Computer Algebra — Problems and Developments

Franz Winkler

RISC-Linz
Johannes Kepler Universität
Linz, Austria

An earlier version has been published in *Computer Algebra in Industry 2*, A.M. Cohen, L. van Gastel, S.V. Lunel (eds.), Wiley (1995)

Contents

Lady Augusta Ada Byron on Charles Babbage's "Analytical Engine":

"Many persons who are not conversant with mathematical studies imagine that because the business of [Babbage's Analytical Engine] is to give its results in numerical notation, the nature of its processes must consequently be arithmetical and numerical rather than algebraic and analytical. This is an error. The engine can arrange and combine its numerical quantities exactly as if they were letters or any other general symbols; and in fact it might bring out its results in algebraic notation were provisions made accordingly."

I. What is computer algebra?

attempt at a definition:

Computer algebra is that subfield of scientific computation, which

- *develops,*
- *analyses,*
- *implements, and*
- *applies*

algebraic algorithms

Computer algebra is contained in the wider field of Symbolic Computation. Computer algebra and computational logic, i.e. computer supported reasoning, are the most important subfields of symbolic computation. But really, symbolic computation is not so much a subfield of mathematics or computer science, but actually a new systematically constructive approach to these fields.

Some characteristic features of computer algebra

- computation in algebraic structures

  basic domains $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}_p, \mathbb{Q}_p$ (p–adic numbers), ...

  algebraic extensions $\mathbb{Q}(\alpha)$, $p(\alpha) = 0$

  polynomials $R[x_1, \ldots, x_n]$

  rational functions $K(t_1, \ldots, t_n)$

  matrices

  differential fields $(K,')$, difference fields (K, ∇)

  finitely presented groups $\{a, b \mid aba = 1, bab = 1\}$

  ...

- manipulation of formulas

typically, in computer algebra we want to compute

$$\int \frac{x}{x^2-1}dx = \frac{1}{2}\ln|x^2-1|$$

instead of

$$\int_0^{\frac{1}{2}} \frac{x}{x^2-1}dx = 0.1438...$$

i.e. in general both the input and the output of algorithms and programs are mathematical expressions or formulas rather than numbers.

- exact computation

exact computation rather than approximate computation is the goal of computer algebra.

So, typically, we want to compute

$$(\frac{\sqrt{3}}{2}, -1/2)$$

instead of

$$(0.86602..., -0.5)$$

as the solution of the system of equations

$$\begin{aligned} x^4 + 2x^2y^2 + 3x^2y + y^4 - y^3 = 0 \\ x^2 + y^2 - 1 = 0 \end{aligned}$$

As a consequence of these characteristics, decision algorithms can be built on computer algebra methods.

Problems that can be decided by computer algebra methods include:

- factorization of polynomials
- equivalence of algebraic expressions
- solvability of systems of algebraic equations
- solvability of integration problems and differential equations problems
- validity of geometric formulae

Some areas of application for computer algebra

- the piano movers problem in robotics

  "find a path that will move a given body B from an initial position to a desired final position. Along this path the body B should not hit any obstacles"

Legal positions of B are represented as a semi–algebraic set L in $\mathbb{R}^m$, i.e. sets of the form

$$\{(x_1,\ldots,x_m) \mid p(x_1,\ldots,x_m) \mathrel{\overset{>}{\underset{<}{=}}} 0\ \}$$

combined by union, intersection, and difference.

The problem is reduced to the question:

“Can two points P_1, P_2 in a semi–algebraic set L be connected by a path, i.e. are they in the same component of L?”

This question can be decided, for instance, by Collins’ cylindrical algebraic decomposition method.

