ S = Z[x_1,...,x_n]
    Output: gcd(f,g)
    local
        M = kumulierter Modul
        p = aktuelle Primzahl
        (h,M) = Lift der modularen Ergebnisse
        d = Leitterm des gcd
        c = integerGCD(lc(f),lc(g))
    c := integerGCD(lc(f),lc(g)), M:=1
    d := lt(f^2) (* garantiert zu groß *)
    repeat
        Wähle Primzahl p > c, (p,M) = 1
        h_p := c · gcd(f_p,g_p) (* normierter S_p-gcd *)
        if (lt h_p < d) then (* alle bisherigen p unlucky *)
            d:=lt h_p
            M:=p
            (h,p) := Lift(h_p,p) mit lc(h) = c
        else if lt h_p > d then continue (* p unlucky *)
        else
            (h,p · M) := Lift((h,M),(h_p,p)) mit lc(h) = c
            if h|f, h|g then return
                integerGCD(cont(f),cont(g)) · primPart(h)
            M:=p · M
    forever
```

Ist B eine Schranke für die Bitlänge der Koeffizienten von $gcd(f,g)$ über $\mathbf{Z}$, $(d_1, \ldots, d_n)$ eine Schranke für $Deg(f), Deg(g)$ und vernachlässigen wir den Aufwand, der durch unglückliche Primzahlen entsteht, so erhalten wir eine ähnliche Abschätzung wie oben. Die Schleife wird dann $O(B)$ mal durchlaufen und d ist durch die Anzahl der Terme in f und g, also $L := max(L(f), L(g)) = O(d_1 \cdot \ldots \cdot d_n)$ zu ersetzen. Bezeichnet schließlich $C(d_1, \ldots, d_n)$ den Aufwand zur Berechnung von $gcd(f_p, g_p)$, erhalten wir insgesamt
Satz: Der Aufwand zur Berechnung von `modularMGCD(f,g)` unter obigen Annahmen ist von der Ordnung

$$B \cdot O(B^2 + B \cdot d_1 \cdot \ldots \cdot d_n + C(d_1, \ldots, d_n))$$

Aufgabe: Überprüfen Sie diese Beziehung.

5.3.2 Die Reduktion der Variablenzahl

Im vorigen Abschnitt ist die Frage offengeblieben, wie man den ggT multivariater Polynome über einem Körper $k = \mathbf{Z}_p$ berechnen kann. Dies erfolgt durch eine andere Anwendung modularer Techniken, nämlich den Übergang von multivariaten zu univariaten Polynomen. Wir wollen auch diese Aufgabe zunächst an einem Beispiel studieren.
Beispiel 1:
Wir betrachten die Polynome

$$\begin{aligned} f(x,y) &= x^3\,y + 2\,x^2\,y^2 + x^2 + x\,y^3 + 2\,x\,y + y^2, \\ g(x,y) &= x^3\,y - 2\,x^2\,y^2 + x^2 + x\,y^3 - 2\,x\,y + y^2 \end{aligned}$$

im Ring $S = k[x,y] = k[y][x]$ über dem Körper $k = \mathbf{Z}_{181}$ und wollen deren ggT $h(x,y)$ daselbst berechnen. Wir betrachten dazu den Morphismus

$$\phi_a : R = k[y] \longrightarrow R' = k \qquad \text{via } y \mapsto a$$

für verschiedene Werte von $a \in k$ und bezeichnen ähnlich wie im vorigen Abschnitt $\phi_a(f)$ kurz durch f_a.
Für verschiedene Werte von a erhalten wir die Polynome

a	$f(x,a)$	$g(x,a)$	$h_a(x)$
1	$x^3+3\,x^2+3\,x+1$	$x^3+180\,x^2+180\,x+1$	$x+1$
2	$2\,(x^3+95\,x^2+6\,x+2)$	$2\,(x^3+87\,x^2+2\,x+2)$	$x+91$
3	$3\,(x^3+127\,x^2+11\,x+3)$	$3\,(x^3+115\,x^2+7\,x+3)$	$x+121$
4	$4\,(x^3+144\,x^2+18\,x+4)$	$4\,(x^3+128\,x^2+14\,x+4)$	$x+136$

Nach unserem Satz dürfen wir nur die Ersetzung $y = 0$ nicht vornehmen, da $lc(f) = lc(g) = y$ in $R[x]$ gilt. Alle Morphismen liefern Polynome vom gleichen Grad, so daß wir annehmen können, daß alle gutartig sind und der richtige ggT $h \in R[x]$ ebenfalls den Grad 1 in x hat, so daß wir versuchen können, diesen durch modulare Rekonstruktion zu finden. Genau wie oben haben wir aber dabei das Normierungsproblem zu lösen. Wir suchen deshalb h in der Form $h = y \cdot x + c_0(y)$, weil $gcd(lc(f), lc(g)) = y$. Skalieren wir die Ergebnisse in unserer Tabelle entsprechend, so erhalten wir für $c_0(y)$ als Bedingung, daß $c_0(1) = c_0(2) = c_0(3) = c_0(4) = 1$ gelten muß, so daß wir vermuten können, daß $h = y\,x + 1$ gilt. Probedivision in $R[x]$ durch f und g zeigt, daß dies in der Tat das korrekte Ergebnis ist.

Aufgabe: Berechnen Sie analog $gcd(x^3 + x^2\,y - x\,y^2 - y^3, x^3 - x^2\,y - x\,y^2 + y^3)$ in $k[x,y]$.

Im allgemeinen haben wir beim Liften einen größeren Aufwand zu treiben, wie das folgende Beispiel zeigt.
