Wolfram *Mathematica*® Tutorial Collection

ADVANCED ALGEBRA

For use with Wolfram *Mathematica*® 7.0 and later.

For the latest updates and corrections to this manual:
visit reference.wolfram.com

For information on additional copies of this documentation:
visit the Customer Service website at www.wolfram.com/services/customerservice
or email Customer Service at info@wolfram.com

Comments on this manual are welcomed at:
comments@wolfram.com

Content authored by:
Adam Strzebonski and John Novak

Printed in the United States of America.

15 14 13 12 11 10 9 8 7 6 5 4 3 2

Contents

Complex Polynomial Systems

Introduction

The *Mathematica* functions `Reduce`, `Resolve`, and `FindInstance` allow you to solve a wide variety of problems that can be expressed in terms of equations and inequalities. The functions use a collection of algorithms applicable to classes of problems satisfying particular properties, as well as a set of heuristics that attempt to reduce the given problem to a sequence of problems that can be solved using the algorithms. This tutorial describes the algorithms used to solve the class of problems known as complex polynomial systems. It characterizes the structure of the returned answers and describes the options that affect various aspects of the methods involved.

A complex polynomial system is an expression constructed with polynomial equations and inequations

$$f(x_1, \ldots, x_n) == g(x_1, \ldots, x_n) \text{ and } f(x_1, \ldots, x_n) \neq g(x_1, \ldots, x_n)$$

combined using logical connectives and quantifiers

$$\Phi_1 \bigwedge \Phi_2,\ \Phi_1 \bigvee \Phi_2,\ \Phi_1 \Rightarrow \Phi_2, \neg \Phi, \forall_x \Phi, \text{ and } \exists_x \Phi.$$

An occurrence of a variable x inside $\forall_x \Phi$ or $\exists_x \Phi$ is called a bound occurrence, and any other occurrence of x is called a free occurrence. A variable x is called a free variable of a complex polynomial system if the system contains a free occurrence of x. A complex polynomial system is quantifier-free if it contains no quantifiers.

Here is an example of a complex polynomial system with free variables x, y, and z.

$$x^2 + y^2 == z^2 \bigwedge \exists_t \left(\forall_u\, t\, x \neq u\, y\, z + 7 \bigvee x^2\, t == 2\, z + 1\right) \tag{1}$$

In *Mathematica,* quantifiers are represented using the functions `Exists` (∃) and `ForAll` (∀).

Any complex polynomial system can be transformed to the prenex normal form

$$Q_{1\, x_1}\, Q_{2\, x_2} \ldots \mathrm{Q}_{n\, x_n}\, \Phi(x_1, \ldots, x_n; y_1, \ldots, y_m),$$

where each Q_i is a quantifier ∀ or ∃, and $\Phi(x_1, \ldots, x_n; y_1, \ldots, y_m)$ is quantifier-free.

Any quantifier-free complex polynomial system can be transformed to the disjunctive normal form

$$(\varphi_{1,1} \wedge \ldots \wedge \varphi_{1,n_1}) \vee \ldots \vee (\varphi_{m,1} \wedge \ldots \wedge \varphi_{m,n_m}),$$

where each $\varphi_{i,j}$ is a polynomial equation or inequation.

`Reduce`, `Resolve`, and `FindInstance` always put complex polynomial systems in the prenex normal form, with quantifier-free parts in the disjunctive normal form, and subtract the sides of the equations and inequations to put them in the form

$$f(x_1, \ldots, x_n) == 0 \text{ and } f(x_1, \ldots, x_n) \neq 0.$$

In all the tutorials for complex polynomial system solving, assume that the system has been transformed to this form.

`Reduce` can solve arbitrary complex polynomial systems. The solution (possibly after expanding $\wedge$ with respect to $\vee$) is a disjunction of terms of the form

$$\begin{gathered} x_1 = r_1 \wedge g_1(x_1) \neq 0 \wedge x_2 = r_2(x_1) \wedge g_2(x_1, x_2) \neq 0 \wedge \ldots \\ {}_{-1}(x_1, \ldots, x_{n-1}) \neq 0 \wedge x_n = r_n(x_1, \ldots, x_{n-1}) \wedge g_n(x_1, \ldots, x_n) \neq 0, \end{gathered} \tag{2}$$

where $x_1, \ldots, x_n$ are the free variables of the system, each g_i is a polynomial, each r_i is an algebraic function expressed using radicals or `Root` objects, and any terms of the conjunction (2) may be absent. Each $r_i(x_1, \ldots, x_{i-1})$ is well defined, that is, no denominators or leading terms of `Root` objects in r_i become zero for any $(x_1, \ldots, x_{i-1})$ satisfying the preceding terms of the conjunction (2).