- theorem proving in elementary geometry

  scope: geometric statements, that can be described by polynomial equations

  Example:

  "the altitude pedal of the hypothenuse of a right–angled triangle and the midpoints of the three sides of the triangle lie on a circle"

hypotheses:

$$\begin{aligned}
h_1 &\equiv\ 2y_3 - y_1 = 0 \qquad\qquad (\text{E is midpoint of } \overline{\text{AC}}) \\
h_2 &\equiv\ (y_7 - y_3)^2 + y_8^2 - (y_7 - y_4)^2 - (y_8 - y_5)^2 = 0 \\
&\qquad\qquad\qquad (\overline{\text{EM}} \text{ and } \overline{\text{FM}} \text{ are equally long}) \\
&\ \vdots \\
&\ \vdots \\
h_m &
\end{aligned}$$

conclusion:

$$c \equiv (y_7 - y_3)^2 + y_8^2 - (y_7 - y_9)^2 - (y_8 - y_{10})^2 = 0$$
$$(\overline{\text{EM}} \text{ and } \overline{\text{HM}} \text{ are equally long})$$

The geometric problem is reduced to the algebraic problem:

“show that $c \in \text{radical}(h_1, \ldots, h_m)$*”*

This algebraic problem can be decided by the method of Gröbner bases.

- analysis of algebraic curves

  We consider the plane curve (tacnode) defined by the equation

  $$f(x,y) = 2x^4 + y^4 - 3x^2y - 2y^3 + y^2 = 0$$

  This curve is an irreducible curve, which can be checked by trying to factor $f(x,y)$ by any computer algebra system.

The tacnode has 2 singular points, where branches intersect. They are the solutions of the system of algebraic equations

$$\begin{aligned} f(x,y) &= 0 \\ \frac{\partial f}{\partial x}(x,y) = 8x^3 - 6xy &= 0 \\ \frac{\partial f}{\partial y}(x,y) = 4y^3 - 3x^2 - 6y^2 + 2y &= 0 \end{aligned}$$

By a Gröbner basis computation this system is transformed into the equivalent system

$$\begin{aligned} 3x^2 + 2y^2 - 2y &= 0 \\ xy &= 0 \\ x^3 &= 0 \end{aligned}$$

So the singular points are $(0,0)$ and $(0,1)$.

We get the tangents at a singular point (a, b) by moving it to the origin with the transformation $T(x, y) = (x + a, y + b)$, factoring the form of lowest degree, and applying the inverse transformation $T^{-1}(x, y) = (x - a, y - b)$.

So the tangents at $(0, 1)$ are

$$y = \sqrt{3}x + 1, \qquad y = -\sqrt{3}x + 1$$

and there is one tangent

$$y = 0$$

of multiplicity 2 at $(0, 0)$.

- modelling in science and engineering

  In science and engineering, it is common to express a problem in terms of integrals or differential equations with boundary conditions. Numerical integration leads to approximations of the values of the solution functions. But, as R.W. Hammings has written, "the purpose of computing is insight, not numbers." So instead of computing tables of values it would be much more gratifying to derive formulas for the solution functions. Computer algebra algorithms can do just that for certain classes of integration and differential equation problems.

Consider, for example, the system of differential equations

$$
\begin{aligned}
-6\frac{dq}{dx}(x) + \frac{d^2p}{dx^2}(x) - 6\sin(x) &= 0,\\
6\frac{d^2q}{dx^2}(x) + a^2\frac{dp}{dx}(x) - 6\cos(x) &= 0
\end{aligned}
$$

subject to the boundary conditions $p(0) = 0, q(0) = 1, p'(0) = 0, q'(0) = 0$. Given this information as input, any of the major computer algebra systems will derive the formal solution

$$
\begin{aligned}
p(x) &= -\frac{12\sin(ax)}{a(a^2-1)} - \frac{6\cos(ax)}{a^2} + \frac{12\sin(x)}{a^2-1} + \frac{6}{a^2},\\
q(x) &= \frac{\sin(ax)}{a} - \frac{2\cos(ax)}{a^2-1} + \frac{(a^2+1)\cos(x)}{a^2-1}
\end{aligned}
$$

for $a \notin \{-1, 0, 1\}$.

II. Some problem areas in computer algebra

Solution of algebraic equations

Solving by resultants

If $f(x), g(x)$ are polynomials (their coefficients might be polynomials themselves), the resultant of f and g, $\mathrm{res}_x(f, g)$, is a constant, the determinant of the Sylvester matrix of f and g.

Fact: $\mathrm{res}_x(f, g) = 0$ if and only if f and g have a common factor, or, in other words, there is a solution to the system of equations $f(x) = g(x) = 0$.

