Preface

Systems of polynomial equations arise throughout mathematics, science, and engineering. Algebraic geometry provides powerful theoretical techniques for studying the qualitative and quantitative features of their solution sets. Recently developed algorithms have made theoretical aspects of the subject accessible to a broad range of mathematicians and scientists. The algorithmic approach to the subject has two principal aims: developing new tools for research within mathematics, and providing new tools for modeling and solving problems that arise in the sciences and engineering. A healthy synergy emerges, as new theorems yield new algorithms and emerging applications lead to new theoretical questions.

This book presents algorithmic tools for algebraic geometry and experimental applications of them. It also introduces a software system in which the tools have been implemented and with which the experiments can be carried out. *Macaulay 2* is a computer algebra system devoted to supporting research in algebraic geometry, commutative algebra, and their applications. The reader of this book will encounter *Macaulay 2* in the context of concrete applications and practical computations in algebraic geometry.

The expositions of the algorithmic tools presented here are designed to serve as a useful guide for those wishing to bring such tools to bear on their own problems. A wide range of mathematical scientists should find these expositions valuable. This includes both the users of other programs similar to *Macaulay 2* (for example, Singular and CoCoA) and those who are not interested in explicit machine computations at all.

The chapters are ordered roughly by increasing mathematical difficulty. The first part of the book is meant to be accessible to graduate students and computer algebra users from across the mathematical sciences and is primarily concerned with introducing *Macaulay 2*. The second part emphasizes the mathematics: each chapter exposes some domain of mathematics at an accessible level, presents the relevant algorithms, sometimes with proofs, and illustrates the use of the program. In both parts, each chapter comes with its own abstract and its own bibliography; the index at the back of the book covers all of them.

One of the first computer algebra packages aimed at algebraic geometry was *Macaulay*, the predecessor of *Macaulay 2*, written during the years 1983-1993 by Dave Bayer and Mike Stillman. Worst-case estimates suggested that trying to compute Gröbner bases might be a hopeless approach to solving problems. But from the first prototype, *Macaulay* was successful surprisingly often, perhaps because of the geometrical origin of the problems attacked. *Macaulay* improved steadily during its first decade. It helped transform the theoretical notion of a projective resolution into an exciting new practical research tool, and became widely used for research and teaching in commutative algebra and algebraic geometry. It was possible to write routines in the top-level language, and many important algorithms were added by David Eisenbud and other users, enhancing the system and broadening its usefulness.

There were certain practical drawbacks for the researcher who wanted to use *Macaulay* effectively. A minor annoyance was that only finite prime fields were available as coefficient rings. The major problem was that the language made available to users was primitive and barely supported high-level development of new algorithms; it had few basic data types and didn't support the addition of new ones.

Macaulay 2 is based on experience gained from writing and using its predecessor *Macaulay*, but is otherwise a fresh start. It was written by Dan Grayson and Mike Stillman with the generous financial support of the U.S. National Science Foundation, with the work starting in 1993[1]. It also incorporates some code from other authors: the package SINGULAR-FACTORY[2] provides for factorization of polynomials; SINGULAR-LIBFAC[3] uses FACTORY to enable the computation of characteristic sets and thus the decomposition of subvarieties into their irreducible components; and GNU MP[4] by Torbjörn Granlund and others provides for multiple precision arithmetic.

Macaulay 2 aims to support efficient computation associated with a wide variety of high level mathematical objects, including Galois fields, number fields, polynomial rings, exterior algebras, Weyl algebras, quotient rings, ideals, modules, homomorphisms of rings and modules, graded modules, maps between graded modules, chain complexes, maps between chain complexes, free resolutions, algebraic varieties, and coherent sheaves. To make the system easily accessible, standard mathematical notation is followed closely.

As with *Macaulay*, it was hoped that users would join in the further development of new algorithms for *Macaulay 2*, so the developers tried to make the language available to the users as powerful as possible, yet easy to use. Indeed, much of the high-level part of the system is written in the same language available to the user. This ensures that the user will find it just as

---

[1] NSF grants DMS 92-10805, 92-10807, 96-23232, 96-22608, 99-70085, and 99-70348.

[2] `SINGULAR-FACTORY`, a subroutine library for factorization, by G.-M. Greuel, R. Stobbe, G. Pfister, H. Schoenemann, and J. Schmidt; available at ftp://helios.mathematik.uni-kl.de/pub/Math/Singular/Factory/.

[3] `SINGULAR-LIBFAC`, a subroutine library for characteristic sets and irreducible decomposition, by M. Messollen; available at ftp://helios.mathematik.uni-kl.de/pub/Math/Singular/Libfac/.

[4] `GMP`, a library for arbitrary precision arithmetic, by Torbjörn Granlund, John Amanatides, Paul Zimmermann, Ken Weber, Bennet Yee, Andreas Schwab, Robert Harley, Linus Nordberg, Kent Boortz, Kevin Ryde, and Guillaume Hanrot; available at ftp://ftp.gnu.org/gnu/gmp/.

easy as the developers did to implement a new type of mathematical object or to modify the high-level aspects of the current algorithms.

