Introduction

Writing, for him who knows it
is better than all other professions;
it pleases more than bread and beer,
more than clothing and ornament.
(20^{th} Dynasty, Egypt)

0.1 What Is This Book About?

The title of this book is "Computational Commutative Algebra 2". It is the natural continuation of "Computational Commutative Algebra 1" written by us in the last millennium. In the introduction to the first volume we wrote that the fundamental ideas of Computational Commutative Algebra were deeply rooted in the development of mathematics in the 20^{th} century, and that we planned to be back with more in the not so distant future. So, here we are!

But why did it take so long? We love deadlines, in particular the whooshing sound they make as they fly by. Therefore we successfully missed every deadline we had set for ourselves. Clearly, one reason is that writing this book pleased us more than pasta and wine. Another reason is that laying the foundations for Computational Commutative Algebra in the third millennium turned out to be more of a job than we had bargained for. Many sections of this book are like small research papers because we tried to present the material in the style that pleases us so much and differs markedly from existing literature.

The result is a volume having almost twice the size of the first: some of its 23 sections are as big as a whole chapter, some of its 55 tutorials are as big as a whole section, and some exercises are as big as a tutorial. Together the two books form a compilation of about 900 pages which includes a vast collection of 99 tutorials to keep you busy for a long, long time. To sustain your interest, we have tried to continue the presentation in the lively style of the first volume. Alas, we have to inform you that this is absolutely and definitively the second and last volume of the trilogy.

0.2 Now, What Is This Book *Really* About?

As I said before,
I never repeat myself.
(Anonymous)

Let us examine some concrete problems whose solutions we shall try to explain in this book. For a similar description of the contents of Volume 1 we refer the reader to its introduction. Let us start with the problem of using Gröbner bases in the homogeneous case. Suppose we are given a polynomial ring $P = K[x_1, \ldots, x_n]$ over some field K.

Question 1 *How can we equip P with gradings which are useful for Computational Commutative Algebra?*

More precisely, we are asking whether it is possible to find a monoid Γ and a Γ-grading on P such that homogeneous ideals and graded modules have additional invariants and are simpler to handle computationally. The answer will turn out to be "positive $\mathbb{Z}^m$-gradings", whatever they are.

Question 2 *How can we relate an arbitrary ideal or module to a homogeneous ideal or a graded module?*

In other words, how can we pass from the inhomogeneous to the homogeneous case? At least three possibilities will be discussed: degree form ideals and Macaulay bases, homogenization, and the homogeneous part of an ideal.

Question 3 *How can we exploit homogeneity to compute Gröbner bases and perform elementary operations on ideals and modules more efficiently?*

Related questions are: Are there special term orderings which can be used to advantage in the graded setting? Are Gröbner bases of homogeneous ideals again homogeneous? If we stop the computation of a homogeneous Gröbner basis after some degree is finished, is the resulting *truncated Gröbner basis* any good?

The fact that the gradings we use are positive implies that all irredundant homogeneous systems of generators of a finitely generated graded module are minimal. Hence the minimal number of generators of such a module is a well-behaved invariant. This leads us to examine the following problems.

Question 4 *How can we compute minimal homogeneous systems of generators, minimal homogeneous presentations, and minimal graded free resolutions of finitely generated graded modules?*

Another aspect of positive gradings is that they force the homogeneous components of a finitely generated graded P-module M to be finite dimensional K-vector spaces. This allows us to define the *Hilbert function* $\mathrm{HF}_M : \mathbb{Z}^m \longrightarrow \mathbb{Z}$ of M by $\mathrm{HF}_M(i) = \dim_K(M_i)$ and to study its generating power series, the *Hilbert series* of M.

Question 5 *How can we compute the Hilbert function and the Hilbert series of a finitely generated graded module? What are their basic properties? How fast do Hilbert functions grow?*

In the standard graded case, i.e. when $\deg(x_1) = \cdots = \deg(x_n) = 1$, one of these basic properties of Hilbert functions is that $\mathrm{HF}_M(i)$ is given by the value of the *Hilbert polynomial* of M for large enough i.

Question 6 *How can we compute the Hilbert polynomial of a graded module M? Can we derive other invariants of M from it?*

It turns out that one of these invariants is the dimension of a graded ring or a graded module. Thus we ask ourselves:

Question 7 *What are the algebraic and geometric interpretations of the dimension of a graded ring?*

These are our questions. And if they are not enough for you ... well, we have others. For instance, we shall address the following existential question:

Question 8 *Can mathematicians be replaced by computers?*

In other words, can computers take over the job of proving theorems? Is there true artificial intelligence?

