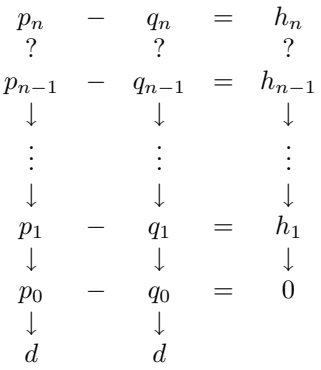
Gröbnerbaser—Ett Verktyg för att konstruera och räkna med matematiska objek



Per Andersson

Luleå University of Technology

Gröbnerbaser—Ett verktyg för att konstruera och räkna med matematiska objekt

Per Andersson
Luleå Tekniska Universitet

16 maj 1997

Abstract

A common situation, not only in mathematics, is that a large and useful class of problems is known to be solvable but no algorithm to actually solve the problem exists.

In this text I will show how Gröbner Bases can be used to construct and calculate with objects derived from ordinary polynomials (more precisely, objects expressed as residue classes to a polynomial ring modulo an ideal).

As an application it will be shown how systems of nonlinear polynomial equations are solved with Gröbner Bases.

Sammanfattning

En vanlig situation, inte bara i matematik, är att en stor och användbar klass av problem är lösbara i princip men ingen algoritm som faktiskt löser problemen är känd.

I denna bok visar jag hur Gröbnerbaser kan användas för att konstruera och räkna med objekt skapade av vanliga polynom (restklasser till en polynomring modulo ett ideal).

Som en tillämpning visas hur system av icke-linjära polynomekvationer löses med Gröbnerbaser.

Förord

En historisk återblick

I sin "ungdom" var algebra *algoritmisk* till sin natur. Att syssla med algebra innebar att hitta på metoder, algoritmer, för att lösa praktiska problem:

> Givet en ekvation av typen $ax + b = c$, hur bestämmer vi det värde på c som satisfierar ekvationen?

Så länge problemen inte var av alltför omfattande natur löste algebraikerna många fundamentalt viktiga problem.

Men ju längre tiden gick, desto mer omfattande och komplicerade blev problemen, lösningarna likaså. I början av artonhundratalet var metoderna algebraikerna konstruerade av så komplex natur att de blev helt ohanterliga, alla problem var ju fortfarande tvungna att lösas med penna och papper!

Till skillnad från algebrans algoritmiska natur hade man inom geometri en *axiomatisk* syn på världen. Istället för att konstruera metoder för att lösa problem intresserade man sig för själva lösningarna och deras egenskaper. Detta synsätt anammades av algebraikerna och man ställde sig frågor som:

> Givet en ekvation av typen $ax + b = c$, vilka egenskaper har de (matematiska) objekt som satisfierar ekvationen?

Detta synsätt gav ofta mycket eleganta resonemang, lätt överblickbara på ett papper. Men i och med detta nya synsätt övergav man också konstruktionen av lösningsmetoder: Man visste hur lösningarna såg ut men man hade ingen metod att lösa konkreta problem med"

Under senare delen av nittonhundratalet har emellertid det algoritmiska synsättet fått en renessans. Två viktiga förändringar ligger bakom detta:

1. 1965 såg Gröbnerbaser dagens ljus i och med Bruno Buchbergers doktorsavhandling. Med Gröbnerbaser knöts de algoritmiska och axiomatiska synsätten ihop i och med att man med hjälp av Gröbnerbaser kan räkna på en stor klass av objekt som man använt sig av för att beskriva lösningars egenskaper.

2. Den otroligt snabba utvecklingen av datorer har inneburit att metoderna med Gröbnerbaser är tillgängliga för den breda massan.

Konstruktion av nya objekt: De rationella talen

Ett mycket vanligt arbetssätt inom t ex matematiken när man vill konstruera något nytt är att man utgår ifrån något gammalt och välkänt, lägger till extra villkor och lär sig räkna på det "nya" man har konstruerat.

Vi ska konstruera de rationella talen utgående från heltalen. Vi brukar oftast se rationella tal i formen

$$\frac{13}{6},$$

så en rimlig utgångspunkt är talpar (a, b) där a och b är heltal. Om vi ser på mängden

$$S = \{(a, b) \mid a, b \in \mathbb{Z}\} \tag{a}$$

av alla dessa talpar ser vi att den *inte* duger som beskrivning av de rationella talen. T ex så ligger både $(4, 2)$ (brukar skrivas som $\frac{4}{2}$) och $(8, 4)$ i S, d v s vi har två olika "element" i S som uttrycker samma "tal" i $\mathbb{Q}$ vilket är helt oacceptabelt. Om vi har två tal (a, b) och (c, d) från S, när uttrycker de "samma sak"? Jo, om

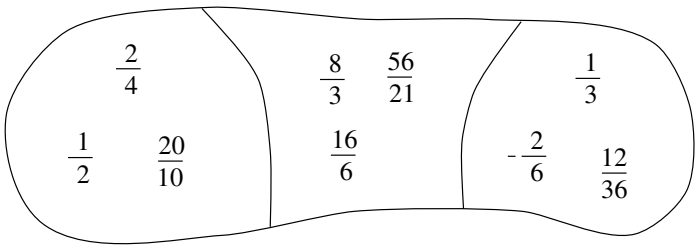
$$\frac{a}{b} = \frac{c}{d} \Longleftrightarrow ad = bc$$

vill vi att (a, b) och (c, d) ska referera till samma tal. Med andra ord, vi vill dela upp S i ett antal ekvivalensklasser som uppfyller att (a, b) och (c, d) ligger i samma klass om och endast om $ad = bc$, matematiskt uttrycker vi det som

$$(a, b) \sim (c, d) \Longleftrightarrow ad = bc \tag{b}$$

vilket illustreras i figur 0.1.

Eftersom alla tal från S som ligger i en och samma klass "betyder" samma sak vore det bra om vi ur varje klass kunde plocka ut ett speciellt tal som får



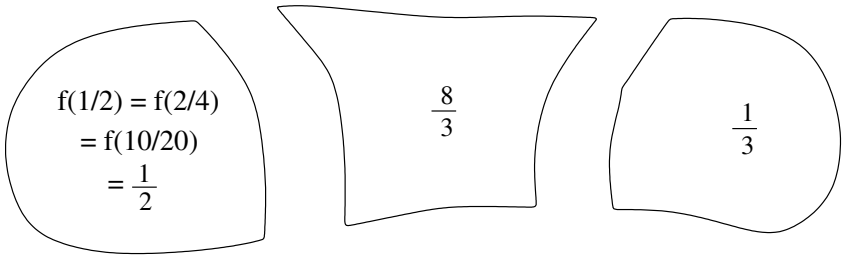
Figur 0.1: Uppdelning av S i ekvation (a) i ekvivalensklasser enligt ekvation (b).

utgöra en klassrepresentant för den klassen. Vi kan förstås välja precis vilket tal som helst i en klass som klassens representant, men eftersom vi sysslar med matematik vill vi, givet en klass, kunna *räkna* oss fram till denna representant.

Se på funktionen f definierad genom

$$f : (a, b) \mapsto \left(\frac{a}{\gcd(a, b)}, \frac{b}{\gcd(a, b)} \right) .$$

Vi har t ex att $f(4, 2) = (2, 1)$ och $f(8, 4) = (2, 1)$. Vi ville ju att t ex talen $(4, 2)$ och $(8, 4)$ skulle betraktas som samma tal och i och med funktionen f har vi funnit en lösning på problemet: Funktionen $f(a, b)$ ger oss en representant ur den ekvivalensklass som (a, b) tillhör och denna representan är dessutom unik, alla talpar som ligger i en och samma ekvivalensklass har samma bild under f vilket illustreras i figur 0.2.

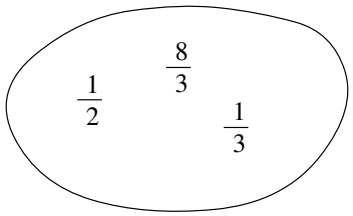


Figur 0.2: Plocka ut en representant ur varje restklass.

För att återknyta till den ursprungliga frågeställningen: De rationella talen vi var ute efter är helt enkelt mängden av alla dessa klassrepresentanter, matematiskt uttrycker vi det som

$$\mathbb{Q} = \{f(a,b) \mid (a,b) \in S\} \ .$$

vilket illustreras i figur 0.3.



Figur 0.3: Låt $\mathbb{Q}$ vara mängden av alla klassrepresentanter.

Hur räknar vi på dessa "nya" tal? Eftersom alla tal som ligger i en ekvivalensklass sägs vara ekvivalenta kan man gissa sig till att det inte spelar någon roll vilket av dem man väljer vid ett visst tillfälle. Vi hat t ex att

$$\frac{3}{12} \cdot \frac{2}{4} = \frac{6}{48}$$

och för att få ett tal tillhörande $\mathbb{Q}$ applicerar vi f på $(6, 48)$ och får

$$f(6, 48) = (1, 8)$$

vilket inte kommer som någon överraskning eftersom

$$\frac{6}{48} = \frac{1}{8} \ .$$

Om vi istället räknar med element i $\mathbb{Q}$ har vi ($f(3, 12) = (1, 4)$, $f(2, 4) = (1, 2)$)

$$\frac{1}{4} \cdot \frac{1}{2} = \frac{1}{8} \ .$$

Det spelar m a o ingen roll vilket element ur en ekvivalensklass vi räknar med eftersom funktionen f *reducerar* resultatet till ett element i $\mathbb{Q}$.

Om vi beräknar

$$\frac{3}{12} \cdot \frac{2}{4} = \frac{6}{48}$$

men talet $(6, 48)$ ligger *inte* i $\mathbb{Q}$. Räddningen är dock enkel: Vi beräknar $f(6, 48) = (1, 8)$ som är en av de representanter vi tidigare kommit fram till och alltså ligger i $\mathbb{Q}$.

En generalisering

Konstruktion enligt ovan av $\mathbb{Q}$ kan sägas bestå av ett antal generella steg:

1. Välj en samling objekt att utgå från (för att konstruera de rationella talen valde vi att utgå från heltalen).
2. Bilda de nya objekten som ekvivalensklasser.
3. Konstruera en funktion för att beräkna unika klassrepresentanter.

I det här kompendiet ska vi se hur denna konstruktion ser ut när vi utgår ifrån mängden av alla polynom istället för mängden av alla heltal. Det var Bruno Buchberger som fick i uppgift av sin handledare Wolfgang Gröbner att "konstruera en bas för restklassringen till en polynomring modulo ett ideal" vilket innebär att studera hur objekten ser ut som blir resultatet av en konstruktion liknande den vi genomförde ovan.

En tillämpning av Gröbnerbaser: Lösning av ekvationssystem

Säg att vi vill lösa ekvationssystemet

$$y^2x^2 - 1 = 0 \tag{0.1}$$
$$xy^2 + x - 2 = 0 \tag{0.2}$$

Vi försöker eliminera variabeln y. Vi tar den första ekvationen minus x gånger den andra och får

$$y^2x^2 - 1 - x \cdot (xy^2 + x - 2) = -x^2 + 2x - 1 = 0 \tag{0.3}$$

som enkelt kan lösas. Genom att kombinera (0.1) och (0.3) får vi

$$y^2(2x-1)-1=0$$

och löser vi den med avseende på x får vi

$$x=\frac{1}{2}\cdot\frac{y^2+1}{y^2}$$

vilket insatt i (0.1) efter förenkling ger

$$y^4-2y^2+1=0\ .$$

I mer komplicerade fall blir naturligtvis proceduren för att lösa ett ekvationssystem betydligt mer komplicerad. Men finns det då inte något bättre sätt att göra detta? Visst finns det det.

Ekvationerna i exemplet ovan är intimt sammankopplade med begreppet *ideal*. I fallet med lösning av ekvationssystem kan vi faktiskt säga att de är likvärdiga. Att söka nollställen till systemet ovan, d v s att lösa det, är samma sak som att söka nollställen till idealet genererat av $\{y^2x^2-1, xy^2+x-2\}$. Nu hjälper det oss inte speciellt mycket, men om vi beräknar detta ideals Gröbnerbas får vi

$$\{y^2+2x-3, x^2-2x+1\} \tag{0.4}$$

och detta ideal har samma nollställen som det ursprungliga. Och att lösa ekvationssystemet

$$y^2+2x-3=0 \tag{0.5}$$
$$x^2-2x+1=0 \tag{0.6}$$

bereder oss inga som helst problem.

Översikt

Lösning av system av ickelinjära polynomekvationer ett av många exempel på tillämpning av Gröbnerbaser, men vägen dit är inte helt enkel.

Vi nämnde i förbifarten begreppet *ideal* och det är hämtat från den gren av matematiken som kallas *abstrakt algebra*. Vårat första steg mot Gröbnerbaser blir alltså att kika en del på vad abstrakt algebra är i allmännhet och vad dessa ideal är i synnerhet.

Utrustade med abstrakt algebra från kapitel 1 och 2 ger vi oss så in på en formalisering av de operationer vi nyss genomförde när vi skapade funktionen f i början av detta kapitel, *reduktion*. Denna reduktion kommer att användas mycket flitigt i kapitel 4 som är huvudmålet för detta kompendium.

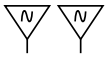
Resterande kapitel ägnas åt olika tillämpningar av Gröbnerbaser från de mest elementära (och också de mest tillämpbara) inom idealteori till mer komplexa som lösning av ekvationssystem.

Typografiska konventioner

En del stycken är markerade med symbolen

Dessa stycken ligger vid sidan av huvudspåret och är inte nödvändiga för att förstå resten. De innehåller (oftast) bakgrundsinformation, djupare motiveringar och liknande. Anledningen till att de överhuvudtaget finns med är att jag tycker de är intressanta och att de passar in just där.

Några få av dessa stycken är av så exotisk natur att de har dekorerats med *två* varningstrianglar



Målgrupp

Kompendiet är lämpligt för alla med grundläggande högskole/universitetsmatematik, speciellt förutsätts att diskret matematik och linjär algebra finns i någorlunda färskt minne.

Tack ...

Först och främst vill jag rikta ett stort tack till min handledare, Dr Thomas Gunnarsson för stöd och visat förtroende. Våra filosofiska diskussioner om matematikens innersta väsen har varit direkt avgörande för det här kompendiets konstruktion.

Stort tack också till Niklas Grip och Elisabeth Söderström som båda två har korrekturläst och gett ovärderliga synpunkter på mitt sätt att skriva och formulera (matematik).

Tack till Roger Isaksson och Gunnar Östlund för era intressanta frågor—hoppas ni kommer ihåg dem för här kommer svaren!

Nämnas skall också: Dr Hans Johansson för hans hjälp med lokaler och datorer. Dr Pavel Kurasov för uppmuntring i svåra stunder. Christer Björklund och Håkan Targama vid Ericsson Utvecklings AB i Östersund för att jag har fått resurser tillgängliga för att slutföra arbetet.

Och sist men inte minst, tack till mig själv för att jag tog mig an den här uppgiften och genomförde den på *mitt* sätt.

Per Andersson

Innehåll

Algoritmer

Tabeller

Del I

Verktygen

Kapitel 0

Grunder

0.1 Notationer

Vi använder de vanliga beteckningarna för

$\emptyset$ Den *tomma mängden*, skrivs ibland som $\{\}$.

$\mathbb{N}$ Mängden av de *naturliga talen*, d v s $\{0, 1, 2, 3, \dots\}$.

$\mathbb{Z}$ Mängden av alla *heltal*, $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$.

$\mathbb{Q}$ De *rationella talen*.

$\mathbb{R}$ De *reella talen*.

$\mathbb{C}$ De *komplexa talen*.

Med $a \in A$ menar vi att a är ett element som ligger i mängden A, med $A \subset B$ menar vi att A är en äkta delmängd av B, d v s det finns minst ett element i B som *inte* finns i A och slutligen menar vi med $A \subseteq B$ att A är en delmängd av B men att de också kan vara lika.

Utifrån ett antal givna mängder kan vi skapa nya mängder som är kombinationer av de ursprungliga:

$A \cup B$ *Unionen* av A och B, om $c \in A \cup B$ har vi att $c \in A$ eller $c \in B$ (eller $c \in A$ *och* $c \in B$).

$A \cap B$ *Snittet* av A och B, om $c \in A \cap B$ har vi att $c \in A$ och $c \in B$.

$A \setminus B$ *Mängddifferensen* av A och B, om $c \in A \setminus B$ har vi att $c \in A$ men $c \notin B$.

$A \otimes B$ *Direkta produkten* av mängderna A och B. Ett element $c \in A \otimes B$ skrivs som (a, b) där $a \in A$ och $b \in B$.

$A \oplus B$ *Direkta summan* av mängderna A och B. Att ett element $c \in A \oplus B$ betyder att *antingen* $c \in A$ *eller* $c \in B$, d v s $A \cap B = \emptyset$. Ofta används den direkta summan "omvänt" på så vis att om man har en mängd C menar man genom att skriva $C = A \oplus B$ att C kan delas upp i två distinkta delar A och B.

Förutom de vanliga notationerna för mängder kommer vi ofta att använda dessa:

$\forall$ *Allkvantorn*, "för alla ... "

$\exists$ *Existenskvantorn*, "det finns minst ett ... "

$\vee$ Logisk operator, " ... *eller* ... "

$\wedge$ Logisk operator, " ... *och* ... "

$\simeq$ *Isomorfi*, $A \simeq B$ betyder ungefär att A och B är lika så när som på namnen på elementen, lika många element, samma struktur.

$|$ *Delar*, med $a \mid b$ menas att $b = a \cdot c$ för något c. Detta är naturligtvis bara av intresse om a, b och c är t ex heltal och helt meingslöst om de är t ex reella tal.

$\circ$ *Komposition*, t ex om f är en funktion av en variabel betyder

$$(f \circ f)(x) \equiv f(f(x)) \ .$$

$a = (a_1, \ldots, a_n)$ *Multiindex*, $|a| = a_1 + \cdots + a_n$, $x^a = x_1^{a_1} \cdots x_n^{a_n}$.

mod *Rest* t ex vid heltalsdivision.

lcm *Minsta gemensamma nämnare* (eng. "Least Common Multiple"). En *gemensam nämnare* till a och b är ett c sådant att $a \mid c$ och $b \mid c$. En *minsta* gemensam nämnare till a och b är det minsta c som är en gensam nämnare till a och b.

T ex så är $\text{lcm}(6, 8) = 24$ och $\text{lcm}(5, 7) = 35$.

0.2 Idén med abstrakt algebra

Vi ska i den här sektionen börja i (a) med att se på den del av matematiken som kallas *abstrakt algebra* ur en lite filosofisk synvinkel. Syftet är att bygga en bro mellan "räkning" (calculus) och den axiomatiska matematiken.

Därefter ska vi i (b) se lite på skillnaden mellan vad man brukar kalla "intuituva förklaringar" och de definitioner och satser de förklarar.

Till sist ska vi i (c) diskutera vad bevisföring går ut på.

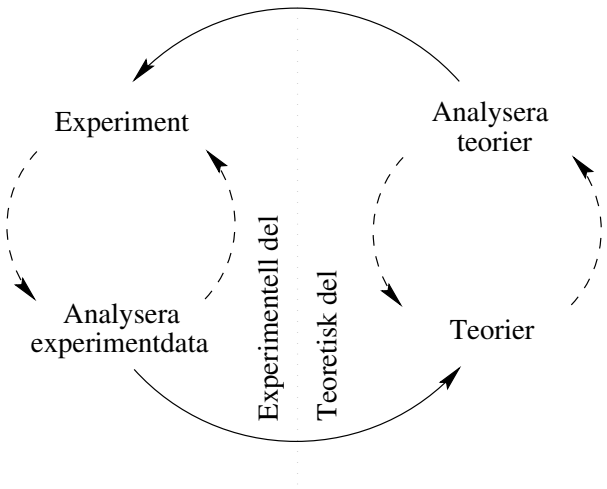
(a) Abstrakt algebra

Forskning i olika vetenskaper kan man ofta dela in i två olika delar: En experimentell del och en teoretisk del.

I den experimentella delen jobbar man med "verkligheten" inom sitt forskningsområde. Det kan innebära att undersöka t ex vad som händer när partiklar kolliderar efter att ha accelererats i en partikelaccelerator eller hur människor reagerar på stress. Ett experiment kan i sin tur ge upphov till idéer för nya experiment. I den teoretiska delen jobbar man istället med en *modell av* verkligheten. Det kan innebära att genom att i en dator simulera en modell av partiklar eller bara genom resonemang formulera nya teorier utifrån de gamla. Mellan dessa två delar pågår ett ständigt informationsutbyte. Experimentalisten kan experimentellt undersöka om en ny teori håller och teroretikern kan formulera teorier utifrån experimentdata. Inom vissa vetenskaper kan uppdelningen mellan teoretiker och experimentalister vara ganska tydlig men inom andra områden flytande eller praktiskt taget obefintlig. Detta kan åskådliggöras i ett diagram enligt figur 0.1.

I matematik finns egentligen ingen experimentdel. Man kan inte skicka integraltecken genom en accelerator och man kan inte se hur differentialekvationer påverkas av stress. En uppdelning som fungerar bättre är att se på tillämpad matematik, d v s matematik för att lösa problem inom olika områden (inklusive matematiken själv), och teoretisk matematik där man ser på den tillämpade matematiken och dess grundläggande ideer.

Abstrakt algebra (den teoretiska delen) syftar till att (1) se på t ex "räkning" (i mängder) i alla dess former, skrapa bort allt "onödigt" och ta fram en eller flera minsta gemensamma nämnare för alla sätt att räkna. Dessa minsta gemensamma nämnare formulerar man matematiskt, undersöker genom att formulera och bevisa satser vad dessa matematiska formuleringar leder till (2) och tillämpar slutligen resultaten av undersökningarna på praktikfall (3). Man kan upptäcka



Figur 0.1: Koppling mellan experimentell och teoretisk del i olika vetenskaper.

samband mellan olika praktikfall och arbete man har lagt ner på dessa minsta gemensamma nämnare kan tillämpas på många olika praktikfall och detta förfarande är alltså mycket arbetsbesparande!

(b) Intuitiva förklaringar

En vanlig källa till missförstånd i matematik är att blanda ihop ett matematiskt begrepp, t ex en sats, med en förklaring till detsamma.

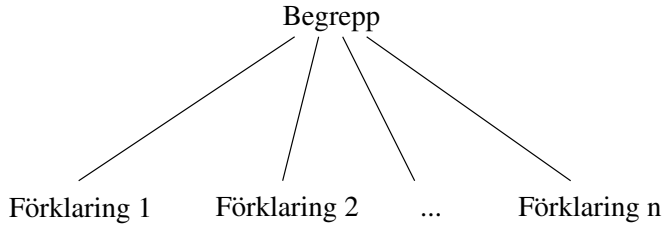
Exempel. •

Förklaringen är till för att förklara men man får inte under några omständigheter dra slutsatser från den! *Slutsatser måste ovillkorligen komma från de ursprungliga definitionerna och satserna!*

Generellt kan man säga att ett begrepp

Begrepp

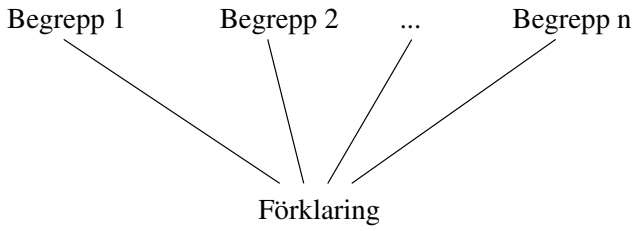
kan förklaras på många olika sätt



och en förklaring

Förklaring

kan vara en förklaring till många olika begrepp



och detta leder till ett många-till-många förhållande mellan begrepp och förklaringar. Håller man inte isär dessa har man ganska snart ingen aning om vad som verkligen gäller! Man måste alltså alltid testa sina ideer mot den formella definitionen. Testar man mot en förklaring testar man i själva verket mot *alla* definitioner som kan förklaras med den förklaringen!

I texten kommer det att finnas en hel del förklaringar till saker och ting men det gäller att hela tiden vara på sin vakt och inse vad som verkligen ÄR matematiska begrepp och vad som bara ser ut att vara det. Oavsett hur väl man tycker att en intuitiv förklaring passar in på ett "praktikfall" måste man gå tillbaka till definitionerna och satserna för att övertyga sig själv om att den intuitiva bilden man har verkligen går att tillämpa.

(c) Bevisföring

Vad menas med ett bevis? När en författare presenterar ett påstående (t ex en sats) vill författaren (oftast) att läsaren ska tror på vad författaren skriver. Det finns många sätt att övertyga läsare: Man kan formulera sig på ett sätt så läsaren inte alls ifrågasätter, man kan med statistik troliggöra sitt påstående o s v.

Att bevisa något matematiskt innebär också det att övertyga läsaren om att ett påstående är sant. Författaren utgår från att läsaren "tror" på vissa grundläggande påståenden och skriver ett "bevis" som ska få läsaren att tro att det verkligen finns ett bevis (utan fnuttar). Det "bevis" författaren presenterar är aldrig fullständigt utan vitsen är att *läsaren* utifrån "beviset" och sina tidigare kunskaper ska kunna konstruera ett bevis som är tillräckligt för att själv tro på författarens påstående.

0.3 Ordningar

För att kunna ordna objekt måste dessa kunna jämföras med varandra och för att två objekt ska kunna jämföras måste det finnas en *relation* mellan dem.

Låt M vara en mängd objekt vi vill ordna. Bilda produkten $M \otimes M$. Med en relation $\Re$ på M menar vi en delmängd $\Re \subseteq M \otimes M$ och vi säger att $a \,\Re\, b$ (a relaterar b) om $(a, b) \in \Re$.

Exempel. Den vanliga $\leqslant$ är en relation på $\mathbb{N}$. Om vi har två tal $a, b \in \mathbb{N}$ definieras relationen $\Re$ av

$$a \leqslant b \iff (a, b) \in \Re \ .$$

Relationen kan åskådliggöras i en tabell (ett $\times$ betyder att den kombinationen tillhör relationen):

$a \leqslant b$		b				
		0	1	2	3	...
	0	$\times$	$\times$	$\times$	$\times$	$\times$
	1		$\times$	$\times$	$\times$	$\times$
a	2			$\times$	$\times$	$\times$
	3				$\times$	$\times$
	$\vdots$					$\times$

•

Olika sorts relationer och ordningar

Låt $\Re$ vara en relation på M.

Att $\Re$ är **reflexiv** innebär att

$$\triangle(M) \subseteq \Re$$

($\triangle(M)$ är diagonalen från övre vänstra till nedre högra hörnet i $M \otimes M$) d v s att för alla $a \in M$ har vi att $a \,\Re\, a$, a relaterar sig själv (kan jämföras med sig själv).

Att r är **symmetrisk** innebär att

$$\Re \subseteq \Re^{-1}$$

($\Re^{-1} = \{(b,a) \mid (a,b) \in \Re\}$) vilket betyder att om $a \,\Re\, b$ så måste också $b \,\Re\, a$ vilket också innebär att vi tillåter element som är likvärdiga (ur relationen $\Re$:s synvinkel) i den mening att $a \,\Re\, b$ och $b \,\Re\, a$.

Att $\Re$ är **transitiv** innebär att

$$\Re \circ \Re \subseteq \Re$$

vilket är detsamma som att för $a, b, c \in M$ gäller att om $a \,\Re\, b$ och $b \,\Re\, c$ så har vi också att $a \,\Re\, c$, vi kan "gå direkt" utan att ta omvägen via b.

Att $\Re$ är **antisymmetrisk** innebär att

$$\Re \cap \Re^{-1} \subseteq \triangle(M)$$

och det betyder att det *inte* kan finnas några element som är likvärdiga i den mening att $(a \neq b)$ $a \,\Re\, b$ och $b \,\Re\, a$, d v s (nästan) motsatsen till symmetrisk.

Att $\Re$ är **konnex** innebär att

$$r \cup r^{-1} = M \otimes M$$

d v s att för $a, b \in M$ har vi att $a \,\Re\, b$ eller $b \,\Re\, a$ (eller båda), man kan se det som att vi inte tillåter några "öar" med element som relaterar varandra utan att ha någon koppling till resten av elementen eller att alla element är jämförbara med varandra.

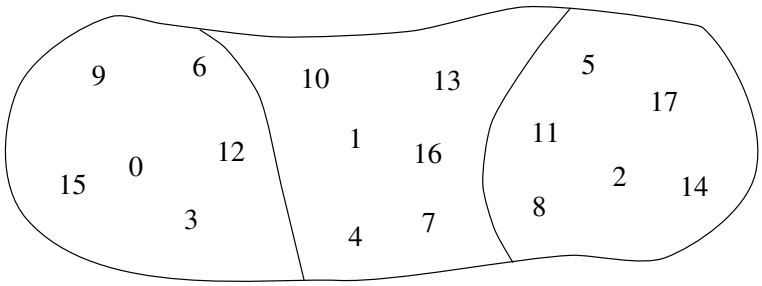
Dessa grundläggande karaktäriseringar av relationer används inte så ofta direkt utan man brukar istället kombinera dem för att skapa mer "kraftfulla" relationer.

Den första kombinationen vi ska använda oss av kallas för **ekvivalensrelation**, betecknas vanligen med $\sim$, och den är reflexiv, transitiv och symmetrisk. En ekvivalensrelation $\sim$ på en mängd M delar upp M i ett antal klasser (högar) med inbördes likvärdiga (enligt $\sim$) element.

Exempel. Ekvivalensrelationen $\sim$ som definieras av

$$a \sim b \Longleftrightarrow a \equiv b \bmod 3$$

(d v s samma rest vid heltalsdivision med 3) delar upp mängden $\{0, 1, 2, \ldots, 17\}$ i tre klasser enligt figur 0.2. •



Figur 0.2: Illustration av ekvivalensklasser.

Den andra kombinationen vi ska använda oss av kallas för **partiell ordning**, betecknas vanligen med $\leqslant$, och den är reflexiv, transitiv och antisymmetrisk. I en partiell ordning finns inga likvärdiga element. De element som relaterar varandra (motsvarande elementen i en ekvivalensklass) ordnas sinsemellan och visualiseras som en eller flera grenar i ett Hassediagram, se figur 0.3. T ex så bildar talen $\{3, 6, 24, 1080\}$ en gren och talen $\{2, 3, 5\}$ utgör diagrammets löv.

Exempel. Den partiella ordningen $\leqslant$ på elementen

$$\{2, 3, 4, 5, 6, 15, 24, 45, 1080\}$$

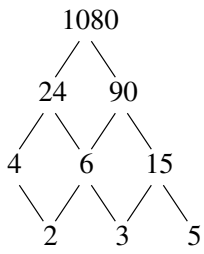
som definieras av

$$a \leqslant b \iff a \mid b$$

(d v s b är en multipel av a) kan åskådliggöras i ett Hassediagram, se figur 0.3. •

0.4 Ekvivalens och kanonisk förenklare

Ett ofta förekommande problem är att avgöra om två givna uttryck i själva verket uttrycker samma sak. Att se att polynomen $p_1 = (x + 1)^2$ och $p_2 = x^2 + 2x + 1$ i själva verket är samma polynom är inte svårt, men generellt sett är det inte alltid så lätt. Vad vi gör när vi vill ta reda på om två uttryck är lika är att på något listigt sätt skriva om dessa för att få uttrycken på en form där vi direkt kan avgöra om de är lika eller ej.



Figur 0.3: Illustration av en partiell ordning.

En generell formulering av ovanstående problem består av två komponenter: Den första komponenten är en ekvivalensrelation $\sim$ på en mängd uttryck E som bestämmer vilka uttryck som ska anses vara lika. Den andra komponenten är vår strategi för förenkling som vi kan se som en funktion som givet ett uttryck som argument ger ett annat, "förenklat", uttryck.

Att avgöra om två givna uttryck a och b är ekvivalenta under den givna ekvivalensrelationen, d v s om $a \sim b$ eller inte, kallar vi för att lösa ett *ekvivalensproblemet*. Ett specialfall av ekvivalensproblemet får vi om $b = 0$, d v s vi vill ta reda på om $a \sim 0$, vi kallar det för *nollekvivalensproblemet*.

Om vi börjar med nollekvivalensproblemet, vilka krav ställer vi på "förenklingsfunktionen" (som vi kallar för S)? Om a är ett uttryck måste naturligtvis $S(a) \sim a$, d v s om vi förenklar a förändrar vi inte dess betydelse. Eftersom vi speciellt är intresserade av att avgöra om ett uttryck a betyder samma sak som noll (skrivs som $a \sim 0$) vill vi (om så är fallet) att $S(a)$, uttrycket a efter förenkling, och $S(0)$, 0 efter förenkling, ska se lika ut. Vi skriver detta som att $S(a) \equiv S(0)$.

Detta kan vi sammanfatta i en definition:

Definition 0.1 (Normal förenklare) *Låt E vara en mängd uttryck och $\sim$ en ekvivalensrelation på E. En* **normal förenklare** *på $(E, \sim)$ är en beräkningsbar funktion $S : E \to E$ som uppfyller följande:*

(i) För alla $a \in E$ gäller att $S(a) \sim a$.

(ii) För alla $a \in E$ gäller att $a \sim 0 \Rightarrow S(a) \equiv S(0)$.

I det mer generella ekvivalensproblemet, vilka krav ställer vi på "förenklingsfunktionen" (som vi här kallar för C)? Eftersom ekvivalensproblemet "inkluderar" nollekvivalensproblemet ska C vara en normal förenklare. Dessutom är vi nu intresserade av att avgöra om två uttryck a och b betyder samma sak (skrivs som $a \sim b$) och vi vill (om så är fallet) att $C(a)$ efter förenkling ser ut precis som $C(b)$. Vi skriver detta som $C(a) \equiv C(b)$.

Detta kan vi sammanfatta i en definition:

Definition 0.2 (Kanonisk förenklare) *Låt $(E, \sim)$ vara som ovan. En* **kanonisk förenklare** *på $(E, \sim)$ är en normal förenklare $C : E \to E$ som dessutom uppfyller följande:*

(iii) För alla $a \in E$ gäller att $a \sim b \Rightarrow C(a) \equiv C(b)$.