This solves the system (1).

```
In[1]:= Reduce[x^2 + y^2 == z^2 && ∃t (∀u t x ≠ u y z + 7 || x^2 t == 2 z + 1), {x, y, z}]
Out[1]= (y == 0 && z == -x) || (y == 0 && z == x) || ((y == -i x || y == i x) && z == 0) ||
  ((z == -Sqrt[x^2 + y^2] || z == Sqrt[x^2 + y^2]) && x ≠ 0) || (x == 0 && y == -1/2 && z == -1/2) || (x == 0 && y == 1/2 && z == -1/2)
```

`Resolve` can eliminate quantifiers from arbitrary complex polynomial systems. If no variables are specified, the result is a logical combination of terms

$$f(x_1, \ldots, x_n) == 0 \text{ and } g(x_1, \ldots, x_n) \neq 0,$$

where f and g are polynomials, and each x_i is a free variable of the system. With variables specified in the input, `Resolve` gives the same answer as `Reduce`.

This eliminates quantifiers from the system (1).

```
In[2]:= Resolve[x^2 + y^2 == z^2 && ∃t (∀u t x ≠ u y z + 7 || x^2 t == 2 z + 1)]
Out[2]= (y == 0 && x - z == 0) || (y == 0 && x + z == 0) || (x^2 + y^2 == 0 && z == 0) || (x == 0 && -1 + 2 y == 0 && 1 + 2 z == 0) ||
  (x == 0 && 1 + 2 y == 0 && 1 + 2 z == 0) || (x^2 + y^2 - z^2 == 0 && y - z ≠ 0 && y + z ≠ 0)
```

`FindInstance` can handle arbitrary complex polynomial systems giving instances of complex solutions, or an empty list for systems that have no solutions. If the number of instances requested is more than one, the instances are randomly generated from the full solution of the system, and therefore they may depend on the value of the `RandomSeed` option. If one instance is requested, a faster algorithm that produces one instance is used, and the instance returned is always the same.

This finds a solution for the system (1).

```
In[3]:= FindInstance[x^2 + y^2 == z^2 && ∃t (∀u t x ≠ u y z + 7 || x^2 t == 2 z + 1), {x, y, z}]
Out[3]= {{x → 0, y → 0, z → 0}}
```

The main tool used in solving complex polynomial systems is the Gröbner basis algorithm [1], which is available in *Mathematica* as the `GroebnerBasis` function.

Gröbner Bases

Theory

This section gives a very brief introduction to the theory of Gröbner bases. It presents only the properties that are necessary to describe the algorithms used by *Mathematica* in solving complex polynomial systems. For a more complete presentation see, for example, [1, 2]. Note that what [2] calls a monomial, [1] calls a term, and vice versa. This tutorial uses the terminology of [1].

A monomial in $x_1, \ldots, x_n$ is an expression of the form $x_1^{e_1} \ldots x_n^{e_n}$ with non-negative integers e_i.

Let $M = M(x_1, \ldots, x_n)$ be the set of all monomials in $x_1, \ldots, x_n$. A monomial order is a linear order $\preccurlyeq$ on M, such that $1 \preccurlyeq t$ for all $t \in M$, and $t_1 \preccurlyeq t_2$ implies $t_1\, s \preccurlyeq t_2\, s$ for all $t_1, t_2, s \in M$.

Let R be a field, the domain of integers, or the domain of univariate polynomials over a field. Let $Quot$ and Rem be functions $R^2 \longrightarrow R$ defined as follows. If R is a field, $Quot(a, b) = a/b$, and $Rem(a, b) = 0$. If R is the domain of integers, $Quot$ and Rem are the integer quotient and remainder functions, with $-|b|/2 < Rem(a, b) \leq |b|/2$. If R is the domain of univariate polynomials over a field, $Quot$ and Rem are the polynomial quotient and remainder functions.

A product $t = a\,m$, where a is a nonzero element of R and m is a monomial, is called a term.

Let $\preccurlyeq$ be a monomial order on M, and let $f \in R[x_1, \ldots, x_n]\backslash\{0\}$. The leading monomial $LM(f)$ of f is the $\preccurlyeq$-largest monomial appearing in f, the leading coefficient $LC(f)$ of f is the coefficient at $LM(f)$ in f, and the leading term $LT(f)$ of f is the product $LC(f)\,LM(f)$.

A Gröbner basis of an ideal I in $R[x_1, \ldots, x_n]$, with respect to a monomial order $\preccurlyeq$, is a finite set G of polynomials, such that for each $f \in I$, there exists $g \in G$, such that $LT(g)$ divides $LT(f)$. Every ideal I has a Gröbner basis (see [1] for a proof).