Example:

$$f_1 = xz - xy^2 - 4x^2 - \frac{1}{4} = 0$$
$$f_2 = y^2 z + 2x + \frac{1}{2} = 0$$
$$f_3 = x^2 z + y^2 + \frac{1}{2}x = 0$$

first we eliminate the variable z:

$$g_1(x,y) = \mathrm{res}_z(f1, f2) =$$
$$y^4 x + 4x^2 y^2 + \frac{1}{4}y^2 + 2x^2 + \frac{1}{2}x$$

$$g_1(x,y) = \mathrm{res}_z(f_2, f_3) =$$
$$y^4 - 2x^3 + \frac{1}{2}xy^2 - \frac{1}{2}x^2$$

finally we eliminate the variable y:

$$h = \mathrm{res}_y(g_1, g_2) =$$
$$\frac{1}{1024} \cdot x^4 \cdot (4x+1)^2 \cdot$$
$$(32x^5 - 216x^4 + 64x^3 - 42x^2 + 32x + 5)^2$$

So every solution of the system has to have an x–coordinate which is a root of $h(x)$. Unfortunately, not every root of $h(x)$ can be extended to a solution of the original system.

There are certain *extraneous factors* in the resultant. In our example, only the last factor gives the correct x–coordinates.

Solving by Gröbner bases

If

$$F = \{f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)\}$$

is given (as the basis for a polynomial ideal, or in other words all the linear combinations of $f_1, \ldots, f_m$, or in other words all the consequences of relating these polynomials to 0), then

the Gröbner basis of F (w.r.t. some ordering of the power products)

$$GB(F)$$

is another basis for this same ideal with certain nice properties.

Fact: The Gröbner basis of F w.r.t. a lexicographic ordering is a triangularized system of polynomials having the same solutions as the original system F. After solving the polynomials with the least number of variables in the Gröbner basis, all these solutions can be extended to solutions of the whole system.

We compute a Gröbner basis for

$$F = \{f_1, f_2, f_3\}$$

This results in the new (and equivalent) system

$$\begin{aligned}
65z + 64x^4 - 432x^3 + 168x^2 - 354x + 104 &= 0 \\
26y^2 - 16x^4 + 108x^3 - 16x^2 + 17x &= 0 \\
32x^5 - 216x^4 + 64x^3 - 42x^2 + 32x + 5 &= 0
\end{aligned}$$

The univariate polynomial $g_3(x)$ in the Gröbner basis is irreducible.

There are 5 solutions of this univariate polynomial, and therefore exactly 10 solutions of the whole system (counting multiple solutions).

Now we could either stay in an algebraic mode of computation, extend the rational numbers by a root of g_3, and solve the system in such a way.

Or we could easily approximate the roots of the system numerically. An approximation of one of the roots is

$$(-0.128475, 0.321145, -2.356718)$$

Indefinite summation

Given a sequence

$$a_1, a_2, \ldots$$

(i.e. and expression generating this sequence), we want to find an expression $S(j)$, in which the summation sign has been eliminated and such that

$$S(j) = \sum_{i=1}^{j} a_i.$$

This is a discrete analogue to indefinite integration. $S(j)$ may be thought of as an 'anti–derivative'. Having obtained $S(j)$ we then have

$$\sum_{i=m}^{n} a_i = S(n) - S(m-1)$$

where $S(0) = 0$.

E.g.

$$S(n) = \frac{3n^2 + 5n}{4(n^2 + 3n + 2)} \quad \text{for} \quad \sum_{i=1}^{\infty} \frac{1}{i(i+2)}.$$

Gosper's algorithm can produce indefinite summation for hypergeometric series, i.e. for series

$$\sum_{i=1}^{\infty} a_i$$

such that

$$\frac{a_n}{a_{n-1}}$$

is a rational function in n.

For instance,

$$\sum_{i=1}^{n} i \cdot x^i = \frac{n \cdot x^{n+2} - (n+1) \cdot x^{n+1} + x}{(x-1)^2}$$

Every major computer algebra system contains an implementation of this algorithm due to Gosper.

Indefinite integration

Let a (real or complex) function

$$f(x)$$

be given by an expression.