Question 9 *What is there beyond Gröbner bases?*

Which other kinds of problems can be addressed using Computational Commutative Algebra? Are there situations in which the theory of Gröbner bases can be imitated or generalized?

Let us end this discussion by pointing out one unimportant choice we have made. We have decided to change the title of Chapter 6 slightly and call it simply *"Further Applications"* rather than mentioning Gröbner bases explicitly. The justification is that the chapter has in fact a much wider scope. Although Gröbner bases are always sneaking in one way or another, we *cover all the bases:* binomial bases of toric ideals, Hilbert bases of monoids, liftings and distractions of monomial bases, SuperG bases, bases of vanishing ideals of finite sets of points, border bases of zero-dimensional ideals, standard bases of filtered modules, SAGBI bases of subalgebras, and a basis for automatic theorem proving. You name it, we've got it! *(Well, almost.)*

Last, but not least, let us henceforth limit the use of "let us", lest we run the risk of repeating ourselves.

Repetita iuvant.
(Ancient Latin Adage)

0.3 What Is This Book *Not* About?

All is the same
A year has gone by
Some day you came
Some day you'll die.
(Cesare Pavese)

Had we written this sentence in Volume 1, this very moment would have occurred about four years ago. In those days we mentioned that the list of topics which we did *not* talk about contained soccer, chess, gardening, and our other favourite pastimes. Since life is too short and precious to miss out on the good stuff, we decided to remedy these shortcomings. The current volume includes two tutorials related to chess, and in the introductions of the last two chapters we meet a gardener and a chess player engaged in an animated *trilogue* with an amateur mathematician. Unfortunately, they are so immersed in the discussion that the gardener forgets to advise us about the right time to sow tomato seeds and the correct composition of the soil for growing azaleas.

What else is *not* in this book? There is still nothing about computability or complexity. We use the expression *effectively computable* loosely to mean that there is some algorithm to perform the computation. And when we write that some algorithm is *efficient*, we are usually indicating a personal opinion. In fact, analyzing the worst-case or average-case complexity of algorithms is not only beyond the scope of this book, but in our experience it would tell little about their *practical complexity*, i.e. their running time for those examples we happen to be interested in.

Even the terms *algorithm* and *procedure* are applied in a purely intuitive and informal manner. In fact, the way we present algorithms in this book differs from computer science texts. For us, an algorithm is really a more general form of theorem where the instructions do not necessarily follow linearly one after another, but conditionals and recursive constructions are allowed. Consequently, it is usually not sufficient to prove correctness alone (as for a theorem) but also finiteness. Later in this book we shall encounter *enumerating procedures*. Roughly speaking, we think of them as algorithms with only partial finiteness proofs. You will not find Turing machines or recursive sets mentioned anywhere because for all algorithms and procedures we explain there exist practical implementations (for instance in CoCoA), so that a theoretical consideration of their workability is superfluous.

Finally, in Volume 1 we wrote that we did not cite anything anywhere. This is still true. However, if you have devoured this book and crave for more, you can find some suggestions for further reading in Appendix D.

0.4 Are There any Applications of This Theory?

There is no branch of mathematics, however abstract,
which may not some day be applied
to phenomena of the real world.
(Nikolai Lobachevsky)

Allow us to be bold and brazen: Computational Commutative Algebra is going to be one of the pillars of the applications of mathematics to the real world in the third millennium. With the advance of computing technology it has become feasible to grapple with industrial sized problems using symbolic computations. One of the main purposes of this volume is to continue laying the theoretical foundations for this development. Scattered throughout it you can discover clues and suggestions relating the topics we discuss to actual applications.

One aspect of this trend is nevertheless conspicuously absent here: symbolic methods should really be viewed as complementary to the numerical methods which have predominated up to now. Although a few computer algebra systems, including CoCoA, offer you some possibilities of performing *hybrid* computations, i.e. computations mixing symbolic and numerical methods, the area of numerical polynomial algebra is still in its infancy. Therefore we leave to future treatises the questions of how to represent measured data using numerical polynomials and of how to compute Gröbner bases, border bases, SAGBI bases etc. with numerical polynomials. Let us just say that actual industrial applications of Computational Commutative Algebra exist, are being continually developed, and will become more important in the coming years.