Beispiel 2:
Wir betrachten wieder in $k[x,y]$ mit $k = \mathbf{Z}_{181}$ die Polynome

$$\begin{aligned} f(x,y) &= x^3\,y + 62\,x^2\,y^2 + 122\,x^2\,y - x^2 + 61\,x\,y^3 + 123\,x\,y^2 + 120\,x\,y + 59\,x + y^3 - y, \\ g(x,y) &= x^3\,y + x^2\,y^2 - x^2 - x\,y^2 - y^3 + y \end{aligned}$$

Die entsprechenden modularen Reduktionen ergeben in diesem Fall

a	$f(x,a)$	$g(x,a)$	$h_a(x)$
1	$x^3+2\,x^2+x$	x^3-x	x^2+x
2	$2\,(x^3+155\,x^2+6\,x+3)$	$2\,(x^3+92\,x^2+179\,x+178)$	$x+92$
3	$3\,(x^3+6\,x^2+32\,x+8)$	$3\,(x^3+63\,x^2+178\,x+173)$	$x+63$
4	$4\,(x^3+53\,x^2+19\,x+15)$	$4\,(x^3+49\,x^2+177\,x+166)$	$x^2+51\,x+98$

Die Morphismen ϕ_1 und ϕ_4 liefern einen ggT von zu hohem Grad, sind also unglücklich. Wir vermuten deshalb, daß der richtige ggT $h(x,y) \in R[x]$ wiederum vom Grad 1 in x ist und suchen ihn in der Form $h = y\,x + c_0(y)$ mit $c_0(2) = 3, c_0(3) = 8$. $c_0(y)$ ist also ein nicht-konstantes Polynom in y und als Absolutglied von $h \in R[x]$ ein Teiler von $abs(f) \sim abs(g) = y^3 - y$, also $deg\ c_0(y) \leq 3$. Ein solches Polynom kann man aber durch Interpolation aus seinen Werten an vier verschiedenen Stellen bestimmen. ϕ_5 und ϕ_7 liefern zusätzlich zu unseren bereits bekannten Werten $c_0(5) = 24$, $c_0(7) = 48$, was zusammen $c_0(y) = y^2 - 1$ und $h = y\,x + y^2 - 1$ ergibt. Die Probedivision zeigt, daß das wirklich ein gemeinsamer Teiler der beiden Polynome f und g und damit auch ihr ggT ist.

Aufgabe ([9, S. 95]): Berechnen Sie analog für

$$f = x^3\,y\,(2\,y - 1) + x^2\,y^3 + x\,(2\,y^4 - y^3 - 6\,y + 3) + y^2\,(y^3 - 3),$$
$$g = x^3\,(2\,y - 1) + x^2\,y\,(-y + 1) + x\,(-y^3 + 4\,y - 2) + 2\,y^2$$

$gcd(f,g)$ in $k[x,y]$.

Beschreiben wir nun den allgemeinen Fall. Wir wollen dies gleich für multivariate Polynome ausführen. Sei $\phi_a : R = k[x_0] \to R' = k$ wie oben ein durch $x_0 \mapsto a \in k$ induzierter *Evaluationshomomorphismus* und

$$\phi_a : R[x_1, \ldots, x_n] \longrightarrow k[x_1, \ldots, x_n]$$

der entsprechende Morphismus multivariater Polynomringe, der durch Koeffizientenreduktion entsteht. Ist $f = \sum c_\alpha(x_0)\mathbf{x}^\alpha$ mit $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}$, so erhalten wir $f_a = \sum c_\alpha(a)\mathbf{x}^\alpha$.
Der folgende Algorithmus reduziert nach obigem Schema die ggT-Berechnung in $S = k[x_0, \ldots, x_n]$ auf die in $S' = k[x_1, \ldots, x_n]$. Wir fixieren dazu wie im vorigen Abschnitt eine (noethersche) Termordnung auf $T(x_1, \ldots, x_n)$.

```
evalMGCD(f,g);
      Input: f,g ∈ S
      Output: gcd(f,g)
      local
            a = aktueller Evaluationspunkt für x_0
            M = Liste der modularen Ergebnisse
            d = Leitterm des gcd
            c = gcd(lc(f),lc(g))
      c(x_0) := gcd_{k[x_0]}(lc(f),lc(g))
      d := lt(f^2) (* garantiert zu groß *)
      repeat
            Wähle a ∈ k, so daß c(a) ≠ 0
            h_a := c(a) · gcd_{S'}(f_a,g_a) (* normierter S'-gcd *)
            if (lt h_a < d) then (* alle bisherigen a unlucky *)
                  d:=lt h_a
                  M:={(h_a,a)}
            else if lt h_a > d then continue (* a unlucky *)
            else
                  M:=cons((h_a,a),M)
                  h := Lift(M) mit lc(h) = c(x_0)
                  if h|f, h|g then return
                        gcd_{k[x_0]}(cont(f),cont(g)) · primPart_{k[x_0]}(h)
      forever
```

Aufgabe: Zeigen Sie, daß `evalMGCD(f,g)` die angegebene Spezifikation erfüllt.

Untersuchen wir nun die Komplexität dieses Algorithmus. Sei dazu $Deg(f), Deg(g)$ durch $d = (d_0, \ldots, d_n)$ beschränkt und $C(d_0, \ldots, d_n)$ die zu bestimmende Komplexität, wobei wir wiederum den Aufwand, der eventuell durch schlechte Evaluationspunkte entsteht, d.h. wo $lt\ h_a > lt\ h$ ist, nicht berücksichtigen wollen. Im Gegensatz zum ganzzahligen Liften haben in dieser Situation gemeinsame Teiler von f, g einen x_0-Grad, der d_0 nicht übersteigt. $2d_0$ ist also eine Gradschranke für die Koeffizienten in $h \in R[x_1, \ldots, x_n]$, da ja h nicht der normierte ggT ist, sondern auf den Leitkoeffizienten $c = c(x_0)$ getrimmt wurde. Wir benötigen also höchstens $O(d_0)$ Schleifendurchläufe, bis wir das korrekte Ergebnis durch Liften gefunden haben.

Die Liste M enthält $O(d_0)$ Polynome aus $k[x_1, \ldots, x_n]$, deren Multigrad stets durch $(d_1, \ldots, d_n)$ beschränkt ist. Die Anzahl der Terme jedes dieser Polynome wie auch von f und g ist damit durch $L = L(d_1, \ldots, d_n) = O(d_1 \cdot \ldots \cdot d_n)$ beschränkt. Der Aufwand $C_M(d_0, \ldots, d_n)$ zum Liften dieser Liste ergibt sich aus dem Interpolationsaufwand $C_I(d_0)$, der in $k[x_0]$ notwendig ist, um ein Polynom vom Grad $O(d_0)$ aus seinen Werten an einer entsprechenden Anzahl von Stützstellen zu rekonstruieren, als $C_M = L \cdot C_I$, da diese Interpolation für jeden der Koeffizienten von h auszuführen ist.