Let $p \in R[x_1, \ldots, x_n]\backslash\{0\}$, and let $m \in R[x_1, \ldots, x_n]$ be a monomial. A term $t = a\,m$ is reducible modulo p, if $LM(p)$ divides m, and $a \neq Rem(a, LC(p))$. If t is reducible modulo p, the reduction of t modulo p is the polynomial

$$Red(t,\ p) = t - Quot(a,\ LC(p))\,\frac{m}{LM(p)}\,p.$$

Note that if $Rem(a, LC(p)) \neq 0$, then $LT(Red(t, p)) = Rem(a, LC(p))\,m$; otherwise, $LM\,(Red\,(t, p)) \preccurlyeq m$.

Let $f \in R[x_1, \ldots, x_n]$, and let P be an ordered finite subset of $R[x_1, \ldots, x_n]\backslash\{0\}$. f is reducible modulo P if f contains a term reducible modulo an element of P. The reduction $Red(f, P)$ of f modulo P is defined by the following procedure. While the set RT of terms of f reducible modulo an element of P is not empty, take the term $t \in RT$ with the $\preccurlyeq$-largest monomial, take the first $p \in P$, such that t is reducible modulo p, and replace the term t in f with $Red(t, p)$. Note that the monomials of terms t chosen in subsequent steps of the procedure form a $\preccurlyeq$-descending chain, and each monomial can appear at most k times, where k is the number of elements of P, hence the procedure terminates.

A Gröbner basis G is semi-reduced if for all $g \in G$, g is not reducible modulo $G\backslash\{g\}$, and if R is the domain of integers, $LC(g) > 0$.

The *Mathematica* function `GroebnerBasis` returns semi-reduced Gröbner bases. In the following discussion, all Gröbner bases are assumed to be semi-reduced. Note that this is not the same as reduced Gröbner bases defined in the literature, since here the basis polynomials are not required to be monic. For a fixed monomial order, every ideal has a unique reduced Gröbner basis. Semi-reduced Gröbner bases defined here are only unique up to multiplication by invertible elements of R (see Property 2).

Property 1: Let G be a Gröbner basis of an ideal I in $R[x_1, \ldots, x_n]$, and let $f \in R[x_1, \ldots, x_n]$. Then $f \in I$ iff $Red(f, G) = 0$.

This is a simple consequence of the definitions.

Property 2: Let $G = \{g_1, \ldots \mathrm{g}_k\}$ and $H = \{h_1, \ldots \mathrm{h}_m\}$ be two Gröbner bases of an ideal I with respect to the same monomial order $\preccurlyeq$, and suppose that elements of G and H are ordered by their leading monomials. Then $k = m$, and for all $1 \leq i \leq k$, if R is the domain of integers, $g_i = h_i$; otherwise, $g_i = c_i h_i$ for some invertible element c_i of R.

Proof: If $LM(f) = LM(g)$, then $LT(f)$ is reducible modulo g or $LT(g)$ is reducible modulo f. Hence the leading monomials of the elements of a Gröbner basis are all different. Without loss of generality, assume $k \leq m$. For induction, fix $j \leq k$ and suppose that for all $i < j$, $g_i = c_i h_i$ for some invertible element c_i of R. If R is the domain of integers, $c_i = 1$. Without loss of generality, assume $LM(g_j) \preccurlyeq LM(h_j)$. Since g_j belongs to I, there exists i such that $LT(h_i)$ divides $LT(g_j)$. Then $LM(h_i) \preccurlyeq LM(g_j)$, and so $i \leq j$. If $i < j$, then g_j would be reducible modulo h_i and also modulo $g_i = c_i h_i$, which is impossible, since G is semi-reduced. Hence $i = j$, and $LM(g_j) = LM(h_j)$, and $LT(h_j)$ divides $LT(g_j)$. Similarly, $LT(g_j)$ divides $LT(h_j)$. Therefore, there exists an invertible element c_j of R, such that $LT(g_j) = c_j LT(h_j)$. If R is the domain of integers, $LC(g_j)$ and $LC(h_j)$ are positive, and so $c_j = 1$. Let $r = c_j h_j - g_j$. Suppose $r \neq 0$. Since r belongs to I, $\mathrm{LT}(r)$ must be divisible by $LT(g_i)$, for some $i < j$. Let α and β be the coefficients at $LM(r)$ in g_j and h_j. If R is a field, the term $\alpha LM(r)$ of g_j is reducible modulo g_i, which contradicts the assumption that G is semi-reduced. If R is the domain of univariate polynomials over a field,

