Appendix C

Computer Algebra Systems

This appendix will discuss several computer algebra systems that can be used in conjunction with the text. We will describe AXIOM, Maple, Mathematica and REDUCE in some detail, and then mention some other systems. These are all amazingly powerful programs, and our brief discussion will not do justice to their true capability.

It is important to note that we will *not* give a general introduction to any of the computer algebra systems we will discuss. This is the responsibility of your course instructor. In particular, we will assume that you already know the following:

- How to enter and exit the program, and how to enter commands and polynomials. Some systems require semicolons at the end of commands (such as Maple and REDUCE), while others do not. Also, some systems (such as Mathematica) are case sensitive, while others are not. Some systems require an asterisk for multiplication (such as AXIOM), while others do not.
- How to refer to previous commands, and how to save results in a file. The latter can be important, especially when an answer fills more than one computer screen. You should be able to save the answer in a file and print it out for further study.
- How to work with lists. For example, in the Groebner basis command, the input contains a list of polynomials, and the output is another list which is a Groebner basis for the ideal generated by the polynomials in the input list. You should be able to find the length of a list and extract polynomials from a list.
- How to assign symbolic names to objects. In many computations, the best way to deal with complicated data is to use symbolic names for polynomials, lists of polynomials, lists of variables, etc.

If a course being taught from this book has a laboratory component, we would suggest that the instructor use the first lab meeting to cover the above aspects of the particular computer algebra system being used.

§1 AXIOM

AXIOM is a version of SCRATCHPAD, which was developed by IBM over a period of many years. AXIOM is now freely available from `http://www.nongnu.org/axiom`. Our discussion applies to version 2.0. For us, the most important AXIOM commands

are `normalForm`, for doing the division algorithm, and `groebner`, for computing a Groebner basis.

A distinctive feature of AXIOM is that every object has a specific type. In particular, this affects the way AXIOM works with monomial orders: an order is encoded in a special kind of type. For example, suppose we want to use lex order on $\mathbb{Q}[x, y, z]$ with $x > y > z$. This is done by using the type `DMP([x,y,z],FRAC INT)` (remember that AXIOM encloses a list inside brackets [...]). Here, `DMP` stands for "Distributed Multivariate Polynomial," and `FRAC INT` means fractions of integers, i.e., rational numbers. Similarly, grevlex for $\mathbb{Q}[x, y, z]$ with $x > y > z$ means using the type `HDMP([x,y,z],FRAC INT)`, where `HDMP` stands for "Homogeneous Distributed Multivariate Polynomial." At the end of the section, we will explain how to get AXIOM to work with grlex order.

To see how this works in practice, we will divide $x^3 + 3y^2$ by $x^2 + y$ and $x + 2xy$ using grevlex order with $x > y$. We first give the three polynomials names and declare their types:

```
-> f : HDMP([x,y],FRAC INT) := x^3+3*y^2
-> g : HDMP([x,y],FRAC INT) := x^2+y
-> h : HDMP([x,y],FRAC INT) := x+2*x*y
```

(Here, `->` is the AXIOM prompt, and the colon : indicates a type declaration. You can save typing by giving `HDMP([x,y],FRAC INT)` a symbolic name.) Then the remainder is computed by the command:

```
-> normalForm(f,[g,h])
```

The output is the remainder of `f` on division by `g,h`. In general, the syntax for this command is:

```
-> normalForm(poly,polylist)
```

where `poly` is the polynomial to be divided by the polynomials in the list `polylist` (assuming that everything has been declared to be of the appropriate type).

To do the same computation using lex order with $x > y$, first issue the command:

```
-> Lex := DMP([x,y],FRAC INT)
```

to give `DMP([x,y],FRAC INT)` the symbolic name Lex, and then type:

```
-> normalForm(f::Lex,[g::Lex,h::Lex])
```

Here, we are using AXIOM's type conversion facility :: to convert from one type to another.

The syntax for the `groebner` command is:

```
-> groebner(polylist)
```

This computes a Groebner basis for the ideal generated by the polynomials in `polylist` (of the appropriate type). The answer is reduced in the sense of Chapter 2, §7. For example, if `g,h` are as above, then the command:

```
-> gb := groebner([g,h])
```

computes a list (and gives it the symbolic name `gb`) which is a Groebner basis for the ideal $\langle x^2 + y, x + 2xy\rangle \subset \mathbb{Q}[x, y]$ with respect to grevlex for $x > y$. Also, if you want information about the intermediate stages of the calculation, you can include the options `"redcrit"` or `"info"` in the `groebner` command. For example, the command:

```
-> groebner([g,h], "redcrit")
```

will print out the remainders of S-polynomials (only one in this case) generated during the course of the computation. Adding the `"info"` option yields even more information.