Aufgabe: Zeigen Sie, daß die Lagrange-Interpolationsformel $C_I(d) = O(d^3)$ ergibt.

Damit entsteht bis zum return-Befehl der Schleife ein Aufwand von

$$O(d_0) \cdot (C(d_1, \ldots, d_n) + C_M(d_0, \ldots, d_n))$$

und danach zur Adjustierung der Koeffizienten nochmals ein Aufwand von $L \cdot O(d_0^2)$. Verwenden wir obige (nicht optimale) Abschätzung für C_I, so erhalten wir für den Gesamtaufwand die folgende Rekursionsformel:

$$C(d_0, \ldots, d_n) = O\left(d_0\, C(d_1, \ldots, d_n) + (d_1 \ldots d_n) \cdot d_0^4\right)$$

Aufgabe: Zeigen Sie, daß unter diesen Voraussetzungen

$$C(d_0, \ldots, d_n) = (d_0^3 + \ldots + d_n^3) \cdot O(d_0 \cdot \ldots \cdot d_n) = O(d_0^3 + \ldots + d_n^3) \cdot L(d_0, \ldots, d_n)$$

gilt.

5 Faktorisierung von Polynomen

Im Gegensatz zur ggT-Berechnung hängt das Ergebnis der Faktorisierung wesentlich vom Grundkörper K, über dem die Polynome betrachtet werden, ab. So läßt sich etwa über den komplexen Zahlen jedes univariate Polynom in Linearfaktoren zerlegen, während über den reellen Zahlen bekanntlich jedes irreduzible Polynom höchstens vom Grad 2 ist. Aus dem Eisensteinkriterium folgt, daß es über den rationalen Zahlen irreduzible Polynome von beliebig hohem Grad gibt.

Andererseits hatten wir gesehen, daß das Faktorisierungsproblem in einem Polynomring $R[x]$ über einem Integritätsbereich R äquivalent ist zur Faktorisierung über R und über $K[x]$, wobei $K = Q(R)$ der Quotientenkörper von R ist.

Wendet man den folgenden Satz rekursiv an, so erkennen wir, daß das Faktorisierungsproblem in einer großen Klasse von Ringen (mit eindeutiger Faktorzerlegung) algorithmisch lösbar ist.

Satz: Ist R ein Ring mit eindeutiger Faktorzerlegung, der unendlich viele Elemente enthält, die Menge der Einheiten R^* endlich und das Faktorisierungsproblem in R algorithmisch lösbar. Dann ist das Faktorisierungsproblem auch in $R[x]$ algorithmisch lösbar.

Beweis: Sei $f(x) \in R[x]$ ein Polynom vom Grad $d > 1$. Ist dieses Polynom reduzibel, so besitzt es einen Faktor vom Grad $q \leq \frac{d}{2}$. Ist $g(x) \in R[x]$ ein solcher Faktor, so gilt für alle $c \in R$, daß auch $g(c)$ ein Teiler von $f(c)$ ist. Wählen wir deshalb mit $q = \left[\frac{d}{2}\right]$ Elemente $c_i \in R$, $i = 0, \ldots, q$ mit $f(c_i) \neq 0$ und berechnen jeweils aus der Faktorzerlegung von $f(c_i) \in R$ die (endliche) Menge

T_i der Teiler von $f(c_i)$. Wählen wir nun Elemente $g_i \in T_i$ und bestimmen durch Interpolation das eindeutig bestimmte monische Polynom $g(x) \in K[x]$ vom Grad $\le q$, für welches $g(c_i) = g_i$ gilt. Unter all diesen Polynomen finden wir einen Teiler von $f(x)$, wenn ein solcher existiert. □

Dieses Verfahren ist für praktische Zwecke allerdings wenig geeignet, da die Anzahl der zu betrachtenden Zweige sehr stark wächst. In praktischen Anwendungen kann ein überlegter Einsatz dieser Technik allerdings manchmal weiterhelfen.

Beispiel: Zerlege $f := x^4 + 5\,x^3 + 13\,x^2 + 17\,x + 12 \in \mathbf{Z}[x]$ in Faktoren.

Mögliche Gradkombinationen sind $4 = 0 + 4 = 1 + 3 = 2 + 2$. Ein linearer Faktor von f hätte die Form $g = x - a$, wobei a ein Teiler von 12 sein muß. Da keiner der Teiler von 12 eine Nullstelle von f ist, scheidet diese Möglichkeit aus. Folglich ist f irreduzibel oder in zwei Faktoren je vom Grad 2 zerlegbar. Ein entsprechender Ansatz $f = (x^2 + a\,x + b)\,(x^2 + c\,x + d)$ führt auf das (nichtlineare) Gleichungssystem

$$polys := \{b\,d - 12, a\,d + b\,c - 17, a\,c + b + d - 13, a + c - 5\},$$

für das wir ganzzahlige Lösungen suchen. Setzen wir für b nacheinander alle möglichen Teiler von 12 ein und reduzieren jeweils das entstehende Gleichungssystem, so erhalten wir in genau zwei Fällen nicht widersprüchliche Antworten, die jeweils (bis auf die Reihenfolge der Faktoren) die Lösung $f = (x^2 + 2x + 3)(x^2 + 3x + 4)$ liefern.

Um zu effektiveren Verfahren zu kommen, können wir versuchen ähnlich wie im letzten Kapitel modulare Techniken anzuwenden. Bei der Reduktion kann es dabei allerdings passieren, daß sich ein über den ganzen Zahlen irreduzibles Polynom modular für jedes p in mehrere Faktoren zerlegen läßt. Wir haben also keine Theorie "guter" Reduktionen mehr.

Beispiel: $f = x^4 + 1$.