$$deg(LC(g_i)) \leq deg(LC(r)) \leq max(deg(\alpha), deg(\beta))$$

and so either g_j is reducible modulo g_i, or h_j is reducible modulo $h_i = c_i\, g_i$, which contradicts the assumption that G and H are semi-reduced. Finally, let R be the domain of integers. Since neither g_j is reducible modulo g_i nor h_j is reducible modulo $h_i = g_i$, $-LC(g_i)/2 < \alpha \le LC(g_i)/2$ and $-LC(g_i)/2 < \beta \le LC(g_i)/2$. Hence $-LC(g_i) < LC(r) = \beta - \alpha < LC(g_i)$, which is impossible, since $LT(r)$ is divisible by $LT(g_i)$. Therefore $r = 0$, and so $g_j = c_j\, h_j$. By induction on j, for all $j \le k$, $g_j = c_j\, h_j$. If $k < m$, then h_{k+1} would be reducible modulo some g_j, with $j \le k$, and hence h_{k+1} would be reducible modulo $h_j = c_j^{-1}\, g_j$. Therefore $k = m$, which completes the proof of Property 2.

Property 3: Let I be an ideal in $R[x_1, \ldots, x_n]$, let $f \in R[x_1, \ldots, x_n]$, and let G be a Gröbner basis of the ideal $< I, 1 - y\, f >$ in $R[x_1, \ldots, x_n, y]$. Then f belongs to the radical of I iff $G = \{c\}$ for an invertible element c of R.

If an ideal contains invertible elements of R, `GroebnerBasis` always returns $\{1\}$.

Proof: Note first that

$$1 - y^{2^k} f^{2^k} = (1 - y\, f)\,(1 + y\, f) \ldots \left(1 + y^{2^{k-1}} f^{2^{k-1}}\right)$$

belongs to the ideal $J = < I, 1 - y\, f >$ for any non-negative integer k. Hence, if f belongs to the radical of I, then 1 belongs to J. Since G is a Gröbner basis of J, it must contain an element c whose leading coefficient divides 1. Hence c is an invertible element of R. Since G is semi-reduced and c divides any term, $G = \{c\}$. Now suppose that $G = \{c\}$ for an invertible element c of R. Then 1 belongs to J, and so

$$1 = a_0 + a_1\, y + \ldots + a_m\, y^m + (1 - y\, f)\left(b_0 + b_1\, y + \ldots + b_{m-1}\, y^{m-1}\right),$$

where each a_i belongs to I, and each b_i belongs to $R[x_1, \ldots, x_n]$. Hence comparing coefficients at powers of y leads to the following equations modulo I: $b_0 \equiv 1$, $b_i \equiv b_{i-1}\, f$, for $1 \le i \le m - 1$, and $b_{m-1}\, f \equiv 0$. Then, $b_i \equiv f^i$, for $0 \le i \le m - 1$, and $f^m \equiv 0$ modulo I. Therefore, f belongs to the radical of I, which completes the proof of Property 3.

The following more technical property is important for solving complex polynomial systems.

Property 4: Let G be a Gröbner basis of an ideal I in $\mathbb{C}[x_1, \ldots, x_n, y]$ with a monomial order that makes monomials containing y greater than monomials not containing y, let h be the element of G with the lowest positive degree d in y, let $c(x_1, \ldots, x_n)$ be the leading coefficient of h in y, and let

$\{h_1, \ldots, h_s\}$ be all elements of G that do not depend on y. Then for any polynomial $p \in I$ and any point $(a_1, \ldots, a_n, b)$ if $c(a_1, \ldots, a_n) \neq 0$, $h_i(a_1, \ldots, a_n) = 0$, for $1 \leq i \leq s$, and $h(a_1, \ldots, a_n, b) = 0$, then $p(a_1, \ldots, a_n, b) = 0$.

Proof: Consider the pseudoremainder r of the division of p by h as polynomials in y.

$$c(x_1, \ldots, x_n)^e\, p(x_1, \ldots, x_n, y) = q(x_1, \ldots, x_n, y)\, h(x_1, \ldots, x_n, y) + r(x_1, \ldots, x_n, y) \tag{1}$$

Since p and h belong to I, so does r. By Property 1, reduction of r by G must yield zero. Since the degree of r in y is less than d, r cannot be reduced by any of the elements of G that depend on y. Hence

$$r(x_1, \ldots, x_n, y) = p_1(x_1, \ldots, x_n, y)\, h_1(x_1, \ldots, x_n) + \ldots + p_s(x_1, \ldots, x_n, y)\, h_s(x_1, \ldots, x_n)$$

and so $r(a_1, \ldots, a_n, b) = 0$. Since $c(a_1, \ldots, a_n) \neq 0$, (1) implies that $p(a_1, \ldots, a_n, b) = 0$, which completes the proof of Property 4.

Mathematica Function GroebnerBasis

The *Mathematica* function `GroebnerBasis` finds semi-reduced Gröbner bases. This section describes `GroebnerBasis` options used in the solving of complex polynomial systems.

option name	*default value*	
`CoefficientDomain`	`Automatic`	the type of objects assumed to be coefficients
`Method`	`Automatic`	the method used to compute the basis
`MonomialOrder`	`Lexicographic`	the criterion used for ordering monomials

`GroebnerBasis` options used in the solving of complex polynomial systems.

