Chapter 1
Computer Algebra Systems

First attempts to use computers for calculations not only with numbers but also with mathematical expressions (e.g., symbolic differentiation) were made in the 1950s. In the 1960s research in this direction became rather intensive. This area was known under different names: symbolic calculations, analytic calculations, and computer algebra. Recently this last name is most widely used. Why algebra and not, say, calculus? The reason is that it is most useful to consider operations usually referred to calculus (such as differentiation) as algebraic operations in appropriate algebraic structures (differential fields).

First universal (i.e., not specialized for some particular application area) computer algebra systems appeared at the end of the 1960s. Not many such systems have been constructed; they are shown in the Table 1.1. Creating a universal computer algebra system is a huge amount of work, at the scale of hundreds of man-years. Some projects of this kind were not sufficiently developed and subsequently died; they are not shown in Table 1.1.

Table 1.1 Universal computer algebra systems

System	Year	Implementation language	Current name	Status
REDUCE	1968	Lisp		Free (BSD)
Macsyma	1969		Maxima	Free (GPL)
Scratchpad	1974		Axiom OpenAxiom FriCAS	Free (BSD)
muMATH	1979		Derive	Dead
Maple	1983	C, C++		Proprietary
Mathematica	1988			Proprietary
MuPAD	1992		MATLAB symbolic toolbox	Proprietary

Theoretical physicist A. Hearn (known to specialists for the Drell–Hearn sum rule) has written a Lisp program REDUCE to automatize some actions in

A. Grozin, *Introduction to Mathematica® for Physicists*, Graduate Texts in Physics,
DOI 10.1007/978-3-319-00894-3_1,

calculating Feynman diagrams. It quickly grew into a universal system. At first, it was distributed free (it was sufficient to ask for Hearn's permission) and became widely used by physicists. Later it became commercial. At the end of 2008 it has become free, with a modified BSD license.

Macsyma was born in the MAC project at MIT (1969), the name means MAC SYmbolic MAnipulator. The project has nothing to do with Macintosh computers, which appeared much later. Its name had several official meanings (Multiple-Access Computer, Man And Computer, Machine Aided Cognition) and some unofficial ones (Man Against Computer, Moses And Company, Maniacs And Clowns, etc.). The work was done on a single PDP-6, later PDP-10 computer (about 1 MByte memory; there were no bytes back then, but 36-bit words). One of the first time-sharing operating systems, ITS, was written for this computer, and many users at once worked on it interactively. Later this computer became one of the first nodes of ARPANET, the ancestor if Internet, and users from other universities could use Macsyma.

The company Symbolics was spun off MIT. It produced Lisp machines—computers with a hardware support of Lisp, as well as software for these computers, including Macsyma—the largest Lisp program at that time. Later production of Lisp machines became unprofitable, because general-purpose workstations (Sun, etc.) became faster and cheaper. Symbolics went bankrupt; Macsyma business was continued by Macsyma Inc., who sold Macsyma for a number of platforms and operating systems. Its market share continued to shrink because of the success of Maple and *Mathematica*, and finally the company was sold in 1999 to Andrew Topping. The new owner stopped Macsyma development and marketing. Then he died, and the rights to the commercial Macsyma now belong to his inheritors. All efforts spent on improving this branch of Macsyma are irreversibly lost.

Fortunately, this was not the only branch. Macsyma development at MIT was largely funded by DOE, and MIT transferred this codebase to DOE who distributed it. This version was ported to several platforms. All these ports died except one. Professor William Schelter ported DOE Macsyma to Common Lisp, the new Lisp standard, and developed this version until he died in 2001. This version was called Maxima, to avoid trademark problems. In 1998 he obtained permission from DOE to release Maxima under GPL. He also developed GCL (GNU Common Lisp). Currently Maxima is an active free software project and works on many Common Lisp implementations.

Macsyma has played a huge role in the development of computer algebra systems. It was the first system in which modern algorithms for polynomials, integration in elementary functions, etc., were implemented (REDUCE and Macsyma influenced each other strongly and are rather similar to each other). Macsyma was designed as an interactive system. For example, if the form of an answer depends on the sign of a parameter, it will ask the user

Is *a* positive or negative?