We want to compute another expression describing a function

$$F(x)$$

such that

$$\int f(x)dx = F(x).$$

Computer algebra is able to do just that for a certain wide class of functional expressions (*elementary functions*), which can be roughly described as

- start with rational functions $p(x)/q(x)$
- add logarithms and exponentials of expressions, and therefore trigonometric functions
- add algebraic elements, i.e. roots of polynomials

This class of functional expressions contains most of the situations treated in integral tables.

Integration of rational functions

Example:

$$R(x) = \frac{3x^{11} - 2x^{10} + 7x^9 + 2x^7 + 23x^6 - 10x^5 + 18x^4 - 9x^3 + 8x^2 - 3x + 1}{3x^9 - 2x^8 + 7x^7 - 4x^6 + 5x^5 - 2x^4 + x^3} =$$

We obtain

$$\int R(x)\,dx = \frac{2\log(3x^2 - 2x + 1)}{3} - \frac{\log(x^2+1)}{2} + \frac{4\arctan((6x-2)/2\sqrt{2})}{3\sqrt{2}} + \log(x) + \arctan(x) + \frac{4x^3 - 4x^2 + 2x - 1}{2x^4 + 2x^2} + \frac{x^3}{3}.$$

Integration of transcendental elementary functions

Risch's algorithm can take a transcendental elementary function $f(x)$, decide whether its integral can be expressed as an elementary function $g(x)$, and if so compute $g(x)$.

Example:

$$f(x) =$$
$$2x \cdot \exp(x^2) \cdot \log(x) + \frac{\exp(x^2)}{x} + \frac{\log(x) - 2}{(\log^2(x) + x)^2} +$$
$$\frac{(2/x) \cdot \log(x) + 1/x + 1}{\log^2(x) + x}$$

$$\int f(x) =$$
$$\exp(x^2) \cdot \log(x) - \frac{\log(x)}{\log^2(x) + x} + \log(\log^2(x) + x)$$

Risch's algorithm is implemented in many computer algebra systems.

Greatest common divisors of polynomials

As we have seen above, the computation of greatest common divisors (gcd) of polynomials is ubiquitous in computer algebra. Therefore we need very fast algorithms for gcd computation.

One of the problems is the explosion of the coefficients. Over the last 2 decades sophisticated algorithms have been developed for tackling this problem.

Example:
consider the two bivariate polynomials

$$f(x,y) = y^6 + xy^5 + x^3y - xy + x^4 - x^2,$$
$$g(x,y) = xy^5 - 2y^5 + x^2y^4 - 2xy^4 + xy^2 + x^2y$$

with integral coefficients. Let y be the main variable, so that the coefficients of powers of y are polynomials in x.

Euclid's algorithm yields the polynomial remainder sequence

$r_0 = f,$

$r_1 = g,$

$r_2 =$

$(2x - x^2)y^3 + (2x^2 - x^3)y^2 + (x^5 - 4x^4 + 3x^3 +$

$4x^2 - 4x)y + x^6 - 4x^5 + 3x^4 + 4x^3 - 4x^2,$

$r_3 =$

$(-x^7 + 6x^6 - 12x^5 + 8x^4)y^2 +$

$(-x^{13} + 12x^{12} - 58x^{11} + 136x^{10} - 121x^9 -$

$117x^8 + 362x^7 - 236x^6 - 104x^5 + 192x^4 - 64x^3)y$

$- x^{14} + 12x^{13} - 58x^{12} + 136x^{11} - 121x^{10} -$

$116x^9 + 356x^8 - 224x^7 - 112x^6 + 192x^5 - 64x^4,$

$$
\begin{aligned}
&r_4 = \\
&(-x^{28} + 26x^{27} - 308x^{26} + 2184x^{25} - 10198x^{24} + \\
&32188x^{23} - 65932x^{22} + 68536x^{21} + 42431x^{20} - \\
&274533x^{19} + 411512x^{18} - 149025x^{17} - 431200x^{16} \\
&+ 729296x^{15} - 337472x^{14} - 318304x^{13} + \\
&523264x^{12} - 225280x^{11} - 78848x^{10} + 126720x^{9} - \\
&53248x^{8} + 8192x^{7})y \\
&- x^{29} + 26x^{28} - 308x^{27} + 2184x^{26} - 10198x^{25} + \\
&32188x^{24} - 65932x^{23} + 68536x^{22} + 42431x^{21} - \\
&274533x^{20} + 411512x^{19} - 149025x^{18} - 431200x^{17} \\
&+ 729296x^{16} - 337472x^{15} - 318304x^{14} + \\
&523264x^{13} - 225280x^{12} - 78848x^{11} + 126720x^{10} \\
&- 53248x^{9} + 8192x^{8}.
\end{aligned}
$$

The greatest common divisor of f and g is obtained by eliminating common factors $p(x)$ in r_4.