Definition 0.3 (Normal och kanonisk form) *Låt S respektive C vara en normal respektive kanonisk förenklare på mängden E och låt a vara ett element i E. Om vi har att $S(a) \equiv a$ så är a på* **normal form**. *Om vi har att $C(a) \equiv a$ så är a på* **kanonisk form**.

Vi har alltså att $a \sim 0$ om och endast om $S(a) \equiv 0$ och $a \sim b$ om och endast om $C(a) \equiv C(b)$. Vi säger att nollekvivalensproblemet respektive ekvivalensproblemet är löst om vi kan hitta beräkningsbara funktioner (algoritmer) S respektive C enligt resonemanget ovan.

Exempel. Låt E vara mängden av alla polynom i en variabel x med heltalskoefficienter. Vi skapar två funktioner f_1 och f_2 enligt följande:

f_1:	(i)	Multiplicera ihop alla produkter av polynom och
	(ii)	samla ihop termer av samma grad.
f_2:	(i)	Multiplicera ihop alla produkter av polynom,
	(ii)	samla ihop termer av samma grad och
	(iii)	ordna termerna efter stigande gradordning.

Då är f_1 en normal förenklare men ej en kanonisk och f_2 är en kanonisk förenklare. En normal form för polynom med avseende på f_1 är

$$a_1 x^{e_1} + a_2 x^{e_2} + \cdots + a_m x^{e_m} \qquad \text{där} \quad e_i \neq e_j \quad \text{när} \quad i \neq j \ ,$$

t ex så är $2x + 3x^3 - x^2$ på normal form med avseende på f_1 men $2x^2 + 3x + x^2$ är det inte. En kanonisk form för polynom med avseende på f_2 är

$$a_1 x^{e_1} + a_2 x^{e_2} + \cdots + a_m x^{e_m} \qquad \text{där} \quad e_i < e_j \quad \text{när} \quad i > j \ ,$$

t ex så är $x + 3x^2 + x^4$ på kanonisk form med avseende på f_2 men $x^4 + 1 - x^2$ är det inte.

Om två polynom p_1 och p_2 båda är på kanonisk form ser vi att de betyder samma sak (som polynom betraktade) om och endast om de också *ser* lika ut (rent textuellt)!

•

Kapitel 1

Abstrakt algebra

Inom alla fackområden finns det mer eller mindre vedertagna fackspråk. För de som är insatta i området är fackspråket guld värt—man kan snabbt, enkelt och precist använda sig av olika begrepp och det utan att bli alltför missförstådd. För en utomstående är fackspråket ibland till besvär. Det är inte nog med att man inte förstår vad "experterna" säger, det kan också vara svårt att få en vettig förklaring till ett begrepp så om man vill ge sig i kast med ett område är det mer eller mindre ett måste att först bli förtrogen med dess språk.

I grunden för all matematik (i alla fall den som presenteras här) finns mängder i en eller annan form[1]. Det kan röra sig om mängder av heltal, mängder av funktioner eller kanske mängder av mängder. För att kunna gå vidare behöver man förr eller senare kunna göra något med elementen i dessa mängder, utföra någon operation på ett, flera eller kanske alla element i en mängd. En sorts operationer är de s k *binära operationerna* och dessa ska vi stifta bekantskap med i sektion 1.1. Med dessa operationer ska vi ge oss på något så bekant som vanlig räkning! De fyra räknesätten har för länge sedan blivit något naturligt för oss utan att vi egentligen har brytt oss om vad vi *egentligen* gör. I sektion 1.1 börjar vi från början, d v s i årskurs ett i grundskolan. Sakta men säkert bygger vi på vår reportoar av räknesätt för att slutligen (återigen) kunna räkna precis som vi har gjort i flera år. Men, och det är ett viktigt men, nu *vet* vi vad vi gör! Och med denna kunskap kommer vi att kunna räkna på sätt som för den oinvigde ser helt obegripliga ut.

Räkning i delmängder av olika slag är ingen nyhet. Ett enkelt exempel är

räkning med rest:

$$\frac{17}{5} = 3 \text{ rest } 2 \qquad \text{eller} \qquad 17 \bmod 5 = 2$$

Här räknar man med en delmängd $\{0, 1, 2, 3, 4\}$ av heltalen. Vad vi ska göra i sektion 1.2 och 1.3 är att generalisera det här till att gälla för delmängder i allmänhet. Sektion 1.2 handlar om en abstrakt beskrivning av delmängder av olika strukturer och sektion 1.3 om hur man räknar i dem.

Målet för kapitlet är att känna sig någotsånär hemma bland begreppen binära operationer, monoider, grupper, ringar, kroppar, ideal och restklasser. Eftersom det här inte är en kurs i abstrakt algebra utan bara en introduktion till de mest centrala begreppen finns inte utrymme för några djuplodande diskussioner utan vi håller oss på ett ganska populärt plan, dock utan att försaka matematisk stringens.

VIKTIGT!!! Det ska sägas att den abstrakta algebran vi nu ska gå igenom inte alls är lätt. Tvärtom, eftersom det här är något *helt nytt* för de flesta kan det kännas *väldigt tungt*. Men ha hela tiden i åtanke att vad vi håller på med är (nästan) ingenting annat än helt vanlig räkning, precis som vi har gjort sedan första klass i grundskolan. Den enda skillnaden är att vi nu riktar in oss på de *grundläggande ideerna* bakom denna räkning och formulerar dessa så att det vi vet om räkning får en *större giltighet*. Och med denna nya formulering kommer vi att lära oss räkna på ett flertal nya sätt. Men allt vi tidigare lärt oss om räkning gäller (givetvis) fortfarande!

1.1 Räkning i mängder

Vad innebär räkning i mängder? I princip innebär det att man för en mängd definierar regler för hur olika element i mängden ska associeras till varandra. När vi lärde oss den vanliga multiplikationstabellen fick vi lära oss utantill att om läraren sa ”nio gånger sju” skulle vi svara ”sextiotre”. Den delen av matematiken syftade till att lära oss hur regeln ”gånger” associerade två tal (i exemplet nio och sju) till ett tredje (sextiotre). Efter lång träning lyckades vi lära oss att till alla kombinationer av sifforna 0–10 associera vad som kallas för ”produkten” av dessa kombinationer och att ”multiplicera” (regeln kallades multiplikation) två tal med varandra har sedan dess skett helt automatiskt.

En stor hjälp vid inlärandet av multiplikation var *multiplikationstabellen*, en liten del av den kunde se ut så här:

$a \cdot b$		b		
		1	2	3
	1	1	2	3
a	2	2	4	6
	3	3	6	9

Multiplikationstabellen består av ett par olika delar: Rutan längst upp till vänster visar hur vi betecknar regeln som tabellen definierar. Det kan naturligtvis finnas många olika sätt att beteckna en och samma regel, t ex så brukar man oftast inte skriva ut $a \cdot b$ utan nöja sig med ab. Regeln multiplikation omfattar två *operander* (en annan regel kanske omfattar fler operander). Kolumnen längst till vänster kallar vi för *definitionskolumn* och översta raden för *definitionsrad*. Om en av operanderna hålls konstant får vi en "ny" regel med endast en operand och den regelen definieras av en kolumn eller en rad i den ursprungliga tabellen. Vi kallar en sådan rad för en *operandrad* och en sådan kolumn för en *operandkolumn*. T ex så menar vi i exemplet med multiplikationstabellen med *operandraden till 2* talen 2, 4 och 6.

Det finns många andra sätt att se på "regler". Välbekant är avbildningar (funktioner) mellan mängder som spelar en stor roll inom matematik. En generell avbildning som avbildar element ur mängden A till element i mängden B betecknar vi med $f : A \to B$. Hur mängderna A och B ser ut kan förstås variera i det oändliga. Om f är en funktion av en variabel som avbildar ett reellt tal på ett reellt tal skriver vi $f : \mathbb{R} \to \mathbb{R}$, ett exempel på en sådan funktion är avbildningen $f : x \mapsto 3x^2 - 3$ (eller i "dagligt tal" $f(x) = 3x^2 - 3$). Själva definitionsmängden kan vara mer komplext uppbyggd, en funktion av ett reellt tal och ett heltal som ger ett reellt tal skriver vi som $f : \mathbb{R} \times \mathbb{N} \to \mathbb{R}$ och ett exempel på en sådan funktion är $g : (x, n) \mapsto x^n$ (d v s $g(x, n) = x^n$).

En regel enligt ovanstående kan i sin mest generella form se ut hur som helst. Ingenting hindrar oss från att definiera en regel genom tabellen

$a * b$		b		
		mfkel	kluyk	hgu
	jir	rtqe	yui	cfzs
a	jeio	vxgs	voc	kort
	poi	zxrq	ueo	oireu

och kalla den för "multiplikation", men man kan ju fråga sig vad det skulle vara

bra för. Nu är vi inte intersserade av liknande regler utan vi kommer att ställa vissa krav på de regler vi ska använda oss av.

Det lägsta krav vi ställer på en operation är att den ska vara en *binär operation* och vad vi menar med det ska vi se i sektion (a).

Vi presenterar sedan i sektionerna (b)–(e) begreppen *monoider*, *grupper*, *ringar* och *kroppar*. Dessa begrepp definierar olika algebraiska strukturer som ärver varandras egenskaper med monoid som den mest generella och kropp som den mest specialiserade. För varje struktur börjar vi med att beskriva den i relativt lösa termer för att definiera den matematiskt. Vi tittar på hur tabeller liknande multiplikationstabellen ser ut för strukturerna och avslutar varje sektion med både kända och okända exempel.

Avslutningsvis sammanställer vi i sektion (f) dessa strukturers relationer till varandra.

(a) Binär operation

En speciell sorts avbildningar är de som är *slutna* i den mening att funktionsvärdet av ett element ur en mängd tillhör den ursprungliga mängden, vi kan alltså inte "komma ut" ur definitionsmängden. En avbildning $f : \mathbb{N} \to \mathbb{N}$ definierad som $f : x \mapsto 2x$ är sluten men avbildningen $f : \mathbb{N} \to \mathbb{Q}$ definierad av $f : x \mapsto 1/x$ är det inte.

Definition 1.1 (Binär operation) *En binär operation på en mängd S är en avbildning $S \times S \to S$ som till varje par $(a, b) \in S \times S$ ordnar exakt ett element $c \in S$.*

Eftersom en binär operation är sluten kan operationens tabell bara innehålla element ur operationens definitionsmängd. Och eftersom båda operanderna har samma definitionsmängd måste definitionskolumnen och definitionsraden vara lika.

Exempel. Den booleska operatorn `AND` och subtraktion, $(\{0,1\}, \texttt{AND})$ och $(\mathbb{Z}, -)$:

a `AND` b		b	
		0	1
a	0	0	0
	1	0	1

$a-b$		b					
		$\cdots$	-1	0	1	2	$\cdots$
	$\vdots$			$\vdots$			
	-1		0	-1	-2	-3	
a	0	$\cdots$	1	0	-1	-2	$\cdots$
	1		2	1	0	-1	
	2		3	2	1	0	
	$\vdots$			$\vdots$			

Andra exempel är addition av naturliga tal $(\mathbb{N}, +)$, derivering av en funktion som är oändligt deriverbar, $(C^\infty, \frac{d}{dx})$ och multiplikation av polynom.

Observera att den "vanliga" multiplikationstabellen från 1–10 *inte* är en binär operation eftersom den innehåller tal som är större än 10. Subtraktion av två naturliga tal är *inte* en binär operation, t ex så är $2 - 4 = -2$ och -2 är ej ett naturligt tal. •

Anmärkning. Tänk på att det i definition 1.1 finns *tre* mycket viktiga krav: (1) Operationen måste vara *definierad*, d v s fungera för *alla* par $(a, b) \in S \times S$. (2) Operationen måste vara *väldefinierad*, d v s till ett par $(a, b) \in S \times S$ får operationen bara associera *exakt ett* element $c \in S$. (3) Operationen måste vara *sluten*, d v s elementet c som operationen associerar med (a, b) måste verkligen ligga i S.

Vid vanlig multiplikation kan man utan vidare byta plats på operanderna, d v s $ab = ba$. Detta gäller dock inte alltid, t ex så kan man inte alltid kasta om ordningen på två matriser vid matrismultiplikation. Om man kan kasta om ordningen på detta sätt säger vi att operationen är *kommutativ*:

Definition 1.2 (Kommutativ operation) *En binär operation $*$ på mängden S är* **kommutativ** *om $a * b = b * a$ för alla $a, b \in S$.*

Den kommutativa operationen gör att den till operationen hörande tabellen är symmetrisk relativt diagonalen från över vänstra till nedre högra hörnet.

Exempel. Multiplikation av heltal, $(\mathbb{Z}, \cdot)$, är kommutativ:

$a \cdot b$		b					
		$\cdots$	-1	0	1	2	$\cdots$
	$\vdots$				$\vdots$		
	-1		1	0	-1	-2	
a	0	$\cdots$	0	0	0	0	$\cdots$
	1		-1	0	1	2	
	2		-2	0	2	4	
	$\vdots$				$\vdots$		

Andra exempel är de logiska operatorerna `OR` och `AND` samt medelvärdesbildning i $\mathbb{Q}$. •

Anmärkning. Observera att tecknet som betecknar operationen kan skrivas precis som man vill och att man skriver operationen som ett + behöver inte betyda addition i vanlig mening. Dock brukar man använda + för att beteckna en kommutativ operation och $\cdot$ för en operation som ej är kommutativ.[2]

Anmärkning. Kom ihåg att även om vi talar om addition och multiplikation så är det bara för att skilja de båda operationerna åt. Det *kan* ju vara så att additionsoperationen faktiskt *är* vanlig addition, men oftast är det inte så.

En annan vanlig räkneregel är att man vid en serie likadana operationer kan utföra dem i vilken ordning man vill, t ex så är $a+b+c = (a+b)+c = a+(b+c)$, denna egenskap kallas för *associativitet*.

Definition 1.3 (Associativ operation) *En binär operation $*$ på mängden S är* **associativ** *om $a*(b*c) = (a*b)*c$ för alla $a, b, c \in S$.*

Att se i en tabell om den definierar en associativ operation eller ej kan man inte göra på något enkelt sätt, man måste då kontrollera hela tabellen, ruta för ruta, om alla kombinationer uppfyller kraven för en associativ operation.

Exempel. Vanlig addition och multiplikation är associativa men subtraktion och division är det inte, t ex så är $(1-1)-1 = -1$ och $(30/5)/2 = 3$ vilket inte är samma sak som $1-(1-1) = 1$ och $30/(5/2) = 12$.

Funktionerna $\min(a,b)$ och $\max(a,b)$ är associativa och kommutativa men "projektionen" $(a,b) \mapsto a$ bara är associativ. •

(b) Monoider

Den enklaste formen av struktur vi ska använda för att räkna i består av en mängd samt en binär operation. Denna operation kan definieras precis som man vill så länge man uppfyller vissa villkor. Det ena villkoret är att operationen ska vara associativ, d v s att $(a + b) + c = a + (b + c)$. Det andra villkoret är att det ska finnas ett s k *enhetselement*.

Definition 1.4 (Monoid) *En monoid $(M, *)$ är en mängd M tillsammans med en binär operation, $*$, som uppfyller följande villkor:*

G 1 *Den binära operationen $*$ är* **associativ**.

G 2 *Det existerar ett (entydigt)* **enhetselement** *$e \in M$ som uppfyller $e * x = x * e = x$ för alla $x \in M$.*

Vi ska enligt definitionen ha en tabell som beskriver en associativ binär operation. Förutom detta ska vi ha ett enhetselement och ett enhetselement e känner man igen genom att operandkolumnen respektive operandraden till e är identisk med definitionskolumnen respektive definitionsraden.

Exempel. Addition av naturliga tal (med 0 som enhetselement), $(\mathbb{N}, +)$, och mängden av alla potenser av variabeln X, $(\{X^r \mid r \geqslant 0\}, \cdot)$ med positiva exponenter (med $1 = X^0$ som enhetselement):

$a+b$	b				
	0	1	2	3	$\cdots$
0	0	1	2	3	
1	1	2	3	4	$\cdots$
2	2	3	4	5	
3	3	4	5	6	
$\vdots$		$\vdots$			

a

$a \cdot b$	b				
	1	X	X^2	X^3	$\cdots$
1	1	X	X^2	X^3	
X	X	X^2	X^3	X^4	$\cdots$
X^2	X^2	X^3	X^4	X^5	
X^3	X^3	X^4	X^5	X^6	
$\vdots$		$\vdots$			

a

Operatorerna `AND` och `OR` är kommutativa monoider med 1 respektive 0 som enhetselement.

Matrismultiplikation bildar en monoid $(M_n, \cdot)$ som *inte* är kommutativ. •

(c) Grupper

En naturlig specialisering av monoider är att kräva att vi ska kunna ”ta inversen” av operationen. Man kan se detta som att vi nu får två räknesätt där de är

inversen av varandra. I själva verket tillför vi ingen ny operation utan vi kräver istället att det till varje element a ska finnas ett inverterat element a^{-1} som uppfyller villkoret $a*a^{-1} = e$ där $*$ är operationen och e är enhetselementet. Vad innebär detta? Om vi ser på heltalen under addition skriver vi t ex $3 + 3^{-1} = 0$ och vi ser att om vi skriver 3^{-1} som -3 får vi $3 + (-3) = 0$ och vi har genom dessa inverterade element fått en (skenbar) extra operation.

Definition 1.5 (Grupp) *En grupp $\langle G, * \rangle$ är en monoid som dessutom uppfyller villkoret:*

G 3 *För varje $x \in G$ existerar det ett element $x^{-1} \in G$ som uppfyller $x*x^{-1} = x^{-1} * x = e$ där e är enhetselementet.*

Om operationen $*$ dessutom är kommutativ är G en **abelsk grupp**.

En tabell för en grupp ser ut som en tabell för en monoid. Eftersom det till varje element a i gruppen ska finnas en "invers" a^{-1} som uppfyller att $a*a^{-1} = e$ måste varje rad och varje kolumn innehålla enhetselementet.

Exempel. Addition av heltal modulo 4, $\langle \{0,1,2,3\}, + \bmod 4 \rangle$, och addition av termer i en variabel X, $\langle \{aX \mid a \in \mathbb{Z}\}, + \rangle$, bildar abelska grupper (t ex så är $1 + 3 = 0 \bmod 4$ så 3 är det inversa elementet till 1):

$a+b$ mod4		b			
		0	1	2	3
a	0	0	1	2	3
	1	1	2	3	0
	2	2	3	0	1
	3	3	0	1	2

$a+b$		b					
		$\cdots$	$-X$	0	X	$2X$	$\cdots$
	$\vdots$			$\vdots$			
a	$-X$		$-2X$	$-X$	0	X	
	0	$\cdots$	$-X$	0	X	$2X$	$\cdots$
	X		0	X	$2X$	$3X$	
	$2X$		X	$2X$	$3X$	$4X$	
	$\vdots$			$\vdots$			

Heltalen bildar under addition en grupp $\langle \mathbb{Z}, + \rangle$ (med 0 som enhetselement och $-a$ som inverterat element a^{-1} liksom de rationella talen förutom noll under multiplikation $\langle \mathbb{Q} - \{0\}, \cdot \rangle$ (med 1 som enhetselement och $1/a$ som inverterat element a^{-1}).

Mängden av alla inverterbara matriser (av en viss storlek) bildar en grupp (som *ej* är abelsk) under matrismultiplikation, $\langle S_n, \cdot \rangle$. •

En grupp sägs vara **cyklisk** om det existerar ett element $a \in G$ så att alla element tillhörande G kan skrivas (multiplikativt) som a^n för något heltal n (eller additivt som $a \cdot n$). Man säger att a är en **cyklisk generator** eller att a **genererar** G eller att G **genereras av** a vilket skrivs som $G = \langle a \rangle$.

Exempel.

$$\begin{aligned} \langle X \rangle &= \{X^n, n \in \mathbb{Z}\} \\ &= \{\dots, X^{-2}, X^{-1}, X^0, X^1, X^2, \dots\} \\ \langle 1 \rangle &= \{1 \cdot n, n \in \mathbb{Z}\} \\ &= \{\dots, -2, -1, 0, 1, 2, \dots\} \end{aligned}$$

•

För fortsatt läsning, se [Fra89] kapitel 1.

(d) Ringar

Skillnaden mellan en monoid och en grupp är ett *skenbart* nytt räknesätt. Nu ska vi titta närmare på en struktur som till en abelsk grupp $\langle R, + \rangle$ tillför ett *verkligt* nytt räknesätt. Dessa två räknesätt knyts samman genom den s k *distributiva lagen* som, om den nya operationen betecknas med "$\cdot$", säger att $a \cdot (b + c) = a \cdot b + a \cdot c$. Vilket tecken vi väljer för att representera den andra operationen spelar ingen roll så länge vi kan skilja de två operationerna från varandra.

På samma sätt som vi kan se på en grupp som en mängd med två räknesätt kan vi se på denna nya struktur som en mängd med tre räknesätt:

Definition 1.6 (Ring) *En ring $\langle R, +, \cdot \rangle$ är en mängd R tillsammans med två binära operationer $+$ och $\cdot$, som vi kallar addition och multiplikation, som uppfyller följande villkor:*

R 1 *$\langle R, + \rangle$ bildar en abelsk grupp.*

R 2 *Multiplikation är associativ och har ett enhetselement*[†]*.*

R 3 *För alla $a, b, c \in R$ gäller den distributiva lagen:*

$$(a + b)c = ac + bc$$

och

$$c(a + b) = ca + cb$$

Om multiplikationen är kommutativ kallas R en **kommutativ ring**.

[†]Vanligt förekommande är också att man i definitionen av en ring *inte* kräver ett multiplikativt enhetselement. Ringen i definition 1.6 kallas då *ring med enhetselement*.

Eftersom en ring innehåller två olika operationer måste vi ha två tabeller. Den ena tabellen (kallas för den additiva tabellen eftersom operationen är kommutativ) ser ut precis som en tabell för en grupp och den andra (kallas den multiplikativa tabellen eftersom operationen inte behöver vara kommutativ) som för en monoid. Dessa två tabeller kopplas samman genom den distributiva lagen.

Exempel. Genom att kombinera exempel från monoider och grupper ser vi att $\langle \{aX^r \mid a \in \mathbb{Z}, r \in \mathbb{N}\}, +, \cdot \rangle$ bildar en ring.

Heltalen bildar under vanlig addition och multiplikation en ring $\langle \mathbb{Z}, +, \cdot \rangle$, *heltalsringen*.

Matriser bildar en icke kommutativ ring under vanlig matrisaddition och matrismultiplikation, $\langle M_n, +, \cdot \rangle$.

Polynom bildar under polynomaddition och polynommultiplikation en ring, *polynomringen*. •

För fortsatt läsning, se [Fra89] kapitel 4.

(e) Kroppar

På samma sätt som vi fick en grupp utifrån en monoid genom att införa en skenbar "inversfunktion" kan vi skapa ytterligare en ny struktur genom att införa en (skenbar) inversfunktion till ringens multiplikationsoperation. Resonemanget är helt analogt med utökningen av monoider till grupper, fast nu vill vi skapa en invers med avseende på det *multiplikativa* enhetselementet.

Vi kräver att det till varje element (noll, det additiva enhetselementet, undantaget) a i ringen ska finnas ett inverterat element a^{-1} som uppfyller villkoret $a \cdot a^{-1} = 1$ där $\cdot$ är den multiplikativa operationen och 1 är det multiplikativa enhetselementet. Vad innebär detta? Om vi sätter in siffror i uttrycket ovan får vi t ex $3 \cdot 3^{-1} = 1$ och vi ser att om vi skriver 3^{-1} som $1/3$ får vi $3 \cdot (1/3) = 1$ och vi har genom dessa inverterade element fått en (skenbar) extra operation. Detta är detsamma som att säga att alla element förutom 0 (noll) bildar en grupp under operationen multiplikation.

Enligt tidigare resonemang kan man se denna nya struktur som en mängd med fyra räknesätt, dock är det inte fråga om fyra helt skilda räknesätt utan två skilda räknesätt med tillhörande inversa operationer.

Definition 1.7 (Kropp) *En kropp är en kommutativ ring*[†] *sådan att* $1 \neq 0$ *och den multiplikativa monoiden av element skilda från noll bildar en grupp.*

[†]I de fall en ring har definierats *utan* multiplikativt enhetselement kräver en kropp istället en *kommutativ ring med enhetselement*, se fotnot till definition 1.6.

Eftersom en kropp också är en ring behöver vi två tabeller som till skillnad från en allmänn ring måste ha *olika enhetselement*. Den additiva tabellen bildar en grupp och om man stryker raden och kolumnen till det additiva enhetselementet bilder också den multiplikativa tabellen en grupp.

Exempel. De utan jämförelse vanligast förekommande kropparna är de mycket välbekanta $\mathbb{Q}$, $\mathbb{R}$ och $\mathbb{C}$. Definitionen på en kropp säger att vi kan addera, subtrahera och multiplicera helt fritt och om vi tar bort 0 (noll) kan vi också dividera. Ej singulära matriser bildar *ej* en kropp, matrismultiplikation är ej kommutativ. •

För fortsatt läsning, se [Fra89] kapitel 4.

(f) Översikt

Dessa fyra centrala strukturer—monoider, grupper, ringar och kroppar— hänger starkt samman med varandra.

I tabell 1.1 finns en sammanställning över strukturerna och vilka "räknesätt" som finns tillgängliga. Ofta vill man i en text referera till t ex en ring utan att mena någon speciell ring utan bara en ring i allmännhet och i kolumnen "standardbeteckning" finns de beteckningar som vi då kommer att använda oss av.

Som objekt betraktade kan man säga att t ex en ring har ärvt en grupps egenskaper. Figur 1.1 visar hur strukturerna är relaterade till varandra. Man kan säga att ju längre ner i hierarkin man kommer desto mer struktur får man. En monoid är en mängd med lite struktur, en grupp är en monoid med lite mer struktur o s v.

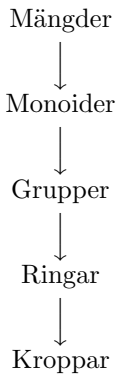
Vi har också sammanställt ett antal olika strukturer i tabell 1.2 som visar vilka egenskaper de har.

1.2 Delmängder

Precis som man kan dela upp en mängd i olika delmängder kan man dela upp monoider, grupper, ringar och kroppar i olika delar. Vad gäller delmonoider och delgrupper ska vi bara nämna att dessa finns. Vi ska istället lägga tyngdpunkten på *ideal* som är en delmängd av ringar. Vi ska också nämna en del om delkroppar.

Tabell 1.1: Räknesätt för olika algebraiska strukturer.

Antal räknesätt	Algebraisk struktur	Räknesätt	Standard-beteckning
1	Monoid	$+$	M
2	Grupp	$+,-$	G
3	Ring	$+,-,*$	R
4	Kropp	$+,-,*,/$	K



Figur 1.1: Arv av egenskaper

(a) Delgrupper

Om vi har en grupp $\langle G, * \rangle$ och ur denna grupp kan plocka ut en delmängd H som också är en grupp, d v s $\langle H, * \rangle$ uppfyller villkoren för en grupp, säger vi att H är en **delgrupp** av G och skriver $H \leqslant G$.

Operationerna för en grupp och en delgrupp kan ha olika beteckningar men ska vara lika som funktioner. Om $\langle G, * \rangle$ och H är som ovan är $\langle H, \cdot \rangle$ en delgrupp till G om $\cdot$ och $*$ är lika som funktioner. Algoritmerna för att beräkna dessa operationer behöver dock ej vara lika.

För fortsatt läsning, se [Fra89] kapitel 1.

Struktur	Kropp								
	Ring								
	Grupp								
	Monoid								
	+ ass	+ enhet	+ invers	+ komm	· ass	· enhet	distr	· komm	· invers
medelvärde				x					
$(a, b) \mapsto a$	x								
min/max	x			x					
$(M_n, \cdot)$	x	x							
$(\{0, 1\}, \texttt{AND})$	x	x		x					
$(\{0, 1\}, \texttt{OR})$	x	x		x					
$(\{X^r \mid r \geqslant 0\}, \cdot)$	x	x		x					
$(\mathbb{N}, +)$	x	x		x					
$(\{0, 1, 2, 3, 4\}, + \bmod 5)$	x	x	x	x					
$\langle S_n, \cdot \rangle$	x	x	x						
$(M_n, +)$	x	x	x	x					
$\langle \mathbb{Z}, + \rangle$	x	x	x	x					
$\langle \{aX \mid a \in \mathbb{Z}\} \rangle$	x	x	x	x					
$\langle M_n, +, \cdot \rangle$	x	x	x	x	x	x	x		
$\langle \{aX^r \mid r \geqslant 0, a \in \mathbb{Z}\}, +, \cdot \rangle$	x	x	x	x	x	x	x	x	
$\langle \mathbb{Z}, +, \cdot \rangle$	x	x	x	x	x	x	x	x	
$(\mathbb{Q}, +, \cdot)$	x	x	x	x	x	x	x	x	x
$(\mathbb{R}, +, \cdot)$	x	x	x	x	x	x	x	x	x
$(\mathbb{C}, +, \cdot)$	x	x	x	x	x	x	x	x	x

Tabell 1.2: Sammanställning över strukturer och deras egenskaper där ass = operationen är associativ, enhet = det finns ett enhetselement m a p operationen, invers = till varje element finns det en invers m a p operationen, komm = operationen är kommutativ och distr = de två operationerna satisfierar den distributiva lagen.

(b) Ideal

Efter denna korta glimt på delgrupper går vi nu vidare till ideal som är det begrepp vi kommer att använda mest i fortsättningen.

För att skapa en del av en ring $\langle R, +, \cdot \rangle$ måste vi dels plocka ut en delmängd av elementen i R, dels se till att operationerna $+$ och $\cdot$ fungerar på ett användbart sätt. Den del av en ring vi kommer att använda oss av kallas *ideal* och definieras som:

Definition 1.8 (Ideal) *En delmängd I av en ring R som uppfyller villkoren*

$$aI \subseteq I \text{ och } Ia \subseteq I \text{ för alla } a \in R$$

kallas ett **ideal**. *Om I är ett ideal till $\langle R, +, \cdot \rangle$ är $\langle I, + \rangle$ en delgrupp till $\langle R, + \rangle$.*

Eftersom ett ideal "ärver" både addition och multiplikation av den ursprungliga ringen måste vi ha två tabeller. Den additiva tabellen ska enligt definitionen se ut som för en abelsk grupp (d v s sluten, additivt enhetselement, additiva inverser och symmetrisk). Eftersom vi kräver att

$$aI = \{an \mid n \in I\} \subseteq I \qquad (*)$$

ska multiplikationen vara sluten men i en vidare mening än förut: För att kvalificera in som en binär operation räcker det om (*) gäller för alla $a \in I$ men här ska den gälla för alla $a \in R$ så den multiplikativa tabellen ska vara sluten men inte bara med avseende på idealets element, den ska vara sluten med avseende på *alla element i hela ringen*.

Exempel. Låt $I = \{\text{alla jämna heltal}\}$ och låt addition och multiplikation vara den vanliga heltalsvarianten. Addition och multiplikation av jämna tal ger jämna tal så operationerna är slutna. Men multiplikation av ett udda tal och ett jämnt tal ger även den ett jämnt tal så multiplikationen är sluten för *alla* heltal! •

Det finns många typer av ideal, vi ska använda oss av en som kallas *huvudideal*.

Definition 1.9 (Huvudideal, generator för ideal) *Om A är en ring och $a \in A$ så är aA ett ideal. Detta ideal består av alla produkter $\{ax, x \in A\}$. a kallas en* **generator** *för idealet Aa. I stället för aA skriver vi $\langle a \rangle$.*[†]

Exempel. $2\mathbb{Z}$, mängden av alla jämna heltal, bildar en ring under addition och multiplikation. För alla heltal $a \in \mathbb{Z}$ gäller att $a2\mathbb{Z} = \{2 \cdot x \mid x \in 2\mathbb{Z}\} \subseteq 2\mathbb{Z}$ och alltså är $2\mathbb{Z}$ ett ideal i $\mathbb{Z}$, idealet genererat av 2, uttryckt som $\langle 2 \rangle$. •

Exempel. Låt R vara mängden av alla funktioner $f : \mathbb{R} \to \mathbb{R}$ och I alla $f \in R$ för vilka $f(0) = 0$. Om vi tar $f, g \in I$ och bildar $f + g$ ser vi att $f(0) + g(0) = 0$ så addition av element i I är sluten. Addition är associativ, funktionen $h(x) = 0$ tjänar som ett additivt enhetselement och till en given funktione f finns det alltid en funktion $-f$. Om vi har två funktioner f och g som uppfyller $f(0) = g(0) = 0$ har vi att $f(0) \cdot g(0) = 0$ men vi har också att oavsett vilken funktion $p \in R$ vi än tar att produkten $f(0) \cdot p(0) = 0 \cdot p(0) = 0$ så I är sluten med avseende på *alla* funktioner i R och I är därmed ett ideal i R. •

Definitionen av huvudideal med en generator kan generaliseras till flera generatorer:

Definition 1.10 (Huvudideal, flera generatorer) *Om A är en ring och $a_1, \ldots, a_n \in A$ så är*

$$I = \langle a_1, \ldots, a_n \rangle = \{\sum_{i=1}^{n} a_n x_n \mid x_n \in A\}$$

bestående av alla linjärkombinationer av $a_1, \ldots, a_n$ ett ideal som genereras av $(a_1, \ldots, a_n)$.

För fortsatt läsning, se [Fra89] kapitel 5.