Scratchpad was born in IBM research laboratories (1974). At first it did not differ from other systems (Macsyma, REDUCE) very much and borrowed chunks of code from them. It was radically redesigned in the version Scratchpad II (1985).

And this design, perhaps, still remains the most beautiful one from a mathematical point of view. It is a strongly typed system (the only one among universal computer algebra systems). Any object (formula) in it belongs to some domain (e.g., it is a single-variable polynomial with integer coefficients). Each domain belongs to some category (e.g., it is a ring, or a commutative group, or a totally ordered set). New domains can be constructed from existing ones. For example, a matrix of elements belonging to any ring can be constructed. It is sufficient to program a matrix multiplication algorithm once. This algorithm calls the operations of addition and multiplication of the elements. If matrices of rational numbers are being multiplied, then addition and multiplication of rational numbers are called; and if matrices of polynomials—then addition and multiplication of polynomials.

Scratchpad was never distributed to end users by IBM. At last, IBM decided to stop wasting money for nought (or for basic research) and sold Scratchpad II to the English company NAG (famous for its numerical libraries). It marketed this system under the name Axiom. However, the product did not bring enough profit and was withdrawn in 2001. Axiom development took about 300 man-years of work of researchers having highest qualification. All this could easily disappear without a trace. Fortunately, one of old-time Scratchpad II developers at IBM, Tim Daly, has succeeded in convincing NAG to release Axiom under the modified BSD license. Now it is a free software project and still the most beautiful system from mathematical point of view. But unfortunately, due to incompatible visions of the directions of the future development, two forks appeared—OpenAxiom and FriCAS. And it is not clear which one is better.

muMATH (Soft Warehouse, Hawaii, 1979) got to the list of universal computer algebra systems with some stretch. It was written for microprocessor systems with a very limited memory (later called personal computers); mu in its name, of course, means μ, i.e., micro. This system never implemented advanced modern algorithms. It used heuristic methods instead, as taught in university calculus courses: let's try this and that, and if you can't get it, you can't get it. But it was surprisingly powerful at its humble size. The system has been essentially rewritten in 1988 and got a menu interface, graphics, and the new name, Derive. Then Soft Warehouse was bought by Texas Instruments, who presented a calculator with a (Derive-based) computer algebra system in 1995. Derive was withdrawn from market in 2007.

All these systems can be referred to the first generation. They are all written in various dialects of Lisp. They were considered related to the area of artificial intelligence.

The first representative of the second generation is the Canadian system Maple. It has a small kernel written in C, which implements an interpreted procedural language convenient for writing computer algebra algorithms. The major part of its mathematical knowledge is contained in the library written in this language. Maple can work on many platforms. It quickly became popular. In 2009 Maplesoft (Waterloo Maple Inc.) has been acquired by the Japanese company Cybernet Systems Group; development of Maple is not affected. By the way, numerical program MathCAD used a cut-down version of Maple to provide some computer algebraic capabilities.

In the beginning of the 1980s, a young theoretical physicist Steven Wolfram, an active Macsyma user, together with a few colleagues, has written a system SMP (Symbolic Manipulation Program). The project was a failure (I still have a huge SMP manual sent to me by S. Wolfram). After that, he understood what mass users want—they want a program to look pretty. He, together with a few colleagues, has rewritten the system, paying a lot of attention to the GUI and graphics (the symbolic part was largely based on SMP). The result was *Mathematica*, version 1 (1988). And Wolfram got his first million in three months of selling it.

Mathematica heavily relies on substitutions. Even a procedure call is a substitution. Pattern matching and their replacing by right-hand sides of substitutions are highly advanced in *Mathematica*. Often a set of mathematical concepts can be easily and compactly implemented via substitutions. On the other hand, this can lead to inefficiency: pattern matching is expensive.

The latest arrival in the list of universal computer algebra systems is MuPAD (its name initially meant Multi-Processor Algebra Data tool, and indeed early versions contained experimental support of multiprocessor systems, which later disappeared). The system was designed and implemented by a research group at the University of Paderborn in Germany (this is one more meaning of PAD in the name) in 1992 and later was distributed commercially by the company SciFace. Initially, MuPAD was quite similar to Maple. Later it borrowed many ideas from Axiom (domains, categories; however, MuPAD is dynamically typed). During a long period, it was allowed to download and use MuPAD Light for free; it had no advanced GUI, but its symbolic functionality was not cut down. Funding of the University project was stopped in 2005; in 2008, SciFace was bought by Mathworks, the makers of MATLAB. After that, MuPAD is available only as a MATLAB addon.