The final result is

$$\gcd(f, g) = y + x.$$

Although the inputs and the output are small, the intermediate expressions get very big. The biggest polynomial in this computation happens to occur in the pseudo–division of r_3 by r_4. The intermediate polynomial has degree 70 in x.

Modular gcd algorithm

The most efficient algorithm for computing gcd's of multivariate polynomials is a modular algorithm. The basic idea is to apply homomorphisms to the coefficients, compute the gcd's of the evaluated polynomials, and use the *Chinese remainder algorithm* to reconstruct the actual coefficients in the gcd.

If the input polynomials are univariate, we can take homomorphisms H_p, mapping an integer a to a mod p. If the input polynomials are multivariate, we can take evaluation homomorphisms of the form $H_{x_1=r_1}$ for reducing the number of variables.

In our example we get

$$\begin{aligned}\gcd(H_{x=2}(f), H_{x=2}(g)) &= y+2,\\ \gcd(H_{x=3}(f), H_{x=3}(g)) &= y+3.\end{aligned}$$

So the gcd is $y+x$. Never during this algorithm did we have to consider big coefficients.

Factorization of polynomials

Like gcd computation, factorization of polynomials is a basic building block of computer algebra algorithms.

For factoring a univariate polynomial

$$f(x)$$

with integer coefficients, we proceed in the following way:

- factor $f(x)$ modulo a prime p
- determine a bound B for the coefficients of factors of f and an integer k s.t. $p^k > B$
- lift the factorization of f mod p to a factorization of f mod p^k
- combine factors in order to get the factors over the integers

Example: consider the polynomial

$$f(x) = 6x^7 + 7x^6 + 4x^5 + x^4 + 6x^3 + 7x^2 + 4x + 1.$$

Taking $p = 5$, we get the factorization

$$f \equiv (x-2)(x^2-2)(x^2+2)(x^2-x+2) \pmod{5}.$$

Lifting this to a factorization modulo 5^2 we obtain

$$f \equiv (2x+1)(x^2-7)(x^2+7)(3x^2+2x+1) \pmod{25}.$$

Now we can combine the 2nd and 3rd factors to obtain the factorization over the integers

$$f = (2x+1)(x^4+1)(3x^2+2x+1).$$

Simplification of expressions

As we have seen above, the swell of intermediate expressions can be a serious problem in computer algebra algorithms.

This may lead to exhaustion of the storage space and it may make the final result all but unreadable.

Therefore, all the expressions and formulas need to be “simplified”.

There a basically two approaches to simplification:

non–canonical simplification: the expressions are somehow made shorter or more readable. However, it is not always easy to determine what "simpler" should mean. E.g.

$$\frac{x^2-1}{x-1} \sim x+1,$$

$$\text{but}$$

$$\frac{x^{100}-1}{x-1} \sim x^{99}+\ldots$$

canonical simplification: formulas which are equivalent are simplified to the same normal form. If that is possible, then the underlying equivalence can be decided by reducing to normal forms and checking for syntactical equality.
This is for instance possible for equivalences described by polynomial equations (Gröbner bases).
However, for some other very simple classes of formulas this is theoretically impossible.

III. Limitations of computer algebra

(Un)Decidable simplification problems

Simplification problem for radical expressions

Radical expressions are built from
- variables $x_1, \dots, x_n$
- rational constants
- the arithmetic function symbols $+, -, \cdot, /$
- and the radical sign $\sqrt[q]{\ }$, or, equivalently, rational powers (*"radicals"*) s^r for $r \in \mathbb{Q}$

Two radical expressions are *equivalent* iff they describe the same (meromorphic) functions.