CoefficientDomain

This option specifies the domain R of coefficients. With the default `Automatic` setting, the coefficient domain is the field generated by numeric coefficients present in the input.

`Integers`	the domain of integers
`InexactNumbers[`*prec*`]`	inexact numbers with precision *prec*
`Polynomials[`x`]`	the domain of polynomials in x
`RationalFunctions`	the field of rational functions in variables not on the variable list given to `GroebnerBasis`
`Rationals`	the field of rational numbers

Available settings for `CoefficientDomain`.

Note that the coefficient domain R also depends on the setting of the `Modulus` option of `GroebnerBasis`. With `Modulus -> `p, for a prime number p, the coefficient domain is the field $\mathbb{Z}_p$, or the field of rational functions over $\mathbb{Z}_p$ if `CoefficientDomain -> RationalFunctions`.

Method

With the default setting `Method -> Automatic`, `GroebnerBasis` normally uses a variant of the Buchberger algorithm. Another algorithm available is the Gröbner walk, which computes a Gröbner basis in an easier monomial order and then transforms it to the required harder monomial order. This is often faster than directly computing a Gröbner basis in the required order, especially if the input polynomials are known to be a Gröbner basis for the easier order. With the `Method -> Automatic` setting, `GroebnerBasis` uses the Gröbner walk for the default `CoefficientDomain -> Rationals` and `MonomialOrder -> Lexicographic`.

`GroebnerBasis[`*polys*`,`*vars*`, Method->{"GroebnerWalk","InitialMonomialOrder"->`$order_1$`},MonomialOrder->`$order_2$`]`	find a Gröbner basis in $order_1$ and use the Gröbner walk algorithm to transform it to a Gröbner basis in $order_2$

Transforming Gröbner bases using the Gröbner walk algorithm.

MonomialOrder

This option specifies the monomial order. The value can be either one of the named monomial orders or a weight matrix. The following table gives conditions for $x_1^{d_1} \ldots x_n^{d_n} \preccurlyeq x_1^{e_1} \ldots x_n^{e_n}$.

`Lexicographic`	$d_1 == e_1 \bigwedge \ldots \bigwedge d_{i-1} == e_{i-1} \bigwedge d_i < e_i$
`DegreeLexicographic`	$d_1 + \ldots + d_n < e_1 + \ldots + e_n \bigvee (d_1 + \ldots + d_n == e_1 + \ldots + e_n \bigwedge d_1 == e_1 \bigwedge \ldots \bigwedge d_{i-1} == e_{i-1} \bigwedge d_i < e_i)$
`DegreeReverseLexicographic`	$d_1 + \ldots + d_n < e_1 + \ldots + e_n \bigvee (d_1 + \ldots + d_n == e_1 + \ldots + e_n \bigwedge d_n == e_n \bigwedge \ldots \bigwedge d_{i+1} == e_{i+1} \bigwedge d_i > e_i)$

Monomial orders.

Quantifier elimination needs an order in which monomials containing quantifier variables are greater than monomials not containing quantifier variables. The `Lexicographic` order satisfies this condition, but the following `EliminationOrder` usually leads to faster computations.

$$m_1(X)\, n_1(Y) \preccurlyeq m_2(X)\, n_2(Y) \Longleftrightarrow d(n_1(Y)) < d(n_2(Y)) \bigvee (d(n_1(Y)) == d(n_2(Y)) \bigwedge m_1(X)\, n_1(Y) \preccurlyeq_{\mathrm{DRL}} m_2(X)\, n_2(Y),$$

where d denotes total degree, X denotes free variables, Y denotes quantifier variables, m_i and n_i are monomials, and $\preccurlyeq_{\mathrm{DRL}}$ denotes the `DegreeReverseLexicographic` order.

Using `EliminationOrder` requires the `GroebnerBasis` syntax with elimination variables specified.

`GroebnerBasis[`*polys*`,`*xvars*`,`*yvars*`,MonomialOrder->EliminationOrder]`	find a Gröbner basis in `EliminationOrder`

Gröbner basis in elimination order.

By default, `GroebnerBasis` with `MonomialOrder -> EliminationOrder` drops the polynomials that contain *yvars* from the result, returning only basis polynomials in *xvars*. To get all basis polynomials, the value of the system option `EliminateFromGroebnerBasis` from the `GroebnerBasisOptions` group must be changed. (*Mathematica* changes the option locally in the quantifier elimination algorithm.) The option value can be changed with

```
SetSystemOptions[
 "GroebnerBasisOptions" -> {"EliminateFromGroebnerBasis" -> False}]
```
.

option name	default value	
"EliminateFromGroebnerBasis"	True	whether GroebnerBasis with MonomialOrder -> EliminationOrder should remove polynomials containing elimination variables

System option EliminateFromGroebnerBasis.