AXIOM can also work with coefficients in a variety of fields besides $\mathbb{Q}$. This is easily done by replacing `FRAC INT` in the type declaration. For instance, to compute Groebner bases over the field of rational functions in polynomials with integer coefficients, one uses `FRAC POLY INT`. To see how this works, let us compute a Groebner basis for the ideal $\langle vx^2 + y, uxy + y^2\rangle \subset \mathbb{Q}(u, v)[x, y]$ using lex order with $x > y$. This is accomplished by the following AXIOM commands:

```
-> m : List DMP([x,y],FRAC POLY INT)
-> m := [v*x^2+y,u*x*y+y^2]
-> groebner(m)
```

Notice that this illustrates another method for declaring the type of the polynomials used in a Groebner basis computation.

Other fields are just as easy: one uses `FRAC COMPLEX INT` for the field of Gaussian rational numbers $\mathbb{Q}(i) = \{a + bi : a, b \in \mathbb{Q}\}$ (note that AXIOM writes $i = \sqrt{-1}$ as `%i`) and `PrimeField(`p`)` for a finite field with p elements (where p is a prime). It is also possible to compute Groebner bases over arbitrary finite fields. AXIOM's method of working with finite fields is explained in Section 8.11 of JENKS and SUTOR (1992). The ability to simply "insert" the field you want to compute Groebner bases over is a good illustration of the power of AXIOM.

Besides working with lists of polynomials, AXIOM also allows the user to declare a list of polynomials to be an ideal. The syntax of the ideal command is:

```
-> ideal polylist
```

where `polylist` is a list of polynomials of the appropriate type. This is useful because AXIOM has a number of commands which apply to ideals, including:

- `intersect`, which computes the intersection of a list of ideals.
- `zeroDim?`, which determines (using the methods of Chapter 5, §3) if the equations have finitely many solutions over an algebraically closed field.
- `dimension`, which computes the dimension of the variety defined by an ideal.
- `prime?`, which determines whether an ideal is prime.
- `radical`, which computes the radical of an ideal.
- `primaryDecomp`, which computes the primary decomposition of an ideal.

Examples of how to use these and other related AXIOM commands can be found in Section 8.12 of JENKS and SUTOR (1992). We should also mention that there are the commands `leadingMonomial` and `leadingCoefficient` for extracting the leading term and coefficient of a polynomial.

All of the commands described so far require that you declare in advance the type of polynomial you'll be using. However, if you only need Groebner bases in lex or grevlex order with rational coefficients, then a simpler approach is to use the AXIOM commands `lexGroebner` and `totalGroebner`. For example, the command:

```
-> lexGroebner([2*x^2+y,2*y^2+x], [x,y])
```

computes a Groebner basis (reduced up to constants) for the ideal $\langle 2x^2 + y, 2y^2 + x\rangle \subset \mathbb{Q}[x, y]$ using lex order with $x > y$. Notice that we didn't have to declare the type of the polynomials in advance—`lexGroebner` takes care of this. To do the same computation using grevlex, simply replace `lexGroebner` with `totalGroebner`.

We will end this section by explaining how to get AXIOM to work with grlex order. All of the raw material needed is present in AXIOM, though it takes a little work to put it together. For concreteness, suppose we want grlex order on $\mathbb{Q}[x, y]$ with $x > y$. Then issue the commands:

```
-> )set expose add constructor GDMP
-> )set expose add constructor ODP
-> Grlex := GDMP([x,y],FRAC INT,ODP(2,NNI,totalLex$ORDFUNS
  (2,NNI)))
```

The basic idea here is that `GDMP` stands for "General Distributed Multivariate Polynomial," which can be used to create an AXIOM type for *any* monomial order, and `totalLex` is the function which orders exponent vectors using grlex. By declaring polynomials to be of type `Grlex`, you can now compute Groebner bases using grlex with $x > y$. We should caution that type conversion doesn't work between Grlex and the monomial orders created by `DMP` and `HDMP`, though it is possible to write type conversion routines. Using the AXIOM concept of a package, one could write a package which knows all of the monomial orders mentioned in the exercises to Chapter 2, §4, along with commands to convert from one type to the other.