0.5 How Was This Book Written?

Those are my principles,
and if you don't like them ...
well, I have others.
(Groucho Marx)

A short answer to this question is that this book was written using LaTeX and the Springer macro package. Another quick reply is that this book is written in the same style as Volume 1. Colloquially speaking, it contains more of the same, much more. Furthermore, we tried to adhere to the rules formulated in Section 0.7 of the first volume. Browsing this second volume, you will also notice that we spruced it up with a rather generous sprinkling of quotes and occasional literary efforts. Although this may be rare in a mathematical book, we believe that a good mathematical joke is better than a dozen mediocre papers. Moreover, some things are so serious that you can

only joke about them. And if you don't like our quotes ..., well, we have others.

One of the problems we faced when we wrote Volume 1 is still as unsolved as ever, namely the problem of differing notations. The following case in point arose when we drafted Section 6.4. In accordance with books in commutative algebra, we called a non-empty set of terms an *order ideal* if it is closed under forming divisors. When we were criticized for this choice of name, we did some research and found out that the same concept had been "discovered" in several branches of mathematics. Of course, everybody had "introduced" a different name, and few authors seemed to be aware of the alternative terminologies. The following table summarizes our findings.

	order ideal	Computational Commutative Algebra
1)	canonical term basis	computer algebra
2)	Δ-set	coding theory
3)	Ferrer diagram	differential algebra
4)	footprint	coding theory
5)	normal set	numerical mathematics
6)	poset ideal	combinatorics
7)	set of terms connected to 1	computer algebra
8)	staircase	algebraic geometry
9)	standard set	commutative algebra
10)	term filter	set theory

After the publication of the first volume, some questions arose concerning the prerequisites for reading these books. Not surprisingly, for Volume 2 we assume that you have mastered the essential parts of Volume 1. But what do you need to know to be able to read Volume 1? In principle, a good working knowledge of basic algebraic structures and techniques should suffice: groups, rings, modules and fields, as well as homomorphisms, residue classes, exact sequences and a few basic proof methods. Although we tried to keep everything as self-contained as possible, it surely won't hurt if you peek into a standard algebra textbook (e.g. [La70]) in case of need. At times we also assume that you apply your common sense and produce a reasonable guess at what we mean, rather than expecting a formal definition of every single word. Isn't it clear what a *subtuple* should be?

Finally, let us point out one "rule" which has not yet been mentioned. Normally, every section or subsection in this book has some global hypotheses. Unless we specifically mention something else, they apply to everything in that section or subsection. We tried not to change these global hypotheses frequently during the course of the presentation. For propositions and theorems we deem important, we repeat the global hypotheses to aid the reader in looking up references. Of course we were not able to impose this rule unwaveringly. There are no exceptions to the rule that every theorem likes to be an exception to the rule.

0.6 What Is a Tutorial?

The hardest thing to understand is the income tax.
(Albert Einstein)

The second hardest thing to understand could be some of the tutorials in this book. Fortunately, there exists another rule: it is not necessary to do and understand the tutorials in order to continue reading the main text. Besides containing links to a wide variety of applications, tutorials provide fodder for student projects, lab classes, term papers, and so on. But teachers beware! The complexity of solving a tutorial may vary dramatically from one to the next. Some tutorials require a lot of programming, while others expect the reader to find rather tricky proofs. For some tutorials you can find the solution in research papers or other literature; maybe there even exists a pointer in Appendix D.

So, what are the tutorials good for? As we wrote in Volume 1, they are mainly intended to help bridge the gap between the theory and actual computations. What is computer algebra without a computer? Just another dry part of algebra. Gröbner bases, gradings, Hilbert functions, all these notions only come to life when you actually compute them. The tutorials should entice you into experimenting, playing with CoCoA, computing special cases, and thinking about what is going on inside the machine. If you use this book for teaching, we believe it is important that you try your hand at the tutorials before you assign them. Do not skip the computational parts of this book. Do not teach according to the motto:

He who can, does. He who cannot, teaches.

0.7 What Is CoCoA?

With computers you can
waste time a lot faster.
(Darryl Mc Cullough)

If you want to heed our advice and do some actual computations, you need access to a computer and a computer algebra system. As for the computer, we are not offering advice. But as for the computer algebra system, we suggest that you visit the web page

`http://cocoa.dima.unige.it/`

and download the free computer algebra system CoCoA. In fact, we hope that you have already done so.