Analogt med basvektorer som spänner upp ett vektorrum har vi:

Definition 1.11 (Spänna upp, bas) *Om ett ideal $I = \langle a_1, \ldots, a_n \rangle$, $a_i \in I$, så sägs $a_1, \ldots, a_n$* **spänna upp** *idealet I eller att $a_1, \ldots, a_n$ är en* **bas** *för I.*

Och med samma analogi kan vi också här ha "överflödiga" (linjärt beroende) generatorer (basvektorer):

Sats 1.12 *Låt $I = \langle p_1, \ldots, p_k \rangle$ vara ett ideal i en ring A. Om $p_k = a_1 p_1 + \cdots + a_{k-1} p_{k-1}$, $a_i \in A$ så genereras I av $p_1, \ldots, p_{k-1}$.*

[†]I det allmänna fallet skiljer man på aA och $Aa = \{xa, x \in A\}$. aA kallas då ett *högerideal* och Aa ett *vänsterideal*. Vi kommer uteslutande att arbeta med kommutativa ringar och då är $aA = Aa$, vi säger därför att aA är *idealet* (utan att skilja på höger och vänster) genererat av a.

Bevis: Om $b \in I$ har vi att $b = \sum_{i=1}^{k} b_i p_i$. Men $p_k = \sum_{j=1}^{k-1} a_j p_j$ vilket ger att vi kan skriva b som

$$\begin{aligned} b &= \sum_{i=1}^{k-1} b_i p_i + b_k \sum_{j=1}^{k-1} a_j p_j \\ &= \sum_{i=1}^{k-1} b_i p_i + \sum_{j=1}^{k-1} (b_k a_j) p_j \\ &= \sum_{i=1}^{k-1} c_i p_i \ , \quad c_i \in A \end{aligned}$$

•

Anmärkning. I linjär algebra är elementen i en bas linjärt oberoende men här behöver de *inte* vara det. Om $\{a_1, a_2\}$ är en bas för ett ideal kallas även t ex $\{a_1, a_2, a_1 + a_2\}$ för en bas trots att den innehåller linjärt beroende element.

(c) Delkroppar

Precis som man kan prata om delar av grupper och delar av ringar kan man prata om delar av kroppar:

Definition 1.13 (Delkropp) *Om F är en kropp, E en delmängd av F och E är en kropp så är E en delkropp till F.*

Exempel. De rationella talen $\mathbb{Q}$ är en delkropp till de reella talen $\mathbb{R}$ som i sin tur är en delkropp till de komplexa talen $\mathbb{C}$. •

1.3 Restklasser

Om man undersöker ett matematiskt objekt finner man att det karaktäriseras av ett antal olika egenskaper. Talet 4 har t ex egenskaperna att det är ett heltal, det är positivt, det är jämnt delbart med två och det är det minsta positiva heltal som inte är ett primtal. Vi kan säga att 4 på detta vis består av ett antal (eventuellt redundanta) *komponenter*.

Om vi bara är intesserade av en eller ett par av dessa komponenter skulle vi vilja kunna *reducera bort* alla andra komponenter. Kanske är vi bara intresserade av att givet ett heltal a avgöra om det är jämnt delbart med fyra eller inte?

Eftersom det problemet löses genom att se på resten av a vid division med fyra kan vi likväl säga att problemet löses genom att reducera bort kvoten vid division med fyra för att på så vis bara ha resten kvar.

Vi börjar i (a) med ett familjärt exempel: Räkning modulo 4. Detta leder till en uppdelning av $\mathbb{Z}$ i två komponenter—rest och kvot.

I (b) formaliserar vi dessa ideer inom ramarna för grupper och ringar. Det visar sig att så fort vi har en normal delgrupp eller ett ideal kan vi skapa en kvotgrupp eller kvotring och i och med det visa på ett sätt att räkna med restklasser.

I (c) har vi samlat en del notationer som är användbara i sammanhanget.

Slutligen formulerar vi i (d) nollekvivalens- och ekvivalensproblemen i ljuset av kvotgrupper och kvotringar och vi får fram villkor som definierar vad en normal respektive en kanonisk förenklare är.

(a) Specialfall: Räkning modulo 4

Ett av de enklaste exemplen är räkning med heltal modulo ett heltal. Säg att vi vill beräkna $10 + 3$ modulo 4. Vi har att $10 + 3 = 13 = 1 + 3 \cdot 4$ där 1 är *resten* och 3 är *kvoten* vid division av 10 med 4. På det här viset kommer många tal i $\mathbb{Z}$ att identifieras med 1: $\{\dots, -3, 1, 5, 9, 13, \dots\}$ ger alla resten 1 vid division med 4. Alla dessa tal bildar en ekvivalens*klass* och eftersom vi räknar med resten kallar vi den för en *restklass*.

Vad är det egentligen vi gör när vi räknar med restklasser? Varje tal $a \in \mathbb{Z}$ kan vi skriva som $a = r + k \cdot 4$ där r betecknar resten vid division med fyra och k heltalskvoten. Vi har fått en uppdelning av $\mathbb{Z}$ enligt

$$\mathbb{Z} \cong \mathbb{Z}_4 \times 4\mathbb{Z} \cong \{0, 1, 2, 3\} \times \{\dots, -4, 0, 4, 8, \dots\} \tag{1.1}$$

där $4\mathbb{Z}$ är alla heltal multiplicerade med 4. Vi kallar $\mathbb{Z}_4 = \{0, 1, 2, 3\}$ för *restkomponenter* modulo 4 och $\{\dots, -4, 0, 4, 8, \dots\}$ för *kvotkomponenter* modulo 4. Vad vi gör när vi räknar modulo fyra är att *stryka kvotkomponenten* och bara se på restkomponenten och det uttrycker vi som

$$\mathbb{Z}/4\mathbb{Z} \ .$$

Insatt i ekvation (1.1) får vi att

$$\mathbb{Z}/4\mathbb{Z} \cong (\mathbb{Z}_4 \times 4\mathbb{Z})/4\mathbb{Z} \cong \mathbb{Z}_4$$

och vi ser att vad vi får kvar är just $\mathbb{Z}_4 = \{0, 1, 2, 3\}$.

Vi vet nu att vi kan skriva $\mathbb{Z}$ som en mängd talpar (a, b) där a är resten och b är kvoten vid division med 4. När vi räknar modulo 4 är, eftersom vi bara är intresserade av resten, alla talpar som kan skrivas t ex på formen $(1, i)$, $i \in \mathbb{Z}$, likvärdiga. Vi får på detta vis fyra olika "tal" till förfogande, $\{\{(0, i)\}, \{(1, j)\}, \{(2, k)\}, \{(3, l)\}\}$ där $i, j, k, l \in \mathbb{Z}$, men för att spara utrymme skriver vi $[0]$ för $\{0, i\}$, $i \in \mathbb{Z}$ och $[1]$, $[2]$ och $[3]$ på samma sätt och vi får att $\{\{0, i\}, \dots\} = \{[0], [1], [2], [3]\}$.

Anmärkning. I stället för t ex $[1]$ inom klamrar kan vi lika gärna skriva $[13]$, $[29]$ eller något annat tal som ger resten 1 vid division med fyra, men eftersom det blir ganska krångligt att avgöra om t ex $[1373] = [1]$ eller inte vill vi gärna ha ett bestämt sätt att "ge namn" åt restklassen och vi väljer att kalla den för $[1]$ eftersom $0 \leqslant 1 \leqslant 3$.

Vid praktisk räkning modulo 4 räknar man i två steg: Först utför man sin räkneoperation (här: addition av två heltal) och sedan *reducerar* man resultatet till ett tal mellan 0 och 3. Det ser det ut som att vi räknar med talen 0, 1, 2 och 3 men vad vi i själva verket räknar med är restklasserna $[0]$, $[1]$, $[2]$ och $[3]$. Att beräkna $[2] + [3]$ går till på så vis att man först räknar som vanligt med representanter ur restklasserna, $2 + 3 = 5$, och sedan "gör om" 5 (som i sin tur är en representant till en restklass) till en restklass genom att "reducera" det modulo 4 och vi får att $[2] + [3] = [1]$.

Vi räknar i första steget på vanligt vis med representanter från två restklasser men i och med att vi sedan reducerar resultatet till en restklass utgör denna räkning egentligen *räkning med restklasser*.

Anmärkning. Vi sa tidigare att vi med restklasser reducerar *bort* en eller flera komponenter. Eftersom räkning med restklasser tydligen fungerar som räkning med representanter ur restklasserna ser vi att restklasserna har *ärvt* alla egenskaper och dessutom fått en *ny* egenskap: En slags ihopklumpning där alla element i restklassen anses vara likvärdiga.

(b) Allmänna fallet: En generalisering

En naturlig generalisering är att ta ett "valfritt" matematiskt objekt, på något vis "dela upp" det i ett antal komponenter och sedan "reducera bort" ett par av dessa komponenter. Vi ska nu visa hur detta går till för grupper och ringar.

När vi räknade modulo ett heltal fick vi mängder som kunde set ut som $\{\dots,-3,1,5,9,\dots\}$ och vi kallade dem för restklasser. Dessa restklasser är en speciell form av *sidoklasser*:

Definition 1.14 (Sidoklass) *Låt H vara en delgrupp av $\langle G,*\rangle$. Delmängden $a*H=\{a*h \mid h\in H\}$, $a\in G$, av G kallas* **vänster sidoklass** *som innehåller a och $H*a=\{h*a \mid h\in H\}$ kallas analogt för* **höger sidoklass** *som innehåller a.*

Exempel. Låt $G=\langle\mathbb{Z},+\rangle$ och $H=\{\dots,-4,0,4,8,\dots\}$. Hur ser sidoklasserna ut?

Eftersom en sidoklass enligt definitionen är relaterad till ett element $a\in G=\mathbb{Z}$ tar vi ett element ur $\mathbb{Z}$ på måfå, säg $a=7$. Vi får då att vänster sidoklass som innehåller 7 är

$$\begin{aligned}7+H=\{7+h \mid h\in H\}=\{7+h \mid h\in\{\dots,-8,-4,0,4,\dots\}\ \}\\ =\{\dots,-1,3,7,11,\dots\}=[3]\ .\end{aligned}\quad(1.2)$$

Finns det fler sidoklasser? Om vi fortsätter att plocka element ur $\mathbb{Z}$ som *inte* finns i någon sidoklass vi har "beräknat" kommer vi till slut mycket riktigt att få en uppdelning av $\mathbb{Z}$ i fyra sidoklasser som (givetvis) är de vanliga restklasserna modulo 4. •

Kan vi räkna med sidoklasser? Vi vet att det går bra att räkna modulo ett heltal genom att räkna med restklassrepresentanter, men kan vi göra så också med sidoklasser, d v s räkna med sidoklassrepresentanter?

Sats 1.15 *Låt H vara en delgrupp till G. Sidoklassaddition är väl definierad genom*

$$(a*H)*(b*H)=(a*b)*H$$

om och endast om vänster och höger sidoklasser sammanfaller så att $a*H=H*a$ för alla $a\in G$.

Bevis: $\Rightarrow$: Anta att $(a*H)*(b*H)=(a*b)*H$ ger en väldefinierad binär operation på vänster sidoklasser. Låt $a\in G$. Vi vill visa att $a*H$ och $H*a$ är samma mängder.

Låt $x\in a*H$. Om vi väljer representanter $x\in a*H$ och $a^{-1}\in a^{-1}*H$ har vi att $(x*H)*(a^{-1}*H)=(x*a^{-1})*H$. Om vi å andra sidan väljer representanter

$a \in a * H$ och $a^{-1} \in a^{-1} * H$ ser vi att $(a * H) * (a^{-1} * H) = e * H = H$. Eftersom vi antog att vänster sidoklassaddition var väldefinierad måste vi ha att $x * a^{-1} = h \in H$. Då är $x = h * a$ så $x \in H * a$ och $a * H \subseteq H * a$.

Anta nu att $y \in H * a$ så att $y = h_1 * a$ för något $h_1 \in H$. Då är $y^{-1} = a^{-1} * h_1^{-1} \in a^{-1} * H$. Eftersom $(a^{-1} * H) * (a * H) = H$ och vänster sidoklassaddition antas vara väl definierad ser vi att $y^{-1} * a = h_2 \in H$ så $y = a * h_2^{-1}$. Det här visar att $H * a \subseteq a * H$ och följdaktligen är $a * H = H * a$.

$\Leftarrow$: Eftersom vänster och höger sidoklasser sammanfaller kan vi istället prata om *sidoklasser* utan att specifiera höger eller vänster. Anta att vi vill beräkna $(a * H) * (b * H)$. Om vi väljer representanter $a \in a * H$ och $b \in b * H$ får vi sidoklassen $(a * b) * H$. Om vi istället väljer andra representanter $a * h_1 \in a * H$ och $b * h_2 \in b * H$ får vi sidoklassen $(a * h_1 * b * h_2) * H$. Vi måste visa att dessa är samma sidoklasser. Vi har att $h_1 * b \in H * b = b * H$, så $h_1 * b = b * h_3$ för något $h_3 \in H$ och alltså

$$(a * h_1) * (b * h_2) = a * (h_1 * b) * h_2 = a * (b * h_3) * h_2 = (a * b) * (h_3 * h_2)$$

och $(a * b) * (h_3 * h_2) \in (a * b) * H$. Därför ligger $a * h_1 * b * h_2$ i $(a * b) * H$ och beviset är klart. •

När vi nu har en binär operation på sidoklasserna undrar vi om det är så att dessa också bildar en grupp?

Sats 1.16 *Låt H vara en delgrupp till G vars vänster och höger sidoklasser sammanfaller. Då bildar sidoklasserna till H en grupp G/H under den binära operationen $(a * H) * (b * H) = (a * b) * H$.*

Bevis: Eftersom vi vet att operationen är en binär operation har vi kvar att visa att den är associativ, att det finns ett enhetselement och att det finns inverser. Eftersom vi räknar med sidoklasser genom att räkna med representanter i G och G är en grupp följer att också G/H är en grupp. •

Om H är en delgrupp till en *abelsk* grupp G får vi följande specialfall av sats 1.16.

Korollarium 1.17 *Låt H vara en delgrupp till en abelsk grupp G vars vänster och höger sidoklasser sammanfaller. Då bildar sidoklasserna till H en abelsk grupp G/H under den binära operationen $(a+H)+(b+H) = (a+b)+H$.*

En delgrupp H som uppfyller ovanstående krav (vänster och höger sidoklasser sammanfaller) kallas en **normal delgrupp**.

Vi har nu gett mening åt följande definition:

Definition 1.18 (Faktorgrupp) *G/H i sats 1.16 är en grupp och kallas* **faktorgruppen** *eller* **kvotgruppen** *av G modulo H.*

Tidigare pratade vi om att "dela upp i komponenter" och att "reducera bort" vissa av dessa komponenter. När vi bildar en kvotgrupp av en grupp G modulo en delgrupp H är det just det vi gör—de "komponenter" som specifieras av H kommer att "reduceras bort" och de komponenter vi får kvar är sidoklasserna.

Vi ska nu gå vidare och konstruera samma sak som ovan fast för ringar. Eftersom en ring dessutom är en grupp (under addition) gäller speciellt att vi av en ring kan bilda kvotgrupper.

Analogt med sats 1.15 kan vi också multiplicera sidoklasser:

Sats 1.19 *I är ett ideal till en ring R om och endast om multiplikation av additiva sidoklasser till I enligt*

$$(a + I)(b + I) = ab + I$$

är väldefinierad.

Bevis: Att I är ett ideal är detsamma som att $ai \in I$ och $ib \in I$ för alla $a, b \in R$ och $i \in I$.

Anta först att $ai \in I$ och $ib \in I$ för alla $a, b \in R$ och alla $i \in I$. Låt $i_1, i_2 \in I$ vara sådana att $a + i_1$ och $b + i_2$ också är representanter till sidoklasserna $a + I$ och $b + I$ innehållande a och b. Då gäller att

$$(a + i_1)(b + i_2) = ab + ai_2 + i_1b + i_1i_2 \ .$$

Eftersom ai_2 och i_1b ligger i I ser vi att $(a + i_1)(b + i_2) \in ab + I$.

Anta å andra sidan att multiplikation av additiva sidoklasser är väl definierad. Låt $a \in R$ och se på sidoklassprodukten $(a + I)I$. Om vi väljer representanter så att $a \in (a + I)$ och $0 \in I$ ser vi att $(a + I)I = a0 + I = 0 + I = I$. Eftersom vi även kan beräkna $(a + I)I$ genom att välja $a \in (a + I)$ och vilket som helst $i \in I$ ser vi att $ai \in I$ för alla $i \in I$. Ett likartat resonemang som utgår från produkten $I(b + I)$ visar att $ib \in I$ för alla $i \in I$. •

Analogt till sats 1.16 har vi följande:

Sats 1.20 *Låt I vara ett ideal till ringen R. Då bildar (de additiva) sidoklasserna till I en ring R/I med en binär operation definierad av*

$$(a+I)+(b+I)=(a+b)+I$$

och

$$(a+I)(b+I)=ab+I$$

Och analogt till definition 1.18:

Definition 1.21 (Kvotring) *Ringen R/I i sats 1.20 är* **faktorringen** *eller* **kvotringen** *till R modulo I.*

(c) Några notationer

Ofta stöter man inom matematiken på många notationer som till sitt yttre ser helt olika ut men som ändå uttrycker samma sak. Varför är det så? En förklaring kan förstås vara att olika författare föredrar olika notationer, men det finns fler anledningar än så. Vi har tidigare (se definition 1.21) kallat alla restklasser till en ring modulo ett ideal för uttryck av typen R/I och om a ligger i R har vi med $[a]$ menat den restklass modulo I som innnehåller a. Vi ska nu visa ett par sätt till att uttrycka (nästan) samma sak, fördelen med dessa sätt är att de är mer kompakta skrivsätt än vad man skulle kunna få på det "gamla" viset.

Vi börjar med att se på vanlig räkning modulo 4.

Vi såg tidigare hur detta ledde till en uppdelning av ett heltal i en kvot och en rest. Om ett tal a är jämnt delbart med fyra skriver vi

$$a \equiv 0 \bmod 4$$

och det är samma sak som att säga att $a \in \{\dots, -4, 0, 4, 8, \dots\}$.

Om vi har två tal a och b och de ligger i *samma* restklass modulo ett tal, t ex 4 skriver vi det som

$$a \equiv b \bmod 4\ .$$

Om vi flyttar b till vänstra sidan av $\equiv$-tecknet får vi

$$a - b \equiv 0 \bmod 4$$

och det innebär förstås, precis som förut, att a och b ligger i samma restklass modulo 4. Om vi sätter $c = a - b$ får vi att $c \equiv 0 \mod 4$ och eftersom det betyder att c är jämnt delbart med fyra måste också $a-b$ vara det, m a o $a - b \in \{\dots, -4, 0, 4, 8, \dots\}$.

I det mer generella fallet med grupper och ringar menar vi "samma sak".

Om vi har en ring† R, ett ideal I och $a, b \in R$ menar vi med

$$a \equiv 0 \bmod R \tag{1.3}$$

att $a \in R$. Om man ser på kvotringen som en uppdelning av elementen i R i komponenter betyder det att alla a:s komponenter som vi *inte* reducer bort är noll.

Om a och b ligger i samma restklass modulo R skriver vi det som

$$a \equiv b \bmod R \tag{1.4}$$

och enligt komponentresonemanget betyder det att a och b är lika i de komponenter som *inte* reduceras bort. Precis som med heltalen kan vi flytta över b till vänstra ledet och få

$$a - b \equiv 0 \bmod R\ . \tag{1.5}$$

Vi ser att det betyder att alla komponenter i differensen $a - b$ som vi inte reducerat bort är noll och det är ju helt naturligt eftersom det ligger i samma restklass. D v s om vi bara ser till de komponenter vi inte reducerar bort är a och b identiska.

(d) Ekvivalens och kanonisk förenklare

I sektion 0.4 diskuterade vi nollekvivalens- och ekvivalensproblemen, nu ska vi precisera vad vi menar med dessa i det här sammanhanget. Gemensamt för båda problemen är att vi endast är intresserade av de komponenter som blir kvar efter reduktion modulo en delgrupp eller ett ideal. Vad vi i den här sektionen säger om kvotringar gäller även för kvotgrupper.

För att lösa nollekvivalensproblemet skulle vi skapa en funktion som vi kallade för S som om $a \sim 0$ ger att $S(a) \equiv 0$. Ekvation 1.3 talar om att det i R/I är

†Vad som sägs här om ringar och ideal gäller också grupper och delgrupper.

samma sak som att säga att $a \in I$. En normal förenklare är alltså en funktion S som uppfyller att

$$a \equiv 0 \bmod I \Leftrightarrow a \in I \Leftrightarrow S(a) \equiv 0 \ , \tag{1.6}$$

d v s den normala formen av a är noll.

För att lösa ekvivalensproblemet skulle vi skapa en funktion som vi kallade för C som om $a \sim b$ ger att $C(a) \equiv C(b)$. Ekvation 1.4 talar om att det i R/I är samma sak som att säga att a och b ligger i samma restklass modulo I eller att $a - b \in I$. En kanonisk förenklare är alltså en funktion C som uppfyller att

$$a \equiv b \bmod I \Leftrightarrow a - b \in I \Leftrightarrow C(a) \equiv C(b) \ , \tag{1.7}$$

d v s den kanoniska formen av a är samma som den kanoniska formen av b.

1.4 Sammanfattning

Vi har i det här kapitlet skaffat oss ett bra grundordförråd inom den abstrakta algebran. Strukturerna monoider, grupper, ringar och kroppar har fått sin förklaring och vi har också tittat på hur man kan räkna på delar av dessa strukturer med hjälp av restklasser.

I nästa kapitel ska vi spinna vidare på samma tråd. Vi ska titta närmare på en speciell ring, polynomringen, och dess delar (ideal). Denna ring kommer att spela en central roll i fortsättningen.

Noter

[1] [BW93] sidan 13, "A common foundation for ... almost all of modern mathematics ... the theory of sets."

[2] [Fra89] sidan 52

Kapitel 2

Polynom

Att studera Gröbnerbaser innebär att studera ideal till polynomringen och speciellt idealens baser. Vad ett polynom är torde vara välbekant för de flesta. Allmännt brukar ett polynom beskrivas som en ändlig summa[1]

$$p(x) = \sum a_i x^i$$

där a_i är koefficienter i en talmängd A. Detta sätt att se på polynom är i många fall fullt tillräckligt men här måste vi gå ett par steg djupare.

Hela detta kapitel är som ett stort exempel på (stora delar av) kapitel 1 i fallet med polynom.

Vi börjar i sektion 2.1 med att definiera vad vi menar med ett polynom i en variabel.

Ofta klarar man sig inte med en variabel utan man behöver polynom i flera variabler. I sektion 2.2 utgår vi från polynom i en variabel och generaliserar dessa till polynom i flera variabler. En speciell sorts polynom är de som endast består av en enda term. Dessa kallas för monom och dessa kommer att spela en viktig roll längre fram.

I sektion 2.3 ska vi titta på hur ett ideal spänns upp av sina generatorer, när två ideal som spänns upp av olika uppsättningar generatorer är lika och vad som menas med basbyte för ideal.

VIKTIGT!!! Det vi nu kommer att säga om polynom kan se väldigt konstigt ut. Det kan t o m se så konstigt ut att man kan tro att det är något helt nytt, men

det är det inte! Det här är *precis samma polynom* som vi har räknat med sedan flera år tillbaka. Addition, subtraktion, multiplikation och division av polynom kommer vi att använda precis som vanligt. Skillnaden är att vi nu ska beskriva polynom och vad vi vet om dem på ett sätt som liksom räkningen i kapitel 1 har *större giltighet*. Och vitsen med detta är att vi ska kunna räkna med polynom på andra sätt än förut. Men allt vi hittills har lärt oss om polynom gäller (givetvis) fortfarande!

2.1 Polynom i en variabel

Låt A vara en kommutativ ring.

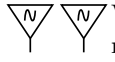
Om vi skriver om ett polynom som en lista av talpar, ett talpar för varje term i polynomet, där ett talpars första element är termens variabel och ett talpars andra element är koefficient framför termens variabel får vi en representation av polynom som även kan ses som en funktion. T ex så motsvaras polynomet $p(X) = 4X^3 + 7X - 2$ av listan $[\dots, (X^4, 0), (X^3, 4), (X^2, 0), (X^1, 7), (X^0, -2)]$. Listan definierar en funktion $f(s) : S \to A$ där $f(X^3) = 4$, $f(X^1) = 7$, $f(X^0) = -2$ och $f(s) = 0$ för övrigt, $S = \{X^0, X^1, X^2, \dots\}$ och A är koefficientringen (den ring som koefficienterna tillhör).

Definition 2.1 (Polynom) *Ta en oändligt cyklisk grupp genererad av ett element X. Låt S vara delmängden bestående av potenser X^r, $r \geq 0$. Då är S en monoid. Med mängden polynom $A[X]$, A en ring, menas mängden funktioner $S \to A$ som är 0 för alla förutom ett ändligt antal element av S. För varje element $a \in A$ menar vi med aX^n funktionen som har värdet a för X^n och noll för alla andra element i S.*

Det är klart att ett polynom kan skrivas som en ändlig summa

$$f(X) = a_0X^0 + \cdots + a_nX^n \tag{2.1}$$

för något tal $n \in \mathbb{N}$ och $a_i \in A$. För a_0X^0 skriver vi a_0, för a_1X^1 skriver vi a_1X och för $1X^n$ skriver vi X^n.


Vi skriver här ”variablerna” med versaler i stället för gemener. I sammanhanget spelar det ingen roll, men ibland kan man vilja skilja mellan $A[X_1, \dots, X_n]$ där X_i betecknar en variabel och $A[x_1, \dots, x_n]$ där x_i betecknar algebraiskt oberoende element över $A[x_1, \dots, x_{i-1}]$[2].

Addition och multiplikation av polynom med koefficienter i en ring R definieras på ett naturligt sätt. Om

$$f(X) = a_0 + a_1X + a_2X^2 + \cdots$$

och

$$g(X) = b_0 + b_1X + b_2X^2 + \cdots$$

så definierar vi summan som

$$f(X) + g(X) = c_0 + c_1X + c_2X^2 + \cdots$$

där $c_n = a_n + b_n$ och produkten som

$$f(X)g(X) = d_0 + d_1X + d_2X^2 + \cdots$$

där $d_n = \sum_{i=0}^{n} a_i b_{n-i}$.

När vi nu har en mängd (polynom) och två operationer definierade på mängdens element (polynomaddition och polynommultiplikation) vore det ju trevligt om vi kunde prata om en *polynomring* och nu visar det sig att alla villkor för en ring är uppfyllda och vi säger att

Sats 2.2 *$A[X]$ i definition 2.1 är en ring. Om A är kommutativ är $A[X]$ en kommutativ ring.*

Bevis: Som additivt nollelement har vi polynomet $0X^0 = 0$ och som multiplikativt $1X^0 = 1$. För varje polynom f kan vi bilda en additiv invers genom att skifta tecken på alla koefficienter. Addition av polynom sker "komponentvis" och är därför associativ. För multiplikation har vi att

$$\begin{aligned}
[a(X)b(X)]c(X) &= (\sum_i a_iX^i \sum_j b_jX^j) \sum_k c_kX^k \\
&= (\sum_n (\sum_{i=0}^{n} a_i b_{n-i})X^n) \sum_k c_kX^k = \sum_s (\sum_{n=0}^{s} (\sum_{i=0}^{n} a_i b_{n-i})c_{s-n})X^s \\
&= \sum_s (\sum_{i+j+k=s} a_i b_j c_k)X^s = \sum_s (\sum_{m=0}^{s} a_{s-m} (\sum_{j=0}^{m} b_j c_{m-j}))X^s \\
&= \sum_i a_iX^i (\sum_m (\sum_{j=0}^{m} b_j c_{m-j})X^m) = \sum_i a_iX^i (\sum_j b_jX^j \sum_k c_kX^k) \\
&= a(X)[b(X)c(X)]
\end{aligned}$$

så multiplikation är också associativ. Vidare har vi att

$$\begin{aligned} c(X)[a(X)+b(X)] &= \sum_i c_i X(\sum_j a_j X^j + \sum_k b_k X^k) \\ &= \sum_i c_i X^i \sum_j (a_j + b_j) X^j = \sum_s (\sum_{i=0}^{s} c_i (a_{s-i} + b_{s-i})) X^s \\ &= \sum_s (\sum_{i=0}^{s} c_i a_{s-i}) X^s + \sum_s (\sum_{i=0}^{s} c_i b_{s-i}) X^s = c(X)a(X) + c(X)b(X) \end{aligned}$$

vilket visar att distributiva lagen gäller och vi är klara! •

Exempel. Polynomen med heltalskoefficienter betecknas $\mathbb{Z}[X]$ och med rationella koefficienter $\mathbb{Q}[X]$. Om vi menar en polynomring där koefficienterna bildar en kropp men inte någon speciell kropp skriver vi $K[X]$. •

För fortsatt läsning, se [Fra89] kapitel 4.5.

2.2 Polynom i flera variabler

Nu ska vi övergå till polynom i flera variabler. Ett polynom i två variabler X och Y kan genom en enkel omskrivning ses som ett polynom i en variabel Y med koefficienter som är polynom i variabeln X eller tvärtom:

$$\begin{aligned} f(X,Y) &= 3XY^2 + 7X^3Y + 7X - 3Y \\ &= (3X)Y^2 + (7X^3 - 3)Y + (7X)1 \\ &= (7Y)X^3 + (3Y^2 + 7)X - (3Y)1 \end{aligned} \tag{2.2}$$

Man kan alltså se på polynom i två (oberoende) variabler som ett polynom i den andra variabeln och med koefficienter som är polynom i den första variabeln eller vice versa. Vi kan skriva att

$$(A[X])[Y] = (A[Y])[X] = A[X,Y]. \tag{2.3}$$

Sats 2.3 *$A[X,Y]$ i ekvation (2.3) är en (kommutativ) ring.*

Bevis: Eftersom $A[X]$ enligt sats 2.2 är en (kommutativ) ring och Y är en variabel över $A[X]$ följer att $(A[X])[Y] = A[X,Y]$ är en (kommutativ) ring. •

Med "variabel" menas generatorn till monoiden i definition 2.1 av polynom.

På samma sätt kan vi skapa en polynomring i godtyckligt antal variabler:

$$A[X_1, \dots, X_n] = (A[X_1][X_2] \dots [X_{n-1}])[X_n] = A[X]^{\dagger} \tag{2.4}$$

där varje element $f \in A[X]$ kan uttryckas som en *unik* summa

$$\sum a_{(\nu)} T_{(\nu)}(X) = \sum a_{(\nu)} X_1^{\nu_1} \cdots X_n^{\nu_n} \tag{2.5}$$

Sats 2.3 och induktion ger att $A[X_1, \dots, X_n]$ enligt ekvation (2.4) är en (kommutativ) ring.

Definition 2.4 (Polynom i flera variabler) *Ringen $A[X_1, \dots, X_n]$ i ekvation (2.4) kallas* **polynomringen i n variabler**.

De enskilda delarna i summan i ekvation (2.5) brukar benämnas **monom**:

Definition 2.5 (Monom, term) *En del*

$$a_{(\nu)} T_{(\nu)}(X)$$

i summan (2.5) kallas för ett **monom**. *Produkterna*

$$T_{(\nu)}(X) = X_1^{\nu_1} \cdots X_n^{\nu_n}$$

kallas för **primitiva monom** *eller* **termer** *och $a_{(\nu)}$ dess* **koefficienter**. *Med*

$$M = M(X) = M(X_1, \dots, X_n)$$

menar vi alla monom i variablerna $X_1, \dots, X_n$ och med

$$T = T(X) = T(X_1, \dots, X_n)$$

alla termer i variablerna $X_1, \dots, X_n$.

Exempel. $3X^2Y$ är ett monom i variablerna X och Y. Monomet är en produkt av koefficienten 3 och termen X^2Y. •

[†]Med X i ekvation (2.4) menar vi i själva verket variablerna $X_1, \dots, X_n$. Vi kommer i fortsättningen att använda X för att beteckna både en och flera variabler—av sammanhanget framgår vilka.

Ibland vill man tala om en polynomring i allmännhet, till skillnad från en speciell ring, och då brukar man använda sig av olika standardbeteckнngar beroende på vilka slags koefficienter de olika termerna kan ha, se tabell 2.1.

Polynomringen visar sig vara en specialisering av en mer generell struktur—monoidringen. Det finns många sätt att beroende på dessa ringars egenskaper klassificera dem. En huvudidealring har vi redan sett, andra är Euklidiska områden och ringar som medger unik faktorisering. Det finns många algoritmer med gamla anor för polynom, t ex Euklides algoritm för att bestämma två polynoms största gemensamma divisor och kinesiska restsatsen för att räkna i restklasser. För fortsatt läsning, se [BW93] kapitel 2.

Tabell 2.1: Standardbeteckningar för polynomringar.

Typ av koefficienter	Standardbeteckning
Ingen speciell typ	$P[X]$
Kommutativ	$A[X]$
Kropp	$K[X]$

Nollställe för polynom

Definition 2.6 (Nollställe) *Låt $f \in K[X_1, \ldots, X_n]$. Med $\mathrm{Z}(f)$ menar vi alla nollställen till f, d v s alla n-tupler $(a_1, \ldots, a_n) \in K^n$ som uppfyller att $f(a_1, \ldots, a_n) = 0$.*

2.3 Ideal i polynomringen

Ta ett polynom, t ex $g(X) = X + 2 \in A[X]$ och bilda sedan en mängd $I = \{ag \mid a \in A[X]\}$. Om vi nu tar ett element $i \in I$ och multiplicerar detta med ett valfritt polynom $p(X) \in A[X]$ får vi en produkt pi. Men varje element i I var ju i sig en produkt ag av ett polynom $a \in A[X]$ och g så vi har att $pi = pag$ för något a. Eftersom produkten av två polynom är ett polynom kan vi skriva att $pi = pag = p_1 g$ för något $p_1 \in A[X]$. Nu definierade vi I som mängden av alla produkter av ett polynom och g, så speciellt måste gälla att $p_1 g \in I$. Eftersom polynomet p var valfritt ser vi att I är ett ideal enligt definition 1.8.