§2 Maple

Our discussion applies to Maple 9.5. For us, the most important part of Maple is the `Groebner` package. To have access to the commands in this package, type:

```
> with(Groebner);
```

(here > is the Maple prompt, and as usual, all Maple commands end with a semicolon). Once the `Groebner` package is loaded, you can perform the division algorithm, compute Groebner bases, and carry out a variety of other commands described below.

In Maple, a monomial ordering is called a termorder. Of the monomial orderings considered in Chapter 2, the easiest to use are lex and grevlex. Lex order is called `plex` (for "pure lexicographic"), and grevlex order is called `tdeg` (for "total degree"). Be careful not to confuse `tdeg` with grlex. Since a monomial order depends also on how the variables are ordered. Maple needs to know both the termorder you want (`plex` or `tdeg`) *and* a list of variables. For example, to tell Maple to use lex order with variables $x > y > z$, you would need to input `plex (x,y,z)`.

This package also knows an elimination order, as defined in Exercise 5 of Chapter 3, §1. To eliminate the first k variables from $x_1, \ldots, x_n$, one can use `lexdeg` ([`x_1`, ..., `x_k`], [`x_{k + 1}`, . . . , `x_n`]) (remember that Maple encloses a list inside brackets [. . .]). This order is similar (but not identical to) to the elimination order of Bayer and Stillman described in Exercise 6 of Chapter 3, §1.

The Maple documentation for the `Groebner` package also describes how to use certain weighted orders, and we will explain below how matrix orders give us many more monomial orderings.

The most commonly used commands in Maple's `Groebner` package are `normalf`, for doing the division algorithm, and `gbasis`, for computing a Groebner basis. The name `normalf` stands for "normal form", and the command has the following syntax:

```
> normalf(f,polylist,term_order);
```

The output is the remainder of `f` on division by the polynomials in the list `polylist` using the monomial ordering specified by `term_order`. For example, to divide $x^3 + 3y^2$ by $x^2 + y$ and $x + 2xy$ using grevlex order with $x > y$, one would enter:

```
> normalf(x^3+3*y^2,[x^2+y,x+2*x*y], tdeg(x,y));
```

The base field here is the rational numbers $\mathbb{Q}$. Note that `normalf` does not give the quotients in the division algorithm.

As you might expect, `gbasis` stands for "Groebner basis", and the syntax is as follows:

```
> gbasis(polylist, term_order);
```

This computes a Groebner basis for the ideal generated by the polynomials in `polylist` with respect to the monomial ordering specified by `term_order`. The answer is a reduced Groebner basis (in the sense of Chapter 2, §7), except for clearing denominators. As an example of how `gbasis` works, consider the command:

```
> gb := gbasis([x^2+y,2*x*y+y^2],plex(x,y));
```

This computes a list (and gives it the symbolic name `gb`) which is a Groebner basis for the ideal $\langle x^2 + y, 2xy + y^2 \rangle \subset \mathbb{Q}[x, y]$ using lex order with $x > y$.

If you use polynomials with integer or rational coefficients in `normalf or gbasis`, Maple will assume that you are working over the field $\mathbb{Q}$. Note that there is no limitation on the size of the coefficients. Maple can also work with coefficients that lie in rational function fields. To tell Maple that a certain variable is in the base field (a "parameter"), you simply omit it from the variable list in the `term_order`. Thus,

```
> gbasis([v*x^2+y,u*x*y+y^2],plex(x,y));
```

will compute a Groebner basis for $\langle vx^2 + y, uxy + y^2 \rangle \subset \mathbb{Q}(u, v)[x, y]$ for lex order with $x > y$. The answer is reduced up to clearing denominators (so the leading coefficients of the Groebner basis are polynomials in u and v).

The Groebner package can also work with matrix orders by using the `Ore_algebra` package, which is loaded via the command:

```
> with(Ore_algebra);
```

Using Maple notation for matrices, suppose that [u_1, ..., u_n] is invertible matrix, where each u_i = [u_i1, ..., u_in] is a vector in $\mathbb{Z}^n_{\geq 0}$ Then define $x^\alpha > x^\beta$ if

$$\text{u\_1} \cdot \alpha > \text{u\_1} \cdot \beta, \text{ or } \text{u\_1} \cdot \alpha = \text{u\_1} \cdot \beta \text{ and } \text{u\_2} \cdot \alpha > \text{u\_2} \cdot \beta, \text{ or } \ldots.$$

Order of this type are discussed (from a slightly more general point of view) in the remarks following Exercise 12 of Chapter 2, §4.