p	Faktorzerlegung mod p
3	$\left(x^2 + x + 2\right)\left(x^2 + 2\,x + 2\right)$
5	$\left(x^2 + 2\right)\left(x^2 + 3\right)$
7	$\left(x^2 + 4\,x + 1\right)\left(x^2 + 3\,x + 1\right)$
11	$\left(x^2 + 8\,x + 10\right)\left(x^2 + 3\,x + 10\right)$
13	$\left(x^2 + 5\right)\left(x^2 + 8\right)$
17	$(x + 8)\,(x + 9)\,(x + 2)\,(x + 15)$
19	$\left(x^2 + 6\,x + 18\right)\left(x^2 + 13\,x + 18\right)$
23	$\left(x^2 + 18\,x + 1\right)\left(x^2 + 5\,x + 1\right)$
29	$\left(x^2 + 17\right)\left(x^2 + 12\right)$
31	$\left(x^2 + 8\,x + 1\right)\left(x^2 + 23\,x + 1\right)$

Aufgabe: Zeigen Sie, daß dieses Polynom wirklich für jede Primzahl p in wenigstens zwei Faktoren zerfällt. Hinweis: Untersuchen Sie, für welche p Zerlegungen der Form $(x^2 + a)(x^2 - a)$ bzw. $(x^2 + ax + b)(x^2 - ax + b)$ existieren.

Obiges Beispiel zeigt zugleich, daß auch die Anzahl der Faktoren von der jeweiligen Reduktion abhängt. Selbst wenn die Anzahl und Grade der Faktoren übereinstimmen, so ist nicht gewährleistet, daß sie die Reduktion derselben Faktoren über den ganzen Zahlen sind, so daß auch die Liftungstechniken, die wir bei der ggT-Berechnung verwenden konnten, hier nicht einsetzbar sind.

5.1 Quadratfreie Zerlegung

Untersuchen wir deshalb zuerst einen Teilschritt hin zur vollständigen Faktorzerlegung, die Zerlegung in quadratfreie Bestandteile. Sei dazu R ein Integritätsbereich mit eindeutiger Faktorzerlegung und $K = Q(R)$ dessen Quotientenkörper.

Definition: Ein Polynom $0 \neq f(x) \in K[x]$ heißt *quadratfrei*, wenn in seiner Primfaktorzerlegung jeder Faktor mit der Multiplizität 1 auftritt.

Satz: Ist $char(K) = 0$, so ist ein Polynom $0 \neq f(x) \in K[x]$ quadratfrei genau dann, wenn $gcd(f(x), f'(x)) = 1$.
Beweis: Ist $f(x) = a(x)^2 b(x)$ nicht quadratfrei, so gilt

$$f'(x) = 2a(x)a'(x)b(x) + a(x)^2 b'(x) = a(x)\left(2a'(x)b(x) + a(x)b'(x)\right),$$

d.h. f und f' haben a als gemeinsamen Teiler. Ist umgekehrt $f = \prod_i f_i$ eine Zerlegung von f in (paarweise teilerfremde) irreduzible Faktoren, so ist

$$f' = \sum_i \left(f_i' \prod_{j \neq i} f_j \right).$$

Da f_i in allen bis auf einem dieser Summanden als Faktor vorkommt, erhalten wir $gcd(f_i, f') = gcd(f_i, f_i') = 1$, weil f_i irreduzibel ist. □

Die Voraussetzung $char(K) = 0$ ist notwendig, weil für $char(K) = p$ gilt

$$g'(x) = 0 \quad <=> \quad g(x) = \sum a_i x^{pi} = h(x^p)$$

In diesem Fall kann also $f_i' = 0$ sein.
Aufgabe: Zeigen Sie, daß für *irreduzible* Polynome in $g \in \mathbf{Z}_p[x]$ stets $g'(x) \neq 0$ gilt. Hinweis: Zeigen Sie, daß in $\mathbf{Z}_p$ die Beziehung $h(x^p) = h(x)^p$ gilt.
Als *quadratfreie Zerlegung* eines Polynoms $f \in K[x]$ bezeichnet man eine Folge paarweise teilerfremder quadratfreier Polynome $g_i \in K[x]$, so daß

$$f = \prod_{i=1}^{k} g_i^i$$

gilt.
Sei $char(K) = 0$ und $f_1 := f$. Wie oben zeigt man, daß dann

$$f_2 := gcd(f_1, f_1') = \prod_{i=2}^{k} g_i^{i-1}$$

und generell

$$f_m := gcd(f_{m-1}, f_{m-1}') = \prod_{i=m}^{k} g_i^{i-m+1}$$

für $m > 1$ gilt. Damit ist

$$h_m = \frac{f_m}{f_{m+1}} = \prod_{i=m}^{k} g_i$$

und schließlich

$$g_m = \frac{h_m}{h_{m+1}}.$$

Dies erlaubt es, die quadratfreie Zerlegung von f durch Operationen im Ring $K[x]$ zu berechnen.

Beispiel:

$$\begin{aligned}
&f := x^8 - 2\,x^6 + 2\,x^2 - 1, \\
&F := [x^8 - 2\,x^6 + 2\,x^2 - 1, x^4 - 2\,x^2 + 1, x^2 - 1, 1] \\
&H := [x^4 - 1, x^2 - 1, x^2 - 1, 1] \\
&G := [x^2 + 1, 1, x^2 - 1],
\end{aligned}$$

so daß wir in diesem Fall die quadratfreie Zerlegung

$$f = (x^2 + 1) \cdot (x^2 - 1)^3$$

erhalten.
Aufgabe: Implementieren Sie die quadratfreie Zerlegung eines unitären Polynoms in einem CAS.

5.2 Faktorisierung über endlichen Körpern

In diesem Abschnitt wollen wir zunächst das "modulare Problem" untersuchen, d.h. einen Algorithmus zur Faktorisierung univariater Polynome über einem Restklassenkörper $k = \mathbf{Z}_p$ angeben. Dieser Algorithmus geht auf Berlekamp [1] zurück und kombiniert auf elegante Weise die Theorie endlicher Körper und lineare Algebra über k.

Fassen wir deshalb zuerst einige Rechenregeln in $k = \mathbf{Z}_p$ zusammen:

Lemma: In $k = \mathbf{Z}_p$ gilt

1. $a^p = a$ für $a \in k$.
2. $(a + b)^p = a^p + b^p$ für $a, b \in k$.
3. $\prod_{c \in k} (x - c) = x^p - x$.
4. $v(x)^p = v(x^p)$ für $v(x) \in k[x]$.