It seems that *Mathematica* dominates the market of commercial computer algebra systems, with Maple being number two. *Mathematica* is highly respected for the huge amount of mathematical knowledge accumulated in its libraries. It is not bug-free (this is true for all systems). Often it requires more resources (memory, processor time) for solving a given problem than other systems. But it is very convenient and allows a user to do a lot in a single framework.

In addition to universal systems, there are a lot of specialized computer algebra systems. Here we'll briefly discuss just one example important for theoretical physics.

In the 1960s, a well-known Dutch theoretical physicist M. Veltman, a future Nobel prize winner, has written a system Schoonschip in the assembly language of CDC-6000 computers (in Dutch Schoonschip means "to clean a ship," in a figurative sense "to put something in order," "to throw unneeded things overboard"). This system was designed for handling very long sums (millions of terms) whose size can be much larger than the main memory and is limited only by the available disk space. All operations save one are local: they are substitutions which replace a single term by several new ones. The system gets a number of terms from the disk, applies the substitution to them, and puts the results back to the disk. The only unavoidable nonlocal operation is collecting similar terms; it is done with advanced disk sorting algorithms. Built-in mathematical knowledge of the system is very limited; the

user has to program everything from scratch. Many nontrivial algorithms, such as polynomial factorization, are highly nonlocal and impossible to implement. On the other hand, this was the only system which could work with very large expressions, orders of magnitude larger than in other systems. Later Schoonschip was ported to IBM-360 (in PL/I; you can guess that this was not done by Veltman :–). Then Veltman has rewritten it from the CDC assembly language to the 680x0 assembly language. When 680x0-based personal computers (Amiga, Atari) became extinct, it became clear that something similar but more portable is needed.

In 1989 another well-known Dutch theoretical physicist, Vermaseren, has written (in C) a new system, Form. It follows the same ideology, but many details differ. It was distributed free of charge as binaries for a number of platforms; recently it became free software (GPL). Development of Form continues. A parallel version for multiprocessor computers and for clusters with fast connections now exists. Many important Feynman diagram calculations could never have been done without Schoonschip and later Form.

The percentage of theoretical physicists among authors of computer algebra systems is suspiciously high. Some of them remained physicists (and even got a Nobel prize); some completely switched to development of their systems (and even became millionaires).

In conclusion we'll discuss a couple of important computer algebra concepts. For some (sufficiently simple) classes of expressions an algorithm of reduction to a *canonical form* can be constructed. Two equal expressions reduce to the same canonical form. In particular, any expression equal to 0, in whatever form it is written, has the canonical form 0.

For example, it is easy to define a canonical form for polynomials of several variables with integer (or rational) coefficients: one has to expand all brackets and collect similar terms. What's left is to agree upon an unambiguous order of terms, and we have a canonical form (this can be done in more than one way).

It is more difficult, but possible, to define a canonical form for rational expressions (ratios of polynomials). One has to expand all brackets and to bring the whole expression to a common denominator (collecting similar terms, of course). However, this is not sufficient: one can multiply both the numerator and the denominator by the same polynomial and obtain another form of the rational expression. It is necessary to cancel the greatest common divisor (gcd) of the numerator and the denominator. Calculating polynomial gcd's is an algorithmic operation, but it can be computationally expensive. What's left is to fix some minor details—an unambiguous order of terms in both the numerator and the denominator and, say, the requirement that the coefficient of the first term in the denominator is 1, and we obtain a canonical form.

A *normal form* for a class of expressions satisfies a weaker requirement: any expression equal to 0 must reduce to the normal form 0. For example, bringing to common denominator (without canceling gcd) defines a normal form for rational expressions.

For more general classes of expressions containing elementary functions, not only canonical but even normal form does not exist. Richardson has proved that it is algorithmically undecidable if such an expression is equal to 0.