So, for instance,

$$\frac{\sqrt{2}}{\sqrt{x+1} \cdot \sqrt[4]{24x+24}} \quad \sim \quad \frac{\sqrt[4]{6^3} \cdot \sqrt[4]{x+1}}{6x+6}$$

The equivalence of unnested radical expressions (radicals do not contain other radicals) can be decided by an algorithm due to Caviness and Fateman.

Simplification probl. for transcendental expressions

A certain class of transcendental expressions, R2, is defined in the following way:

- one variable x
- rational constants
- π
- function symbols $+, \cdot, \sin, \mathrm{abs}$

Two expressions are equivalent if and only if they describe the same functions on $\mathbb{R}$.

Based on work by Richardson and Matijasevic's proof of the undecidability of Hilbert's tenth problem, Caviness has shown that the equivalence of expressions in R2 is undecidable.

Problem of constants

In algorithms for indefinite integration we need to build up a tower of fields in which the integrand can be expressed, e.g.

$$x \cdot e^x \quad \in \quad \mathbb{Q}(x, \exp(x)).$$

A new exponential or logarithm, e.g. $\exp(x+1)$ may fail to be in the previous field by virtue of just a constant.
It may appear reasonable to enlarge the constant field to

$$\mathbb{Q}(\exp(1)).$$

The problem is, that another such adjoined constant, e.g. $2\pi i$, may be algebraic over this field without our knowledge. This, of course, means that we cannot affirmatively decide whether a constant is zero or not. Our integration algorithm might then output a closed form solution which is none since one of the produced denominators vanishes.

Various theorems and conjectures may help tackle particular cases:

Theorem (Lindemann, 1882): If $a_1, \ldots, a_n \in \overline{\mathbb{Q}}$ are linearly independent over $\mathbb{Q}$, then $\exp(a_1), \ldots, \exp(a_n)$ are algebraically independent over $\mathbb{Q}$, i.e.

$$\text{tr deg}_{\mathbb{Q}}(\mathbb{Q}(\exp(a_1), \ldots, \exp(a_n))) = n.$$

Schanuel's conjecture: If $c_1, \ldots, c_n \in \mathbb{C}$ are linearly independent over $\mathbb{Q}$, then

$$\text{tr deg}_{\mathbb{Q}}(\mathbb{Q}(c_1, \ldots, c_n, \exp(c_1), \ldots, \exp(c_n))) \geq n.$$

This conjecture implies that e and π are algebraically independent, but even this has not been established.

Rewrite rules in computer algebra

The computer algebra system *Mathematica* uses pattern matching and transformation rules / rewrite rules as a basic concept underlying all other programming constructs, control structures, procedure and function definitions.

So, for instance, the definition of our own natural logarithm `log` may look like

```
log[E]=1
log[1]=0
log[x_ y_] := log[x] + log[y]
log[x_^y_] := y log[x]
```

There are several problems in connection with such rewrite rules. The two most important ones are

- termination and
- completeness.

Termination of systems of rewrite rules

Given a TRS (term rewriting system) R, we want to know whether all sequences of rewrites finally terminate.

This is an undecidable problem !!!

Reason: the halting problem for Turing machines can be reduced to the problem of termination of certain TRS's

Completeness of TRSs

a TRS R is basically a system of equations E together with an orientation of these equations. Whereas in general transformations w.r.t. E can be done in both directions, transformations w.r.t. R are allowed only from left to right.

So the question is whether all equivalences

$$s = t$$

which can be proved by E can also be proved by using only the rules in R.

Example: theory of free groups
In mathematics a *group* is a set with operations 1 (arity 0), $\cdot$ (arity 2), $^{-1}$ (arity 1), satisfying the group axioms

$$\begin{array}{ll} \text{(G1)} & 1 \cdot x = x, \\ \text{(G2)} & x^{-1} \cdot x = 1, \\ \text{(G3)} & (x \cdot y) \cdot z = x \cdot (y \cdot z). \end{array}$$

We are interested in deciding equations of the form $s = t$, where both s and t are terms constructed from variables and the group operations, for instance

$$(x^{-1} \cdot (x \cdot y))^{-1} = (x^{-1} \cdot y)^{-1} \cdot x^{-1} \quad ?$$

Orienting the group axioms leads to the rewrite rule system RG:

$$\begin{array}{ll} (RG1) & 1 \cdot x \to x, \\ (RG2) & x^{-1} \cdot x \to 1, \\ (RG3) & (x \cdot y) \cdot z \to x \cdot (y \cdot z). \end{array}$$

Unfortunately, the above terms cannot be shown to be equivalent using RG.