To see how such an order can be entered into Maple, suppose that we want to use grlex with $x > y > z$. This is done via the commands:

```
> B:= poly_algebra(x,y,z);
> M:= [[1,1,1],[1,0,0],[0,1,0]];
> GL:= termorder(B,'matrix'(M,[x,y,z]));
```

(It is a good exercise to show that the monomial ordering given by `GL` is grlex with $x > y > z$.) Using `GL` as the `term_order` in the `normalf` or `gbasis` commands, one can now compute remainders or Groebner bases in $\mathbb{Q}[x, y, z]$. Using matrix orders, one can create *all* of the monomial orderings described in the book.

When combined, the `Groebner` and `Ore_algebra` packages of Maple 9.5 allow one to do Groebner basis computations for a wide variety of monomials orderings over algebraic number fields, finite fields, or even certain noncommutative rings such as rings of differential operators. Further information can be found in the Maple documentation.

Some other useful Maple commands in the Groebner package are:

- `leadmon`, which computes LM(f) and LC(f) for a polynomial f with respect to `term_order`. Related commands are `leadterm`, which finds LM(f), and `leadcoeff`, which finds LC(f).
- `spoly`, which computes the S-polynomial $S(f, g)$ of two polynomials.
- `is_solvable`, which uses the consistency algorithm from Chapter 4, §1 to determine if a system of polynomial equations has a solution over an algebraically closed field.
- `is_finite`, which uses the finiteness algorithm from Chapter 5, §3 to determine if a system of polynomial equations has finitely many solutions over an algebraically closed field.
- `univpoly`, which given a variable and a Groebner basis, computes the polynomial of lowest degree in the given variable which lies in the ideal generated by the Groebner basis.
- `hilbertpoly`, which given a Groebner basis of an ideal I, computes ${}^aHP_I(s) - -{}^aHP_I(s-1)$ in the notation of Chapter 9, §3. When I is a homogenous ideal, Theorem 12 of Chapter 9, §3 shows that `hilbertpoly` computes the Hilbert polynomial $HP_I(s)$. A related command is `hilbertseries`, which for a homogeneous ideal computes the Hilbert series as defined in Exercise 24 of Chapter 6, §4 of COX, LITTLE, AND O' SHEA (1998).

There is also a `solve` command which attempts to find all solutions of a system of equations. Maple has an excellent on-line help system that should make it easy to master these (and other) Maple commands.

Finally, we should mention the existence of a Maple package written by Albert Lin and Philippe Loustaunau of George Mason University (with subsequent modifications by David Cox and Will Gryc of Amherst College and Chris Wensley of the University of Bangor, Wales) which extends the `Groebner` package. In this package, the program `div_alg` gives the quotients in the division algorithm, and the program `mxgb` computes a Groebner basis together with a matrix telling how to express the Groebner basis in terms of the given polynomials. This package is slow compared to the `Groebner` package, but can be used for many of the simpler examples in the book. There is also a Maple worksheet which explains how to use the package. Copies of the package and worksheet can be obtained from `http://www.cs.amherst.edu/~dac/iva.html`.

§3 Mathematica

Our discussion applies to Mathematica 5.1. There is no special package to load in order to compute Groebner bases: the basic commands are part of the Mathematica kernel.

Mathematica knows all of the monomial orderings considered in Chapter 2. In typical Mathematica fashion, lex order is called `Lexicographic`, `grlex` is

DegreeLexicographic and grevlex is DegreeReverseLexicographic. The monomial order is determined by using the MonomialOrder option within the Mathematica commands described below. If you omit the MonomialOrder option, Mathematica will use the default order, which is lex. Mathematica can also use the weight orders mentioned in the comments at the end of the exercises to Chapter 2, §4.

Since a monomial order also depends on how the variables are ordered, Mathematica also needs to know a list of variables in order to specify the monomial order order you want. For example, to tell Mathematica to use lex order with variables $x > y > z$, you would input {x,y,z} (remember that Mathematica encloses a list inside braces{...}) into the Mathematica command you want to use.