5. Der *Frobeniusmorphismus* $F : v(x) \mapsto v(x)^p$ ist eine lineare Abbildung des Vektorraums $k[x]$ in sich selbst.

Die erste Aussage ist gerade der kleine Fermatsche Satz. Die dritte folgt aus der ersten, da alle $c \in k$ Nullstellen von $x^p - x$ sind, sich also entsprechende Linearfaktoren $x - c$ abspalten lassen.

Aufgabe: Beweisen Sie die anderen Aussagen des Lemmas.

Sei $a(x) \in k[x]$ ein Polynom vom Grad $deg\ f = n$, das wir oBdA als quadratfrei voraussetzen wollen. Dann ist $V := k[x]/(a(x))$ ein n-dimensionaler Vektorraum über k mit der Basis $1, x, \ldots, x^{n-1}$ und wir können ein Element $v(x) = \sum_{i=0}^{n-1} c_i x^i$ mit dem Koeffizientenvektor $[c_0\ c_1\ \ldots\ c_{n-1}]$ identifizieren. V ist sogar ein Ring mit der Multiplikation $v(x) * w(x) := v(x) \cdot w(x)\ (mod\ a(x))$, d.h. bzgl. der üblichen Multiplikation von Polynomen mit nachfolgender Reduktion des Ergebnisses modulo $a(x)$ zu einem Polynom vom Grad $< n$.

Betrachten wir nun die Menge

$$W := \{v \in V\ :\ v^p = v\}.$$

Lemma: $W = Ker(F - id)$ ist ein Unter-k-Vektorraum von V.

W ist sogar ein Unterring von V, da $v_1^p = v_1$, $v_2^p = v_2 \implies (v_1 v_2)^p = v_1 v_2$.

Sei nun $a(x) = a_1(x) \cdot \ldots \cdot a_r(x)$ eine Zerlegung von $a(x)$ in (paarweise teilerfremde, da a quadratfrei ist) irreduzible Faktoren und $V_i := k[x]/(a_i(x))$. Das folgende Lemma ist eine Modifikation des Chinesischen Restklassensatzes.

Satz (Chinesischer Restklassensatz für Polynome):

Sind $a_1(x), \ldots, a_r(x)$ paarweise teilerfremde Polynome aus dem Ring $k[x]$ über dem Körper k und $a(x) = a_1(x) \cdot \ldots \cdot a_r(x)$ ihr Produkt, so ist die natürliche Abbildung

$$\phi :\ k[x]/(a(x)) \longrightarrow k[x]/(a_1(x)) \times \ldots \times k[x]/(a_r(x)) \qquad \text{via } v(x) \mapsto (v(x)\ mod\ a_1, \ldots, v(x)\ mod\ a_r)$$

ein Ringhomomorphismus.

Aufgabe: Beweisen Sie diesen Satz (Hinweis: Vollziehen Sie unter Beachtung, daß $k[x]$ auch ein Euklidscher Ring ist, die Beweisführung des Chinesischen Restklassensatzes über $\mathbf{Z}$ nach.)

Folgerung: ϕ induziert einen Vektorraum-Isomorphismus

$$V \cong V_1 \times \ldots \times V_r.$$

$W_i := \{s \in V_i\ :\ s^p = s\}$ sind dann wiederum Unterräume der Vektorräume V_i. Da ferner $v(x)^p \equiv v(x)\ (mod\ a(x))$ erst recht $v(x)^p \equiv v(x)\ (mod\ a_i(x))$ impliziert, induziert ϕ einen Ringhomomorphismus $\phi_W : W \longrightarrow W_1 \times \ldots \times W_r$.

Lemma:
$$\phi_W : W \longrightarrow W_1 \times \ldots \times W_r$$
ist sogar ein Ring- und somit auch ein k-Vektorraum-Isomorphismus.

Beweis: Die Injektivität von ϕ_W folgt aus der von ϕ. Wir zeigen die Surjektivität von ϕ_W: Sei $(s_1, \ldots, s_r) \in W_1 \times \ldots \times W_r$. Aus der Surjektivität von ϕ folgt die Existenz eines Polynoms $v \in V$ mit $\phi(v) = (s_1, \ldots, s_r)$. Wir wollen zeigen, daß $v \in W$, also $v^p = v$ in V ist. Da $s_i \in W_i$, gilt sicher
$$\phi(v^p) = \phi(v)^p = (s_1^p, \ldots, s_r^p) = (s_1, \ldots, s_r) = \phi(v)$$
und somit $\phi(v^p - v) = 0$. Aus der Injektivität von ϕ schließen wir $v^p = v$, also $v \in W$.

Ist $a(x)$ irreduzibel, so ist $V = k[x]/(a(x))$ sogar ein Körper und wegen $z^p - z = \prod_{c\in k}(z-c)$ gilt für $v(x) \in V$
$$v(x)^p - v(x) = \prod_{c\in k}(v(x) - c) = 0$$
genau dann, wenn einer der Faktoren gleich Null ist, d.h. wenn $v(x) = c$ ein Polynom vom Grad 0 ist. W ist also in diesem Fall der eindimensionale k-Vektorraum der konstanten Polynome.

Folgerung: Ist $a(x) = a_1(x) \cdot \ldots \cdot a_r(x)$ eine Zerlegung von $a(x)$ in irreduzible Faktoren und W wie oben beschrieben, so gilt $dim\ W = r$.

Beweis: W ist das direkte Produkt von r eindimensionalen Vektorräumen.

Damit können wir also mit Mitteln der linearen Algebra die Anzahl der Primfaktoren von $a(x)$ bestimmen. Diese Beobachtung und der folgende Satz sind die Basis des **Berlekamp-Algorithmus** zum Faktorisieren von univariaten Polynomen über endlichen Körpern.