Detta kan vi sammanfatta i följande sats:

Sats 2.7 *Mängden $I = \{ag\} = \langle g \rangle$, $a, g \in A[X]$ är ett ideal,* **idealet genererat av** g.

Att skapa ett ideal i polynomringen är som att "lyfta upp" alla polynom med en generator, det finns med andra ord inget element kvar (förutom 0) som är "mindre" än generatorn.

Exempel. $\langle X \rangle$ är ett ideal där alla polynom har multiplicerats med X och resultatet blir att det inte finns några polynom av grad 0 (konstanter) kvar, det finns inte heller något polynom kvar med en konstantterm. •

Ett ideal kan naturligtvis ha flera generatorer. För $g_1, \ldots, g_n \in A[X]$ gäller att $\langle g_1, \ldots, g_n \rangle = \{a_1 g_1 + \cdots + a_n g_n\}$ där $a_i \in A[X]$, d v s alla linjärkombinationer av generatorerna och polynom i variablerna $X = X_1, \ldots, X_n$.

Om ett ideals generatorer finns mycket att säga, vi börjar med en definition:

Definition 2.8 (Spänna upp ideal i polynomringen) *Om ett ideal $I = \langle p_1, \ldots, p_n \rangle \subseteq P[X]$, $p_i \in P[X]$, så sägs polynomen $p_1, \ldots, p_n$ vara en* **bas** *för idealet I eller* **spänna upp** *idealet I.*

Precis som i fallet med vektorrum där man kan ha linjärt beroende vektorer kan man i ett ideal ha linjärt beroende generatorer. Om $I = \langle X, Y, 3X + 4Y \rangle$ kan vi skriva den tredje generatorn som en linjärkombination av de två första så $\langle X, Y, 3X + 4Y \rangle = \langle X, Y \rangle$. Eftersom koefficienterna a_i i våra linjärkombinationer $\sum a_i g_i$ numera är polynom i stället för "vanliga tal" kan man få lov att tänka till ibland. Idealet genererat av $\{X, Y, 3X + 2XY^2\}$ har bara två linjärt oberoende generatorer ty $3X + 2XY^2 = 3(X) + 2XY(Y)$, alltså är även $\langle X, Y, 3X + 2XY^2 \rangle = \langle X, Y \rangle$.

För ett och samma ideal har vi alltså tre baser:

$$\langle X, Y \rangle = \langle X, Y, 3X + 4Y \rangle = \langle X, Y, 3X + 2XY^2 \rangle$$

Ett vektorrums basvektorer måste som bekant vara linjärt oberoende för att få kallas en bas. Polynomen som spänner upp ett ideal får däremot vara linjärt beroende och ändå kallas bas.

Härnäst ska vi diskutera utförligare om dessa basbyten.

Basbyte för ideal i polynomringen

I linjär algebra har vi sett hur användbara basbyten kan vara för att lösa linjära ekvationssystem, spelar basbyten för ideal en jämförbar roll?

Exempel. Säg att idealet I genereras av $\{xy^2 - y, x^2y - y^2\} = \{p_1, p_2\}$. Att polynomen $q = x^2y^2 - y^3$ och $r = -x^2y^2 + xy$ ligger i I ser man lätt ty

$$q - y \cdot p_2 = x^2y^2 - y^3 - x^2y^2 + y^3 = 0$$
$$r + x \cdot p_1 = -x^2y^2 + xy + x^2y^2 - xy = 0$$

eftersom om ett polynom kan skrivas som en linjärkombination av ett ideals generatorer ligger polynomet i idealet.

Vi vet nu att q och r ligger i I och därför ska också en linjärkombination av dessa enligt definition 1.10 ligga i I. Låt $t = q + r = xy - y^3$. Kan vi skriva t som en linjärkombination av p_1 och p_2? Vi kompletterar basen för I med polynomet $p_3 = x \cdot p_1 - y \cdot p_2 = y^3 - xy$ (eftersom p_3 är en linjärkombination av andra generatorer förändrar vi inte idealet) och har nu att I genereras av

$$\{xy^2 - y, x^2y - y^2, y^3 - xy\}$$

och nu är det inga problem att skriva t som en linjärkombination av generatorerna eftersom $t = -1 \cdot p_3$ och alltså ligger även $t = q + r$ i I, precis som det ska! •

I exemplet såg vi att om vi vill svara på frågan om ett polynom ligger i ett ideal eller inte kan ett basbyte, precis som i fallet med linjära ekvationssystem, förenkla problemet avsevärt. Basbytet gick till på så vis att vi tog två generatorer, ett par (i andra fall kan vi förstås ha många fler), utförde en operation på paret och kompletterade basen med resultatet. Denna procedur är så generellt användbar att den fått ett namn: *Critical pair/completion*, se [BL83]. Operationen vi gjorde var att beräkna parets *s-polynom* (se sektion 4.2) och att på detta vis beräkna s-polynom och komplettera basen är precis vad vi senare ska göra för att beräkna ett ideals Gröbnerbas, men mer om det i del II!

Nollställe för ideal

Definition 2.9 (Nollställe) *Låt I vara ett ideal i $K[X_1, \ldots, X_n]$. Med $\mathrm{Z}(I)$ menar vi alla n-tupler $(a_1, \ldots, a_n) \in K^n$ som för alla $f \in I$ uppfyller att $f(a_1, \ldots, a_n) = 0$.*

Sats 2.10 *Låt $f_1, \ldots, f_m \in K[X_1, \ldots, X_n]$. Då gäller att*

$$\mathrm{Z}(f_1, \ldots, f_m) = \mathrm{Z}(\langle f_1, \ldots, f_m \rangle)$$

Bevis: Vi har att

$$\langle f_1, \ldots, f_m \rangle = \{p_1 f_1 + \cdots + p_m f_m\}$$

där $p_i \in K[X_1, \ldots, X_n]$.

$\Rightarrow$: Om vi för $(a) = (a_1, \ldots, a_n) \in K^n$ har att $f_1(a) = \cdots = f_m(a) = 0$ gäller att $p_1 f_1(a) + \cdots + p_m f_m(a) = 0$ för alla $p_1, \ldots, p_m$.

$\Leftarrow$: Om det finns ett $(a) \in K^n$ så att det för alla $p_1, \ldots, p_m$ gäller att $p_1 f_1(a) + \cdots + p_m f_m(a) = 0$ gäller speciellt att $f_1(a) = \cdots = f_m(a) = 0$ (ta t ex $p_1 = 1$ och alla andra $p_i = 0$). •

Sats 2.11 *Om vi har två ideal $\langle P \rangle$ och $\langle Q \rangle$ gäller att*

$$\langle P \rangle = \langle Q \rangle \Longrightarrow \mathrm{Z}(P) = \mathrm{Z}(Q)$$

Bevis: Enligt sats 2.10 har vi

$$\mathrm{Z}(P) = \mathrm{Z}(\langle P \rangle) = \mathrm{Z}(\langle Q \rangle) = \mathrm{Z}(Q)$$

•

2.4 Restklasser

I kapitel 1.3 diskuterade vi restklasser i ett allmännt perspektiv. Här ska vi titta närmare på specialfallet en polynomring modulo ett ideal.

Vi börjar i (a) med den bekanta polynomdivisionen som fungerar analogt med heltalsdivision, fast med polynom. Vi får också här en uppdelning i en kvot och en rest men till skillnad från fallet med räkning modulo heltal blir det inte lika enkelt att räkna upp restklasserna.

Om vi generaliserar till ideal som genereras av en mängd monom stöter vi problem som vi ska se i (b). Att ta fram uttryck för restklasserna och att räkna med representanter går bra men vi kan inte reducera resultatet till en speciell restklass som vi har gjort tidigare.

I (c) ska vi se på det mest generella fallet när idealet genereras av en mängd polynom. Nu får vi ännu fler problem: Vi kan inte ens ställa upp ett uttryck

för restklasserna! Räddningen är dock inte långt borta: Den heter reduktion och Gröbnerbaser och är vad del II handlar om.

Vi avslutar denna sektion i (d) med en titt på nollekvivalens- respektive ekvivalensproblemen i sammanhang med polynom.

(a) Envariabelfallet: Polynomdivision

När vi räknade med heltal modulo 4 fick vi en uppdelning av heltal i en kvot och en rest och genom att reducera bort kvoten fick vi kvar fyra restklasser som vi kallade för [0], [1], [2] och [3]. Om vi byter ut heltalsdivision mot polynomdivision och 4 mot ett polynom g i en variabel X kan vi få en motsvarande uppdelning av polynom i en kvot och en rest modulo g.

Exempel.

	Rest $p(X)$ mod $g(X)$	$g(X)$		
		$3X$	X^2-1	$2X^3-X^2+3X+1$
$p(X)$	$2X^3-7$	-7	$2X-7$	X^2-3X-8
	$4X^4+3X$	0	$3X+4$	$-5X^2-2X-1$
	$24X^5+3X^2-8$	-8	$24X-5$	$-42X^2+39X+7$

•

För dessa exempel är resten alltid en linjärkombination av de potenser av X som *inte* finns med i det polynom man dividerar med $g(X)$ och dessa potenser ser ut att utgöra en *bas* för restklasserna. Och att det verkligen är så visar följande korollarium:

Korollarium 2.12 *Om $p(X) \in K[X]$ är ett polynom i en variabel så är*

$$U = \{[X^i] \mid 0 \leqslant i < \ grad(p)\}$$

en bas för $K[X]/\langle p\rangle$.

Bevis: Se korollarium 5.3. •

Vi kan med dess hjälp direkt få ett uttryck för basen till en restklassring.

Exempel.

Restklassring	Bas för restklasserna	Mall för element i restklassringen
$K[X]/\langle X\rangle$	$\{1\}$	$a[1]+\langle X\rangle$
$K[X]/\langle X^2\rangle$	$\{1,X\}$	$a_1[1]+a_2[X]+\langle X^2\rangle$
$K[X]/\langle X^n\rangle$	$\{1,X,\dots X^{n-1}\}$	$a_1[1]+a_2[X]+\cdots+a_n[X^{n-1}]+\langle X^n\rangle$
$K[X]$	$\{1,X,\dots\}$	$a_1[1]+a_2[X]+\cdots$

•

Räkning i $K[X]/\langle g(X)\rangle$ fungerar precis analogt med räkning modulo heltal: Räkna med representanter och reducera resultatet till rätt restklass genom att beräkna resten med polynomdivision.

Exempel. Vi ska nu räkna modulo $g(X) = X^3 - 3X^2 + 1$. Om $p_1 = X^4 + 3X^2$ och $p_2 = X^6 + X^5 - 1$ har vi att $p_1p_2 = X^{10} + X^9 - X^4 + 3X^8 + 3X^7 - 3X^2$ och $p_1p_2 \bmod g = 9450X^2 - 1140X - 3283$. För att kontrollera att det går bra att räkna med olika representanter reducerar vi först p_1 och p_2 modulo g: $r_1 = p_1 \bmod g = 12X^2 - X - 3$ och $r_2 = p_2 \bmod g = -36 + 101X^2 - 12X$. Vi får att $r_1r_2 = -723X^2 + 1212X^4 - 245X^3 + 72X + 108$ och slutligen $r_1r_2 \bmod g = 9450X^2 - 1140X - 3283$ vilket naturligtvis är precis samma resultat som $p_1p_2 \bmod g$. Vi kan genom att räkna på detta vis ställa upp en multiplikationstabell för restklassringen där vi har en rad (och en kolumn) för varje element i den tillhörande basen:

$p\cdot q$ mod g		$[q]$		
		1	X	X^2
$[p]$	1	1	X	X^2
	X	X	X^2	$3X^2-1$
	X^2	X^2	$3X^2-1$	$9X^2-X-3$

•

(b) Modulo en mängd monom

Restklasserna i envariabelfallet hade en bas som bestod av alla potenser av X som var mindre än den största potensen av X som förekom i det polynom som generarade idealet. Nu har vi istället för ett polynom i en variabel flera monom i flera variabler. Kan det vara lika enkelt att hitta en bas för restklassringen också nu? Följande korollarium ger oss svaret att det är lika enkelt:

Korollarium 2.13 *Om* $P = \{p_1, \ldots, p_n\}$ *där* $p_i \in K[X_1, \ldots, X_n]$ *är monom så är*

$$U = \{[u] \mid u \in T(X_1, \ldots, X_n), \neg\exists p \in P \text{ så att } p \mid u\}$$

en bas för $K[X_1, \ldots, X_n]/\langle P\rangle$.

Bevis: Se korollarium 5.4. •

Basen består av alla termer som är "mindre" än generatorerna för idealet och med att termen $a \in T(X)$ är mindre än termen $b \in T(X)$ menas att om vi bildar kvoten a/b får vi inga negativa exponenter.

Exempel.

Restklassring	Bas för restklasserna
$K[X_1, X_2]/\langle X_1, X_2\rangle$	$\{1\}$
$K[X_1, X_2]/\langle X_1^2, X_2^2\rangle$	$\{1, X_1, X_2, X_1X_2\}$
$K[X_1, X_2]/\langle X_1^p, X_2^r\rangle$	$\{X_1^iX_2^j \mid 0 \leqslant i \leqslant p-1, 0 \leqslant j \leqslant n-1\}$
$K[X_1, \ldots, X_n]/\langle X_1^{p_1}, \ldots, X_n^{p_n}\rangle$	$\{X_1^{r_1} \cdots X_n^{r_n} \mid 0 \leq r_i < p_i\}$
$K[X]/\langle X_1^3, X_1X_2^3\rangle$	$\{1, X, X^2, X_1X_2^2, X_1X_2, X_2^2, X_2\}$

•

Räkning sker som tidigare med representanter men när vi ska reducera till en restklass får vi problem eftersom polynomdivision bara fungerar när vi har ett ideal genererat av ett polynom i en variabel. Vi behöver alltså en slags "generaliserad polynomdivision" som klarar av reduceringen när vi har flera generatorer i flera variabler. Denna generalisering kallas för *reduktion* och kommer att beskrivas i detalj i kapitel 3.

(c) Modulo en mängd polynom

Det generella fallet där vi har ett ideal genererat av flera polynom i flera variabler kommer vi än så länge ingenstans med. Vi kan som i förra fallet inte utföra den avslutande reduktionen, men inte nog med det. För att kunna skriva ner en bas för restklasserna kräver sats 5.2 att idealet ska vara på en viss form, det ska vara en *Gröbnerbas*. Vad en Gröbnerbas är och hur man gör om ett ideal till en Gröbnerbas ska vi se på i kapitel 4.

(d) Ekvivalens och kanonisk förenklare

Under samma rubrik i sektion 1.3 diskuterade vi nollekvivalens och ekvivalens allmännt för kvotgrupper ock kvotringar och eftersom en polynomring $K[X]$ modulo ett ideal I är en kvotring kan allt som sades där direkt föras över hit.

För att lösa nollekvivalensproblemet måste vi skapa en funktion S som om $a \sim 0$ ger att $S(a) \equiv 0$. Ekvation 1.3 talar om att det i $K[X]/I$ är samma sak som att säga att $a \in I$. En normal förenklare är alltså en funktion S som uppfyller att

$$a \equiv 0 \bmod I \Leftrightarrow a \in I \Leftrightarrow S(a) \equiv 0 \ , \tag{2.6}$$

d v s den normala formen av a är noll.

För att lösa ekvivalensproblemet måste vi skapa en funktion C som om $a \sim b$ ger att $C(a) = C(b)$. Ekvation 1.4 talar om att det i $K[X]/I$ är samma sak som att säga att a och b ligger i samma restklass modulo I eller att $a - b \in I$. En kanonisk förenklare är alltså en funktion C som uppfyller att

$$a \equiv b \bmod I \Leftrightarrow a - b \in I \Leftrightarrow C(a) \equiv C(b) \ , \tag{2.7}$$

d v s den kanoniska formen av a är samma som den kanoniska formen av b. Problemet att inte kunna reducera till en restklass som vi beskrev tidigare kan vi nu formulera som att vi inte har en kanonisk förenklare (och inte heller en normal dito) i kvotringen $K[X]/I$. Hade vi det skulle vi kunna reducera t ex produkten av två polynom till en restklass.

2.5 Sammanfattning

Vi har i detta kapitel stiftat närmare bekantskap med polynomringen i en och flera variabler och sett hur ideal kan spännas upp av generatorer. Kapitlet avslutades med en titt på basbyte för ideal och restklasser till polynomringen modulo ett ideal.

I och med detta kapitel lämnar vi det som hör till ”allmännbildning” inom algebra och går i nästa del in på för detta område mer specifika begrepp: Reduktion och Gröbnerbaser.

Noter

[1] [Fra89] sidan 276

[2] Sillnaden är, se [Lan93] sidan 104, ”more psychological than mathematical”.

Del II

Gröbnerbaserna

Kapitel 3

Reduktion

Ett enkelt sätt att se om ett polynom $p(X)$ i en variabel ligger i ett ideal $I = \langle g \rangle$ är att med polynomdivision dividera p med g och om resten är noll, d v s divisionen gick jämt upp, vet man att $p \in I$. I flervariabelfallet kan man göra på samma sätt fast med en generaliserad variant av polynomdivision som kallas för *reduktion*.

Termerna i ett polynom i *en* variabel ordnas efter sin grad och det är denna ordning som bestämmer i vilken ordning man räknar vid polynomdivision. Också i flervariabelfallet måste man ordna sina termer. Här finns det dock många olika sätt att ordna termerna och i sektion 3.1 ska vi titta närmare på de två vanligaste, den lexiografiska ordningen och den totala gradordningen.

Kapitlets tyngdpunkt ligger i sektion 3.2 som presenterar den ovan nämda reduktionen. Resonemangen leder fram till en algoritm som löser reduktionsproblemet: Att kunna uttrycka ett polynom som en linjärkombination av generatorer till ett ideal plus en (eventuell) rest. Eller matematiskt uttryckt: Finna en representation $\sum a_i p_i$ av ett polynom $g(X)$ där $\{p_i\}$ genererar idealet $I \subset P[X]$ och $a_i \in P[X]$.

Vi kommer längre fram att behöva en algoritm som givet en mängd polynom och ett ideal omvandlar denna mängd till en reducerad mängd. Eftersom algoritmen hänger intimt samman med algoritmen för att reducera ett polynom presenterar vi den i sektion 3.3.

Eftersom vi nu har fått en ”generaliserad” polynomdivision återknyter vi i sektion 3.4 till de problem vi fick tidigare (sektion 2.4) med reduktion till en unik klassrepresentant.

3.1 Term- och monomordningar

Att ordna saker och jämföra vilken som är störst, minst, tyngst, vackrast eller kallast är något man också sysslar med inom matematik. Vi har redan stiftat bekantskap med en slags ordning, den partiella ordningen. Men som namnet antyder är den ingen "riktigt" ordning. Om vi har en mängd element som är partiellt ordnade behöver det ju inte vara så att vi kan jämföra alla element med varandra, det går bara om de ligger i samma gren i ordningens Hassediagram. Att kräva en *ordning* (utan "partiell") innebär att kräva att alla element ska vara jämförbara med varandra, d v s alla element ska ligga i samma gren eller i en "rad" om man så vill. Vi ska i nästa sektion se på ett par olika sätt att på det här viset skapa en ordning för termer i variablerna $X_1, \ldots, X_n$.

Vi ska nu beskriva ett speciellt sätt att förse $T(X)$, mängden av alla termer i variablerna $X_1, \ldots, X_n$, med en ordning så vi kan jämföra dem med varandra och sortera dem inbördes.

Låt $T(X)$ vara alla termer i variablerna $X = X_1, \ldots, X_n$. För att en ordning ska vara godkänd i detta sammanhang ska den vara "tillåten" (eng: *admissible*):

Definition 3.1 (Termordning) *En* **tillåten termordning** *är en ordning* $\leqslant_T$ *på* $T(X)$ *som uppfyller:*

1. $1 \leqslant_T m$ *för alla* $m \in T(X)$.
2. $m_1 \leqslant_T m_2 \Rightarrow s \cdot m_1 \leqslant_T s \cdot m_2$ *för alla* $m_1, m_2, s \in T(X)$

Första villkoret talar om att $1 = X_1^0 \cdots X_n^0$ är det minsta elementet i $T(X)$. Andra villkoret säger att om vi vet att t ex $x < y$ så är också $p(X,Y) \cdot x < p(X,Y) \cdot y$.

Det finns många olika "tillåtna" sätt att ordna termerna på och vi ska i huvudsak använda oss av två: Lexiografisk ordning och total gradordning. Anledningen till att använda just dessa två är att de är av två principiellt olika typer som kommer att leda till resultat som ser ut på två principiellt olika sätt. Dessutom är det oftast just dessa två termordningar som finns implementerade i program för symbolisk algebra.

Definition 3.2 (Lexiografisk ordning) *Den (rent) lexiografiska ordningen*

definieras av

$$s = x_1^{i_1} \cdots x_n^{i_n} \leqslant_L x_1^{j_1} \cdots x_n^{j_n} = t \Longleftrightarrow$$

$$(i_1, \ldots, i_n) = (j_1, \ldots, j_n)$$

eller

$$\exists l \text{ så att } i_l < j_l \text{ och } i_k = j_k, 1 \leqslant k < l$$

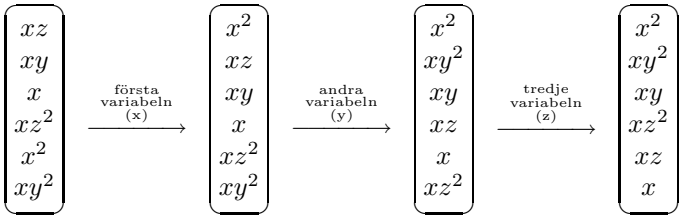
Att sortera termerna i ett polynom i n variabler *lexiografiskt* är en n-stegsprocess: (1) Sortera fallande efter första variabelns exponent, (2) sortera fallande efter andra variabelns exponent, ... , (n) sortera fallande efter n:te variablens exponent. Observera att sorteringen är "lokal" i den meningen att andra stegets sortering inte "förstör" första stegets sortering o s v. En liknande beskrivning för att sortera namn i (omvänd) bokstavsordning skulle lyda: (1) Sortera stigande efter första bokstaven, (2) sortera stigande efter andra bokstaven o s v.

Exempel. Trivialt så är $1 < x < x^2 < \cdots$

Låt $x = x_1$, $y = x_2$ och $z = x_3$. Då är $1 < z < y < x$. För att se detta skriv $z = x^0y^0z^1$, $y = x^0y^1z^0$ och $x = x^1y^0z^0$. Vidare gäller, se figur 3.1, att

$$x < xz < xz^2 < \cdots < xy < xy^2 < \cdots < x^2 < \cdots$$

•



Figur 3.1: Sortering steg för steg enligt den lexiografiska ordningen.

Definition 3.3 (Total gradordning) *Den (totala) gradordningen definieras av*

$$s = x_1^{i_1} \cdots x_n^{i_n} \leqslant_D x_1^{j_1} \cdots x_n^{j_n} = t \iff$$

$$(i_1, \ldots, i_n) = (j_1, \ldots, j_n)$$

eller

$$\deg(s) < \deg(t), \ \textit{eller}$$
$$\deg(s) = \deg(t) \ \textit{och} \ \exists l \ \textit{så att} \ i_l < j_l \ \textit{och} \ i_k = j_k, 1 \leqslant k < l$$

Att sortera termerna i ett polynom i n variabler enligt den totala gradordningen är en n-stegsprocess: (1) Sortera fallande efter totala gradtalet, (2) sortera stigande efter första variablens exponent, (3) sortera stigande efter andra variabelns exponent, ... , (n) sortera stigande efter näst sista variabelns exponent. OBS: Om två polynom med samma gradtal har samma exponent i variablerna $x_1, \ldots, x_{n-1}$ har de också samma exponent i x_n, därav bara n steg. Man kan också säga att termer av samma gradtal sorteras lexiografiskt. Vad som sades om lokal sortering för den lexiografiska sorteringen gäller även här.

Exempel. Trivialt så är $1 = x^0 < x < x^2 < \cdots$

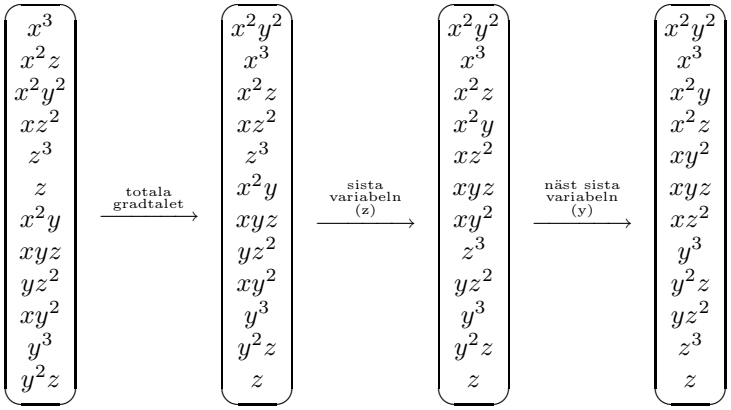
Låt $x = x_1$, $y = x_2$ och $z = x_3$. Då är $1 < z < y < x$. För att se detta skriv $z = x^0y^0z^1$, $y = x^0y^1z^0$ och $x = x^1y^0z^0$. Vidare gäller, se figur 3.2, att

$$\underbrace{\underbrace{x^3}_{y^0}}_{x^3} > > \underbrace{\underbrace{x^2y}_{y^1} \underbrace{x^2z}_{y^0}}_{x^2} > > \underbrace{\underbrace{xy^2}_{y^2} \underbrace{xyz}_{y^1} \underbrace{xz^2}_{y^0}}_{x^1} > > \underbrace{\underbrace{y^3}_{y^3} \underbrace{y^2z}_{y^2} \underbrace{yz^2}_{y^1} \underbrace{z^3}_{y^0}}_{x^0}$$

•

Med en termordning kan vi ordna en delmängd av $T(X)$ och speciellt kan vi ordna termerna i ett polynom. Av speciellt intresse är den term som är "störst" enligt en viss termordning $\leqslant_T$:

Definition 3.4 (Ledande term, ledande monom) *Med* **största** *eller* **ledande termen** *till $p \in K[X]$ med avseende på en termordning $\leqslant_T$ menas den term i polynomet p som är störst. Vi kallar denna term för* $\mathrm{hterm}_T(p)$



Figur 3.2: Sortering steg för steg enligt den totala gradordningen

eller $\mathrm{hterm}(p)$ *om termordningen är underförstådd. Vi kallar den* **ledande termens koefficient** *för* $\mathrm{hcoeff}(f)$ *och det* **ledande monomet** *för* $\mathrm{hmon}(f)$ *så att*

$$\mathrm{hmon}(f) = \mathrm{hcoeff}(f)\,\mathrm{hterm}(f)$$

Vi säger också att $\mathrm{hcoeff}(0) = 0$ *och* $\mathrm{hterm}(0) = 1$.

Exempel. $f(x,y) = 7xy^3 + 3x^2y$. För $\leqslant_D$ gäller

$$\mathrm{hmon}(f) = 7xy^3 \quad \mathrm{hcoeff}(f) = 7 \quad \mathrm{hterm}(f) = xy^3\ .$$

För $\leqslant_L$ gäller

$$\mathrm{hmon}(f) = 3x^2y \quad \mathrm{hcoeff}(f) = 3 \quad \mathrm{hterm}(f) = x^2y\ .$$

•

Vi har här definierat en ordning av monom utifrån en ordning av motsvarande termer. Vi kallar denna ordning den av $\leqslant$ *inducerade* ordningen $\preccurlyeq$ och skriver

$$f \preccurlyeq g \Longleftrightarrow aT(f) \preccurlyeq bT(g) \Longleftrightarrow T(f) \leqslant T(g)$$

där f och g är monom, $T(f)$ och $T(g)$ motsvarande termer (d v s monom med koefficienten 1) samt a och b är koefficienter till termerna $T(f)$ och $T(g)$. Eftersom koefficienterna

inte påverkar monomens inbördes ordning är alla monom med samma term ekvivalenta med avseende på $\preccurlyeq$ och (den inducerade) monomordningen kan därför inte vara en partiell ordning. Men två element kan relatera varandra utan att vara ekvivalenta och $\preccurlyeq$ är därför ej heller en ekvivalensrelation utan en blandning av dessa två, en *kvasiordning*. För fortsatt läsning, se [BW93] sidan 192.

3.2 Reduktion

Målet med division av t ex två polynom p/g är att beräkna en kvot a och en rest q som uppfyller

$$p = ag + q$$

I de fall då resten $q = 0$ säger vi att divisionen gått jämt upp och skriver $p/g = a$. Vi kan också säga att

$$q = 0 \Leftrightarrow p \in \langle g \rangle \tag{3.1}$$

Vi kan alltså använda oss av polynomdivision för att avgöra om ett polynom ligger i ett ideal som genereras av ett polynom.

Men hur gör vi om vi vill se om ett polynom i flera variabler ligger i ett ideal som genereras av *flera* generatorer? Vad vi vill är att med en slags "generaliserad division" skriva ett polynom p som en linjärkombination av ett antal generatorer plus en eventuell rest:

$$p = a_1 g_1 + \cdots + a_k g_k + q$$

På samma sätt som ovan gäller att

$$q = 0 \Leftrightarrow p \in \langle g_1, \ldots, g_k \rangle$$

Vi börjar med ett exempel. Säg att $g = 3xy + 1$ och $p = 9x^3y - 3xy^2 + 6$. Om vi nu låter $q = p - 3x^2 \cdot g = 9x^3y - 3xy^2 + 6 - 9x^3y - 3x^2 = -3xy^2 + 6 - 3x^2$ så har första termen i p "försvunnit" (och vi har fått en ny term $-3x^2$ som är mindre än $9x^3y$) och vi säger att vi har **reducerat** p med g, eller att

$$p \xrightarrow[g]{} q \ .$$

Ett generellt **reduktionssteg** $p \xrightarrow[g]{} q$ betyder att $q = p - ag \Leftrightarrow p = ag + q$ för något $a \in M(X)$, d v s a är ett monom i variablerna $X = X_1, \ldots, X_n$ (se definition 2.5) så att någon av p:s termer elimineras i exemplet ovan.

Definition 3.5 (Reduktion modulo ett polynom) *Om det för $p, q, g \in K[X]$ existerar ett $a \in M(X)$ med $\mathrm{hmon}(ag) \in M(p)$ ($M(p)$ är alla monom i polynomet p) så att $p = q + ag$ säger vi att g* **reducerar** *p (genom att eliminera $t = \mathrm{hmon}(ag)$):*

$$p \underset{g}{\longrightarrow} q \qquad\qquad (p \underset{g}{\longrightarrow} q\ [t])$$

Om $t = \mathrm{hmon}(ag) = \mathrm{hmon}(p)$ säger vi att g **topreducerar** *p.*

Anmärkning. Observera att reduktionen *alltid* måste *eliminera* en term i p annars "räknas" det inte som en reduktion.

Hittills har vi bara reducerat i *ett* steg med *en* på förhand bestämd reducerare. Vi ska nu utvidga reduktionen till att utföras i *flera* steg och med *flera* olika reducerare att välja mellan. Först inför vi en speciell "lika med"-symbol som knyter an till vad som sagts tidigare om olika termordningar. Med symbolen $\overset{\text{sort}}{=}$ menas en sortering av termer enligt en termordning (vilken termordning som gäller framgår av sammanhanget).

Om vi sorterar efter den totala gradordningen får vi till exempel

$$\begin{aligned} xy^3 + x^2y - y^3 + x^4 + x^3y^2 + x^2y^2 &\overset{\text{sort}}{=} \\ x^3y^2 + x^4 + x^2y^2 + xy^3 + x^2y - y^3 & \end{aligned}$$

Rent allmännt kan en reduktionssekvens, p modulo en mängd polynom G ($G = \{g_1, \ldots, g_k\}$), skrivas som

$$\begin{aligned} p_1 &= p - a_1 g_{i_1} \\ &\vdots \\ p_n &= p_{n-1} - a_n g_{i_n} \\ q &= p_n \end{aligned}$$

Detta ger en uppdelning av ett polynom i ett antal "baspolynom" $g_1, \ldots, g_k$ enligt

$$p = a_1 g_{i_1} + \cdots + a_n g_{i_n} + q \tag{3.2}$$

och om $q = 0$ är uppdelningen fullständig, d v s vi kan skriva p som en linjärkombination av polynomen i basen G.

Reduktionsprocessen kan uttryckas på ett smidigare sätt som

$$p \xrightarrow[g_{i_1}]{} p_1 \longrightarrow \cdots \xrightarrow[g_{i_n}]{} p_n = q \tag{3.3}$$

analogt med ekvation (3.2).