For our purposes, the most important commands in Mathematica are PolynomialReduce and GroebnerBasis. One nice feature of PolynomialReduce is that it does the division algorithm from Chapter 2 with quotients. The syntax is as follows:

```
In[1] := PolynomialReduce[f,polylist,varlist,options]
```

(where In[1] := is the Mathematica prompt). This computes the quotients and remainder of f on division by the polynomials in polylist using the monomial order specified by varlist and the MonomialOrder option. For example, to divide $x^3 + 3y^2$ by $x^2 + y$ and $x + 2xy$ using grlex order with $x > y$, one would enter:

```
In[2] := PolynomialReduce[x^3 + 3 y^2,{x^2 + y,x + 2 xy},
          {x,y}, MonomialOrder -> DegreeLexicographic]
```

The output is a list with two entries: the first is a list of the quotients and the second is the remainder.

Of course, the Mathematica command GroebnerBasis is used for computing Groebner bases. It has the following syntax:

```
In[3] := GroebnerBasis[polylist,varlist,options]
```

This computes a Groebner basis for the ideal generated by the polynomials in polylist with respect to the monomial order given by the MonomialOrder option with the variables ordered according to varlist. The answer is a reduced Groebner basis (in the sense of Chapter 2, §7), except for clearing denominators. As an example of how GroebnerBasis works, consider:

```
In[4] := gb = GroebnerBasis[{x^2+y,2xy+y^2},{x,y}]
```

The output is a list (with the symbolic name gb) which is a Groebner basis for the ideal $\langle x^2 + y, 2xy + y^2 \rangle \subset \mathbb{Q}[x, y]$ using lex order with $x > y$. We omitted the MonomialOrder option since lex is the default.

If you use polynomials with integer or rational coefficients in GroebnerBasis or PolynomialReduce, Mathematica will assume that you are working over the field $\mathbb{Q}$. There is no limitation on the size of the coefficients. Another possible coefficient field is the Gaussian rational numbers $\mathbb{Q}(i) = \{a + bi : a, b \in \mathbb{Q}\}$, where $i = \sqrt{-1}$ (note that Mathematica uses I to denote $\sqrt{-1}$. To compute a Groebner basis over a finite field with p elements (where p is a prime number), you need to include the option Modulus -> p in the GroebnerBasis command. (This option also works in PolynomialReduce.)

Mathematica can also work with coefficients that lie in a rational function field. The strategy is that the variables in the base field (the "parameters") should be omitted

from the variable list in the input, and then one sets the `CoefficientDomain` option to `RationalFunctions`. For example, the command:

```
In[5] := GroebnerBasis[{vx^2+y,uxy+y^2},{x,y},
          CoefficientDomain -> RationalFunctions]
```

will compute a Groebner basis for $\langle vx^2 + y, uxy + y^2\rangle \subset \mathbb{Q}(u, v)[x, y]$ for lex order with $x > y$. The answer also clears denominators, so the leading coefficients of the Groebner basis are polynomials in u and v. (The `CoefficientDomain` option is also available in `PolynomialReduce`.)

Here are some other useful Mathematica commands:

- `MonomialList`, which lists the terms of a polynomial according to the monomial order.
- `Eliminate`, which uses the Elimination Theorem of Chapter 3, §1 to eliminate variables from a system of polynomial equations.
- `Solve`, which attempts to find all solutions of a system of equations.

For further descriptions and examples, consult *The Mathematica Book* by WOLFRAM (1996).

Finally, there is a Mathematica package written by Susan Goldstine of Amherst College (with an update by Will Gryc, also of Amherst) which includes many commands relevant to the book. Using this package, students can compute Groebner bases, together with information about the number of nonzero remainders that occur. Other algorithms from the book are included, such as ideal membership, radical membership, and finiteness of solutions. This package is slow compared to the `GroebnerBasis` command, but it can be used for most of the simpler examples in the text. Copies of the package can be obtained from `http://www.cs.amherst.edu/~dac/iva.html`.

§4 REDUCE

Our discussion applies to version 3.5 of REDUCE. To do a Groebner basis calculation with REDUCE, you need to use either the Groebner package or the Cali package.

Groebner

We will describe the version of the Groebner package dated November 18, 1994. To have access to the commands in this package, type:

```
1: load_package groebner;
```

(here, 1: is the REDUCE prompt, and as usual, all REDUCE commands end with a semicolon). Once the Groebner package is loaded, you can perform the division algorithm, compute Groebner bases, and carry out a variety of other commands described below.