Satz: Sei $a(x) \in k[x]$ ein quadratfreies Polynom und $v(x) \in W$ ein nicht-konstantes Polynom. Dann gilt
$$a(x) = \prod_{s\in k} gcd(v(x) - s, a(x)).$$

Beweis: Sei $a(x) = a_1(x) \cdot \ldots \cdot a_r(x)$ die Zerlegung von $a(x)$ in irreduzible Faktoren. Aus $y^p - y = \prod_{s\in k}(y - s)$ folgt mit $y = v(x)$
$$v(x)^p - v(x) = \prod_{s\in k}(v(x) - s).$$
Da $v(x)^p - v(x) \equiv 0\ (mod\ a(x))$, so gilt erst recht
$$a_i(x) \,|\, v(x)^p - v(x) = \prod_{s\in k}(v(x) - s).$$
Da andererseits $v(x) - s_1$ und $v(x) - s_2$ für $s_1 \neq s_2$ zueinander teilerfremd sind, teilt $a_i(x)$ genau einen der Faktoren $v(x) - s$.

Obwohl verschiedene $a_i(x)$ zum selben Faktor $gcd(a(x), v(x) - s)$ gehören können, so erhalten wir doch aus Gradgründen auf diesem Wege eine Zerlegung von $a(x)$ in Faktoren kleineren Grades, wenn $r > 1$ ist.

Aufgabe: Zeigen Sie, daß es für eine fixierte Basis $\{v_1, \ldots, v_r\}$ von W und $i \neq j$ stets $l \in 1 \ldots r$ und $s \in k$ gibt, so daß
$$a_i(x) \,|\, v_l(x) - s, \quad a_j(x) \not|\, v_l(x) - s$$
gilt.

Eine Basis $v_l(x),\ l = 1, \ldots, r$ des Vektorraums W separiert also die verschiedenen Primteiler von $a(x)$. Eine solche Basis gewinnen wir auf folgendem Wege: Hat $a(x)$ den Grad n, so gilt
$$v(x) = \sum_{k=0}^{n-1} a_k x^k \in W \quad <=> \quad \sum_{k=0}^{n-1} a_k \cdot \left(x^k - (x^{k\,p}\ mod\ a(x))\right) = 0$$

Durch Koeffizientenvergleich erhalten wir ein quadratisches homogenes n-reihiges lineares Gleichungssystem in $\{a_0, \ldots, a_{n-1}\}$, dessen Lösungsraum gearde durch eine Basis von W aufgespannt wird.
Beispiel: Faktorisiere $f = x^6 - 3x^5 + x^4 - 3x^3 - x^2 - 3x + 1$ über $k = \mathbf{Z}_{11}$. Mit Reduce erhalten wir

```
> setmod 11; on modular;
> f:=x^6-3x^5+x^4-3x^3-x^2-3x+1;
> q:=for i:=0:5 collect remainder(x^(11*i),f);
> eqns:=coeff(for i:=0:5 sum mkid(a,i)*(part(q,i+1)-x^i),x);
```

als lineares Gleichungssystem (in $a_0, \ldots, a_5$)

$$\begin{aligned}
&a_1 + a_2 + 3 \cdot a_3 + 6 \cdot a_4 + 10 \cdot a_5 = 0\\
&a_1 + 7 \cdot a_2 + a_3 + 2 \cdot a_4 + 8 \cdot a_5 = 0\\
&a_1 + 2 \cdot a_2 + 4 \cdot a_3 + 4 \cdot a_4 + 5 \cdot a_5 = 0\\
&a_1 + 7 \cdot a_2 + 3 \cdot a_3 + a_5 = 0\\
&a_1 + 9 \cdot a_2 + 3 \cdot a_3 + 9 \cdot a_4 + 9 \cdot a_5 = 0\\
&a_1 + 4 \cdot a_3 + 6 \cdot a_4 + 8 \cdot a_5 = 0.
\end{aligned}$$

Seine Lösungsmenge wird durch

$$\{a_1 = a_4,\ a_2 = a_4 + 7 \cdot a_5,\ a_3 = a_4 + 9 \cdot a_5\}$$

beschrieben. Eine Basis dieses Lösungsraums besteht zum Beispiel aus den folgenden Vektoren

$$[1, 0, 0, 0, 0, 0],\ [0, 1, 1, 1, 1, 0],\ [0, 0, 7, 9, 0, 1],$$

was den Polynomen

$$v_1 = 1,\ v_2 = x^4 + x^3 + x^2 + x,\ v_3 = x^5 + 9x^3 + 7x^2$$

in W entspricht.

```
> for i:=0:10 collect gcd(v2-i,f);
```

ergibt

$$\{x + 1,\ x^2 + 5 \cdot x + 3,\ 1,\ 1,\ x^3 + 2 \cdot x^2 + 3 \cdot x + 4,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1\},$$

woraus wir die gesuchte Faktorzerlegung

$$f(x) = (x + 1)\,(x^2 + 5 \cdot x + 3)\,(x^3 + 2 \cdot x^2 + 3 \cdot x + 4)$$

extrahieren können.
Im allgemeinen Fall liefert der folgende Algorithmus die gesuchte Faktorzerlegung:

factorBerlekamp(f(x),p);
  Input: $f(x) \in k[x]$ quadratfrei, $k = \mathbf{Z}_p$.
  Output: Liste der irreduziblen Faktoren von $f(x)$.
  begin
    n:=deg f;
    q:=$\sum_{i=0}^{n-1} a_i \cdot \left(x^{p\,i} - x^i\right) \pmod{a(x)}$;
    E:=coeff(q,x); (* *Koeffizienten von q, lineares Gleichungssystem in $a_0, \ldots, a_{n-1}$* *)
    B:=Basis des Lösungsraums von E $= (v_1 = 1, v_2(x), \ldots, v_r(x))$
    Berechne $gcd(v_l(x) - s, f(x)),\ l \in 1 \ldots r,\ s \in k$,
      bis alle r irreduziblen Faktoren von $a(x)$ gefunden sind.
  end.