Vi har nu gett en mening till följande definition:

Definition 3.6 (Reduktion) *Med* $p \xrightarrow[G]{} q$ *menar vi att det finns ett* $g_i \in G$ *så att* $p \xrightarrow[g_i]{} q$. *Med* $p \xrightarrow[G]{*} q$ *menar vi en reduktionssekvens enligt ekvation 3.3.*

Om $p \xrightarrow[G]{*} q$ betyder det att vi kan skriva

$$p = a_1 g_{i_1} + \cdots + a_n g_{i_n} + q = \sum_{r=1}^{n} a_r g_{i_r} + q$$

Exempel. Vi vill reducera polynomet

$$p = -3xy^5 - 3x^2y^3 + x^3y^2 + xy^3 + 2x^3y + 2xy^2$$

så långt det går med reducerarna

$$g_1 = x^2y + y^2 \quad \text{och} \quad g_2 = xy^2 + x^2$$

och den totala gradordningen. Reduktionen kan t ex göras så här:

$$\begin{aligned}
p =& -3xy^5 - 3x^2y^3 + x^3y^2 + xy^3 + 2x^3y + 2xy^2 \\
p_1 =& p + 3y^3 g_2 = -3xy^5 - 3x^2y^3 + x^3y^2 + xy^3 + 2x^3y + 2xy^2 + 3y^3(xy^2 + x^2) \\
\overset{\text{sort}}{=}& x^3y^2 + xy^3 + 2x^3y + 2xy^2 \\
p_2 =& p_1 - xyg_1 = x^3y^2 + xy^3 + 2x^3y + 2xy^2 - xy(x^2y + y^2) \\
\overset{\text{sort}}{=}& 2x^3y + 2xy^2 \\
p_3 =& p_2 - 2xg_1 = 2x^3y + 2xy^2 - 2x(x^2y + y^2) \\
\overset{\text{sort}}{=}& 0
\end{aligned}$$

Vi har nu fått en uppdelning av p enligt ekvation (3.2) som

$$\begin{aligned}
p =& a_1 g_{i_1} + a_2 g_{i_2} + a_3 g_{i_3} = a_1 g_2 + a_2 g_1 + a_3 g_1 \\
=& -3y^3 g_2 + (xy + 2x) g_1
\end{aligned}$$

•

Anmärkning. Eftersom man i ett steg av reduktionen ibland kan använda mer än en av reducerarna ser man att den beskrivna reduktionsprocessen inte leder till ett entydigt resultat. I kapitel 4 kommer vi att se att om mängden reducerare är en Gröbnerbas spelar det ingen roll i vilken ordning vi reducerar, d v s reduktionen leder till ett entydigt resultat.

Hittills har vi bara sett på exempel där reduktionen "gått jämt upp", men vad händer om man kommer till ett läge då ingen av de tillgängliga reducerarna kan användas till nästa steg? Det naturliga är att flytta över den term som ej kan reduceras till restpolynomet q och sedan försöka reducera på nytt. Om det fortfarande inte går flyttar man ännu en term till restpolynomet o s v.

Exempel. Exempel på reduktion som ej går "jämt upp". De monom som ej kan reduceras bort och som således kommer att utgöra restpolynomet q markeras med $\{\dots\}$ runt sig. I varje reduktionssteg försöker vi alltså reducera bort den enligt sorteringsordningen största termen som *inte* står inom $\{\dots\}$. Termerna sorteras lexiografiskt med $x > y$.

$$g_1 = x^2y + 5x^2 + y^2 \qquad g_2 = 7xy^2 - 2y^3 + 1$$

$$p = 3x^3y + 2x^2y^2 - 3xy + 5x$$

$$
\begin{aligned}
p_1 = p - \boxed{3x \cdot g_1} &= 3x^3y + 2x^2y^2 - 3xy + 5x - 3x^3y - 15x^3 - 3xy^2 \\
&\overset{\text{sort}}{=} \{-15x^3\} + 2x^2y^2 - 3xy^2 - 3xy + 5x \\
p_2 = p_1 - \boxed{2y \cdot g_1} &= \{-15x^3\} + 2x^2y^2 - 3xy^2 - 3xy + 5x - 2x^2y^2 - 10x^2y \\
&\quad - 2y^3 \\
&\overset{\text{sort}}{=} \{-15x^3\} - 10x^2y - 3xy^2 - 3xy + 5x - 2y^3 \\
p_3 = p_2 + \boxed{10 \cdot g_1} &= \{-15x^3\} - 10x^2y - 3xy^2 - 3xy + 5x - 2y^3 + 10x^2y \\
&\quad + 50x^2 + 10y^2 \\
&\overset{\text{sort}}{=} \{-15x^3 + 50x^2\} - 3xy^2 - 3xy + 5x - 2y^3 + 10y^2 \\
p_4 = p_3 + \boxed{\tfrac{3}{7}g_2} &= \{-15x^3 + 50x^2\} - 3xy^2 - 3xy + 5x - 2y^3 + 10y^2 + 3xy^2 \\
&\quad - \frac{6}{7}y^3 + \frac{3}{7} \\
&\overset{\text{sort}}{=} \{-15x^3 + 50x^2 - 3xy + 5x - \frac{20}{7}y^3 + 10y^2 + \frac{3}{7}\}
\end{aligned}
$$

Genom denna reduktion har vi fått ett uttryck för p som lyder

$$
\begin{aligned}
p = &\boxed{(3x + 2y - 10)g_1 + (-\tfrac{3}{7})g_2} \\
&\{-15x^3 + 50x^2 - 3xy + 5x - \frac{20}{7}y^3 + 10y^2 + \frac{3}{7}\}
\end{aligned}
$$

Resten vid reduktion modulo p_1 och p_2 är ej noll och alltså kan vi inte säga att p tillhör idealet $\langle g_1, g_2 \rangle$ (men därmed inte sagt att $p \notin \langle g_1, g_2 \rangle$!). •

En algoritm som löser reduktionsproblemet finns i tabell 3.1.

Huvudloopen på raderna 3–9 repeteras så länge det finns termer kvar i polynomet r att reducera bort.

Vid ett steg i reduktionsprocessen kan det tänkas att det finns flera olika reducerare att reducera med. Om vi till exempel vill reducera polynomet $p_n = x^2y + 1$ med reducerarna $g_1 = x + 1$ och $g_2 = y + 1$ kan vi utföra en reduktion både genom $p_{n+1} = p_n - xyg_1$ och $p_{n+1} = p_n - y^2g_2$. I andra fall när reduktionen som beskrivits ovan ej går jämt upp kan vi inte reducera alls. De polynom som

vi i ett visst steg kan använda för att reducera kan vi skriva som

$$R_{p,G} = \{g \in G - \{0\} \text{ så att } \text{hterm}(g) | \text{hterm}(p)\}$$

där p är polynomet vi vill reducera och G är våra reducerare. Raderna 4–7 repeteras så länge vi kan reducera vårt polynom vidare.

Raderna 5–6 utför den egentliga reduktionen. Funktionen selectpoly($R_{p,G}$) syftar till att i de fall vi kan använda flera olika reducerare välja ut en av dem, t ex den första.

Rad 8 till sist kommer att utföras när vi inte kan reducera vidare, d v s när $R_{p,G}$ i rad fyra är tom. Vad som händer är helt enkelt att man "flyttar" den ledande termen från det polynom man vill reducera till det s k restpolynomet q.

Rad 10 returnerar de termer, i form av ett polynom, som ej kunde reduceras bort. Om $q = 0$ betyder det att p ligger i idealet som genereras av $Q = \{q_1, \ldots, q_k\}$, d v s att p kan skrivas som en linjärkombination $\sum a_i q_i$. Om $q \neq 0$ finns möjligheten att p *ej* ligger i $\langle Q \rangle$, men för att avgöra om $p \in \langle Q \rangle$ eller inte måste Q vara en s k *Gröbnerbas*.

Tabell 3.1: Reduce

```
1.  procedure Reduce(p, Q)
2.      r ← p ; q ← 0
3.      while r ≠ 0 do begin
4.        while R_{r,Q} ≠ {} do begin
5.          f ← selectpoly(R_{r,Q})
6.          r ← r − (hmon(r)/hmon(f)) f
7.        end
8.        q ← q + hmon(r); r ← r − hmon(r)
9.      end
10.     return(q)
11. end
```

3.3 Reduktion av en mängd polynom

Vi har nu sett hur man med en mängd reducerare reducerar ett polynom. En variant på detta är att reducera reducerarna med sig själva för att få en bas på

vad som kallas monisk och reducerad form.

Definition 3.7 (Reducerad monisk mängd) *En mängd* $G \subset K[X]$ *är* **reducerad** *om* $\forall g \in G$ *gäller att* $g = \text{Reduce}(g, G - \{g\})$ *och* **monisk** *om* $\forall g \in G$ *gäller att* $\text{hcoeff}(g) = 1$.

En algoritm som beräknar en reducerad monisk mängd finns i tabell 3.2.

Raderna 3–11 skapar en ny bas P som består av de element ur R som inte går ett reducera bort med elementen (hittils) i P.

Om reduktionen på rad 5 misslyckas betyder det att h är "mindre" än alla $p \in P$ (d v s att det finns inget $p \in P$ som kan reducera h) och alltså borde h ligga i P i stället för de element som är "större" än h (d v s de som kan reduceras av h). Förflyttningen av dessa "större" element från P till R sker på raderna 7–9.

När vi kommit till rad 12 vet vi att inget element i P är "överflödigt" längre, d v s vi kan inte *topreducera* längre, men de kan fortfarande vara *reducibla* modulo varandra. Vi reducerar därför på raderna 13–16 alla element i P modulo de andra och de är därefter alla i normalform (modulo P minus sig självt). Divisionen med $\text{hcoeff}(h)$ på rad 15 syftar till att göra elementen i P moniska.

Exempel. Om

$$E = \{x^2y, xy^2, y^2z, yz^2, x + y, y\}$$

har vi på rad 12 att

$$P = \{x + y, y\}$$

och får till slut att

$$\tilde{E} = \{x, y\}$$

•

3.4 Ekvivalens och kanonisk förenklare

Vi såg i ekvation 3.1 att med vanlig polynomdivision har vi i envariabelfallet att

$$p \in \langle g \rangle \Leftrightarrow p \equiv 0 \bmod g$$

Att det inte alltid är så i flervariabelfallet visar följande exempel:

Tabell 3.2: ReduceSet

```
1.  procedure ReduceSet(E)
2.      R ← E ; P ← ∅
3.      while R ≠ ∅ do begin
4.          h ← selectpoly(R); R ← R − {h}
5.          h ← Reduce(h, P)
6.          if h ≠ 0 then begin
7.              Q ← {q ∈ P så att hterm(h) | hterm(q)}
8.              R ← R ∪ Q
9.              P ← (P − Q) ∪ {h}
10.         end
11.     end
12.     Ẽ ← ∅; S ← P
13.     foreach h ∈ P do begin
14.         h ← Reduce(h, S − {h})
15.         Ẽ ← Ẽ ∪ {h/ hcoeff(h)}
16.     end
17.     return(q)
18. end
```

Exempel. Exempel där $p \in \langle P \rangle \nLeftrightarrow p \xrightarrow[P]{*} 0$.

Låt $p_1 = xy^2z - xyz$, $p_2 = x^2y^2 - z^2$ och $P = \{p_1, p_2\}$. Som polynom att reducera tar vi $q = x^2y^2z - z^3$ och $r = -x^2y^2z + x^2yz$. Vi får nu

$$q - z \cdot p_2 = 0 \Rightarrow q \in \langle P \rangle$$
$$r + x \cdot p_1 = 0 \Rightarrow r \in \langle P \rangle$$

och därmed ska att $q + r \in \langle P \rangle$. Men $q + r = x^2yz - z^3$ som är irreducibelt (går ej att reducera) modulo P.

Med andra ord så har vi hittat ett $p \in \langle P \rangle$ som *ej* går att reducera till noll. •

Vad skulle det innebära om vi krävde samma sak i flervariabelfallet? Om vi byter ut polynomdivision mot reduktion och villkoret att resten ska vara noll mot att vi kan reducera till noll får vi

$$p \in \langle G \rangle \Longleftrightarrow p \xrightarrow[G]{*} 0 \tag{3.4}$$

Vi ser direkt att nollekvivalensproblemet är löst eftersom ett polynom p reducerar till noll modulo G om och endast om polynomet ligger i idealet som genereras av G, så algoritmen Reduce kan tjänstgöra som den normala förenklare S vi har nämnt tidigare (sektion 2.4(d)). Men ekvation (3.4) innebär mer än så vilket följande sats visar:

Sats 3.8 *Om ekvation (3.4) gäller har vi att*

$$\text{Reduce}(p, G) = \text{Reduce}(q, G) \Longleftrightarrow p - q \in \langle G \rangle$$

Bevis: Se sats 4.21. •

Och vad som menas med att $p - q \in \langle G \rangle$ såg vi i kapitel 2.4: p och q ligger i samma restklass. Vi har alltså inte bara en normal förenklare utan också en *kanonisk* förenklare till restklassringen och algoritmen Reduce kan även tjänstgöra som den kanoniska förenklare C vi har nämnt tidigare (sektion 2.4(d)). Ett ideal $\langle G \rangle$ som uppfyller detta villkor[1] är just en Gröbnerbas och i nästa kapitel ska vi rota en hel del i Gröbnerbaser och deras egenskaper.

3.5 Sammanfattning

Vi började kapitlet med att se på olika sätt att sortera termer i ett polynom i flera variabler. Efter att ha ordnat termerna definierade vi reduktion och konstruerade

algoritmen Reduce som uppför sig som en ”generaliserad” polynomdivision. Vi avslutade kapitlet med att se vad som behövs för att lösa nollekvivalens- och ekvivalensproblemen generellt för polynom i flera variabler.

Med grundläggande algebra, förståelse av polynom och ideal samt reduktion är vi nu mogna för att ge oss i kast med kompendiets primära mål, Gröbnerbaser. Där ska vi se hur vi praktiskt kan uppgradera reduktionen till en normal/kanonisk förenklare.

Noter

[1] [GCL92] sidan 439, ”It turns out that this (polynom som tillhör idealet reduceras inte alltid till noll) is not, strictly speaking, due to a deficiency of algorithm ... (här: algoritmen Reduce), but rather the structure of the ideal basis ... ”

Kapitel 4

Gröbnerbaser

Som vi har sett tidigare kan man med reduktionsalgoritmen ”dela upp” ett polynom p i en linjärkombination av ett ideals generatorer. Om den här uppdelningen lyckas, d v s vi kan skriva p som $\sum a_i g_i$ där g_i är generatorer till ett ideal, ser vi att polynomet ligger i idealet, detta följer av att ett ideal definieras just som mängden av alla linjärkombinationer av dess generatorer och speciellt då just vår linjärkombination. Om reduktionsalgoritmen lyckas reducera polynomet till 0 har vi hittat en dylik linjärkombination och vi vet då att $p \in \langle \{g_i\} \rangle$.

I sektion 3.4 såg vi hur antagandet att

$$p \in \langle G \rangle \iff p \xrightarrow[G]{*} 0$$

genom reduktionen gav oss den så efterlängtade kanoniska förenklaren. I det här kapitlet ska vi se hur vi kan åstadkomma detta.

Vi börjar i sektion 4.1 med en definition av Gröbnerbaser. Det finns flera olika vägar att gå för att bevisa existensen av en Gröbnerbas, d v s att det till varje ideal i polynomringen finns en tillhörande Gröbnerbas. Metoden som används här bygger på en speciell sorts partiell ordning, Dickson-partiell ordning.

Existensen av en Gröbnerbas är i och för sig värdefull, men det är först i och med möjligheten att alltid kunna beräkna den som konceptet Gröbnerbaser visar sin verkliga styrka. Hur detta går till ska vi se i sektion 4.2.

Den i sektion 4.2 presenterade algoritmen lider av två stora problem: Till ett givet ideal I finns det många olika Gröbnerbaser, den bas vi beräknar är med andra ord ej unik, och själva beräkningen plågas dessutom av hög tidskomplexitet.

I sektion 4.3 modifierar vi den grundläggande algoritmen så att vi dels får en unik bas, dels sänker komplexiteten avsevärt.

Vi avslutar med en diskussion om algoritmens komplexitet i sektion 4.4.

4.1 Definition och existens

I sektion 3.4 visade vi att för att få den så viktiga kanoniska förenklaren behöver vi kunna säga att

$$p \in \langle G \rangle \iff p \xrightarrow[G]{*} 0$$

och att en idealbas har denna egenskap är just detsamma som att säga att den är en Gröbnerbas:

Definition 4.1 (Gröbnerbas) *En Gröbnerbas till ett ideal I är en bas G sådan att $I = \langle G \rangle$ och som uppfyller*

$$p \in \langle G \rangle \Leftrightarrow p \xrightarrow[G]{*} 0$$

En Gröbnerbas uppfyller alltså våra önskemål att med reduktionsalgoritmen alltid kunna avgöra om ett polynom ligger i ett givet ideal eller inte. Frågor som återstår att besvaras är (1) om det till ett givet ideal *finns* en (entydig) Gröbnerbas, (2) kan vi *beräkna* den och i så fall (3) *hur* beräknar vi den? Första frågan ska vi titta närmare på härnäst och fråga två och tre återkommer vi till i nästa sektion.

Det korta svaret på frågan ”kan vi givet ett ideal genom ett basbyte omvandla dess bas till en Gröbnerbas” är *ja*. Att bevisa existensen ska vi göra i två steg: Först tar vi fram ett förslag till en bas och sedan visar vi att det förslaget verkligen är en Gröbnerbas. För att kunna ta fram förslaget till en bas behöver vi sats 4.5 som säger att $T(X)$ är en s k *Dickson-partiell ordning*. För att bevisa att förslaget till bas verkligen *är* en Gröbnerbas kan man tänka sig att gå direkt på definition 4.1, men vi väljer att istället i sats 4.6(ii) visa på en alternativ karaktärisering av Gröbnerbaser (som också skulle kunna tjäna som definition). Med ett förslag till en bas (ekvation 4.2) och villkoret i sats 4.6(ii) kan vi relativt enkelt bevisa i sats 4.7 att det alltid finns en Gröbnerbas.

Det första steget på vägen är att definiera en speciell partiell ordning kallad *Dickson-partiell ordning*. Välbekanta exempel på partiella ordningar är de naturliga och de reella talen tillsammans med den vanliga relationen $\leqslant$. Men de

naturliga talen och de reella talen skiljer sig åt i minst ett avseende: Om vi tar en delmängd av de naturliga talen kan vi alltid hitta ett minsta element som kan tjäna som en "bas" för de övriga men så är inte alltid fallet med de reella talen. T ex så har mängden $\{x \in \mathbb{R} | x > 0\}$ inget minsta element, hur nära noll vi än väljer ett tal så finns det alltid ett som är närmare noll (om vi väljer a så ligger t ex $a/2$ närmare noll). Om vi kräver att vi alltid för varje "gren" i en partiell ordnings Hasse-diagram ska kunna hitta ett minsta element som vi kan använda som bas och att det bara finns ett ändligt antal grenar kräver vi att den partiella ordningen ska vara en Dickson-partiell ordning:

Definition 4.2 (Dickson-partiell ordning) *Låt $\leqslant$ vara en partiell ordning på M och låt $N \subseteq M$. Då kallas en delmängd B av N en* **Dicksonbas** *för N med avseende på $\leqslant$ om det för varje $a \in N$ finns något $b \in B$ med $b \leqslant a$. Vi säger att $\leqslant$ har* **Dicksons egenskap** *eller är en* **Dickson-partiell ordning** *om varje delmängd $N \subseteq M$ har en ändlig bas $B \subseteq N$ med avseende på $\leqslant$.*

Denna Dicksonbas är det som så småningom kommer att mynna ut i en Gröbnerbas.

Eftersom vi ska jobba uteslutande med polynom vill vi veta om polynomens byggstenar, termerna, har denna Dicksons egenskap. För att i sats 4.5 visa att det är så behöver vi Dicksons lemma (lemma 4.4) och för att bevisa Dicksons lemma behöver vi nästa lemma. Observera att även om ordningsrelationen på M *skrivs* på samma sätt som ordningsrelationen på $\mathbb{N}$ behöver de ej vara lika!

Lemma 4.3 *Låt $(M, \leqslant)$ vara en Dickson-partiellt ordnad mängd och $(\mathbb{N}, \leqslant)$ de naturliga talen med deras naturliga ordning. Då är den direkta produkten $(M \times \mathbb{N}, \leqslant')$ där*

$$(a, b) \leqslant' (c, d) \iff a \leqslant c \text{ och } b \leqslant d$$

en Dickson-partiellt ordnad mängd.

Bevis: Vi ska visa att givet en icke-tom delmängd $S \subseteq (M \times \mathbb{N}, \leqslant')$ finns det en ändlig bas B. Vi gör detta genom att (1) konstruera mängden B och (2) visa att den nyss konstruerade mängden verkligen är en bas.

(1). Givet en delmängd S enligt ovan, låt för varje $n \in \mathbb{N}$,

$$M_n = \{a \in M \mid (a, n) \in S\} \ .$$

Låt B_n vara en ändlig bas till M_n och C en ändlig bas till $\bigcup_{n\in\mathbb{N}} B_n$[†]. Till slut låter vi $r \in \mathbb{N}$ vara sådant att $C \subseteq \bigcup_{i=1}^{r} B_i$. Låt nu

$$B = \{(a, i) \in M \times \mathbb{N} \mid 1 \leqslant i \leqslant r, a \in B_i\} .$$

(2). Eftersom alla B_i är ändliga är också B ändlig. För att B ska vara en bas krävs dessutom att givet ett $(a, n) \in S$ ska det finnas ett $(b, i) \in B$ som uppfyller att $(b, i) \leqslant' (a, n)$. Eftersom $a \in M_n$ kan vi hitta ett $b \in B_n$ så att $b \leqslant a$ och alltså $(b, n) \leqslant' (a, n)$. Om $n \leqslant r$ ligger (b, n) i B och vi är klara. Om inte så finns det (eftersom C är en bas till $\bigcup B_n$) ett $c \in C$ sådant att $c \leqslant b \leqslant a$ och $c \in B_i$ för något $i \leqslant r < n$. Och eftersom $(c, i) \in B$ och $(c, i) \leqslant' (a, n)$ får vi att B mycket riktigt är en bas. •

Eftersom vi nu vet att om vi har en Dickson-partiellt ordnad mängd och "lägger till" en komponent (som är de naturliga talen) fortfarande har en Dickson-partiellt ordnad mängd kan vi definiera en Dickson-partiell ordning för n-tupler av naturliga tal. Definiera $\leqslant'$ som

$$(a_1, \ldots, a_n) \leqslant' (b_1, \ldots, a_n) \iff a_i \leqslant b_i \text{ för alla } 1 \leqslant i \leqslant n. \tag{4.1}$$

Om vi som $\leqslant$ tar dess vanliga betydelse för heltal får vi t ex att $(1, 1) \leqslant' (1, 2) \leqslant' (2, 2)$ men $(2, 2) \not\leqslant' (1, 2000)$. Som en naturlig följd får vi Dicksons lemma:

Lemma 4.4 *Låt $(\mathbb{N}^n, \leqslant')$ vara den direkta produkten av n kopior av de naturliga talen med deras naturliga ordning $(\mathbb{N}, \leqslant)$. Då är $(\mathbb{N}^n, \leqslant')$ en Dickson-partiellt ordnad mängd. Varje delmängd S av $\mathbb{N}^n$ har en ändlig delmängd B sådan att för varje $(m_a, \ldots, m_n) \in S$ finns det ett $(k_1, \ldots, k_n) \in B$ med $k_i \leqslant m_i$ för alla $1 \leqslant i \leqslant n$.*

Exempel. I följande exempel är relationen $\leqslant'$ enligt ekvation (4.1) underförstådd.

$$\{q \in \mathbb{Q} \mid 0 < q \leqslant 1\}$$

är *ej* en DPO (Dickson-partiell ordning) eftersom den inte har någon bas (inget minsta element enligt $\leqslant'$). Däremot är

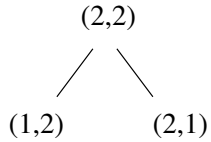
$$\{q \in \mathbb{Q} \mid 0 \leqslant q \leqslant 1\}$$

[†]Eftersom M är en Dickson-partiellt ordnad mängd har M_n och alla delmängder av M_n en ändlig bas.

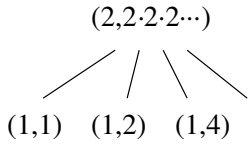
en DPO med $\{0\}$ som bas. Likaså är

$$\{z \in \mathbb{Z} \mid z \geqslant 0\}$$

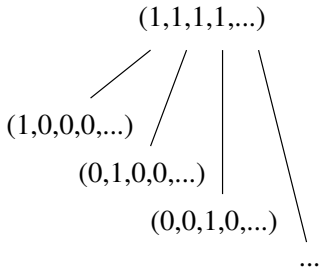
en DPO med $\{0\}$ som bas. Vidare ser vi att



är en DPO med en bas bestående av elementen $\{(1,2),(2,1)\}$. Däremot är



ej en DPO eftersom den har en oändlig bas $\{(1,1),(1,2),(1,4),(1,8),\dots\}$ och på samma sätt är inte heller



en DPO ty dess bas är $\{(1,0,0,0,\dots),(0,1,0,0,\dots),(0,0,1,0,\dots),\dots\}$. •

Men vad har då det här med polynom att göra? Väldigt mycket faktiskt! Se på funktionen $\eta : \mathbb{N}^n \to T$ där $T = T(X_1,\dots,X_n)$ är mängden av alla termer:

$$\eta : (e_1,\dots,e_n) \mapsto X_1^{e_1}\cdots X_n^{e_n}$$

Funktionen η är helt enkelt ett sätt att föra över allt vi tidigare har sagt om n-tupler av naturliga tal till termer i n variabler. Men vi kan inte använda oss

Tabell 4.1: Samband mellan Dickson-partiell ordning
för naturliga tal och termer.

Naturliga tal	Termer
$k = (k_1, \ldots, k_n)$	$s = \eta(k) = X_1^{k_1} \cdots X_n^{k_n}$
$m = (m_1, \ldots, m_n)$	$t = \eta(m) = X_1^{m_1} \cdots X_n^{m_n}$
$\leqslant'$	$\mid$
$(k) \leqslant' (m)$	$s \mid t$
$(1,1) \leqslant' (1,2)$	$X_1X_2 \mid X_1X_2^2$

av den tidigare använda relationen $\leqslant'$ utan måste definiera dess motsvarighet för termer. Låt T vara mängden av alla termer i n variabler. Definiera **delbarhetsrelationen** $\mid$ på T genom $s \mid t$, $s, t \in T$, om och endast om det finns ett $s' \in T$ så att $s \cdot s' = t$. Vi får nu t ex att $X_1X_2 \mid X_1X_2^2 \mid X_1^2X_2^2$ men $X_1^2X_2^2 \nmid X_1X_2^{2000}$. Vi kan nu enkelt formulera om lemma 4.4 till att gälla för termer i stället för n-tupler av naturliga tal:

Sats 4.5 *Delbarhetsrelationen $\mid$ på T definierar en Dickson-partiell ordning på T. Varje icke-tom delmängd S av T innehåller en ändlig delmängd B så att det för varje $s \in S$ finns ett $t \in B$ med $t \mid s$.*

I tabell 4.1 finns en jämförelse mellan Dickson-partiell ordning för naturliga tal och för termer.

Vi ska nu ta fram ett förslag till en bas som vi sedan i sats 4.6 och sats 4.7 visar verkligen *är* en Gröbnerbas.

Låt I vara ett ideal till $K[X]$ och $\mathrm{hterm}(I) = \{\mathrm{hterm}(i) | i \in I\}$. Då följer att varje element i I motsvaras av ett element i $\mathrm{hterm}(I)$. $\mathrm{hterm}(I)$ består då av en mängd termer, d v s är en delmängd av $T(X)$, och har därför en ändlig bas S i enlighet med sats 4.5. Eftersom varje element i $\mathrm{hterm}(I)$ motsvaras av ett element (ett polynom) i I gäller speciellt för alla $s \in S \subset \mathrm{hterm}(I)$ att det finns ett motsvarande polynom $f_s \in I$ så att $\mathrm{hterm}(f_s) = s$. Låt G vara mängden av alla sådana f_s:

$$G = \{f \in I \mid \exists s \in S \text{ med } \mathrm{hterm}(f) = s\} \tag{4.2}$$

För att bevisa att den nyss konstruerade mängden G är en Gröbnerbas behöver vi en sats:

Sats 4.6 *Låt I vara ett ideal till $K[X]$ och G en ändlig delmängd av I så att $0 \notin G$. Då är följande påståenden ekvivalenta med att G är en Gröbnerbas till I:*

(i) $f \xrightarrow[G]{*} 0$ *för alla* $f \in \langle G \rangle$.

(ii) För varje $0 \neq s \in \mathrm{hterm}(I)$ *finns det ett* $t \in \mathrm{hterm}(G)$ *så att* $t|s$.

Bevis: (i)⇒(ii): Ta ett $s \in T(X)$ och bilda

$$S = \{p \in I \mid \mathrm{hterm}(p) = s \text{ och } \neg\exists t \in \mathrm{hterm}(G), t \mid s\} \ ,$$

d v s alla $p \in \langle G \rangle$ som *ej* uppfyller $p \xrightarrow[G]{} h[s]$ för något $h \in \langle G \rangle$. Låt f vara det minsta elementet i S. Men $f \xrightarrow[G]{} h$ och då är $h < f$ samtidigt som $\mathrm{hterm}(h) = \mathrm{hterm}(f)$, h ligger i S och vårat antagande om att f var minimalt leder till en motsägelse.

(ii)⇒(i): Om $0 \neq f \in I$ finns det ett $g \in G$ så att $\mathrm{hterm}(g) \mid \mathrm{hterm}(f)$ och därigenom har vi att

$$f \xrightarrow[g]{} h$$

för något $h \in \langle G \rangle$. Eftersom också $h \in \langle G \rangle$ kan vi på samma sätt reducera bort det ledande monomet i h. Eftersom f bara har ett ändligt antal termer ser vi att denna procedur leder till att $f \xrightarrow[G]{*} 0$. •

Vi kan säga att denna sats visar på två likvärdiga sätt att beskriva en Gröbnerbas. Har vi en bas och vill visa att den är en Gröbnerbas kan vi välja det sätt som för tillfället passar bäst. Vi kommer längre fram att utöka denna lista med fler och—förhoppningsvis—alltmer användbara sätt.

Sats 4.7 *Om I är ett ideal till $K[X]$ och S basen enligt ovanstående resonemang till* $\mathrm{hterm}(I)$ *så bildar mängden*

$$G = \{f \in I \mid \exists s \in S \textit{ med } \mathrm{hterm}(f) = s\}$$

en Gröbnerbas.

Bevis: Eftersom S var en bas för $\mathrm{hterm}(I)$ gäller att det för varje $t \in \mathrm{hterm}(I)$ finns ett $s \in S$ (ett $f \in G$ med $\mathrm{hterm}(f) = s$) så att $s|t$, och eftersom $S = \mathrm{hterm}(G)$ följer av sats 4.6 att G är en Gröbnerbas. •

Vi har nu visat att det till varje ideal i polynomringen finns en Gröbnerbas. Är den entydig, d v s givet två Gröbnerbaser G_1 och G_2, kan vi säga att

$$\langle G_1 \rangle = \langle G_2 \rangle \Longleftrightarrow G_1 = G_2 \ ?$$

Svaret är tyvärr nej. Men dessbättre finns det en variant av Gröbnerbaser, *reducerad* Gröbnerbas, som precis som Gröbnerbaser alltid existerar men som dessutom är entydiga. Hur dessa ser ut återkommer vi till i sektion 4.3 om reducerade Gröbnerbaser.

I nästa sektion ska vi se på de beräkningsmässiga aspekterna. Är Gröbnerbasen beräkningsbar och i så fall hur kan vi beräkna den?

4.2 Konstruktion—Buchbergers algoritm

I föregående sektion såg vi ett existensbevis för Gröbnerbaser. Det tråkiga med det beviset är att det inte säger något om *hur* vi ska kunna *konstruera* en Gröbnerbas. I den här sektionen ska vi utöka sats 4.6 med fler ekvivalenta påståenden. Efter lite trixande kommer vi fram till ett påstående som visar sig vara mycket enkelt att implementera i en algoritm som praktiskt beräknar en Gröbnerbas.

Vi börjar i (a) med ett ordningsrelaterat begrepp, lokal konfluens, som säger att om vi reducerar modulo en Gröbnerbas kan vi i varje reduktionssteg välja fritt bland våra "möjliga reducerare" (funktionen selectpoly i algoritmen Reduce), d v s det spelar ingen roll vilken reducerare vi i ett visst reduktionssteg väljer. I (b) presenterar vi ett "hjälpvillkor" vars enda uppgift är att hjälpa oss med beviset i (c) som i sin tur *äntligen* ger oss ett villkor med vars hjälp vi *praktiskt kan beräkna* en Gröbnerbas.

(a) Lokal konfluens

Först måste vi dock utöka vårt vokabulär. Vi har tidigare sett att om vi har flera olika reducerare att välja mellan kan vi reducera på flera olika sätt. Vi kan t ex ha att $a \underset{g_1}{\longrightarrow} b$ och $a \underset{g_2}{\longrightarrow} c$. En trevlig egenskap vore om vi trots att vi har valt olika "vägar" till en början ändå kan komma till ett och samma slutmål, d v s om det fanns ett d sådant att $a \underset{g_1}{\longrightarrow} b \overset{*}{\underset{G}{\longrightarrow}} d$ och $a \underset{g_2}{\longrightarrow} c \overset{*}{\underset{G}{\longrightarrow}} d$. Om relationen $\underset{G}{\longrightarrow}$ har denna egenskap säger vi att den är *lokalt konfluent*.

Definition 4.8 (Lokalt konfluent) *Om* $a \underset{G}{\longrightarrow} b$ *och* $a \underset{G}{\longrightarrow} c$ *sägs* $\underset{G}{\longrightarrow}$ *vara lokalt konfluent om* $b \downarrow c$, *d v s det existerar ett* d *sådant att* $b \overset{*}{\underset{G}{\longrightarrow}} d \overset{*}{\underset{G}{\longleftarrow}} c$. *(kortare skrivsätt för* $b \overset{*}{\underset{G}{\longrightarrow}}$ *och* $b \overset{*}{\underset{G}{\longrightarrow}}$*).*

Vi kan nu utöka vår lista i sats 4.6 med ekvivalenta definitioner på en Gröbnerbas:

Sats 4.9 *Låt I vara ett ideal till $K[X]$ och G en ändlig delmängd av I så att $0 \notin G$. Då är följande påståenden ekvivalenta med att G är en Gröbnerbas till I:*

(i) $f \xrightarrow[G]{*} 0$ *för alla* $f \in \langle G \rangle$.

(ii) För varje $s \in \mathrm{hterm}(I)$ *finns det ett* $t \in \mathrm{hterm}(G)$ *så att* $t|s$.