In the Groebner package, a monomial ordering is called a term order. Of the monomial orderings considered in Chapter 2, Groebner knows most of them, including lex, grlex and grevlex. Lex order is called `lex`, grlex is called `gradlex`, and grevlex is called `revgradlex`. Groebner also works with product orders (see Exercise 10 of Chapter 2, §4), weight orders (see Exercise 12 of Chapter 2, §4—note that weight orders in

the Groebner package always use lex order to break ties), and more general orders specified by a matrix (see the comments at the end of the Exercises to Chapter 2, §4). These other term orders are described in detail in Section 4.10 of MELENK, MÖLLER and NEUN (1994).

In Groebner, a term order is specified by means of the `torder` command. Since a monomial order depends also on how the variables are ordered, Groebner needs to know both the term order *and* a list of variables. Thus, `torder` commands takes two arguments: a list of variables and the term order. For example, to use grevlex with $x > y > z$, you would type:

```
2: torder ({x,y,z},revgradlex);
```

(remember that REDUCE encloses a list inside braces $\{\dots\}$). In response, REDUCE will print out the previous term order.

The most commonly used commands in the Groebner package are `preduce`, for doing the division algorithm, and `groebner`, for computing a Groebner basis. The name preduce stands for "polynomial reduce," and the command has the following syntax:

```
3: preduce(f,polylist);
```

The output is the remainder of f on division by the polynomials in the list `polylist` using the monomial ordering specified by `torder`. For example, to divide $x^3 + 3y^2$ by $x^2 + y$ and $x + 2xy$ using grlex order with $x > y$, one would enter:

```
4: torder({x,y},gradlex);
5: preduce(x^3+3*y^2,{x^2+y,x+2*x*y},{x,y});
```

In this example, the base field is the rational numbers $\mathbb{Q}$. Note that `preduce` does not give the quotients in the division algorithm.

As you might expect, `groebner` stands for "Groebner basis," and the syntax is:

```
6: groebner(polylist);
```

This computes a Groebner basis for the ideal generated by the polynomials in `polylist` with respect to the monomial ordering specified by `torder`. The answer is a reduced Groebner basis (in the sense of Chapter 2, §7), except for clearing denominators. As an example of how `groebner` works, consider the command:

```
7: gb := groebner({x^2+y,2*x*y+y^2});
```

This computes a list (and gives it the symbolic name `gb`) which is a Groebner basis for the ideal $\langle x^2 + y, 2xy + y^2\rangle \subset \mathbb{Q}[x, y]$, using the term order specified by `torder`.

If you use polynomials with integer or rational coefficients in `preduce` or `groebner`, Groebner will assume that you are working over the field $\mathbb{Q}$. There is no limitation on the size of the coefficients. Another possible coefficient field is the Gaussian rational numbers $\mathbb{Q}(i) = \{a + bi : a, b \in \mathbb{Q}\}$, where $i = \sqrt{-1}$. To work over $\mathbb{Q}(i)$, you need to issue the command:

```
8: on complex;
```

before computing the Groebner basis (note that REDUCE uses `I` to denote $\sqrt{-1}$). Similarly, to compute a Groebner basis over a finite field with p elements (where p is a prime number), you first need to issue the command:

```
9: on modular; setmod p;
```

To return to working over $\mathbb{Q}$, you would type `off modular`.

Groebner can also work with coefficients that lie in a rational function field. To tell Groebner that a certain variable is in the base field (a "parameter"), you simply omit it from the variable list in the `torder` command. Thus, the command:

```
10: groebner({v*x^2+y, u*x*y+y^2});
```

will compute a Groebner basis for $\langle vx^2 + y, uxy + y^2\rangle \subset \mathbb{Q}(u, v)[x, y]$ for the term order given by `torder`. The answer is reduced up to clearing denominators (so the leading coefficients of the Groebner basis are polynomials in u and v).

The Groebner package has two switches which control how Groebner basis computations are done. (In REDUCE, a *switch* is a variable that can be set to `on or off`. Examples of switches you've already seen are `complex` and `modular`.) When computing a Groebner basis, there are a number of choices which can be made during the course of the algorithm, and different choices can have a dramatic effect on the length of the computation. We will describe two switches, `groebopt` and `gsugar`, which can affect how the `groebner` command carries out a computation.