Zur Komplexität dieses Algorithmus: Um $q(x)$ zu berechnen, sind die Polynome $x^{p\,i}$ $(mod\ a(x))$ zu reduzieren. Dies kann man etwa tun, indem man die Reduktion aller Potenzen $x^i,\ i=0,\dots,p(n-1)$ bestimmt. Ist $a(x) = x^n - r(x)$ mit $r \in V$ und

$$x^i \equiv \sum_{k=0}^{n-1} u_k x^k\ (mod\ a(x)),$$

so bekommt man

$$x^{i+1} \equiv x \cdot (\sum_{k=0}^{n-1} u_k x^k) \equiv \sum_{k=0}^{n-2} u_k x^{k+1} + u_{n-1} r(x)\ (mod\ a(x)).$$

Dieser Ausdruck kann mit $O(n)$ Operationen berechnet werden. Für die Berechnung von q benötigen wir also insgesamt $O(pn^2)$ Operationen. Das lineare Gleichungssystem kann in weiteren $O(n^3)$ Operationen gelöst werden. Im letzten Schritt sind schließlich maximal $O(pn)$ ggT-Berechnungen von Polynomen in $k[x]$ von Grad $\leq n$ notwendig. Das liefert einen Gesamtaufwand von $O(pn^3)$.

5.3 Faktorisierung in $\mathbf{Q}[x]$

Sei $f(x) \in \mathbf{Q}[x]$ das zu untersuchende Polynom. Nach dem Gausslemma können wir mit dem gemeinsamen Hauptnenner durchmultiplizieren und annehmen, daß $f(x)$ ganzzahlige Koeffizienten hat und primitiv ist. Zugleich können wir annehmen, daß $f(x)$ quadratfrei ist, da wir das Problem der quadratfreien Zerlegung bereits untersucht und gelöst haben.

Wir wollen zur Lösung des Faktorisierungsproblems wieder auf den bereits entwickelten modularen Algorithmus zurückgreifen. Allerdings hatten wir gesehen, daß ein multimodularer Zugang über eine Rekombination der modularen Ergebnisse mit Hilfe des chinesisschen Restklassensatzes nicht möglich ist. Statt dessen werden wir im folgenden die Technik des *Hensel-Liftens* von einer Faktorisierung $(mod\ p)$ zu einer solchen $(mod\ p^k)$ einsetzen und daraus versuchen, durch Rekombination die ganzzahlige Faktorzerlegung zu gewinnen.

Satz (Hensel-Lemma): Sei p eine Primzahl, $a(x) \in \mathbf{Z}[x]$ so, daß $a(x)$ $(mod\ p)$ in die paarweise relativ primen Faktoren

$$a(x) \equiv a_1(x) \cdot \ldots \cdot a_r(x)\ (mod\ p)$$

zerfällt. Dann existieren für jede natürliche Zahl $k > 0$ Polynome $a_1^{(k)}(x), \dots, a_r^{(k)}(x) \in \mathbf{Z}[x]$, so daß

$$a(x) \equiv a_1^{(k)}(x) \cdot \ldots \cdot a_r^{(k)}(x)\ (mod\ p^k)$$

und

$$a_i^{(k)}(x) \equiv a_i(x)\ (mod\ p), \quad deg(a_i^{(k)} - a_i) < deg(a_i)$$

für $i = 1, \dots, r$ gilt.

Beweis: Mit Induktion nach k. Für $k = 1$ können wir die Ausgangspolynome $a_1, \dots, a_r$ nehmen. Seien nun $a_1^{(k)}, \dots, a_r^{(k)}$ bereits konstruiert. Wir suchen $a_i^{(k+1)}$ in der Form

$$a_i^{(k+1)} = a_i^{(k)} + p^k \cdot q_i(x) \quad \text{für } i = 1, \dots, r.$$

Nach Induktionsvoraussetzung wissen wir, daß es ein Polynom $q(x)$ gibt, so daß

$$a(x) = a_1^{(k)}(x) \cdot \ldots \cdot a_r^{(k)}(x) + p^k \cdot q(x)$$

gilt. Mit

$$A_i(x) := \prod_{j \neq i} a_j(x)$$

ist die Behauptung des Satzes für $a_i^{(k+1)}$, $i = 1, \ldots, r$, genau dann erfüllt, wenn

$$q(x) \equiv \sum q_i(x) \cdot A_i(x) \ (mod\ p)$$

gilt. Die Existenz solcher q_i mit $deg(q_i) < deg(a_i)$ folgt aus dem Chinesischen Restklassensatz für Polynome. $\square$

Der Beweis des Satzes gibt zugleich ein algorithmisches Verfahren an, wie man eine solche Liftung berechnen kann. Dazu findet man eine Lösung

$$1 \equiv \sum u_i(x) \cdot A_i(x) \ (mod\ p),$$

mit der man in jedem Schritt für

$$q(x) = \frac{a(x) - a_1^{(k)}(x) \cdot \ldots \cdot a_r^{(k)}(x)}{p^k} \in \mathbf{Z}[x]$$

die entsprechenden Korrekturterme $q_i = q \cdot u_i \cdot A_i \ (mod\ a_i)$ berechnen kann.
Beispiel: Betrachten wir das Polynom $a(x) = x^4 + 1$, das modulo $p = 3$ in die beiden Faktoren $a_1 = x^2 + x - 1$ und $a_2 = x^2 - x - 1$ zerfällt. Wir erhalten nacheinander

k	$a_1^{(k)}$	$a_2^{(k)}$	q	$q_1 = -q_2$
1	$x^2 + x - 1$	$x^2 - x - 1$	x^2	x
2	$x^2 + 4x - 1$	$x^2 - 4x - 1$	$2x^2$	$-x$
3	$x^2 - 5x - 1$	$x^2 + 5x - 1$	x^2	x
4	$x^2 + 22x - 1$	$x^2 - 22x - 1$	$6x^2$	0
5	$x^2 + 22x - 1$	$x^2 - 22x - 1$	$2x^2$	$-x$
6	$x^2 - 221x - 1$	$x^2 + 221x - 1$	$67x^2$	x

Kennt man eine Schranke für die Größe der Koeffizienten in den einzelnen ganzzahligen Primfaktoren von $a(x)$, so kann man versuchen, diese für genügend großes k aus den Ergebnissen modulo p^k zu rekonstruieren. Dabei ist zu berücksichtigen, daß zwar eine Faktorzerlegung über $\mathbf{Z}$ eine solche $(mod\, p^k)$ induziert, aber Faktoren, die über $\mathbf{Z}$ irreduzibel sind, $(mod\, p^k)$ in weitere Faktoren zerfallen können. Dies ist etwa in obigem Beispiel der Fall, d.h. wir müssen mehrere Faktoren $(mod\ p^k)$ zu einem ganzzahligen Faktor rekombinieren. In obigem Beispiel ist die Auswahl klein. Die Faktoren haben stets die Form $x^2 + ax - 1$ und $x^2 - ax - 1$, ihre Rekombination ergibt also $x^4 - (a^2 + 2)x^2 + 1$ und wir sehen, daß stets $3^k \mid (a^2 + 2)$ gilt, also ein "Grenzwert" dieser Koeffizienten (in der p-adischen Topologie) existiert, während dies für die Koeffizienten der einzelnen Faktoren nicht gilt.