(iii) $\xrightarrow[G]{}$ *är lokalt konfluent.*

Bevis: $(ii) \Rightarrow (iii)$ och $(iii) \Rightarrow (i)$, se [BW93] sats 5.35. •

(b) Translationslemmat

Lemma 4.10 *Låt $p, q, r \in K[X]$ och $S \subseteq K[X]$. Om $p - q \xrightarrow[S]{} r$ finns det $p', q' \in K[X]$ så att $p \xrightarrow[S]{*} p'$ och $q \xrightarrow[S]{*} q'$ som dessutom uppfyller $r = p' - q'$, d v s givet*

$$\begin{array}{ccccc} p & - & q & = & h \\ & & & & \downarrow \\ & & & & r \end{array}$$

kan vi hitta p', q' så att

$$\begin{array}{ccccc} p & - & q & = & h \\ \downarrow & & \downarrow & & \downarrow \\ p' & - & q' & = & r \end{array}$$

Bevis: Reduktion ett steg ger att $r = (p - q) - as$ för något $a \in M(X)$ och $s \in S$. Låt $v = \mathrm{hterm}(as)$ (den term som reduceras bort), c_1 koefficienten till v i p och c_2 koefficienten till v i q. Eftersom termen v reduceras bort i differensen $p - q$ har vi att $c_1 - c_2 = \mathrm{hcoeff}(as) = \mathrm{hcoeff}(a)$ (den sista likheten fås eftersom vi kan anta att $\mathrm{hcoeff}(s) = 1$).

Kan vi hitta $b_1, b_2 \in M(X)$ så att

$$\begin{aligned} p' &= p - b_1 s \\ q' &= q - b_2 s \ ? \end{aligned}$$

Vi får att

$$p' - q' = (p - b_1 s) - (q - b_2 s) = p - q - (b_1 - b_2)s$$

och vi får som villkor att $b_1 - b_2 = a$. Låt

$$b_1 = c_1 \frac{v}{\operatorname{hterm}(s)}$$
$$b_2 = c_2 \frac{v}{\operatorname{hterm}(s)}$$

Vi får nu att

$$\begin{aligned} b_1 - b_2 &= (c_1 - c_2)\frac{\operatorname{hterm}(a)\operatorname{hterm}(s)}{\operatorname{hterm}(s)} = (c_1 - c_2)\operatorname{hterm}(a) \\ &= \operatorname{hcoeff}(a)\operatorname{hterm}(a) = a \end{aligned}$$

•

Lemma 4.11 (Translationslemmat) *Låt $p, q \in K[X]$ och låt $P \subset K[X]$. Då gäller att om $p - q \xrightarrow[P]{*} 0$ så $p \downarrow q$.*

Bevis: Om $p = q$ gäller förstås att $p \downarrow q$, säg $p \longrightarrow d \longleftarrow q$.

$$\begin{array}{ccccc} p & - & q & = & 0 \\ \downarrow & & \downarrow & & \\ d & & d & & \end{array}$$

Om $p - q = h \xrightarrow{*} 0$ finns det en kedja

$$h \longrightarrow h_1 \longrightarrow \cdots \longrightarrow h_n \longrightarrow 0$$

och vi har

$$\begin{array}{ccccc} p & - & q & = & h \\ & & & & \downarrow \\ & & & & h_1 \\ & & & & \downarrow \\ & & & & \vdots \\ & & & & \downarrow \\ & & & & h_n \\ & & & & \downarrow \\ & & & & 0 \end{array}$$

Enligt lemma 4.10 kan vi hitta p_1 och q_1 så att

$$\begin{array}{ccccc} p & - & q & = & h \\ \downarrow & & \downarrow & & \downarrow \\ p_1 & - & q_1 & = & h_1 \\ & & & & \downarrow \\ & & & & \vdots \\ & & & & \downarrow \\ & & & & h_n \\ & & & & \downarrow \\ & & & & 0 \end{array}$$

och vi kan på samma sätt fylla i resten och få

$$\begin{array}{ccccc} p & - & q & = & h \\ \downarrow & & \downarrow & & \downarrow \\ p_1 & - & q_1 & = & h_1 \\ \downarrow & & \downarrow & & \downarrow \\ \vdots & & \vdots & & \vdots \\ \downarrow & & \downarrow & & \downarrow \\ p_n & - & q_n & = & h_n \\ \downarrow & & \downarrow & & \downarrow \\ p_{n+1} & - & q_{n+1} & = & 0 \end{array}$$

Nu är $p_{n+1} = q_{n+1}$ och genom att tillämpa vårat första specialfall är lemmat bevisat. •

Sats 4.12 *Låt I vara ett ideal till $K[X]$ och G en ändlig delmängd av I så att $0 \notin G$. Då är följande påståenden ekvivalenta med att G är en Gröbnerbas till I:*

(i) $f \xrightarrow[G]{*} 0$ *för alla* $f \in \langle G \rangle$.

(ii) För varje $s \in \mathrm{hterm}(I)$ *finns det ett* $t \in \mathrm{hterm}(G)$ *så att* $t|s$.

(iii) $\xrightarrow[G]{}$ *är lokalt konfluent.*

(iv) Om $g_1, g_2 \in G$, $g_1 \neq g_2$ *och* $m_1, m_2 \in M(X)$ *uppfyller att*

$$\mathrm{hmon}(m_1 g_1) = \mathrm{hmon}(m_2 g_2)$$

gäller att $m_1 g_1 - m_2 g_2 \xrightarrow[G]{*} 0$.

Bevis: Att $(i) \Leftrightarrow (ii) \Leftrightarrow (iii)$ har vi visat i sats 4.9.
(iii)⇒(iv): Låt $f, f_1, f_2 \in K[X]$ uppfylla att

$$f_1 \xleftarrow[G]{} f \xrightarrow[G]{} f_2 \ .$$

Då är $f_1 = f - m_1 g_1$ och $f_2 = f - m_2 g_2$ för några $m_1, m_2 \in M(X)$ och $g_1, g_2 \in G$. För att $\xrightarrow[G]{}$ ska vara lokalt konfluent måste $f_1 \downarrow f_2$ och lemma 4.11 säger att detta är sant om $f_1 - f_2 \xrightarrow[G]{*} 0$, d v s om

$$f_1 - f_2 = m_2 g_2 - m_1 g_1 \xrightarrow[G]{*} 0 \ .$$

Vi får två fall:

1. $\mathrm{hterm}(m_1 g_1) \neq \mathrm{hterm}(m_2 g_2)$, en av termerna måste vara större än den andra enligt den aktuella monomordningen, t ex $m_2 g_2 > m_1 g_1$. Vi får då

$$m_2 g_2 - m_1 g_1 \xrightarrow[g_2]{} - m_1 g_1 \xrightarrow[g_1]{} 0$$

2. $\mathrm{hterm}(m_1 g_1) = \mathrm{hterm}(m_2 g_2)$. $m_1 g_1$ reducerar bort termen $\mathrm{hterm}(m_1 g_1)$ från f och $m_2 g_2$ reducerar bort termen $\mathrm{hterm}(m_2 g_2)$ från f. Eftersom $\mathrm{hterm}(m_1 g_1) = \mathrm{hterm}(m_2 g_2)$ måste de reducera bort samma term från f och då måste $\mathrm{hmon}(m_1 g_1) = \mathrm{hmon}(m_2 g_2)$. Vi antog från början att om $\mathrm{hmon}(m_1 g_1) = \mathrm{hmon}(m_2 g_2)$ har vi att $m_2 g_2 - m_1 g_1 \xrightarrow[G]{*} 0$.

(i)⇒(iv): Vi har att $g_1 \in G \subseteq \langle G \rangle \Rightarrow m_1 g_1 \in \langle G \rangle$ och på samma sätt $m_2 g_2 \in \langle G \rangle$. Detta leder till att $m_1 g_1 - m_2 g_2 \in \langle G \rangle$ och följdaktligen, enligt (i), att $m_1 g_1 - m_2 g_2 \xrightarrow[G]{*} 0$. •

(c) S-polynom

Vårat mål för den här sektionen är att komma fram till ett begrepp som kallas *s-polynom*. Med dessa visar det sig bli en enkel uppgift att konstruera en Gröbnerbas.

Definition 4.13 (S-polynom) *S-polynomet av $p, q \in F[X]$ är*

$$\mathrm{Spoly}(p, q) = \mathrm{lcm}(\mathrm{hmon}(p), \mathrm{hmon}(q)) \left[\frac{p}{\mathrm{hmon}(p)} - \frac{q}{\mathrm{hmon}(q)} \right]$$

Varför ska just denna definition leda till något nyttigt? Vi såg i tidigare exempel att om vi hade att $f_1 \xrightarrow[G]{*} 0$ och $f_2 \xrightarrow[G]{*} 0$ kunde det hända att linjärkombinationen $f_1 + f_2$ *inte* reducerade till noll trots att den låg i idealet, se sektion 3.4. Om vi vill att alla linjärkombinationer liknande denna som ligger i idealet ska reducera till noll, vad leder det till?

Vi såg i sats 4.6 att G är en Gröbnerbas är ekvivalent med att

$$(\forall s \in \langle G \rangle)\,(\exists t \in G)\,(\mathrm{hterm}(t) \mid \mathrm{hterm}(s))\ .$$

Vi vet också att

$$s \in \langle G \rangle \Leftrightarrow s = \sum_{i=1}^{n} a_i g_i\ .$$

Om vi tar två element i $\langle G \rangle$ gäller att

$$\begin{aligned} &f_1, f_2 \in \langle G \rangle \Rightarrow \\ &\exists t_1, t_2 \in G \text{ så att } \mathrm{hterm}(t_1) \mid \mathrm{hterm}(f_1) \text{ och } \mathrm{hterm}(t_2) \mid \mathrm{hterm}(f_2)\ . \end{aligned}$$

För dessa två element kan vi skriva

$$\begin{aligned} f_1 &= \sum a_i g_i \qquad \mathrm{hterm}(f_1) = \mathrm{hterm}(a_{f_1} g_{f_1}) \qquad t_{f_1} \in G \qquad t_{f_1} \mid \mathrm{hterm}(f_1) \\ f_2 &= \sum b_i g_i \qquad \mathrm{hterm}(f_2) = \mathrm{hterm}(b_{f_2} g_{f_2}) \qquad t_{f_2} \in G \qquad t_{f_2} \mid \mathrm{hterm}(f_2)\ . \end{aligned}$$

Nu bildar vi en linjärkombination av dessa två element:

$$f = cf_1 + df_2 = c \cdot \sum a_i g_i + d \cdot \sum b_i g_i = \sum (ca_i + db_i) g_i$$

Vi får fyra olika fall:

Fall 1: $\operatorname{hterm}(ca_{f_1}g_{f_1}) >_T \operatorname{hterm}(db_{f_2}g_{f_2})$
$\operatorname{hterm}(f) = \operatorname{hterm}(ca_{f_1}g_{f_1}) \Rightarrow t_{f_1} \mid \operatorname{hterm}(f)$

Fall 2: $\operatorname{hterm}(ca_{f_1}g_{f_1}) <_T \operatorname{hterm}(db_{f_2}g_{f_2})$
$\operatorname{hterm}(f) = \operatorname{hterm}(cb_{f_2}g_{f_2}) \Rightarrow t_{f_2} \mid \operatorname{hterm}(f)$

Fall 3: $\operatorname{hterm}(ca_{f_1}g_{f_1}) =_T \operatorname{hterm}(db_{f_2}g_{f_2}),\ \operatorname{hcoeff}(ca_{f_1}g_{f_1}) \neq_T -\operatorname{hcoeff}(db_{f_2}g_{f_2})$
$\operatorname{hterm}(f) = \operatorname{hterm}(ca_{f_1}g_{f_1}) = \operatorname{hterm}(db_{f_2}g_{f_2}) \Rightarrow \begin{cases} t_{f_1} \mid \operatorname{hterm}(f) \\ t_{f_2} \mid \operatorname{hterm}(f) \end{cases}$

Fall 4: $\operatorname{hmon}(ca_{f_1}g_{f_1}) = -\operatorname{hmon}(db_{f_2}g_{f_2})$
$\operatorname{hterm}(f) \neq \operatorname{hterm}(ca_{f_1}g_{f_1}) \stackrel{?}{\Rightarrow} t_{f_1} \mid \operatorname{hterm}(f)$
$\operatorname{hterm}(f) \neq \operatorname{hterm}(cb_{f_2}g_{f_2}) \stackrel{?}{\Rightarrow} t_{f_2} \mid \operatorname{hterm}(f)$

I de två sista raderna finns ett frågetecken, implikationen behöver inte gälla.

För varje par av generatorer p och q ska vi skapa en linjärkombination som på detta vis eliminerar linjärkombinationens största möjliga monom (d v s leder till fall 4), detta är precis vad vi gör när vi beräknar ett s-polynom till p och q. Eftersom linjärkombinationen tillhör idealet ska vi kunna reducera också den till noll. Om vi lyckas är allt OK, annars utökar vi basen med den reducerade linjärkombinationen för att "rätta till" basen. Vi kommer strax att se i sats 4.16(v) att detta förfarande är både nödvändigt och tillräckligt för att utöka en bas till en Gröbnerbas.

I beviset av sats 4.16 behöver vi två lemman, först en alternativ beskrivning av s-polynom:

Lemma 4.14 *För $i = 1, 2$, låt $0 \neq g_i \in K[X]$, $t_i = \operatorname{hterm}(g_i)$, $a_i = \operatorname{hcoeff}(g_i)$ och $t = s_i t_i = \operatorname{lcm}(t_1, t_2)$ där $s_i \in T$. S-polynomet till g_1 och g_2 är då*

$$\operatorname{Spoly}(g_1, g_2) = a_2 s_1 g_1 - a_1 s_2 g_2 \ . \qquad (*)$$

Bevis: Vi har

$$s_i t_i = \operatorname{lcm}(t_1, t_2) \Longleftrightarrow s_i = \frac{\operatorname{lcm}(t_1, t_2)}{t_i}$$

Instoppat i $(*)$ får vi

$$\begin{aligned}
a_2 s_1 g_1 - a_1 s_2 g_2 &= \operatorname{hcoeff}(g_2)\frac{\operatorname{lcm}(\operatorname{hterm}(g_1),\operatorname{hterm}(g_2))}{\operatorname{hterm}(g_1)} g_1 \\
&\quad - \operatorname{hcoeff}(g_1)\frac{\operatorname{lcm}(\operatorname{hterm}(g_1),\operatorname{hterm}(g_2))}{\operatorname{hterm}(g_2)} g_2 \\
&= \operatorname{lcm}(\operatorname{hterm}(g_1),\operatorname{hterm}(g_2))\left(\frac{\operatorname{hcoeff}(g_2)}{\operatorname{hterm}(g_1)} g_1 - \frac{\operatorname{hcoeff}(g_1)}{\operatorname{hterm}(g_2)} g_2\right) \\
&= \operatorname{lcm}(\operatorname{hterm}(g_1),\operatorname{hterm}(g_2))\operatorname{hcoeff}(g_1)\operatorname{hcoeff}(g_2) \\
&\quad \cdot \left(\frac{g_1}{\operatorname{hcoeff}(g_1)\operatorname{hterm}(g_1)} - \frac{g_2}{\operatorname{hcoeff}(g_2)\operatorname{hterm}(g_2)}\right) \\
&= c \cdot \operatorname{lcm}(\operatorname{hterm}(g_1),\operatorname{hterm}(G_2))\left(\frac{g_1}{\operatorname{hmon}(g_1)} - \frac{g_2}{\operatorname{hmon}(g_2)}\right) \\
&= c \cdot \operatorname{Spoly}(g_1, g_2)
\end{aligned}$$

där $c = \operatorname{hcoeff}(g_1)\operatorname{hcoeff}(g_2)$. •

Lemma 4.15 *Låt $P \subseteq K[X]$, $f \in K[X]$ och $m \in M(X)$. Då gäller att*

$$f \xrightarrow[P]{*} 0 \Longrightarrow mf \xrightarrow[P]{*} 0$$

Bevis: Vi kan skriva

$$f \xrightarrow[P]{*} 0$$

som

$$f = f_0 \xrightarrow[p_{i_0}]{} f_1 \longrightarrow \cdots \xrightarrow[p_{i_{n-1}}]{} f_n = 0\ , \qquad p_i \in P\ .$$

Om vi tittar på ett steg

$$f_k \xrightarrow[p_k]{} f_{k+1}$$

kan vi skriva om det som

$$f_{k+1} = f_k - a_k p_k\ , \qquad a_k \in K[X]\ .$$

Multiplicera med $m \in M(X)$:

$$mf_{k+1} = mf_k - ma_k p_k = m(f_k - a_k p_k)$$

vilket är samma sak som

$$mf_k \xrightarrow[p_k]{} mf_{k+1}$$

och om vi gör på samma sätt med alla steg i sekvensen får vi

$$mf \xrightarrow[p_0]{} mf_1 \longrightarrow \cdots \xrightarrow[p_{n-1}]{} mf_n = m \cdot 0 = 0$$

eller kortare uttryckt som

$$mf \xrightarrow[P]{*} 0 \ .$$

•

Vi kommer nu fram till det ekvivalenta villkor vi så länge väntat på! Problemet hittils har varit att de olika karaktäriseringar vi tagit fram inte varit användbara i praktiken vid beräkningen av Gröbnerbaser. Vi kan nu utöka sats 4.12 med ett villkor till:

Sats 4.16 *Låt I vara ett ideal till $K[X]$ och G en ändlig delmängd av I så att $0 \notin G$. Då är följande påståenden ekvivalenta med att G är en Gröbnerbas till I:*

(i) $f \xrightarrow[G]{*} 0$ *för alla* $f \in \langle G \rangle$.

(ii) För varje $s \in \mathrm{hterm}(I)$ *finns det ett* $t \in \mathrm{hterm}(G)$ *så att* $t|s$.

(iii) $\xrightarrow[G]{}$ *är lokalt konfluent.*

(iv) Om $g_1, g_2 \in G$, $g_1 \neq g_2$ *och* $m_1, m_2 \in M(X)$ *uppfyller att*

$$\mathrm{hmon}(m_1 g_1) = \mathrm{hmon}(m_2 g_2)$$

gäller att $m_1 g_1 - m_2 g_2 \xrightarrow[G]{*} 0$.

(v) $\mathrm{Spoly}(p, q) \xrightarrow[G]{*} 0$ *för alla* $p, q \in G$.

Bevis: Att $(i) \Leftrightarrow (ii) \Leftrightarrow (iii) \Leftrightarrow (iv)$ såg vi i sats 4.12.

(i)$\Longrightarrow$(v): Spoly(p, q) är enligt lemma 4.14 en linjärkombination av två element ur G och ligger alltså i $\langle G \rangle$.

(v)$\Longrightarrow$(iv): Vi behöver bara försäkra oss om att polynom på formen $m_1 g_1 - m_2 g_2$ där $g_1 \neq g_2$ ligger i G, m_1 och m_2 är monom och

$$\operatorname{hmon}(m_1 g_1) = \operatorname{hmon}(m_2 g_2) \tag{4.3}$$

reducerar till 0 modulo G. För $i = 1, 2$, låt $t_i = \operatorname{hterm}(g_i)$, $a_i = \operatorname{hcoeff}(g_i)$ och $m_i = b_i u_i$ där $b_i \in K$ och $u_i \in T$. Vi får då att $\operatorname{hmon}(m_1 g_1) = m_1 \operatorname{hmon}(g_1) = b_1 u_1 a_1 t_1$ och ekvation 4.3 blir

$$\underbrace{b_1 a_1}_{\in K} \underbrace{u_1 t_1}_{\in T} = \underbrace{b_2 a_2}_{\in K} \underbrace{u_2 t_2}_{\in T} \tag{4.4}$$

Identifiering av koefficienter ($\in K$) i vänster- och högerled ger att

$$b_1 a_1 = b_2 a_2 \Longleftrightarrow \frac{a_2}{b_1} = \frac{a_1}{b_2}$$

Identifiering av termer ($\in T$) i vänster- och högerled ger att

$$u_1 t_1 = u_2 t_2$$

och speciellt gäller att $u_1 t_1$ och $u_2 t_2$ är en gemensam multipel till t_1 och t_2 (de är ju t o m lika!). Om vi för $i = 1, 2$ och $s_i \in T$ definierar $s_i t_i = \operatorname{lcm}(t_1, t_2)$ ser vi att eftersom $u_i t_i$ är en gemensam multipel och $s_i t_i$ är den *minsta* gemensamma multipeln finns det ett $v \in T$ så att

$$u_i t_i = v \cdot \operatorname{lcm}(t_1, t_2) = v s_i t_i$$

och vi får att $u_i = v s_i$. Vi får nu att

$$\begin{aligned}
m_1 g_1 - m_2 g_2 &= \lceil m_i = b_i u_i \rceil^{\dagger} = b_1 u_1 g_1 - b_2 u_2 g_2 \\
&= \lceil u_i = v s_i \rceil = b_1 v s_1 g_1 - b_2 v s_2 g_2 = \frac{b_1}{a_2} v \left(a_2 s_1 g_1 - \frac{a_2}{b_1} b_2 s_2 g_2 \right) \\
&= \left\lceil \frac{a_2}{b_1} = \frac{a_1}{b_2} \right\rceil = \frac{b_1}{a_2} v \left(a_2 s_1 g_1 - a_1 s_2 g_2 \right) \\
&= \frac{b_1}{a_2} v \operatorname{Spoly}(g_1, g_2)
\end{aligned}$$

Vi har antagit att $\text{Spoly}(g_1, g_2) \xrightarrow[G]{*} 0$ och detta tillsammans med lemma 4.15 ger att

$$\text{Spoly}(g_1, g_2) \xrightarrow[G]{*} 0 \Longleftrightarrow m_1 g_1 - m_2 g_2 = \frac{b_1}{a_2} v \ \text{Spoly}(g_1, g_2) = m \cdot \text{Spoly}(g_1, g_2) \xrightarrow[G]{*} 0$$

•

Korollarium 4.17 *Om $G = \{g\}$, $g \in K[X_1, \ldots, X_n]$, så är G en Gröbnerbas.*

Bevis: Med bara ett element i basen kan vi bara bilda ett enda s-polynom, $\text{Spoly}(g, g)$, som blir noll. •

Korollarium 4.18 *Om $G = \{g_1, \ldots, g_n\}$ där $g_i \in K[X_1, \ldots, X_n]$ är monom så är G en Gröbnerbas.*

Bevis: Eftersom vi för alla $g \in G$ har att $g = \text{hmon}(g)$ är alla s-polynom noll.
•

Buchbergers algoritm

Villkor (v) i sats 4.16 ger oss en praktiskt användbar väg att gå för att utöka ett ideals bas till en Gröbnerbas.

Satsen säger att G är en Gröbnerbas om och endast om alla till G hörande s-polynom reducerar till noll. Vi beräknar alla s-polynom till G, reducerar dem och om vi träffar på ett som *inte* kan reduceras till noll skapar vi en ny bas G' som är unionen av G och det reducerade s-polynomet. Eftersom ett s-polynom ligger i idealet ser vi att vi inte förändrar själva idealet, vi lägger bara till ett baselement till. Om vi på detta vis lägger till ett nytt baselement upprepar vi hela proceduren från början. Om vi efter ett antal upprepningar på detta vis fått en bas G vars alla s-polynom reduceras till noll har vi enligt sats 4.16 skapat oss en Gröbnerbas.

Ovanstående resonemang kan lätt omsättas i en algoritm, se tabell 4.2.

Rad 3 skapar en mängd med talpar, ett talpar för varje s-polynom som ska räknas ut vilket innebär alla kombinationer av idealet P:s generatorer.

Raderna 4–12 kommer att utföras tills vi har beräknat alla s-polynom.

Rad 5 väljer ett element ur B, för att se vilket man bör välja, se kapitel 4.3. Rad 7 beräknar det motsvarande s-polynomet och reducerar det så långt det går.

Om s-polynomet inte kunde reduceras till noll ska vi utöka basen med det reducerade s-polynomet vilket görs i rad 9. Eftersom vi nu har en generator till måste listan B beräknas igen vilket görs i rad 10.

[†] $a = \lceil P \rceil = b \Leftrightarrow a = b$ under antagandet att P gäller.

Tabell 4.2: Buchberg

```
1.  procedure Gbasis(P)
2.      G ← P; k ← length(G)
3.      B ← {[i, j] : 1 ⩽ i < j ⩽ k}
4.      while B ≠ ∅ do begin
5.          [i, j] ← selectpair(B, G)
6.          B ← B − {[i, j]}
7.          h ← Reduce(Spoly(G_i, G_j), G)
8.          if h ≠ 0 then begin
9.              G ← G ∪ {h}; k ← k + 1
10.             B ← B ∪ {[i, k] : 1 ⩽ i < k}
11.         end
12.     end
13.     return(G)
14. end
```

Vi har:

Sats 4.19 (Buchbergers algoritm) *Låt F vara en ändlig delmängd av $K[X]$. Anta att K är beräkningsbar och att termordningen på T är bestämbar. Då beräknar algoritmen Buchberg i tabell 4.2 med ett ändligt antal steg (d v s den terminerar inom ändlig tid) en Gröbnerbas G i $K[X]$ så att $F \subseteq G$ och $\langle G\rangle = \langle F\rangle$.*

Bevis: Se [BW93] sats 5.53. •

Vi har nu visat att det till varje ideal i polynomringen finns en Gröbnerbas, att den är beräkningsbar och vi har presenterat en algoritm som utför beräkningarna. I nästa sektion ska vi presentera en utökning av algoritmen i tabell 4.2 som beräknar en för ett ideal unik Gröbnerbas.

Ekvivalens och kanonisk förenklare

Vi kan nu bevisa sats 3.8:

Lemma 4.20 *Låt G vara en Gröbnerbas. Om $p \xrightarrow[G]{*} q$ och $p \xrightarrow[G]{*} r$ och p respektive q är irreducibla modulo G så är $q = r$.*

Bevis: Se [GCL92] sats 10.5(iii). •

Sats 4.21 *Om G är en Gröbnerbas och $p, q \in \langle G \rangle$ gäller att*

$$\text{Reduce}(p, G) = \text{Reduce}(q, G) \Longleftrightarrow p - q \in \langle G \rangle$$

Bevis: $\Rightarrow$: Anta att $r = \text{Reduce}(p, G) = \text{Reduce}(q, G)$. Då gäller att $p - r \in \langle G \rangle$ och $q - r \in \langle G \rangle$ och vi har att

$$(p - r) - (q - r) = p - q \in \langle G \rangle \ .$$

$\Leftarrow$: Enligt lemma 4.11 finns det ett d sådant att $p \xrightarrow[G]{*} d$ och $q \xrightarrow[G]{*} d$. Om vi fortsätter att reducera d så långt det går får vi $p \xrightarrow[G]{*} d \xrightarrow[G]{*} e$ och $q \xrightarrow[G]{*} d \xrightarrow[G]{*} f$ (i reduktionsalgoritmen kan man välja reducerare på flera olika sätt). Eftersom vi nu har att $d \xrightarrow[G]{*} e$ och $d \xrightarrow[G]{*} f$ där e och f är irreducibla säger lemma 4.20 att $e = f$, d v s att $\text{Reduce}(p, G) = \text{Reduce}(q, G)$. •

4.3 Reducerad Gröbnerbas—entydighet

Den Gröbnerbas som produceras av 4.2 lider av ett par problem. För det första är den ej unik. Till ett och samma ideal kan vi beroende på de val som måste göras i algoritmen få flera olika Gröbnerbaser. Vi skulle vilja modifiera algoritmen så att den bas vi får bestäms unikt av idealet, matematiskt uttryckt som

$$\langle G_1 \rangle = \langle G_2 \rangle \Longleftrightarrow G_1 = G_2 \ .$$

För det andra är algoritmen av mycket hög komplexitet och den utan jämförelse största delen av tiden spenderar den med att reducera s-polynom[1]. Ironiskt nog visar det sig att de flesta s-polynom algoritmen försöker reducera också reduceras till noll[2]—vi utför en massa kostsamma reduceringar "i onödan". Det vore önskvärt om vi bara genom att "titta" på ett s-polynom kunde avgöra om det reduceras till noll eller inte. Det finns ett par kriterier som om man använder sig av dessa upptäcker de flesta s-polynom som reducerar till noll. Inte alla, men eftersom kriterierna i sig är betydligt "billigare" beräkningsmässigt sett är det en bra affär att använda sig av dem.

Entydighet

Vi ska i denna sektion se hur man kan modifiera algoritmen Buchberg i tabell 4.2 så att den till ett givet ideal beräknar en för detta ideal *unik* Gröbnerbas.

Definition 4.22 (Reducerad Gröbnerbas) *En reducerad Gröbnerbas är en Gröbnerbas (en mängd polynom som är en Gröbnerbas) som, sedd som en mängd, är reducerad och monisk (enligt definition 3.7).*

Vi går tillväga på samma sätt som förut, vi börjar med att bevisa att det alltid finns en reducerad Gröbnerbas och sedan presenterar vi en algoritm som beräknar den.

Vårat första steg på vägen mot en unik Gröbnerbas är att se lite mer på Dicksonbasernas egenskaper. För en allmänn partiell ordning gäller:

Proposition 4.23 *Låt $\leqslant$ vara en partiell ordning på M. Då är följande påståenden ekvivalenta:*

(i) *$\leqslant$ är en Dickson-partiell ordning.*

(ii) *Om $\{a_n\}_{n\in\mathbf{N}}$ är en sekvens av element ur M så finns det ett $i > j$ så att $a_i \leqslant a_j$.*

(iii) *För varje icke-tom delmängd N av M är antalet minimala element i N ändligt och skilt från noll.*

Bevis: (i)⇒(ii): Eftersom $\leqslant$ är en Dickson-partiell ordning kan vi dela upp mängden $\{a_i\}$ i ett ändligt antal delmängder där varje delmängd motsvaras av ett baselement. Vi får på detta vis minst en oändlig delmängd men eftersom varje delmängd har ett minsta element kan den ej vara strikt avtagande.

(ii)⇒(iii): Anta att det finns oändligt många minimala element. Vi kan då skapa en oändlig sekvens $\{b_i\}$ av minimala element[†]. Men (ii) säger att det för varje oändlig sekvens $\{a_i\}$ finns det ett $i < j$ så att $a_i \leqslant a_j$ vilket motsäger elementen $\{b_i\}$:s minimalitet.

(iii)⇒(i): Låt N vara en icke-tom delmängd av M och B mängden av alla minimala element i N. Låt $a \in N$ och bilda

$$N' = \{d \in N \mid d \leqslant a\} \ .$$

N' innehåller då ett minsta element c, detta element måste också vara minimalt i N och det finns då ett $b \in B$ så att $b \leqslant c \leqslant a$ och alltså $b \leqslant a$. •

Vad vi egentligen vill åt är en direkt följdsats till ovanstående proposition:

[†]Enligt urvalsaxiomet (eng. "axiom of choice").

Korollarium 4.24 *Låt $\leqslant$ vara en Dickson-partiell ordning på M. Då har varje icke-tomma delmängd N av M en* **unik minimal och ändlig bas** *B, d v s en bas B sådan att $B \subseteq C$ för alla andra baser C till N. B består av alla minimala element i N.*

Bevis: Låt B vara mängden av alla minimala element i N. Enligt proposition 4.23(iii) är B ändlig och icke-tom. Dessutom gäller att för varje $a \in N$ finns det ett $b \in B$ sådant att $b \leqslant a$, m a o så är B en bas för N. Låt nu C vara en annan bas för N. För varje $b \in B$ finns då ett $c \in C$ sådant att $c \leqslant b$. Eftersom b var minimalt måste det gälla att $c = b$ vilket ger att $b \subseteq C$. •

Med en termordning, t ex $\leqslant_L$, kan vi jämföra två termer med varandra och tala om vilken som är "störst". Kan vi definiera en liknande ordning $\leqslant_P$ för polynom? Om vi ska jämföra två polynom f och g med varandra sorterar vi först termerna enligt en termordning. Börja sedan med att jämföra de högsta termerna ur f respektive g med varandra och fortsätt sedan med succesivt lägre termer. Om båda termerna är lika fortsätter vi med nästa term. Om inte säger vi att det polynom är minst vars term är minst. Med denna procedur har vi definierat en polynomordning $\leqslant_P$ *inducerad* av en termordning $\leqslant_T$.

Lemma 4.25 *Anta att I är ett ideal till $K[X]$, m ett monom och f och g är minimala polynom i I så att* $\mathrm{hmon}(f) = \mathrm{hmon}(g) = m$. *Då är $f = g$.*

Bevis: Enligt resonemanget ovan måste vi ha att $T(f) = T(g)$ annars skulle antingen $f < g$ eller $f > g$. Eftersom både $f \in I$ och $g \in I$ gäller att $f - g \in I$. Antingen är $f - g = 0$ och lemmat bevisat eller så är $s = \mathrm{hterm}(f - g) < m$ (eftersom $\mathrm{hmon}(f) = \mathrm{hmon}(g)$ finns ej denna term med i $f - g$ och de termer som "blir kvar" är alla mindre än den högsta). Vi har att $s \in T(f) = T(g)$. Vi kan i uttrycket

$$h = f - c(f - g) \in I$$

välja $0 \neq c \in K$ så att termen s i h elimineras och därmed $s \notin T(h)$. Eftersom $\mathrm{hterm}(f - g) < m$ är $\mathrm{hmon}(h) = m$. Vad som skiljer h från f är den eliminerade termen s och eftersom $f \longrightarrow h$ har vi att $h < f$ vilket är en motsägelse eftersom vi från början antog att f var minimalt. •

Vi kan nu svara på frågan om det till ett givet ideal I finns en *reducerad* Gröbnerbas:

Sats 4.26 *Om I är ett ideal i $K[X]$ existerar det en reducerad Gröbnerbas G till I med avseende på den inducerade polynomordningen $\leqslant_P$.*

Vi delar upp beviset i tre delar: (1) Först konstruerar vi en mängd vi anser vara en bas sedan visar vi (2) att den är monisk och (3) att den är reducerad.