Naturally there are many other computer algebra systems that you can use to implement the algorithms in this book or to solve the programming parts of the tutorials. If you are expert enough, using several computer algebra

systems in parallel certainly has advantages. However, if you are struggling to come to grips with just one computer algebra system, then CoCoA is a very good choice.

To help you master CoCoA usage and programming, we have given some basic information in Volume 1. In this volume we have added appendices describing its graphical interface and further programming techniques. If this is still not enough for you, have a look at the on-line CoCoA tutorial and the above web site.

Before plunging into action, however, you should carefully plan your time table. Although the speed of time is one-second per second, when you sit in front of your computer time passes a lot faster. And when you are playing with CoCoA, you `Can't Stop!`

0.8 And What Is This Book Good for?

You can fool all the people some of the time,
and some of the people all the time,
but you cannot fool all the people all of the time.
(Abraham Lincoln)

This book is good for learning, teaching, reading, and most of all, enjoying the topic at hand. The theories it describes can be applied to anything from children's toys to oil production. Even browsing it superficially will show you that it is different from other books on mathematics, except for Volume 1 of course. We fool around much more! It is so different that it didn't fit into any of the regular book series of Springer Verlag and had the chance to get its own colourful cover.

In earnest, this book is primarily intended as a textbook for advanced courses in Computational Commutative Algebra. We don't think that we can fool you into believing that the material we cover is easy. But we hope the presentation is comprehensible and pleasant enough to tempt you to read on. Books and research papers in computer algebra differ even from other works in mathematics. Since computers are very difficult to fool, the standard of correctness of a computer algebra text has to be raised until the implementation of the theorems and algorithms works. This is a high mark. We hope we have achieved it.

The advice to go through this book with an open and critical mind applies to Volume 2 as much as it did to Volume 1. Suppose you are playing with CoCoA and you discover the following series of calculations:

$$\begin{array}{lll} 7 \times 15 = 105 & \text{and} & 10 + 5 = 15 \\ 7 \times 18 = 126 & \text{and} & 12 + 6 = 18 \\ 7 \times 21 = 147 & \text{and} & 14 + 7 = 21 \\ 7 \times 24 = 168 & \text{and} & 16 + 8 = 24 \\ 7 \times 27 = 189 & \text{and} & 18 + 9 = 27 \end{array}$$

Have you unearthed a wonderful new method of checking multiplication tables? Not until you provide an actual formulation and a proof of the theorem you claim to have found. This is what this book is really good at: it contains formulations and (hopefully) correct proofs of many results which are "folklore" or otherwise hard to nail down.

You cannot fool all the people all of the time. At some point somebody has to sit down and implement an algorithm for computing graded free resolutions, or a procedure for computing SAGBI bases. And no amount of friendly but imprecise advice is going to deceive that person into believing this is easy: his computer would quickly disabuse him of any such misconception. In this sense, we hope you will find the combination of Volumes 1 and 2 not only valuable for learning or teaching, but also as an ultra-explicit compendium of the state of the art in Computational Commutative Algebra.

0.9 Some Final Words of Wisdom

Drillers shape holes,
bow makers shape arrows,
carpenters shape wood,
and the wise man shapes himself.
(Siddhartha Gotama)

After having typed about 900 pages of densely written mathematics, we still cannot answer profound philosophical questions such as: What is the deeper meaning of Computational Commutative Algebra? How is it going to develop in the future? Instead of philosophizing endlessly, let us end this introduction and send you off to Chapter 4 with a few words of wisdom by Peter Taylor.

> *My premise is that we mathematicians are sitting on a gold mine. In terms of structural beauty, stunning insights, unexpected power, all from simple, accessible ingredients, very little can compare with our wonderful subject. But we do a terrible job at communicating that to most of our students. In the classroom we blow it, and we thereby alienate just about the entire population. And we've no one to blame but ourselves.*

Thus the mathematician shapes everybody. May this book shape you into a fan of Computational Commutative Algebra. May it be a gold mine of material for studying and teaching. May it help you appreciate the beauty and the usefulness of our wonderful subject.

This is not the end
or the beginning of the end,
but it is the end of the beginning.
(Sir Winston Churchill)

http://www.springer.com/978-3-540-25527-7

Computational Commutative Algebra 2
Kreuzer, M.; Robbiano, L.
2005, X, 586 p., Hardcover
ISBN: 978-3-540-25527-7