The language available to the user is interpreted. The interpreter itself is written in a convenient language designed to be mostly type-safe and to handle memory allocation and initialization automatically. For maximum efficiency, the core mathematical algorithms are written in C++ and compiled, not interpreted. This includes the arithmetic operations of rings, modules, and matrices, the Gröbner basis algorithm (in several enhanced versions, tailored for various situations), several algorithms for computing free resolutions of modules, the algorithm for computing the Hilbert series of a graded ring or module, the algorithms for computing determinants and Pfaffians, the basis reduction algorithm, factoring, etc.

In one way *Macaulay 2* is like a standard computer algebra system, such as *Mathematica* or *Maple*: the user enters mathematical expressions at the keyboard, and the program computes the value of the expression and displays the answer.

Here is the first input prompt offered to the user.

```
i1 :
```

In response to the prompt, the user may enter, for example, a simple arithmetic expression.

```
i1 : 3/5 + 7/11

      68
o1 = --
      55

o1 : QQ
```

The answer itself is displayed to the right of the output label

```
o1 =
```

and its type (or class) is displayed to the right of the following label.

```
o1 :
```

The symbol `QQ` appearing in this example denotes the class of all rational numbers, and is meant to be reminiscent of the notation $\mathbb{Q}$.

Macaulay 2 often finds itself being run in a window with horizontal scroll bars, so by default it does not wrap output lines, but instead lets them grow without bound. This book was produced by an automated mechanism that submits code provided by the authors to *Macaulay 2* and incorporates the result into the text. Output lines that exceed the width of the pages of this book are indicated with ellipses, as in the following example.

```
i2 : 100!

o2 = 93326215443944152681699238856266700490715968264381621468592963895 ···
```

Next we describe an important difference between general computer algebra systems (such as *Maple* and *Mathematica*) and *Macaulay 2*. Before entering an expression involving variables (such as `x+y`) into *Macaulay 2* the user must first create a ring containing those variables. Rings are important objects of study in algebraic geometry; quotient rings of polynomial rings, for example, encapsulate the essential information about a system of polynomial equations, including, for example, the field from which the coefficients are drawn. Often one has several rings under consideration at once, along with ring homomorphisms between them, so it is important to treat them as first-class objects in the computer, capable of being named and manipulated the same way numbers and characters can be manipulated in simpler programming languages.

Let's give a hint of the breadth of types of mathematical objects available in *Macaulay 2* with some examples. In *Macaulay 2* one defines a quotient ring of a polynomial ring R over the rational numbers by entering a command such as the one below.

```
i3 : R = QQ[x,y,z]/(x^3-y^3-z^3)

o3 = R

o3 : QuotientRing
```

Having done that, we can compute in the ring.

```
i4 : (x+y+z)^3

       2         2     3     2                 2         2       2     3
o4 = 3x y + 3x*y  + 2y  + 3x z + 6x*y*z + 3y z + 3x*z  + 3y*z  + 2z

o4 : R
```

We can make matrices over the ring.

```
i5 : b = vars R

o5 = | x y z |

             1       3
o5 : Matrix R  <--- R

i6 : c = matrix {{x^2,y^2,z^2}}

o6 = | x2 y2 z2 |

             1       3
o6 : Matrix R  <--- R
```

We can make modules over the ring.

```
i7 : M = coker b

o7 = cokernel | x y z |

                              1
o7 : R-module, quotient of R

i8 : N = ker c

o8 = image {2} | x  0   -y2 -z2 |
           {2} | -y -z2 x2  0   |
           {2} | -z y2  0   x2  |
```

```
                              3
o8 : R-module, submodule of R
```

We can make projective resolutions of modules.

```
i9 : res M

      1      3      4      4      4
o9 = R  <-- R  <-- R  <-- R  <-- R

     0      1      2      3      4

o9 : ChainComplex
```

We can make projective varieties.

```
i10 : X = Proj R

o10 = X

o10 : ProjectiveVariety
```

We can make coherent sheaves and compute their cohomology.

```
i11 : HH^1 cotangentSheaf X

        1
o11 = QQ

o11 : QQ-module, free
```

At this writing, *Macaulay 2* is available for GNU/Linux and other flavors of Unix, and also for Microsoft Windows and the Macintosh operating system. Although it can be used as a free-standing program, it is most convenient to use it in an editor's buffer; Emacs (on Unix or Windows systems) or MPW on Macintosh systems are currently the editors of choice. To obtain *Macaulay 2*, download it from the website[5] and unpack the file. Among the resulting files will be a file called `Macaulay2/README.txt`, which you should read. It will tell you how to run the `setup` script, and how to install a few lines of code in your `emacs` init file to enable you to run `M2` in an emacs buffer and to edit *Macaulay 2* code. A system administrator of a Unix system may optionally arrange for those lines of code to be available to every emacs user.

The editors thank the authors of the chapters for their valuable contributions and hard work, and the National Science Foundation for funding the development of *Macaulay 2* and for partial funding of the authors who have contributed to this volume.

May, 2001

David Eisenbud
Daniel R. Grayson
Michael E. Stillman
Bernd Sturmfels

[5] *Macaulay 2*, a software system for research in algebraic geometry, by Daniel R. Grayson and Michael E. Stillman; available online in source code form and compiled for various architectures at `http://www.math.uiuc.edu/Macaulay2/`.