This eliminates y from $\exists_y \left(x_1^2 + x_2^2 - x_1\, x_2\, y == 1 \bigwedge x_1^2 + x_2^2 + x_1\, x_2\, y + 1 == 0\right)$. The answer is a polynomial whose zeros are the Zariski closure of the projection of the solution set of the two original equations on the (x_1, x_2) plane.

In[4]:=
```
GroebnerBasis[{x1^2 + x2^2 - x1 x2 y - 1, x1^2 + x2^2 + x1 x2 y + 1},
  {x1, x2}, {y}, MonomialOrder → EliminationOrder]
```

Out[4]= $\{x_1^2 + x_2^2\}$

The exact description of the projection of the solution set on the (x_1, x_2) plane depends on all basis polynomials. Note that the second basis polynomial cannot be zero if x_1 or x_2 is zero.

In[5]:=
```
SetSystemOptions[
   "GroebnerBasisOptions" → {"EliminateFromGroebnerBasis" → False}];
GroebnerBasis[{x1^2 + x2^2 - x1 x2 y - 1, x1^2 + x2^2 + x1 x2 y + 1},
  {x1, x2}, {y}, MonomialOrder → EliminationOrder]
```

Out[6]= $\{x_1^2 + x_2^2,\ 1 + y\, x_1\, x_2,\ -x_1 + y\, x_2^3\}$

This resets the system option to its default value.

In[7]:=
```
SetSystemOptions["GroebnerBasisOptions" → {"EliminateFromGroebnerBasis" → True}];
```

Resolve gives the exact description of the projection of the solution set on the (x_1, x_2) plane.

In[8]:=
```
Resolve[∃y (x1^2 + x2^2 - x1 x2 y == 1 ⋀ x1^2 + x2^2 + x1 x2 y + 1 == 0)]
```

Out[8]= $x_1^2 + x_2^2 == 0\ \&\&\ x_2 \neq 0$

Decision Problems

A decision problem is a system with all variables existentially quantified, that is, a system of the form

$$\exists_{x_1} \exists_{x_2} \ldots \exists_{x_n} \Phi(x_1, \ldots, x_n),$$

where $x_1, \ldots, x_n$ are all variables in Φ. Solving a decision problem means deciding whether it is equivalent to `True` or to `False`, that is, deciding whether the quantifier-free system of polynomial equations and inequations $\Phi(x_1, \ldots, x_n)$ has solutions.

Solving this decision problem proves that a quadratic equation with a zero determinant cannot have two different roots.

```
In[9]:= Reduce[∃{a,b,c,x,y} (a x^2 + b x + c == 0 && a y^2 + b y + c == 0 && x ≠ y && b^2 - 4 a c == 0 && a ≠ 0)]
Out[9]= False
```

Given the identities

$$\exists_x (\Phi_1 \vee \ldots \vee \Phi_n) \Longleftrightarrow \exists_x \Phi_1 \vee \ldots \vee \exists_x \Phi_n$$
$$g_1 \neq 0 \wedge \ldots \wedge g_k \neq 0 \Longleftrightarrow g_1 \cdot \ldots \cdot g_k \neq 0$$

solving any decision problem can be reduced to solving a finite number of decision problems of the form

$$\exists_{x_1} \exists_{x_2} \ldots \exists_{x_n} f_1(x_1, \ldots, x_n) == 0 \wedge \ldots \wedge f_k(x_1, \ldots, x_n) == 0 \wedge g(x_1, \ldots, x_n) \neq 0.$$

By Hilbert's Nullstellensatz and Property 3 of Gröbner bases

$$f_1(x_1, \ldots, x_n) == 0 \wedge \ldots \wedge f_k(x_1, \ldots, x_n) == 0 \wedge g(x_1, \ldots, x_n) \neq 0$$

has complex solutions iff

$$\texttt{GroebnerBasis}[\{f_1, \ldots, f_k, 1 - g\, z\}, \{x_1, \ldots, x_n, z\}]$$

with an arbitrary monomial order, is different than $\{1\}$.