In some cases, it is possible to improve efficiency by changing the order of the variables, though keeping the same term order (e.g., using lex with $y > x$ rather than $x > y$). An algorithm for doing this is described in BOEGE, GEBAUER and KREDEL (1986), and to enable this feature in REDUCE, you give the command:

```
11: on groebopt;
```

Once the calculation is done, you can determine how the variables were ordered by typing:

```
12: gvarslast;
```

This will print out the variables in the order used in the computation. There are some cases (especially when doing elimination) when you don't want an arbitrary reordering of the variables. In this situation, you can use the depend command. For example, if you have variables s, t, x, y, z and you want to eliminate s, t, then after giving the on `groebopt` command, you would also type:

```
13: depend s,x,y,z; depend t,x,y,z;
```

With this preparation, the `groebner` command would reorder the variables, but always keeping s, t before x, y, z.

The algorithm used by the `groebner` command uses the concept of sugar, which was mentioned briefly in Chapter 2, §9. To experiment with the effect of sugar, you can turn it on or off by means of the switch `gsugar`. The default is on `gsugar`, so that to turn off sugar for a particular computation, you would issue the command `off gsugar` before giving the `groebner` command.

We should also mention the switches `groebstat, trgroeb` and `trgroebs` for the `groebner` command which print out statistics about the Groebner basis calculation. These switches are described in Section 4.2 of MELENK, MÖLLER and NEUN (1994).

Some other useful commands in the Groebner package are:

- `gsplit`, which computes $\mathrm{LT}(f)$ and $f - \mathrm{LT}(f)$.
- `gsort`, which prints out the terms of a polynomial according to the term order.
- `gspoly`, which computes an S-polynomial $S(f, g)$.
- `greduce`, which computes the remainder on division by the Groebner basis of the ideal generated by the input polynomials.

- `preducet`, which can be used to find the quotients in the division algorithm.
- `gzerodim?`, which tests a Groebner basis (using the methods of Chapter 5, §3) to see if the equations have finitely many solutions over an algebraically closed field.
- `glexconvert`, which, for a Groebner basis for an arbitrary monomial order with finitely many solutions over $\mathbb{C}$, converts it to a lex Groebner basis. This implements the algorithm discussed in Project 5 of Appendix D.
- `groesolve`, which attempts to find all solutions of a system of polynomial equations.
- `idealquotient`, which computes an ideal quotient $I : f$ (using an algorithm more efficient than the one described in Chapter 4, §4).
- `hilbertpolynomial`, which computes the affine Hilbert polynomial of an ideal (as defined in Chapter 9, §3).

These (and many other) commands are described in detail in *Groebner: A package for calculating groebner bases* by MELENK, MÖLLER and NEUN (1994). This document comes with all copies of REDUCE.

Cali

We will discuss Version 2.2.1 of the Cali package. Cali is more mathematically sophisticated than the Groebner package and is a little harder to use for the beginner. On the other hand, it can also do some computations (such as radicals and primary decomposition) which aren't part of the Groebner package. To load Cali, use the command:

```
1: load_package cali;
```

Don't load Groebner and Cali in the same REDUCE session since there are conflicts between them.

In Cali, you first have to declare the variables and monomial order before typing in any polynomials. This is done by the `setring` command, which has the syntax:

```
2: setring(vars,weight,order);
```

Here, vars is the list of variables you will use, `weight` is a list of weight vectors (possibly empty), and `order` is one of `lex or revlex`. For example:

```
3: setring({x,y,z},{},lex);
```

will give lex order on $\mathbb{Q}[x, y, z]$, while:

```
4: setring({x,y,z},{{1,1,1}},lex);
```

gives grlex on the same ring, and you can get grevlex simply by changing `lex to revlex` in the last command. One can also get weight orders, elimination orders and matrix orders as described in Exercise 12 of Chapter 2, §4. See Section 2.1 of GRÄBE (1995) for the details of how monomial orders work in Cali.

Once the ring is established, you can define ideals using lists of polynomials. One difference is that you must explicitly name the ideal. For example, suppose we let `j` denote the ideal generated by $x^2 + y$ and $x + 2xy$. In Cali, this is done by the command:

```
5: setideal(j,{x^2+y,x+2*x*y});
```

Once we know the ideal, we can do various things with it. For example, to divide $x^3 + 3y^2$ by $x^2 + y$ and $x + 2xy$, we use the command:

```
6: x^3+3*y^2 mod j;
```

Also, to compute a Groebner basis of this ideal, the command to use is:

```
7: gbasis j;
```

The output is a Groebner basis for $\langle x^2 + y, x + 2xy\rangle \subset \mathbb{Q}[x, y]$ for the monomial order set by the `setring` command. The answer is reduced (in the sense of Chapter 2, §7), except for clearing denominators.