Insgesamt erhalten wir den folgenden Algorithmus

```
factorBerlekampHensel(f)
      Input: f ∈ Z[x] primitiv, quadratfrei.
      Output: Liste F der Primfaktoren von f.
      begin
            Wähle p, so daß p ∤ lc(f) und f quadratfrei (mod p) bleibt.
            [u_1,...,u_s]=factorBerlekamp(f,p);
            Normalisiere so, daß lc(u_1) = lc(f), lc(u_2) = ... = lc(u_s) = 1.
            Finde eine Schranke K, so daß p^K/2 die zu erwartende
                  Koeffizientengröße in den Primfaktoren übersteigt.
            Lifte [u_1,...,u_s] nach Hensel zu V := [v_1,...,v_s] (mod p^K),
                  wobei das zentrierte minimale Restesystem (mod p^K) verwendet wird.
            F := ∅;
            For v_i ∈ V do
```

```
        if q := v_i | f then
              V := V \ {v_i}; f := f/q; F := F ∪ {q};
    For (v_i1, v_i2) ∈ V^2 do
        if q := v_i1 · v_i2 | f then
              V := V \ {v_i1, v_i2}; f := f/q; F := F ∪ {q};
    ..................
    return F
end.
```

Aufgabe: Führen Sie diese Rechnungen mit dem Polynom

$$f := 6x^7 + 7x^6 + 4x^5 + x^4 + 6x^3 + 7x^2 + 4x + 1$$

durch. (Hinweis: Verwenden Sie $p = 5$)

5.4 Multivariate Polynome

Die folgende Methode, die auf L. Kronecker (1882) zurückgeht, reduziert das Faktorisierungsproblem für multivariate Polynome über einem Integritätsbereich R auf das für univariate Polynome über R. Betrachten wir dazu den Ringhomomorphismus

$$S_d : R[x_1, \ldots, x_n] \longrightarrow R[y] \quad \text{via } h(x_1, \ldots, x_n) \mapsto h(y, y^d, \ldots, y^{d^{n-1}}).$$

S_d ist ein R-Modul-Isomorphismus zwischen den Polynomen in $R[y]$ vom Grad $< d^n$ und den Polynomen in $R[x_1, \ldots, x_n]$, deren Gradvektor in jeder Komponente durch $d-1$ beschränkt ist. In der Tat, ist $a = a_1 + a_2 d + \ldots + a_n d^{n-1}$ für $a < d^n$ dessen (eindeutig bestimmte) Ziffern-Zerlegung im Zahlsystem zur Basis d, so ist $x_1^{a_1} \cdot \ldots \cdot x_n^{a_n}$ das eindeutig bestimmte Urbild unter S_d von y^a, das der genannten Gradbeschränkung genügt. Bezeichne

$$S'_d : \{g \in R[y] \ : \ deg(g) < d^n\} \longrightarrow \{f \in R[x_1, \ldots, x_n] \ : \ \forall\, i \ (deg_{x_i}(f) < d)\}$$

die entsprechende Umkehrabbildung.
Hat also $f \in R[x_1, \ldots, x_n]$ in allen Variablen einen durch $d-1$ beschränkten Grad und ist g ein Faktor von f, so gilt mit $f = g \cdot h$ auch $S_d(f) = S_d(g) \cdot S_d(h)$ und wir können wieder versuchen, ein geeignetes g durch Rekombination der verschiedenen Faktoren von $S_d(f)$ zu finden.
Das führt uns unmittelbar zum folgenden Faktorisierungsalgorithmus für multivariate Polynome:

```
factorKronecker(f)
    Input: f ∈ R[x_1,...,x_n].
    Output: Liste F der Primfaktoren von f.
    begin
        d:=max(deg_{x_i}(f), i = 1,...,n) + 1;
        V:=[v_1,...,v_s]:=Factorization of S_d(f) in R[y];
        F := ∅;
        For v_i ∈ V do
            if q := S'_d(v_i) | f then
                  V := V \ {v_i}; f := f/q; F := F ∪ {q};
        For (v_i1, v_i2) ∈ V^2 do
            if q := S'_d(v_i1 · v_i2) | f then
                  V := V \ {v_i1, v_i2}; f := f/q; F := F ∪ {q};
        ..................
        return F
    end.
```

Eine Lösung, die wie im Fall der ggT-Berechnung die Reduktion durch Evaluation in einem guten Punkt und nachfolgende Henselliftung verwendet, ist ebenfalls möglich und generell weniger aufwendig als der Kroneckeransatz.

Literatur

[1] E.R.Berlekamp: Factoring polynomials over finite fields. *Bell System Technical Journal* **46** (1967), 1853 - 1859.

[2] K.O.Geddes, S.R.Czapor, G.Labahn: Algorithms for Computeralgebra. Kluwer Acad. Publisher, Boston 1992.

[3] N.Koblitz: A course in number theory and cryptography. Springer, New York 1987.

[4] D.E. Knuth: The art of computer programming. Vol. 2, Addison Wesley.

[5] E.Kunz: Algebra. Vieweg Aufbaukurs Mathematik, Braunschweig/Wiesbaden 1991.

[6] R.Loos: Generalized polynomial remainder sequences. In: Computer algebra (ed. B.Buchberger, R.Loos), Springer Wien 1983

[7] B.Mishra: Algorithmic algebra. Springer, New York 1993.

[8] H.Riesel: Prime numbers and computer methods for factorization. Birkhäuser, Basel 1994.

[9] F.Winkler: Polynomial algorithms in computer algebra. Texts and Monographs in Symbolic Computation, Springer, Wien 1996.

[10] R.Zippel: Effective polynomial computation. Kluwer Acad. Publisher, Boston 1993.