This shows that $x^2 + y^2 == 2 \wedge x == y \wedge x \neq -1$ has complex solutions.

```
In[10]:= GroebnerBasis[{x^2 + y^2 - 2, x - y, 1 - (x + 1) z}, {x, y, z}]
Out[10]= {-1 + 2 z, -1 + y, -1 + x}
```

This shows that $x^2 + y^2 == 2 \wedge x == y \wedge x^2 \neq 1$ has no complex solutions.

```
In[11]:= GroebnerBasis[{x^2 + y^2 - 2, x - y, 1 - (x^2 - 1) z}, {x, y, z}]
Out[11]= {1}
```

When *Mathematica* solves a decision problem, the monomial order used by the `GroebnerBasis` computation is `MonomialOrder -> EliminationOrder`, with $\{z\}$ specified as the elimination variable list. This setting corresponds to the monomial ordering in which monomials containing z are greater than those that do not contain z, and the ordering of monomials not containing z is degree reverse lexicographic. If there is no inequation condition, there is no need to introduce z, and *Mathematica* uses `MonomialOrder -> DegreeReverseLexicographic`.

Quantifier Elimination

For any complex polynomial system there exists an equivalent quantifier-free complex polynomial system. This follows from Chevalley's theorem, which states that a projection of a quasi-algebraically constructible set (a solution set of a quantifier-free system of polynomial equations and inequations) is a quasi-algebraically constructible set [3]. Quantifier elimination is the procedure of finding a quantifier-free complex polynomial system equivalent to a given complex polynomial system. In *Mathematica*, quantifier elimination for complex polynomial systems is done by `Resolve`. It is also used by `Reduce` and `FindInstance` as the first step in solving or finding instances of solutions of complex polynomial systems.

Eliminating quantifiers from this system gives a condition for quadratic equations to have at least two different zeros.

```
In[12]:= Resolve[∃{x,y} (a x^2 + b x + c == 0 && a y^2 + b y + c == 0 && x ≠ y)]
Out[12]= (a ≠ 0 && -b^2 + 4 a c ≠ 0) || (a == 0 && b == 0 && c == 0)
```

For complex polynomial systems *Mathematica* uses the following quantifier elimination method. Given the identities

$$\forall_x \Phi \Longleftrightarrow \neg(\exists_x \neg \Phi)$$
$$\exists_x (\Phi_1 \lor \ldots \lor \Phi_n) \Longleftrightarrow \exists_x \Phi_1 \lor \ldots \lor \exists_x \Phi_n$$
$$g_1 \neq 0 \land \ldots \land g_k \neq 0 \Longleftrightarrow g_1 \cdot \ldots \cdot g_k \neq 0,$$

eliminating quantifiers from any complex polynomial system can be reduced to a finite number of single existential quantifier eliminations from systems of the form

$$\exists_y f_1(x_1, \ldots, x_n, y) == 0 \land \ldots \land f_k(x_1, \ldots, x_n, y) == 0 \land g(x_1, \ldots, x_n, y) \neq 0. \quad (1)$$

To eliminate the quantifier from (1), *Mathematica* first computes the Gröbner basis of equations

$$G = \text{GroebnerBasis}[\{f_1, \ldots, f_k\}, \{x_1, \ldots, x_n, y\}]$$

with a monomial order that makes monomials containing y greater than monomials not containing y.

The monomial order used is `EliminationOrder`, with $\{y\}$ specified as the elimination variable list and all basis polynomials kept.

If G contains no polynomials that depend on y, then a quantifier-free system equivalent to (1) can be obtained by equating all elements of G to zero, and asserting that at least one coefficient of g as a polynomial in y is not equal to zero. Otherwise let h be the element of G with the lowest positive degree d in y, let $c(x_1, \ldots, x_n)$ be the leading coefficient of h in y, and let $\{h_1, \ldots, h_s\}$ be all elements of G that do not depend on y. Now (1) can be split into a disjunction of two systems

$$\begin{aligned}\exists_y\, & c(x_1, \ldots, x_n) == 0 \wedge f_1(x_1, \ldots, x_n, y) == 0 \wedge \\ & \ldots \wedge f_k(x_1, \ldots, x_n, y) == 0 \wedge g(x_1, \ldots, x_n, y) \neq 0\end{aligned} \tag{2}$$

and

$$\begin{aligned}\exists_y\, & c(x_1, \ldots, x_n) \neq 0 \wedge f_1(x_1, \ldots, x_n, y) == 0 \wedge \\ & \ldots \wedge f_k(x_1, \ldots, x_n, y) == 0 \wedge g(x_1, \ldots, x_n, y) \neq 0.\end{aligned} \tag{3}$$

To eliminate the quantifier from (2), the quantifier elimination procedure is called recursively. Since the ideal generated by $\{c, f_1, \ldots, f_k\}$ strictly contains the ideal generated by $\{f_1, \ldots, f_k\}$, the Noetherian property of polynomial rings guarantees finiteness of the recursion.