If you use polynomials with integer or rational coefficients, Cali will assume that you are working over the field $\mathbb{Q}$. To compute Groebner bases over a finite field, you use the same commands as for the Groebner package described earlier in this section. Finally, for coefficients that lie in rational function fields, one proceeds as with the Groebner package and simply omits the variables in the base field when giving the `setring` command. For example, if we use the ring set in 4: above, then the commands:

```
8: setideal(m,{v*x^2+y,u*x*y+y^2});
9: gbasis m;
```

will compute a Groebner basis for $\langle vx^2 + y, uxy + y^2\rangle \subset \mathbb{Q}(u, v)[x, y]$ for grlex with $x > y$. The answer is reduced up to clearing denominators (so the leading coefficients of the Groebner basis are polynomials in u and v).

Some other useful commands in Cali are:

- `dimzerop`, which tests a Groebner basis (using the methods of Chapter 5, §3) to see if the equations have finitely many solutions over an algebraically closed field.
- `dim`, which for a Groebner basis for an ideal computes the dimension of the associated variety.
- `idealquotient`, which computes an ideal quotient $I : f$.
- `isprime`, which tests a Groebner basis to see if it generates a prime ideal.
- `radical`, which computes the radical of an ideal.
- `primarydecomposition`, which computes the primary decomposition (as in Chapter 4, §7) of an ideal.

In addition, Cali has commands for dealing with more sophisticated mathematical objects such as modules, blowups, free resolutions and tangent cones. Details of these commands are described in *CALI: A REDUCE package for commutative algebra* by GRÄBE (1995).

§5 Other Systems

Besides the general computer algebra systems discussed so far, there are three more specialized programs, Macaulay 2, CoCoA, and SINGULAR which should be mentioned. These programs were designed primarily for researchers in algebraic geometry and commutative algebra, but less sophisticated users can make effective use of either program. One of their most attractive features is that they are free.

It is a bit more complicated to get started with Macaulay 2, CoCoA or SINGULAR. For example, you have to tell the program in advance what the variables are and what field you are working over. This makes it more difficult for a novice to use Macaulay. Nevertheless, with proper guidance, beginning users should be able to work quite successfully with these programs.

Macaulay 2, CoCoA and SINGULAR give you a choice of working over $\mathbb{Q}$ or a finite field. Over a finite field, some computations go considerably faster. As long as the

coefficient size doesn't exceed the characteristic of the field (which is usually the case in simple examples), there is no problem. However, one must exercise some care in dealing with more complicated problems. This drawback must be weighed against the fact that such problems are often difficult to carry out on other systems because of the extremely large amount of memory that may be required.

For more advanced users, Macaulay 2, CoCoA and SINGULAR offer a wonderful assortment of sophisticated mathematical objects to work with. Many researchers make frequent use of these programs to compute syzygies and free resolutions of modules. Macaulay 2 also includes scripts for computing blowups, cohomology, cotangent sheaves, dual varieties, normal cones, radicals and many other useful objects in algebraic geometry. These programs are available electronically:

- `http://www.math.uiuc.edu/Macaulay2/` for Macaulay 2.
- `http://cocoa.dima.unige.it/` for CoCoA.
- `http://www.singular.uni-kl.de/` for SINGULAR.

In addition to the computer algebra systems described above, there are other systems worth mentioning:

- The system MAS is another computer algebra system available electronically. Besides computing Groebner bases as usual, it can also compute comprehensive Groebner bases and Groebner bases over principal ideal domains (as described in Project 16 of Appendix D). Instructions for obtaining MAS can be found on page xiii of BECKER and WEISPFENNING (1993).
- The computer algebra system Magma can do computations in group theory, number theory, combinatorics and commutative algebra. More information about Magma can be found at the web site:
  `http://magma.maths.usyd.edu.au/magma/`

This list of computer algebra systems for working with Groebner bases is far from complete. As computers get faster and computer algebra software gets more powerful and easier to use, we can expect an ever-increasing range of applications for Groebner bases and algebraic geometry in general.