However, we could give RG as input to the Knuth–Bendix completion procedure, which produces an equivalent TRS RG' (corresponding to the same equational theory G), and which is complete in the sense that every equation that can be proved by G can also be proved by reductions in RG'.

(RG'1)	$1 \cdot x \to x$
(RG'2)	$x^{-1} \cdot x \to 1$
(RG'3)	$(x \cdot y) \cdot z \to x \cdot (y \cdot z)$
(RG'4)	$x^{-1} \cdot (x \cdot y) \to y$
(RG'5)	$x \cdot 1 \to x$
(RG'6)	$1^{-1} \to 1$
(RG'7)	$(x^{-1})^{-1} \to x$
(RG'8)	$x \cdot x^{-1} \to 1$
(RG'9)	$x \cdot (x^{-1} \cdot y) \to y$
(RG'10)	$(x \cdot y)^{-1} \to y^{-1} \cdot x^{-1}$

Using RG' we can prove the equivalence

$$(x^{-1} \cdot (x \cdot y))^{-1} = (x^{-1} \cdot y)^{-1} \cdot x^{-1}$$

The Knuth–Bendix procedure, however, may or may not terminate for a given TRS.

So in general, it might be that no complete system can be constructed which would let us decide the equivalence introduced by a set of rewrite rules.

IV. History of CA systems

- **SAC**: started approx. 1965
  author: G.E. Collins
  Univ. of Wisconsin at Madison, now RISC-LINZ
  development of fast polynomial algorithms

- **MACSYMA**: started approx. 1968
  author: J. Moses
  Massachusetts Institute of Technology
  big library of algebraic algorithms

- **REDUCE**: started approx. 1968
  author: A. Hearn
  Univ. of Utah, now Rand Corp.
  special functions for physics

- **AXIOM**: started approx. 1978
  author: R.D. Jenks
  IBM Yorktown Heights
  generic algorithms

- **DERIVE**: started approx. 1985
  author: R. Stoutemyer
  Univ. of Hawaii
  designed for small computers

- **MAPLE**: started approx. 1980
  authors: K. Geddes and G. Gonnet
  Univ. of Waterloo and (now) ETH Zürich
  small kernel, accommodates many users

- **MATHEMATICA**: started approx. 1985
  author: S. Wolfram
  Wolfram research
  good interface to numerical and graphical comp.

V. Conclusions and future developments

Symbolic computation is the subarea of mathematics and computer science which solves problems on symbolic objects representable on a computer. Typical examples of such objects are algebraic expressions, logical propositions, and programs themselves. The problem solutions are integrated in many advanced software systems for computer algebra, computer aided design and manufacturing, computer supported reasoning, knowledge management, and formal system specifications and verification. Besides playing a fundamental role within mathematics itself, symbolic computation is thus a key technology in many scientific and technical areas.

Within mathematics, the wide availability of computer algebra systems such as Maple or Mathematica has stimulated a variety of new developments towards algorithms and constructiveness. Examples range from classical algebra to analysis, combinatorics, number theory and beyond. The new possibilities to do mathematics with the computer give rise to numerous future challenges for research in symbolic computation. To name only a few examples:

- the combination of numerical and symbolic scientific computing,
- the design of automated systems integrating the mathematical tasks of proving and solving,
- or the development of methods for computer-assisted mathematical knowledge management.

Symbolic algorithms and their implementations are going to play a more and more significant role both in natural sciences and in industrial engineering. Typical examples are

- Gröbner bases in robotics,
- geometry of curves and surfaces in geometric modelling and optimization,
- symbolic methods for differential problems.

Extrapolating from these recent developments, one may anticipate that symbolic computation is going to change the paradigm of how mathematical research is done, how mathematics can be taught and applied and how mathematical knowledge is organized, stored, and made accessible. Thus symbolic computation is not just one particular, special branch of mathematics but a new paradigm that penetrates and changes the entire field of mathematics and its applications.