Bevis: (1) Låt S vara den enligt korollarium 4.24 unika minimala basen till $\text{hterm}(I)$ med avseende på $|$. För varje $t \in S$ finns det ett motsvarande $f_t \in I$ med $\text{hterm}(f_t) = t$. Eftersom I är ett ideal kan vi anta att $\text{hmon}(f_t) = t$, d v s $\text{hcoeff}(f_t) = 1$ och enligt lemma 4.25 kan vi anta att f_t är det unika minimala polynomet i I med den här egenskapen. Låt

$$G = \{f_t \mid t \in S\} \ .$$

Då är G en Gröbnerbas till I enligt samma resonemang som i beviset av sats 4.7.

(2) Eftersom alla $\text{hcoeff}(f_t) = 1$ är G monisk.

(3) Vi antar att det finns $g_1, g_2 \in G$ och $f \in K[X]$ så att $g_1 \underset{g_2}{\longrightarrow} f$ och $g_1 \neq g_2$, d v s att G inte är reducerad. Om reduktionen $g_1 \underset{g_2}{\longrightarrow} f$ är en topreduktion har vi att $\text{hterm}(g_2) \mid \text{hterm}(g_1)$ och $S \setminus \{\text{hterm}(g_1)\}$ skulle i så fall också ha varit en bas till $\text{hterm}(I)$, men S var minimal och vi har en motsägelse. I annat fall har vi att $\text{hmon}(g_1) = \text{hmon}(f)$ och $f < g_1$ men det leder också till en motsägelse eftersom alla element i G var minimala. •

Nu vet vi att det inte bara finns en Gröbnerbas till varje ideal utan det finns dessutom en reducerad Gröbnerbas. Vi har sökt efter en för ett ideal I unik bas och i följande sats visar vi att den redecerade Gröbnerbasen mycket riktigt också är unik:

Sats 4.27 *Om I är ett ideal i $K[X]$ och G är en reducerad Gröbnerbas till I är G den* **unika reducerade Gröbnerbasen** *till I.*

Bevis: Vi antar att vi till I har två olika reducerade Gröbnerbaser G och H.

Låt g vara ett element i den symmetriska mängddifferensen $G \triangle H$ (alla elemenet som ligger antingen i G eller H men inte i båda) så att $\text{hterm}(g)$ är minimal i $\text{hterm}(G \triangle H)$, vi kan anta att $g \in G$. Enligt sats 4.6 finns det ett $h \in H$ så att $\text{hterm}(h) \mid \text{hterm}(g)$. Eftersom $h \neq g$ och G är reducerad måste vi ha att $h \in H \setminus G$. Eftersom g var minimalt måste vi också ha att $\text{hterm}(h) = \text{hterm}(g)$.

Nu ska vi se på differensen $f = g - h$. Vi har att

$$\text{hterm}(f) < \text{hterm}(g) = \text{hterm}(h)$$

eftersom G och H är moniska. Dessutom måste $\mathrm{hterm}(f) \in T(g)$ eller $\mathrm{hterm}(f) \in T(h)$, t ex $\mathrm{hterm}(f) \in T(g)$. Men eftersom $f \in I$ måste det finnas ett $p \in G$ så att $\mathrm{hterm}(p) \mid \mathrm{hterm}(f)$ och vi ser att en term i g (skilt från $\mathrm{hterm}(g)$) delas av den ledande termen av något element i G och det motsäger antagandet att G var reducerad. •

Algoritmen ReducedBuchberg i tabell 4.3 beräknar en reducerad (och enligt sats 4.27 är det *den* reducerade) Gröbnerbas. Det nya (de understrukna raderna) är att vi börjar med att reducera mängden P på rad 2 och avslutar med att reducera bort redundanta element på raderna 13 och 14.

Tabell 4.3: ReducedBuchberg

```
1.  procedure Gbasis(P)
2.      G ← ReduceSet(P);k ← length(G)
3.      B ← {[i, j] : 1 ⩽ i < j ⩽ h}
4.      while B ≠ ∅ do begin
5.        [i, j] ← selectpair(B, G)
6.        B ← B − {[i, j]}
7.        h ← Reduce(Spoly(G_i, G_j), G)
8.        if h ≠ 0 then begin
9.          G ← G ∪ {h}; k ← k + 1
10.         B ← B ∪ {[i, k] : 1 ⩽ i < k}
11.       end
12.     end
13.     R ← {g ∈ G så att R_{g,G} − {g} ≠ ∅}
14.     return(ReduceSet(G − R))
15. end
```

Vi har:

Sats 4.28 *Låt F vara en ändlig delmängd av $K[X]$. Anta att K är beräkningsbar och att termordningen på T är bestämbar. Då beräknar algoritmen Reducedbuchberg i tabell 4.3 den unika reducerade Gröbnerbasen G i $K[X]$ så att $F \subseteq G$ och $\langle G \rangle = \langle F \rangle$.*

Bevis: Se [BW93], korollarium 5.57. •

Onödiga reduceringar

När nu problemet med att bestämma en unik Gröbnerbas är löst kommer vi in på problemet med algoritmernas höga komplexitet. Vi ska här gå igenom de i inledningen utlovade kriterierna för när ett s-polynom reducerar till noll. Observera att dessa kriterier inte är fullständiga, ett s-polynom kan passera båda kriterierna men ändå reducera till noll. Det trevliga med kriterierna är att de flesta s-polynom som reduceras till noll "fångas upp" av dem *utan* att vi behöver reducera dem.

Vi kallar två termer $s, t \in T$ **disjunkta** om de inte har några variabler gemensamt (samma sak som att säga att $\mathrm{lcm}(s,t) = st$).

Sats 4.29 *Om* $\mathrm{hterm}(p)$ *och* $\mathrm{hterm}(q)$ *är disjunkta gäller att*

$$\mathrm{Spoly}(p,q) \xrightarrow[\{p,q\}]{*} 0$$

Bevis: Vi har att

$$\begin{aligned}
\mathrm{Spoly}(p,q) &= \mathrm{lcm}(\mathrm{hmon}(p),\mathrm{hmon}(q))\left[\frac{p}{\mathrm{hmon}(p)} - \frac{q}{\mathrm{hmon}(q)}\right] \\
&= \frac{\mathrm{lcm}(\mathrm{hmon}(p),\mathrm{hmon}(q))}{\mathrm{hmon}(p)\,\mathrm{hmon}(q)}\,[p\cdot\mathrm{hmon}(q) - q\cdot\mathrm{hmon}(p)] \\
&= \alpha(p\,\mathrm{hmon}(q) - q\,\mathrm{hmon}(p)) \\
&= \alpha[(p-\mathrm{hmon}(p))\,\mathrm{hmon}(q) - (q-\mathrm{hmon}(q))\,\mathrm{hmon}(p)]
\end{aligned}$$

Eftersom $\mathrm{hterm}(p)$ och $\mathrm{hterm}(q)$ är disjunkta kan polynomen $p - \mathrm{hmon}(p)$ och $q - \mathrm{hmon}(q)$ ej ha några gemensamma termer och därför elimineras inga termer i subtraktionen. Vi har vidare att

$$\mathrm{hmon}(p) \xrightarrow[p]{} p - \mathrm{hmon}(p) \quad , \qquad \mathrm{hmon}(q) \xrightarrow[q]{} q - \mathrm{hmon}(q)$$

varur resultatet följer. •

Det andra kriteriet följer ur

Sats 4.30 *G är en Gröbnerbas om och endast om för alla $f, g \in K[X]$ gäller att $\exists h \in G, f \neq h \neq g$, så att*

$$\mathrm{hterm}(h) \mid \mathrm{lcm}(\mathrm{hterm}(f),\mathrm{hterm}(g))$$

och

$$\text{Spoly}(f,h) \xrightarrow[G]{*} 0, \text{Spoly}(h,g) \xrightarrow[G]{*} 0 \ .$$

Bevis: Se [GCL92] följdsats 10.6. •

Dessa två satser kan vi formulera om till villkor som säger att vissa s-polynom inte behöver testas om de reducerar till noll. Vi kan i algoritmen för att beräkna Gröbnerbaser hoppa över polynomparet $[i,j]$ om något av nedanstående villkor ej är uppfyllt:

1. criterion1$([i,j],G) \Leftrightarrow$

$$\text{lcm}(\text{hterm}(G_i), \text{hterm}(G_j)) \neq \text{hterm}(G_i)\,\text{hterm}(G_j)$$

2. criterion2$([i,j],B,G) \Leftrightarrow$

$$\neg\exists k, 1 \leqslant k \leqslant \text{length}(G), i \neq k \neq j \tag{4.5}$$

så att

$$\text{hterm}(G_k) \mid \text{lcm}(\text{hterm}(G_i), \text{hterm}(G_j)) \tag{4.6}$$

och

$$[i,k] \notin B, [k,j] \notin B\} \tag{4.7}$$

Med dessa två kriterier får vi den förbättrade varianten av Buchbergers algoritm enligt tabell 4.4.

Ytterligare förbättringar

Det finns många fler sätt att förkorta beräkningstiden.

I [BW93] finns fler villkor liknande de vi har gått igenom här tillsammans med mer utförligare diskussioner. Fler villkor finns också i [Buc85].

I [BW79] diskuterar Buchberger och Winkler hur funktionen selectpair bör fungera. De visar att om vi alltid väljer $[i,j]$ så att

$$\text{lcm}(\text{hterm}(G_i), \text{hterm}(G_j)) =$$
$$\min_{\leqslant_T}\{\text{lcm}(\text{hterm}(G_u), \text{hterm}(G_v)) : [u,v] \in B\}$$

maximerar vi sannolikheten att våra kriterier kan komma till användning. För fortsatt läsning, se [BW93] kapitel 5.5.

Tabell 4.4: ImprovedBuchberger

```
1.  procedure Gbasis(P)
2.      G ← ReduceSet(P);k ← length(G)
3.      B ← {[i, j] : 1 ⩽ i < j ⩽ h}
4.      while B ≠ ∅ do begin
5.        [i, j] ← selectpair(B, G)
6.        if criterion1([i, j], G) and criterion2([i, j], B, G) then begin
7.          B ← B − {[i, j]}
8.          h ← Reduce(Spoly(G_i, G_j), G)
9.          if h ≠ 0 then begin
10.           G ← G ∪ {h}; k ← k + 1
11.           B ← B ∪ {[i, k] : 1 ⩽ i < k}
12.         end
13.       end
14.     end
15.     R ← {g ∈ G så att R_{g,G} − {g} ≠ ∅
16.     return(ReduceSet(G − R))
17. end
```

4.4 Komplexitet

Vi ska här kort se på algoritmernas komplexitet.

En viktig sak att undersöka är gradtalen hos polynomen i Gröbnerbasen. Om man lyckas bestämma (övre) gränser för dessa kan man också få en indikation på antalet polynom i basen och hur många reduktionssteg som kan behövas.[3] För fortsatt läsning, se [BW93] (appendix om komplexitet).

Vilken termordning man väljer har stor inverkan på beräkningstiderna. Buchberger visar i [Buc83] att för varje naturligt tal n finns det en mängd polynom $P \subset K[X_1, X_2]$ med $D(P) = n$ ($D(P)$ är gradtalet hos det polynom $p \in P$ som har högst grad i P) så att

(a) för alla Gröbnerbaser till P med avseende på $<_D$, $D(G) \geq 2n - 1$ och

(b) för alla Gröbnerbaser till P med avseende på $<_L$, $D(G) \geq n^2 - n + 1$.

Komplexiteten för den totala gradordningen är alltså lägre än för den lexiografiska ordningen.[4]

För en specifik termordning spelar den inbördes sorteringen av variabler stor roll. För den lexiografiska ordningen är skillnaden mellan olika permutationer av variablerna så stor att man kan tala om en "instabil algoritm".[5] Den totala gradordningen är jämförelsevis stabil i det här avseendet och enligt Buchberger är den totala ordningen (oavsett ordning av variabler) lika snabb som den snabbaste permutationen för den lexiografiska ordningen.[6] Lazard visar i [Laz83] att beräkning med den totala gradordningen är $O(d^{nk})$ för något $k \in \mathbb{N}$ medans man i [CGH89b, CGH89a] kan läsa att motsvarande för den lexiografiska ordningen är $O(d^{n^k})$.

Trots dessa resultat är den lexiografiska ordningen ofta att föredra eftersom den leder till "bättre" baser i den meningen att t ex lösning av ekvationssystem blir mycket enklare. [7]

4.5 Sammanfattning

Vi har i det här kapitlet visat en mängd saker om Gröbnerbaser: Den existerar alltid och den är alltid beräkningsbar. Vi har visat att det alltid finns

en beräkningsbar, unik Gröbnerbas. Vi har även sett ett par sätt att optimera algoritmen på.

Vi har nu många, fina algebraiska verktyg och en speciell bas för ideal. I nästa del ska vi se vad allt detta kan användas till.

Noter

[1] ([GCL92] sidan 449), "It is also clear that most of the computational cost of the algorithm is in the polynomial arithmetic of the reductionstep."

[2] ([GCL92] sidan 449), "Quite typically, though, only a relatively small proportion of the S-polynomials which are reduced will yield new (nonzero) results."

[3]Se [GCL92] sidan 451

[4] [GCL92] sidan 451, "Apparently, the complexity of the algorithm is lower when using $<_D$ than when using $<_L$."

[5]Se [GCL92] sidan 462

[6]Se [Buc85] sidan 228

[7]Man får med den lexiografiska ordningen en "triangularisering" av ekvationssystemet; en ekvation i endast en variabel, en ekvation i denna variabel och en variabel till o s v. Se sektion 5.5

Del III

Tillämpningarna

Kapitel 5

Tillämpningar inom idealteori

När vi nu har ägnat så mycket möda åt konstruktion och beräkning av Gröbnerbaser vore det ju bra om de gick att använda till något vettigt. Och det finns mycket och många vettiga saker att göra med Gröbnerbaser!

I det här kapitlet ska vi se på de mest grundläggande tillämpningarna, d v s hur man räknar med ideal. Resultaten kommer vi att använda i nästföljande kapitel på mer komplexa problem.

Vi presenterar problemen som ”fyrstegsraketer”:

(1) Allmänn presentation av problemet,

(2) Formulera problemet som ett ”idealteoretiskt” problem.

(3) Presentera en metod för att lösa problemet

(4) Algoritm som löser problemet

(5) Implementation i Maple (se appendix B)

I det här kapitlet kommer vi att referera till funktionerna (algoritmerna) Reduce och Gbasis. Reduce ska vara en funktion som implementerar algoritmen Reduce i tabell 3.1. Gbasis ska vara en funktion som beräknar en monisk,

reducerad Gröbnerbas, *t ex* en implementation av algoritmen ImprovedGbasis i tabell 4.4.

5.1 Idealtillhörighet

Vi har tidigare sett i kapitel 3.4 att $\xrightarrow[G]{*}$ är en normal förenklare för restklassringen $K[X]/\langle G\rangle$, vi såg också att frågan om idealtillhörighet besvaras genom att se på ett polynoms normala form. Är den noll ligger polynomet i idealet. Eftersom vi enligt sats 4.28 kan beräkna en Gröbnerbas till vilket ideal (i polynomringen) som helst betyder det att vi alltid kan ta reda på om ett polynom ligger i ett ideal eller inte.

Metoden för att lösa problemet blir mycket enkel: Givet ett ideal I och ett polynom p, beräkna Gröbnerbasen till I och reducera p med denna bas. Kan vi reducera till noll ligger p i idealet, annars inte, se algoritm InIdeal i tabell 5.1.

Tabell 5.1: InIdeal

```
1.  procedure InIdeal(p, P)
2.      G ← Gbasis(P)
3.      h ← Reduce(p, G)
4.      if h = 0 then
5.          return(true)
6.      else
7.          return(false)
8.      end
9.  end
```

5.2 Kontrollera om två polynom ligger i samma restklass modulo ett ideal

På samma sätt som ovan såg vi i kapitel 3.4 att $\xrightarrow[G]{*}$ också är en kanonisk förenklare för restklassringen $K[X]/\langle G\rangle$. Vi såg också att frågan om restklasstillhörighet besvaras genom att se på ett polynoms kanoniska form. Har två polynom samma kanoniska form ligger de i samma restklass. Eftersom vi enligt

sats 4.28 kan beräkna en Gröbnerbas till vilket ideal (i polynomringen) som helst betyder det att vi alltid kan ta reda på om två polynom ligger i samma restklass.

Metoden för att lösa problemet blir mycket enkel: Givet ett ideal I och två polynom p och q, beräkna Gröbnerbasen till I och reducera p och q med denna bas. Resterna vi (eventuellt) får kvar blir våra (minimala) klassrepresentanter, se algoritm InSameClass i tabell 5.2.

Tabell 5.2: InSameClass

```
1.  procedure InSameClass(p, q, P)
2.      G ← Gbasis(P)
3.      p' ← Reduce(p, G)
4.      q' ← Reduce(q, G)
5.      if(p' = q') then
6.        return (true)
7.      else
8.        return (false)
9.      end
10. end
```

5.3 Räkna med restklasser

Vi har tidigare i sektion 2.4 sett hur vi kan se restklassringen modulo ett ideal, $K[X]/I$, som ett vektorrum. Vi såg också att i vissa fall var det lätt att plocka fram en bas för detta vektorrum, t ex så fick vi:

$$K[X_1, \ldots, X_n]/\langle X_1^{p_1}, \ldots, X_n^{p_n}\rangle = \{X_1^{r_1} \cdots X_n^{r_n} \mid 0 \leq r_i < p_i\}$$

Basen består av alla termer som är "mindre" än (i den mening att de inte delas av) idealets generatorer. Enligt korollarium 4.18 är idealet $\langle X_1^{p_1}, \ldots, X_n^{p_n}\rangle$ en Gröbnerbas och man frågar sig då om det alltid blir lika enkelt så fort man har en Gröbnerbas?

Lemma 5.1 *Låt G vara en Gröbnerbas till idealet $\langle P\rangle \subseteq K[X_1, \ldots, X_n]$ och låt H vara mängden*

$$H = \{\text{hterm}(g) \mid g \in G\} \ .$$

Då är $dim(\langle P\rangle) = 0$ *om och endast om det för alla* $1 \leqslant i \leqslant n$ *finns ett* $m \in \mathbb{N}$ *så att* $(x_i)^m \in H$.

Bevis: Se [BW93] T6.54. •

Sats 5.2 *Anta att* G *är en Gröbnerbas och definiera*

$$U = \{[u],\ \textit{där } u \in T(X) \textit{ så att } \neg\exists g \in G \textit{ med } \mathrm{hterm}(g) \mid u\}$$

där $[u]$ *är restklassen till* u *modulo* G. *Om* $dim(\langle G\rangle) = 0$ *är* U *en linjärt oberoende (vektorrums-) bas för* $K[X]/\langle G\rangle$.

Bevis: Anta att vi har att

$$a_1[u_1] + \cdots + a_m[u_m] = 0$$

där $a_i \in K$ och $u_i \in U$. Eftersom vi vet att för $p \in K[X]$ gäller att

$$[p] = 0 \Longleftrightarrow p \in \langle G\rangle$$

och att reduktion modulo G är en kanonisk förenklare måste det finnas ett polynom $g = a_1u_1 + \cdots + a_mu_m \in \langle G\rangle$. Men enda sättet att $\mathrm{Reduce}(g, G) = 0$ är att $a_1 = \cdots = a_m = 0$. •

Korollarium 5.3 *Om* $p(X) \in K[X]$ *är ett polynom i en variabel så är*

$$U = \{[X^i] \mid 0 \leqslant i < \ \textit{grad}(p)\}$$

en bas för $K[X]/\langle p\rangle$.

Bevis: p bildar enligt korollarium 4.17 en Gröbnerbas, resten följer av sats 5.2.
•

Korollarium 5.4 *Om* $P = \{p_1, \ldots, p_n\}$ *där* $p_i \in K[X_1, \ldots, X_n]$ *är monom så är*

$$U = \{[u] \mid u \in T(X_1, \ldots, X_n), \neg\exists p \in P \textit{ så att } p \mid u\}$$

en bas för $K[X_1, \ldots, X_n]/\langle P\rangle$.

Bevis: P bildar enligt korollarium 4.18 en Gröbnerbas, resten följer av sats 5.2.
•

Tabell 5.3: QuotientBase

1. **procedure** QuotientBase(P)
2. $G \leftarrow$ Gbasis(P)
3. $U \leftarrow \{u$ så att $u \in T[X]$ och $\neg\exists g \in G : \text{hterm}(g) \mid u\}$
4. **return**(U, G)
5. **end**

För att bestämma basen till restklassringen $K[X]/\langle P\rangle$ beräknar vi Gröbnerbasen till P och applicerar sats 5.2, se algoritm QuotientBase i tabell 5.3.

I sektion 2.4 fick vi problem med reduktionen när vi försökte räkna i restklassringen men eftersom vi med Gröbnerbaser har fått oss en kanonisk funktion möter vi nu inga problem. Proceduren blir helt analog med t ex heltalsräkning modulo 4: Givet två restklasser $[p]$ och $[q]$ samt ett ideal $\langle P\rangle$ som bestämmer restklasserna adderar eller multiplicerar vi (eller vad vi nu vill göra) representanter från restklasserna (t ex p och q) med varandra och reducerar resultatet modulo Gbasis(P). Eftersom reduktion modulo en Gröbnerbas är en kanonisk förenklare får vi som resultat en restklass (egentligen en minimal representant ur restklassen), se algoritm QuotientMult i tabell 5.4.

Tabell 5.4: QuotientMult

1. **procedure** QuotientMult(G, q_1, q_2)
2. # G = Gbasis(P) där P är idealet som bestämmer restklasserna.
3. **return**(Reduce($q_1 \cdot q_2, G$))
4. **end**

Exempel. Gröbnerbasen till (total gradordning med $x < y$)

$$F = \{x^3 + 7xy - 1, y^2x + y, x^2y + x\}$$

är

$$G = \{8y^2 - x, x^2 + 8y, xy + 1\}$$

och

$$\text{hterm}(G) = \{y^2, x^2, xy\} \ .$$

Vi får vidare att

$$U = \{1, x, y\} \ .$$

Vi kan nu ställa upp en multiplikationstabell för restklassringen:

$p \cdot q$ $\bmod \langle G \rangle$		$[q]$		
		1	x	y
	1	1	x	y
$[p]$	x	x	$-8y$	-1
	y	y	-1	$x/8$

•

Exempel. Gröbnerbasen till (total gradordning med $x < y < z$)

$$\{x^2 - y, z^2 - \frac{x^2}{2} - \frac{y^2}{2}, xz - 2\,z + xy\}$$

är

$$\begin{aligned} G = \{&y^2 - 2z^2 + y, 2xy + yz + 2z^2 - y - 4z, x^2 - y, \\ &2xz - yz - 2z^2 + y, yz^2 + 6yz - 18z^2 + 3y + 8z, \\ &z^3 - 5yz + 12z^2 - 2y - 6z\} \end{aligned}$$

och

$$\text{hterm}(G) = \{y^2, xy, x^2, xz, yz^2, z^3\} \ .$$

Vi får att

$$U = \{1, x, y, z, yz, z^2\}$$

och i tabell 5.5 finns en multiplikationstabell för restklassringen. •

För fortsatt läsning, se [BW93] kapitel 9.

$p \cdot q$ mod$\langle G\rangle$		$[q]$					
		1	x	y	z	yz	z^2
$[p]$	1	1	x	y	z	yz	z^2
	x	x	y	$-\frac{yz}{2} - z^2 + \frac{y}{2} + 2z$	$\frac{yz}{2} + z^2 - \frac{y}{2}$	$-\frac{3yz}{2} + 5z^2 - \frac{y}{2} - 2z$	$\frac{3yz}{2} - 3z^2 + \frac{y}{2} + 2z$
	y	y	$-\frac{yz}{2} - z^2 + \frac{y}{2} + 2z$	$2z^2 - y$	yz	$9yz - 24z^2 + 4y + 12z$	$-6yz + 18z^2 - 3y - 8z$
	z	z	$\frac{yz}{2} + z^2 - \frac{y}{2}$	yz	z^2	$-6yz + 18z^2 - 3y - 8z$	$5yz - 12z^2 + 2y + 6z$
	yz	yz	$-\frac{3yz}{2} + 5z^2 - \frac{y}{2} - 2z$	$9yz - 24z^2 + 4y + 12z$	$-6yz + 18z^2 - 3y - 8z$	$-170yz + 462z^2 - 75y - 216z$	$123yz - 332z^2 + 54y + 156z$
	z^2	z^2	$\frac{3yz}{2} - 3z^2 + \frac{y}{2} + 2z$	$-6yz + 18z^2 - 3y - 8z$	$5yz - 12z^2 + 2y + 6z$	$123yz - 332z^2 + 54y + 156z$	$-88yz + 240z^2 - 39y - 112z$

Tabell 5.5: Multiplikationstabell för restklassringen $K[X]/\langle G\rangle$, $G = \{\mathbf{y^2} - 2z^2 + y,\ 2\mathbf{xy} + yz + 2z^2 - y - 4z,\ \mathbf{x^2} - y,\ 2\mathbf{xz} - yz - 2z^2 + y,\ \mathbf{yz^2} + 6yz - 18z^2 + 3y + 8z,\ \mathbf{z^3} - 5yz + 12z^2 - 2y - 6z\}$ (ledande termer i fetstil)

5.4 Delideal

Hur vi kan avgöra om ett ideal är en delmängd av ett annat ideal framgår av följande sats:

Sats 5.5 $\langle F_1 \rangle \subseteq \langle F_2 \rangle \Leftrightarrow \forall f \in F_1 : f \xrightarrow[\text{Gbasis}(F_2)]{*} 0$

Bevis: $\Rightarrow$: $f \in F_1 \Rightarrow f \in \langle F_1 \rangle \subseteq \langle F_2 \rangle \Leftrightarrow f \xrightarrow[\text{Gbasis}(F_2)]{*} 0$

$\Leftarrow$: $f \xrightarrow[\text{Gbasis}(F_2)]{*} 0 \Leftrightarrow f \in \langle F_2 \rangle$ •

En algoritm som löser problemet finns i tabell 5.6.

Tabell 5.6: SubIdeal

```
1.  procedure SubIdeal(F1, F2)
2.      G ← Gbasis(F2)
3.      forall f ∈ F1 do begin
4.          if Reduce(f, G) ≠ 0 then
5.              return(false)
6.          end
7.      end
8.      return(true)
9.  end
```

5.5 Eliminationsideal

Att beräkna ett eliminationsideal syftar till att beräkna en del av ett ideal som beror endast av en delmängd av de ursprungliga variablerna. Om vi har ett ideal $I \subseteq K[X_1, \ldots, X_5]$ kan vi t ex beräkna den del som beror endast på variablerna $U = \{X_1, X_4\} \subset \{X_1, \ldots, X_5\}$. Denna del skriver vi som $I_U = I \cap K[U]$.

Vad menas med detta? I fallet med linjära ekvationssystem kan vi genom ett

basbyte genomföra en diagonalisering av ett ekvationssystem som kan se ut som

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 7 \\ x_2 - 7x_3 &= 1 \\ 8x_3 &= 2 \end{aligned}$$

Eliminationsideal är en teknik att åstadkomma diagonalisering av icke-linjära ekvationssystem (d v s polynom i flera variabler som ej behöver vara linjära).

För att beräkna ett eliminationsideal behöver vi en för varje tillfälle specifik termordning som uppfyller att $U \ll X \setminus U$ och med det menas att om $s \in K[U]$ och $t \in K[X \setminus U]$ gäller alltid att $s < t$. En termordning som uppfyller detta krav är den lexiografiska ordningen. Om I är ett ideal i $K[X_1, \ldots, X_5]$ och vi vill beräkna eliminationsidealet $I_{\{X_1,X_4\}}$ kan vi t ex välja den lexiografiska termordningen med $X_1 < X_4 < X_2 < X_3 < X_5$.

Det visar sig att beräkningen av eliminationsidealet blir väldigt enkel om vi först beräknar dess Gröbnerbas. Om vi har att G är en Gröbnerbas till idealet I kan vi istället för att beräkna snittet $I_U = \langle G \rangle_U = I \cap K[U]$ 'flytta ut" snittet till idealens generatorer, d v s

$$I \cap K[U] = \langle G \rangle \cap K[U] = \langle G \cap K[U] \rangle \qquad (*)$$

och det sista ledet i $(*)$ är mycket enkelt att beräkna. För att visa detta behöver vi först ett lemma:

Lemma 5.6 *Anta att $\{U_1, \ldots, U_r\} \subseteq \{X_1, \ldots, X_n\}$ och $\leqslant$ är en termordning som uppfyller att $U \ll X \setminus U$. Då gäller följande:*

(i) Om $s \in T(X)$ och $t \in T(U)$ så att $s < t$ gäller att $s \in T(U)$.

(ii) Om $f \in K[U]$ och $p, g \in K[X]$ så att $f \xrightarrow[p]{} g$ gäller att $p \in K[U]$ och $g \in K[U]$.

Bevis: (i) Om $s \in T(X)$ men $s \notin T(U)$ innehåller s någon variabel ur $X \setminus U$. Skriv s som $s = uv$ där $u \in T(U)$ och $v \in T(X \setminus U)$. Eftersom $s \notin T(U)$ gäller det att $v \neq 1$. Vi får $uv = s < t < v$, en motsägelse ty $uv < v \Rightarrow u = 1$.

(ii) p reducerar f och $\mathrm{hterm}(p)$ delar därigenom någon term t i f. Alla termer i f ligger i $K[U]$ och eftersom $\mathrm{hterm}(p) \in K[U]$ måste enligt del (i) alla andra termer i p ligga i $K[U]$ och därmed $p \in K[U]$. Reduktion definieras som att en av termerna i f reduceras bort och eftersom både $f \in K[U]$ och $p \in K[U]$ följer att $g \in K[U]$. •

Proposition 5.7 *Låt I vara ett ideal i $K[X]$ och $\{U\} = \{U_1, \ldots, U_r\}$ en delmängd av $\{X\} = \{X_1, \ldots, X_n\}$. Anta vidare att $\leqslant$ är en termordning på $T(X)$ sådan att $U \ll X \setminus U$ och G en Gröbnerbas till I med avseende på $\leqslant$. Då är $G \cap K[U]$ en Gröbnerbas till eliminationsidealet I_U.*

Bevis: Låt $0 \neq f \in I_U$. Då gäller att $f \in I$ och att $f \underset{g}{\longrightarrow} h$ för något $g \in G$ (G är en Gröbnerbas för I). Eftersom $f \in I_U \subset K[U]$ följer av lemma 5.6 även att $g \in K[U]$. $g \in G$ och $g \in K[U]$ är detsamma som att $g \in G \cap K[U]$ och f är alltså reducibelt modulo $G \cap K[U]$. •

Korollarium 5.8 *Anta att K är beräkningsbar, låt F vara en delmängd av $K[X]$ och låt $\{U\} = \{U_1, \ldots, U_r\} \subseteq \{X_1, \ldots, X_n\} = \{X\}$. Då beräknar algoritmen Elimination i tabell 5.7 en Gröbnerbas till eliminationsidealet $\langle F \rangle_U$.*

Exempel. Gröbnerbasen till

$$\{x^2y + 1, xy^2 - 2, x^3z^2 + 3y\}$$

beräknad med lexiografisk ordnig och $z < y < x$ är

$$\{12x - z^2, z^2 + 6y, -864 + z^6\} \ .$$

Enligt proposition 5.7 har vi att $I \cap K[z] = \{z^6 - 864\}$, $I \cap K[y, z] = \{z^2 + 6y, z^6 - 864\}$ och $I \cap K[x, y, z] = \{12x - z^2, z^2 + 6y, z^6 - 864\}$. Eftersom den inbördes ordningen mellan variablerna kan väljas helt fritt kan vi eliminera bort vilka variabler som helst. Vill vi t ex beräkna $I \cap K[x, y]$ väljer vi den lexiografiska ordningen med $x < y < z$. •

Eftersom lösning av ett diagonaliserat linjärt ekvationssystem är mycket enkel kan man fråga sig om det blir lika enkelt i det icke-linjära fallet, och som vi ska se i kapitel 6 blir det också så. Nästan i alla fall, istället för en linjär ekvation i en variabel får man ett polynom i en variabel som man sedan måste bestämma nollställen till o s v.

5.6 Äkta delmängd

Givet en mängd polynom $P \subseteq K[X]$, kan vi avgöra om $\langle P \rangle \neq K[X]$ (alltså $\langle P \rangle$ är en äkta delmängd) eller om $\langle P \rangle = K[X]$?

Tabell 5.7: Elimination

```
1.  procedure Elimination(F, U_1, ..., U_r)
2.      set termorder (U, U \ X) : U ≪ U \ X
3.      G ← Gbasis(F)
4.      return(G ∩ K[U])
5.  end
```

Lemma 5.9 *Låt I vara ett ideal till $K[X]$. Då gäller att*

$$I = K[X] \Longleftrightarrow K \subseteq I \ .$$

Bevis: $\Rightarrow$: Följer direkt av att $K \subseteq K[X]$.
$\Leftarrow$: Eftersom $K \subseteq I$ och K är en kropp har vi att $1 \in I$. Definitionen av ideal säger att $aI \subseteq I$ för alla $a \in K[X]$ och eftersom $1 \in I$ har vi att $a \in I$ för alla $a \in K[X]$, d v s att $K[X] \subseteq I$ och eftersom ett ideal till en ring inte kan vara större än ringen har vi att $I = K[X]$. •

Sats 5.10 *Låt I vara ett ideal till $K[X]$, G en Gröbnerbas till I (vilket innebär att $I = \langle G \rangle$). Då gäller att*

$$I = K[X] \Longleftrightarrow G \textit{ innehåller en konstant} \ .$$

Bevis: Enligt proposition 5.7 med $U = \{\}$ har vi att $I \cap K = \langle G \cap K \rangle$.
$\Rightarrow$: Lemma 5.9 ger att $K \subseteq I$. För $a \in K$ gäller att $a \in I$ och eftersom $I = \langle G \rangle$ finns det ett $b \in K[X]$ och ett $g \in G$ så att $bg = a$. Eftersom $a \in K$ (är en konstant) måsste också g vara en konstant.
$\Leftarrow$: Låt $a \in K \cap G$. Vi får då att $K \subseteq \{ap \mid p \in K[X]\} \subseteq I$. Lemma 5.9 ger att $I = K[X]$. •

För att bestämma om ett ideal är en äkta delmängd eller inte beräknar vi först idealets Gröbnerbas. Om Gröbnerbasen innehåller en konstant säg a har vi att $a \in K$ och $\{a\} \cap K \neq \emptyset$, se algoritmen Proper i tabell 5.8.

5.7 Snitt av ideal

Givet ett antal ideal $I_1, \ldots, I_m$ där $I_i = \langle F_i \rangle$, $F_i \subseteq K[X_1, \ldots, X_n]$, kan vi beräkna snittet $\cap_{i=1}^{m} I_i$? För att kunna göra det måste vi införa nya variabler, en

Tabell 5.8: Proper

```
1.  procedure Proper(F)
2.      G ← Gbasis(F)
3.      if(G ∩ K = ∅) then
4.        return(true)
5.      else
6.        return(false)
7.      end
8.  end
```

för varje ideal I_i. Vi kallar dessa variabler $Y_1, \ldots, Y_m$.

Proposition 5.11 *Låt $I_1, \ldots, I_m$ vara ideal till $K[X_1, \ldots, X_n]$ och låt*

$$J = \langle \{1 - (Y_1 + \cdots + Y_m)\} \cup \bigcup_{i=1}^{m} Y_i I_i \rangle$$

i ringen $K[X_1, \ldots, X_n, Y_1, \ldots, Y_m]$. Då är $\cap_{i=1}^{m} I_i$ lika med eliminationsidealet J_X.

Bevis: Se [BW93] P6.19. •

Korollarium 5.12 *Anta att K är beräkningsbar och låt $F_1, \ldots, F_m$ vara ändliga delmängder av $K[X]$. Då beräknar algoritmen Intersection i tabell 5.9 en Gröbnerbas till idealet $\cap_{i=1}^{m} \langle F_i \rangle$ i $K[X]$.*

Tabell 5.9: Intersection

```
1.  procedure Intersection(F_1, ..., F_m)
2.      G ← Elimination({1 − Σ_{i=1}^{m} Y_i} ∪ ⋃_{i=1}^{m} Y_i F_i, X)
3.      return(G)
4.  end
```

Anmärkning. Eftersom algoritmen Elimination anropas kommer vi där att behöva en termordning $\ll$ som uppfyller $\{X_1, \ldots, X_n\} \ll \{Y_1, \ldots, Y_m\}$, d v s för $s \in K[X_1, \ldots, X_n]$ och $t \in K[Y_1, \ldots, Y_m]$ gäller att $s < t$. Detta uppfylls som vi tidigare sagt av den lexiografiska ordningen med $X_1 < \cdots < X_n < Y_1 < \cdots Y_m$.

5.8 Snitt av restklasser

Givet ideal $I_1, \ldots, I_m$ i $K[X]$ och polynom $f_1, \ldots, f_m$ i $K[X]$, forma restklasserna $f_i + I_i$, $1 \leqslant i \leqslant m$ ($f_i + I_i$ är namnet på den restklass modulo I_i som innehåller f_i). Vi vill svara på följande frågor:

(a) Finns det något polynom i snittet $\cap_{i=1}^m (f_i + I_i)$?

(b) Ligger ett givet $g \in K[X]$ i snittet $\cap_{i=1}^m (f_i + I_i)$?

Låt $Y = Y_1, \ldots, Y_m$ vara nya variabler. Bilda

$$J = \langle \bigcup_{i=1}^{m} Y_i I_i \cup \{1 - (Y_1 + \cdots + Y_m)\} \rangle$$

och

$$f^* = Y_1 f_1 + \cdots Y_m f_m \in K[X, Y] \ .$$

Vi säger också att

$$A_f = \cap_{i=1}^m (f_i + I_i)$$

Sats 5.13 (Kinesiska restsatsen) *Låt $\leqslant$ vara en termordning på $T(X, Y)$ som uppfyller $X \ll Y$, låt G vara en Gröbnerbas till idealet J i $K[X, Y]$ med avseende på $\leqslant$ och låt h vara den unika normala formen av f^* modulo G. Då är följande villkor ekvivalenta:*

(i) $A_f \neq \emptyset$.

(ii) $h \in K[X]$.

(iii) $h \in A_f$.

Dessutom är $A_f = h + \cap_{k=1}^m I_k$ där h är minimal i A_f med avseende på ordningen på $K[X]$ inducerad av $\leqslant$ och för alla $g \in K[X]$ har vi att $g \in A_f$ om och endast om h är den normala formen av g modulo Gröbnerbasen $G \cap K[X]$.

Bevis: Se [BW93] T6.23. •

Korollarium 5.14 *Anta att K är beräkningsbar och låt $\leqslant$ vara en bestämbar termordning på $T(X)$. Om $F_1, \ldots, F_m$ är ändliga delmängder av $K[X]$ och $f_1, \ldots, f_m \in K[X]$ så avgör algoritmen CRT i tabell 5.10 om*

$$\cap_{i=1}^{m} f_i + \langle F_i \rangle$$

är tom, och om inte beräknar CRT ett element i snittet som är minimalt med avseende på ordningen inducerad av $\leqslant$.

Tabell 5.10: CRT

```
1.  procedure CRT(F_1, ..., F_m, f_1, ..., f_m)
2.      G ← Gbasis({∑_{i=1}^m Y_i I_i} ∪ {1 − Y_1 − ··· − Y_m})
3.      f ← ∑_{i=1}^m
4.      h ← Reduce(f, G)
5.      if h ∈ K[X] then
6.        return(true, h)
7.      else
8.        return(false)
9.      end
10. end
```

Korollarium 5.15 *Anta att K är beräkningsbar och låt $\leqslant$ vara en bestämbar termordning på $T(X)$. Om $F_1, \ldots, F_m$ är ändliga delmängder av $K[X]$ och $g, f_1, \ldots, f_m \in K[X]$ så avgör algoritmen CRT-in i tabell 5.11 om*

$$g \in \cap_{i=1}^{m} f_i + \langle F_i \rangle \ .$$

5.9 Sammanfattning

Vi har i det här kapitlet tittat en del på de mest grundläggande tillämpningarna av Gröbnerbaser.

Tabell 5.11: CRT-in

```
1.  procedure CRT-in(g, F_1, ... , F_m, f_1, ... , f_m)
2.      G ← Gbasis({∑_{i=1}^m Y_i I_i} ∪ {1 − Y_1 − ··· − Y_m})
3.      f ← ∑_{i=1}^m
4.      h ← Reduce(f, G)
4.      G' ← G ∩ K[X]
4.      h' ← Reduce(g, G')
5.      if h = h' then
6.        return(true)
7.      else
8.        return(false)
9.      end
10. end
```

Tillämpningar och algoritmer är i stort hämtade från [BW93] och där finns ännu fler tillämpningar inom idealteori beskrivna som beräkning av syzygies, idealkvoter och räkning med radikalideal.

I nästa kapitel ska vi se på en mer komplex tillämpning: Lösning av system av icke-linjära polynomekvationer i flera variabler.

Kapitel 6

Ekvationslösning

I det här kapitlet ska vi se på en tillämpning av Gröbnerbaser på ett mer omfattande problem, närmare bestämt lösning av ekvationssystem bestående av icke-linjära polynomekvationer. Linjära ekvationssystem kan enkelt lösas genom en *diagonalisering* av systemet och vi ska här visa att dessa ideer utan problem kan överföras på icke-linjära fallet, proceduren blir dock lite mer komplicerad.

Eftersom resultaten i det här kapitlet till stor del bygger på mer avancerad matematik än vad som ryms inom detta kompendiums ramar kommer vi till skillnad från föregående kapitel inte att presentera några bevis, den som önskar belägg för påståenden och bevis till satser får istället konsultera [GCL92] och [BW93].

6.1 Problemet

Vi vill lösa ett ekvationssystem av typen

$$\begin{cases} f_1 = 0 \\ \quad \vdots \\ f_m = 0 \end{cases} \tag{6.1}$$

där $f_i \in K[X_1, \ldots, X_n]$ och med lösa menar vi hitta alla $a = (a_1, \ldots, a_n) \in K^n$ där $f_1(a) = \cdots = f_m(a) = 0$.

Exempel. Två ekvationssystem som upfyller villkoret i ekvation (6.1) är

$$\begin{cases} xy + 1 = 0 \\ x^2 + x = 0 \end{cases} \tag{6.2}$$

och

$$\begin{cases} zxy + 8z^2y = 0 \\ x^2y^2 - 7z = 0 \\ x^3z - 2x - 2y + 1 = 0 \end{cases} \tag{6.3}$$

Båda dessa är, som vi ska se senare, lösbara och har ett ändligt antal lösningar. •

6.2 Formulering som problem i idealteori

För att kunna använda teknikerna med Gröbnerbaser måste vi formulera om problemet till "rätt språk", d v s vi måste uttrycka problemet i form av ideal till en polynomring.

Om vi vill söka nollställen till ett ekvationssystem av typen (6.1) kan vi enligt sats 2.10 som säger att

$$\mathrm{Z}(f_1, \ldots, f_m) = \mathrm{Z}(\langle f_1, \ldots, f_m \rangle)$$

lika väl söka nollställen till det ideal som genereras av de enskilda polynomen $f_1, \ldots, f_m$. Sats 2.11 säger oss vidare att eftersom

$$\langle f_1, \ldots, f_m \rangle = \langle \{\mathrm{Gbasis}(f_1, \ldots, f_m)\} \rangle$$

har idealet genererat av de enskilda polynomen samma nollställen som idealet genererat av Gröbnerbasen till samma polynom och om vi använder sats 2.10 igen ser vi att alla nollställen till det ursprungliga ekvationssystemet finns bevarade i generatorerna till Gröbnerbasen, d v s

$$\mathrm{Z}(f_1, \ldots, f_m) = \mathrm{Z}(\mathrm{Gbasis}(f_1, \ldots, f_m)) \ .$$

Sats 6.1 *Låt G vara en Gröbnerbas till $\langle P \rangle \subseteq K[X_1, \ldots, X_n]$. Om $dim(\langle P \rangle) = 0$ finns det för varje $1 \leqslant i \leqslant n$ ett polynom p_i i endast variabeln X_i som innehåller alla nollställen m a p den variabeln.*

Bevis: Se [BW93] lemma 6.50. •

Vi kan göra om samma konstruktion för vilken variabel $X_1, \ldots, X_n$ som helst så vi vet att det till varje variabel finns ett polynom i $\langle G \rangle$ som innehåller alla nollställen m a p den variabeln. Hur vi beräknar dessa polynom ska vi återkomma till.

6.3 Lösningsmetod och algoritmer

Från linjär algebra vet vi att vi kan, förutsatt att "konstantmatrisen" ej är singulär, med ett basbyte omvandla ekvationssystemet

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}$$

till

$$\begin{pmatrix} b_{11} & b_{12} & b_{13} \\ 0 & b_{22} & b_{23} \\ 0 & 0 & b_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

Här har vi nu fått en ekvation i endast x_3

$$b_{33}x_3 = d_3$$

och en ekvation i x_2 och x_3

$$b_{22}x_2 + b_{23}x_3 = d_2$$

men eftersom x_3 redan är känd (genom förra ekvationen) blir detta en ekvation i endast *en ny* obekant. Man kan se detta som att vi tar fram en ekvation först för x_3, löser den, stoppar in denna lösning i ekvation två, löser ut x_2 (x_3 är ju redan bestämd), stoppar in lösningen till x_2 och x_3 i första ekvationen och löser ut x_1.

I fallet med ickelinjära ekvationssystem kan vi numera göra på samma sätt. Hur vi bär oss åt för att ta fram en ekvation för en av de obekanta beror på vilken termordning vi använder. I fallet med den lexiografiska ordningen blir det enkelt eftersom vi kan använda eliminationsideal. I fallet med den totala gradordningen blir det dock lite krångligare, mer om detta senare. När vi nu kan ha ickelinjära ekvationssystem kan vi för en variabel inte bara få en utan

flera lösningar. Algoritmen måste då "grena upp" sig och undersöka resten av systemet en gång för varje möjlig lösning.

Innan vi ger oss på att lösa ett ekvationssystem måste vi försäkra oss om två saker: (1) Att ekvationssystemet är lösbart och (2) att det bara har ett ändligt antal lösningar.

Sats 6.2 *Låt G vara en monisk Gröbnerbas till $\langle P\rangle = \langle p_1, \ldots, p_m\rangle \subseteq K[X]$. Då är P (sett som ett ekvationssystem enligt 6.1) lösbart om och endast om $1 \notin G$.*

Anmärkning. Om G dessutom är en *reducerad* Gröbnerbas blir villkoret istället $\{1\} \neq G$.

Sats 6.3 *Låt G vara en Gröbnerbas till $\langle P\rangle = \langle p_1, \ldots, p_m\rangle \subseteq K[X]$. Då har ekvationssystemet associerat med P ändligt många lösningar om och endast om $\dim(\langle P\rangle) = 0$.*

Bevis: Se [BW93]. •

Exempel. Till ekvationssystemet (som uppenbarligen ej är lösbart)

$$\begin{cases} xy^2 = 0 \\ 1 + x = 0 \\ y + 1 = 0 \end{cases}$$

associerar vi idealet $F = \langle xy^2, 1+x, y+1\rangle$ och Gröbnerbasen till F (gradordning, $x < y < z$) är

$$G_1 = \{1\}$$

och vi ser direkt att systemet enligt sats 6.2 ej är lösbart. •

Exempel. Vi har att Gröbnerbasen till

$$\begin{cases} zy = 0 \\ x + y = 0 \\ zx = 0 \\ xy + z^2x = 0 \end{cases}$$

är (lexiografisk, $x < y < z$)

$$G_2 = \{x + y, y^2, zy\}\ .$$

Systemet är enligt sats 6.2 lösbart, men har det ändligt antal lösningar? En snabb titt på systemets Gröbnerbas ger att

$$\text{hterm}(G_2) = \{x, y, zy\}$$

och systemet har enligt sats 6.3 oändligt många lösningar. •

Exempel. Gröbnerbasen till ekvation (6.2) är (gradordning, $x < y < z$)

$$G_3 = \{-1 + y, 1 + x\}$$

vilket är lösbart och har ändligt många lösningar. •

Exempel. Gröbnerbasen till ekvation (6.3) är (lexiografisk ordning, $x < y < z$)

$$\begin{aligned} G_4 = &16x + 2zy + 16y - 16z^2 + 4096z^4 + 512z^5 - z - 8, \\ &4y^4 - 4y^3 + y^2 - 28672z^7 + 1792z^4 - 28z, \\ &2y^2z - zy - 128z^3 + 8192z^6 - 8z^2 - 131072z^9 + 256z^5, \\ &-16z^3 + 512z^6 + 2z^2y - z^2, \\ &512z^4 - 16384z^7 + 16z^3 + 4096z^5 - 262144z^8 + 4194304z^{11} - 7z^2 \end{aligned}$$

så systemet är lösbart. Ur

$$\text{hterm}(G_4) = \{x, y^4, y^2z, z^2y, z^{11}\}$$

ser vi att det också bara har ett ändligt antal nollställen. •

(a) Termordning som uppfyller $U \ll U \setminus X$

För att analogt med det linjära fallet få en ekvation i endast *en* variabel, t ex x_3, använder vi den lexiografiska termordningen med $x_3 < x_2 < x_1$ och beräknar eliminationsidealet $I_{\{X_3\}}$. Vi får då ett ideal som genereras av ett polynom i variabeln x_3 och detta polynom är det polynom p_i som beskrevs i sats 6.1.

Vi bestämmer nollställen för detta polynom på "vanligt" sätt. Nästa steg blir att stoppa in dessa nollställen, ett i taget, i ekvationssystemet och för varje nollställe beräkna eliminationsidealet $I_{\{X_3,X_2\}}$ och ur detta beräkna (eventuella) lösningar till X_2. Fortsätt på samma sätt för resterande obekanta.

Tillhörande algoritm finns i tabell 6.1.

Tabell 6.1: SolveLex

```
1.  procedure SolveLex(P)
2.      G ← Gbasis(P)
3.      roots ← ∅
4.      p ← selectpoly(Elimination(G, X_n))
5.      roots ← roots ∪ {(α) | p(α) = 0}
6.      for k from n − 1 by −1 to 1 do {
7.        S ← ∅
8.        G_k ← G ∩ K[X_k, ... , X_n] − K[X_{k+1}, ... , X_n]
9.        foreach (α_{k+1}, ... , α_n) ∈ roots do
10.         G' ← {g(X_k, α_{k+1}, ... , α_n) | g ∈ G_k}
11.         G' ← Gbasis(G')
12.         p ← selectpoly(G' ∩ K[X_k])
13.         if p ≠ 1 then
14.           S ← S ∪ {(α, α_{k+1}, ... , α_n) | p(α) = 0}
15.         end
16.       end
17.       roots ← S
18.     end
19.     return(roots)
20. end
```

(b) Allmänna fallet

I fallet när vår termordning *ej* uppfyller $U \ll U \setminus X$ måste vi ersätta eliminationsidealet med något annat.

För att beräkna det polynom p_i som nämns i sats 6.1 kan vi använda algoritmen i tabell 6.2. I övrigt är strategin för att lösa hela ekvationssystemet helt analog med fallet för den lexiografiska ordningen. En komplett algoritm finns i tabell 6.3.

Tabell 6.2: SolveOneDeg

```
1.  procedure SolveOneDeg(P, x)
2.      G ← Gbasis(P)
3.      k ← 0
4.      do {
5.          p_k ← Reduce(x^k, G)
6.          if ∃(a_0, ..., a_k) ≠ (0, ..., 0) så att Σ_{j=0}^{k} a_j p_j = 0 then
7.              return(a_k^{-1} · Σ_{j=0}^{k} a_j x^j)
8.          else
9.              k ← k + 1
10.         end
11.     end
12. end
```

6.4 Sammanfattning

Att lösa ekvationssystem med Gröbnerbaser är utan tvekan en mycket kraftfull metod.

Några av dess fördelar: Den är, bara systemet är lösbart och har ändligt antal lösningar, alltid genomförbar och ger möjlighet att beräkna exakta nollställen. I fallet med oändligt många lösningar kan man ta fram parameterlösningar.

Den har också nackdelar: Algoritmen för beräkning av Gröbnerbaser plågas som vi sett tidigare av hög tidskomplexitet. Vi kan också få envariabelpolynom av hög grad som kan vara mycket svåra att lösa.

Tabell 6.3: SolveDeg

```
1.  procedure SolveDeg(P, x)
2.      Q ← {[P, ()]}
3.      for k from n by -1 to 1 do {
4.          S ← ∅
5.          foreach [G, (α_{k+1}, ... , α_n)] ∈ Q do  {
6.              G' ← {g(x_1, ... , x_k, α_{k+1}, ... , α_n) | g ∈ G}
7.              G' ← Gbasis(G')
8.              p ← SolveOneDeg(G', x_k)
9.              if p ≠ 1 then
10.                 S ← s ∪ {[G', (α, α_{k+1}, ... , α_n)] | p(α) = 0}
11.             end
12.         end
13.         Q ← S
14.     end
15.     roots ← ∅
16.     foreach [G, (α_1, ... , α_n)] ∈ Q do {
17.         roots ← roots ∪ {(α_1, ... , α_n)}
18.     end
19.     return(roots)
20. end
```

Lämpliga (allmänna) källor för ytterligare information är [GCL92], [BW93] och [Buc85]. I [Cza89] diskuteras en variant på Buchbergers algoritm som är bättre lämpad för ekvationslösning.

Bilagor

Bilaga A

Gröbnerbaser på WWW

Datoralgebrans hemsida **CAIN** (Computer Algebra Information Network) på

`http://math-www.uni-paderborn.de/CAIN/`

samt, naturligtvis, **RISC** (Research Institute for Symbolic Computation) som drivs av upphovsmannen bakom Gröbnerbaser Professor Bruno Buchberger på

`http://www.risc.uni-linz.ac.at`

Ett urval av program som stödjer Gröbnerbaser:

- **Maple** (kommersiellt)
  Package "grobner". Se "help browser" under "Mathematics/Algebra/ Polynomials/Grobner Bases".

- **MuPAD** (gratis)
  Multi Processing Algebra Tool
  Jämförbart med Maple.
  `http://math-www.uni-paderborn.de/MuPAD/`

- **SACLIB** (gratis)
  Symbolic Algebraic Computation LIBrary
  Ett C-bibliotek för symbolisk algebra.
  `http://www.can.nl/SystemsOverview/Special/Algebra/SACLIB.html`

- **Groebner** (gratis)
  Ett C-bibliotek för att beräkna Gröbnerbaser, bygger på SACLIB.
  `http://www.can.nl/SystemsOverview/Special/Algebra/GROEBNER/productinfo.html`
- **Macaulay** (gratis)
  Algebraisk geometri och datoralgebra. `http://www.math.uiuc.edu/Macaulay2/`
- **Bergman** (gratis)
  `http://www.can.nl/SystemsOverview/Special/Algebra/bergman.html`

Bilaga B

Implementation av algoritmer

I detta kapitel finns flertalet av de presenterade algoritmerna implementerade i Maple. Dessa implementationer gör inga som helst anspråk på att vara särskilt effektiva, deras enda syfte är att åskådliggöra hur en implementation *kan* se ut.

B.1 Algoritmer kapitel 3

```
GBvar:=[x,y,z];
GBorder:=tdeg;


###################
# Algoritm Reduce #
###################

###################
# leadmon

leadmon := proc(p)
  local t;

  t := grobner[leadmon](p, GBvar, GBorder);
```

```
  expand(t[1]*t[2])
end;

####################
# hterm

hterm := proc(p)
  local q;

  if p=0 then RETURN(0) fi;

  q := leadmon(p);
  q / lcoeff(q)
end;

####################
# divides

divides := proc(p, r)
  local t, q;

  # Only checks the term part
  # of r and p.

  if(not(type(p, monomial) and type(r, monomial))) then
    RETURN(false)
  fi;

  if (p=0 or r=0) then
    RETURN(false)
  fi;

  type(r/p, monomial)
end;

####################
# reducer_set
```

```
reducer_set := proc(p, Q)

  map( (x, htp) ->
         if(divides(hterm(x), htp)) then
           x
         fi,
       Q minus {0}, hterm(p))
end;

####################
# selectpoly

selectpoly := proc(S)

  # The most simple implementation of
  # selectpoly: just pick the first one.

  S[1]
end;

####################
# Reduce

Reduce := proc(p, Q)
  local r, q, R, f, k, L;

  r := p;
  q := 0;
  L := [];

  while(r<>0) do
    R := reducer_set(r, Q);
    while(nops(R)>0) do
      f := selectpoly(R);
      k := leadmon(r) / leadmon(f);
      r := expand(r - k*f);
      L := [ op(L), [k,f] ];
      R := reducer_set(r, Q);
```

```
    od;
    q := expand(q + leadmon(r));
    r := expand(r - leadmon(r));
  od;

  RETURN(q, L);
end;
```

```
######################
# Algoritm ReduceSet #
######################

####################
# list_minus

list_minus := proc(L, l)
  local LL;

  LL := [];

  for p in L do
    if not(member(p, l)) then
      LL := [op(LL), p]
    fi;
  od;

  RETURN(LL);

end;

####################
# ReduceSet

ReduceSet := proc(E)
  local R, P, h, Q, EE, s, k;

  R := E;
  P := {};

  while(nops(R)>0) do
    h := selectpoly(R);
    R := list_minus(R, {h});
    h := Reduce(h, P)[1];
    if h<>0 then
      Q := map( (q, hth) ->
                  if( divides(hth, hterm(q)) ) then
```

```
                  q
                fi,
              P, hterm(h));
      R := [op(R), op(Q)];
      P := (P minus Q) union {h};
    fi;
  od;

  EE := [];
  S := P;
  for h in P do
    h := Reduce(h, S minus {h})[1];
    k := lcoeff(leadmon(h));
    h := simplify(h/k);
    EE := [op(EE), h];
  od;

  RETURN(EE);
end;
```

B.2 Algoritmer kapitel 4

```
####################
# Algoritm Buchberg #
####################

###################
# selectpair

selectpair := proc(B, G)

  # The most simple implementation of
  # selectpair: just pick the first one.

  B[1]
end;

###################
# Spoly

Spoly := proc(p, q)

  simplify(lcm(leadmon(p), leadmon(q))
              *((p/leadmon(p))-(q/leadmon(q))))
end;

###################
# Gbasis

Gbasis := proc(P)
  local G, k, B, i, j, h, ij, sp;

  G := P;
  k := nops(G);

  B := map( (x) ->
              if(x[1]<x[2]) then
                x
```

```
            fi,
          {seq(seq([i,j], j=1..k), i=1..k)});

  while(nops(B)>0) do
    ij := selectpair(B, G);
    B := B minus {ij};
    sp := Spoly(G[ij[1]], G[ij[2]]);
    h := Reduce(sp, { op(G) })[1];
    if h<>0 then
      G := [ op(G), h ];
      k := k+1;
      B := B union { seq([i,k], i=1..(k-1))}
    fi;
  od;

  RETURN(G);
end;
```

```
############################
# Algoritm ReducedBuchberg #
############################

RGbasis := proc(P)
  local G, k, B, i, j, h, ij, sp;

  G := Gbasis(ReduceSet(P));

  R := map( (g,G) ->
              if(nops(reducer_set(g,G) minus {g})) > 0 then
                g
              fi,
            {op(G)}, {op(G)}
       );

  RETURN(ReduceSet({op(G)} minus R));
end;
```

```
############################
# Algoritm ImprovedBuchberg #
############################

crit1 := proc(i, j, G)

  if( not(simplify(lcm(hterm(G[i]),hterm(G[j])) ) =
          simplify(hterm(G[i]), hterm(G[j])) ) ) then
    true
  else
    false
  fi;
end;

crit2 := proc(i, j, B, GB)
  local S, n, t;

  S := map( (k) ->
              if not(i=k) and not(k=j)
              and divides(hterm(GB[k]), lcm(hterm(GB[i]),hterm(GB[j])))
              and not(member([i,k], B))
              and not(member([k,j], B)) then
                [i,j]
              fi,
            {seq(t, t=1..nops(GB))}
       );

  if nops(S)=0 then
    true
  else
    false
  fi;
end;

IGbasis := proc(P)
  local G, k, B, i, j, h, ij, sp;

  G := P;
```

```
  k := nops(G);

  B := map( (x) ->
              if(x[1]<x[2]) then
                x
              fi,
            {seq(seq([i,j], j=1..k), i=1..k)});

  while(nops(B)>0) do
    ij := selectpair(B, G);
    if crit1(ij.i, ij.j, B, G) and crit2(ij.i, ij.j, B, G) then
      B := B minus {ij};
      sp := Spoly(G[ij[1]], G[ij[2]]);
      h := Reduce(sp, { op(G) })[1];
      if h<>0 then
        G := [ op(G), h ];
        k := k+1;
        B := B union { seq([i,k], i=1..(k-1))}
      fi;
    fi;
  od;

  RETURN(G);
end;
```

```
RIGbasis := proc(P)
  local G, k, B, i, j, h, ij, sp;

  G := IGbasis(ReduceSet(P));

  R := map( (g,G) ->
              if(nops(reducer_set(g,G) minus {g})) > 0 then
                g
              fi,
            {op(G)}, {op(G)}
       );

  RETURN(ReduceSet({op(G)} minus R));
end;
```

B.3 Algoritmer kapitel 5

Här använder vi Maples inbyggda funktioner för beräkning av Gröbnerbaser. Funktionen

$$\text{grobner[gbasis]}(F, X, \text{termorder})$$

beräknar Gröbnerbasen till F. Termordningen kan antingen vara `plex` för lexiografisk ordning eller `tdeg` för gradordning. Den inbördes ordningen mellan variablerna sätts med X. För att reducera ett polynom skriver vi

$$\text{grobner[normalf]}(\text{poly}, F, X, \text{termorder})$$

där poly är polynomet vi vill reducera, F den Gröbenrbas vi vill reducera med, X och termorder som ovan.

```
GBvar:=[x,y,z];


###################
# CanonicalForm

CanonicalForm := proc(p, P)
  local G, h;

  G := grobner[gbasis](P, GBvar, tdeg);
  h := grobner[normalf](p, G, GBvar, tdeg);

  RETURN(h);
end;

###################
# InIdeal

InIdeal := proc(p, P)
  local G, h;

  if CanonicalForm(p, P)=0  then
    true
  else
```

```
    false
  fi;
end;
```

```
########################
# Algoritm InSameClass #
########################

####################
# InSameClass

InSameClass := proc(p, q, P)
  local G;

  G := grobner[gbasis](P, GBvar, tdeg);

  if grobner[normalf](p, G, GBvar, tdeg) =
     grobner[normalf](q, G, GBvar, tdeg) then
    true
  else
    false
  fi;
end;
```

```
#########################
# Algoritm QuotientBase #
#########################

####################
# QuotientBase

_combine:=proc(A,B)

  C:={};
  for i from 1 to nops(A) do
    for j from 1 to nops(B) do
      C:= [op(C), [op(A[i]), B[j] ] ];
    od;
  od;

  RETURN (C);
end;

_QuotientBase := proc(G, X, k)
  local R, T, t, i, j;

  HT := map( (x) -> hterm(x), G);
  t:=[seq(j, j=0..k[1]) ];
  for i from 2 to nops(k) do
    t := _combine(t, [seq( j , j =0..k[i])] );
  od;

  T:=[];
  for i from 1 to nops(t) do
    tt:=t[i];
    term:=1;
    for j from 1 to nops(X) do
        term:=term*X[j]^tt[j];
    od;
    T := [op(T), term];
  od;
```

```
  R := {};
  for i from 1 to nops(T) do
    f := 0;
    for j from 1 to nops(HT) do
      if divides(HT[j], T[i]) then
        f := 1;
      fi;
    od;
    if(f=0) then
      R := R union {T[i]};
    fi;
  od;

  RETURN(HT, R);
end;

QuotientBase := proc(P)
  local G, maxdeg;

  # maxdeg = maximum degree of each variable in GBvar.
  #          Here it is assumed that 10 is an upper limit.

  maxdeg := map( (var) ->
                   10,
                 GBvar);
  G := grobner[gbasis](P, GBvar, tdeg);
  print(G, maxdeg);
  _QuotientBase(G, GBvar, maxdeg);
end;
```

```
####################
# Algoritm Subset #
####################

Subset:=proc(F, P)
    local G, R;

    G:=grobner[gbasis](P, GBvar, tdeg);

    R:=map( (f, GB) ->
              if not(grobner[normalf](f, GB, GBvar, tdeg)=0)
              then f fi, F, G);

    if nops(R)=0 then
true
    else
false
    fi;
end;
```

Litteraturförteckning

[AL94] William W. Adams and Philippe Loustaunau. *An Introduction to Gröbner bases*. Number 3 in Graduate Studies in Mathematics. American Mathematical Society, 1994. ISBN 0-8218-3804-0.

[BL83] Bruno Buchberger and R. Loos. Algebraic simplification. In Bruno Buchberger, G. Collins, and R. Loos, editors, *Computer Algebra - Symbolic and Algebraic Computation*, pages 11–43. Springer Verlag, Wein - New York, second edition, 1983.

[Buc65] Bruno Buchberger. *Ein algoritmus zum auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. Doctoral dissertion, University of Innsbruk, 1965.

[Buc83] Bruno Buchberger. A note on the complexity of constructing Gröbner bases. In H van Hulzen, editor, *Proc. EUROCAL '83*, number 162 in Lecture Notes in Computer Science, pages 137–145. Springer Verlag, 1983.

[Buc85] Bruno Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional System Theory*, Multidimensional Systems Theory, chapter 6, pages 184–232. Reidel, 1985.

[BW79] Bruno Buchberger and F Winkler. Miscellaneous results on the construction of Gröbner bases for polynomial ideals. Technical Report 137, Univ. of Linz, Math. Inst., 1979.

[BW93] Thomas Becker and Volker Weispfenning. *Gröbner Bases*. Graduate Texts in Mathematics. Springer Verlag, New York, 1993. ISBN 0-387-97971-9.

[CGH89a] L Caniglia, A Galligo, and J Heintz. How to compute the projective closure of an affine algebraic variety in subexponential time. In *Proc. AAECC-7 (to appear)*, 1989.

[CGH89b] L Caniglia, A Galligo, and J Heintz. Some new effectivity bounds in computational geometry. In *Proc. AAECC-6*, number 357 in Lecture Notes in Computer Science, pages 131–152. Springer Verlag, 1989.

[Cza89] S. R. Czapor. Solving algebraic equations: Combining buchberger's algorithm with multivariate factorization. *Journal of Symbolic Computation*, (7):49–53, 1989.

[Fra89] John B. Fraleigh. *A First Course in Abstract Algebra*. World Student Series Edition. Addison-Wesley, fourth edition, 1989. ISBN 0-201-52821-5.

[GCL92] Keith O. Geddes, Stephen R. Czapor, and George Labahn. *Algorithms in computer algebra*. Kluwer Academic Publishers, 1992. ISBN 0-7923-9259-0.

[KB67] Donald E Knuth and Peter B Bendix. Simple word problems in universal algebras. In John Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Science Research Council Atlas Computer Laboratory, Pergamon Press, 1967.

[Lan93] Serge Lang. *Algebra*. Addison-Wesley, third edition, 1993. ISBN 0-201-55540-9.

[Laz83] D Lazard. Gröbner bases, gaussian elimination, and resolution of systems of algebraic equations. In H van Hulzen, editor, *Proc. EUROCAL '83*, number 162 in Lecture Notes in Computer Science, pages 146–156. Springer Verlag, 1983.

Notationer

Index

S

T

U

V