If c belongs to the radical of the ideal generated by $\{f_1, \ldots, f_k\}$, which is exactly when 1 belongs to

```
GroebnerBasis[{h1, ..., hs, 1 - c z}, {x1, ..., xn, z}],
```

(3) is equivalent to `False`. Otherwise let

$$r = r_{d-1}(x_1, \ldots, x_n)\, y^{d-1} + \ldots + r_0(x_1, \ldots, x_n) == c(x_1, \ldots, x_n)^e\, g(x_1, \ldots, x_n, y)^d - q(x_1, \ldots, x_n, y)\, h(x_1, \ldots, x_n, y)$$

be the pseudoremainder of the division of g^d by h as polynomials in y. Then (3) is equivalent to the quantifier-free system

$$\begin{aligned}& c(x_1, \ldots, x_n) \neq 0 \wedge h_1(x_1, \ldots, x_n) == 0 \wedge \ldots \wedge \\ & h_s(x_1, \ldots, x_n) == 0 \wedge (r_{d-1}(x_1, \ldots, x_n) \neq 0 \vee \ldots \vee r_0(x_1, \ldots, x_n) \neq 0).\end{aligned} \tag{4}$$

To show that (3) implies (4), suppose that $(a_1, \ldots, a_n)$ satisfies (3). Then $c(a_1, \ldots, a_n) \neq 0$ and there exists b, such that

$$f_1(a_1, \ldots, a_n, b) == 0 \bigwedge \ldots \bigwedge f_k(a_1, \ldots, a_n, b) == 0 \bigwedge g(a_1, \ldots, a_n, b) \neq 0.$$

Since $\{h_1, \ldots, h_s\}$ and h belong to the ideal generated by $\{f_1, \ldots, f_k\}$,

$$h_1(a_1, \ldots, a_n) == 0 \bigwedge \ldots \bigwedge h_s(a_1, \ldots, a_n) == 0$$

and $h(a_1, \ldots, a_n, b) == 0$. Hence

$$r(a_1, \ldots, a_n, b) == r_{d-1}(a_1, \ldots, a_n)\, b^{d-1} + \ldots + r_0(a_1, \ldots, a_n) == c(a_1, \ldots, a_n)^e\, g(a_1, \ldots, a_n, b)^d \neq 0,$$

which implies that

$$r_{d-1}(a_1, \ldots, a_n) \neq 0 \bigvee \ldots \bigvee r_0(a_1, \ldots, a_n) \neq 0.$$

To show that (4) implies (3), suppose that $(a_1, \ldots, a_n)$ satisfies (4). Then

$$r(a_1, \ldots, a_n, y) == r_{d-1}(a_1, \ldots, a_n)\, y^{d-1} + \ldots + r_0(a_1, \ldots, a_n) ==$$
$$c(a_1, \ldots, a_n)^e\, g(a_1, \ldots, a_n, y)^d - q(a_1, \ldots, a_n, y)\, h(a_1, \ldots, a_n, y).$$

Since $h(a_1, \ldots, a_n, y)$ is a polynomial of degree d, and $r(a_1, \ldots, a_n, y)$ is a nonzero polynomial of degree less than d, there is a root b of $h(a_1, \ldots, a_n, y)$ such that $(y-b)^m$ divides $h(a_1, \ldots, a_n, y)$ but not $r(a_1, \ldots, a_n, y)$ for some $1 \leq m \leq d$. If $g(a_1, \ldots, a_n, b)$ were zero, then $(y-b)^m$ would divide $g(a_1, \ldots, a_n, y)^d$, which is impossible because it would imply that $(y-b)^m$ divides $r(a_1, \ldots, a_n, y)$. Therefore $g(a_1, \ldots, a_n, b) \neq 0$. Property 4 shows that $p(a_1, \ldots, a_n, b) == 0$ for any polynomial $p \in G$. Since G is a Gröbner basis of the ideal generated by $\{f_1, \ldots, f_k\}$,

$$f_1(a_1, \ldots, a_n, b) == 0 \bigwedge \ldots \bigwedge f_k(a_1, \ldots, a_n, b) == 0,$$

which completes the proof of correctness of the quantifier elimination algorithm.

This eliminates the quantifier from $\exists_y\, x_1^2 + x_2^2 + y^2 == 1 \bigwedge x_1 + x_2 == y$. Here $g = 1$, $h = -y + x_1 + x_2$, and $c = -1$. Since c is a nonzero constant, (2) is `False` and the equivalent quantifier-free system is given by (4). Since g is a nonzero constant, (4) becomes $1 - 2x_1^2 - 2x_1 x_2 - 2x_2^2 == 0$.

```
In[13]:= SetSystemOptions[
           "GroebnerBasisOptions" → {"EliminateFromGroebnerBasis" → False}];
         GroebnerBasis[{x1^2 + x2^2 + y^2 - 1, x1 + x2 - y}, {x1, x2}, {y},
          MonomialOrder → EliminationOrder]

Out[14]= {-1 + 2 x1^2 + 2 x1 x2 + 2 x2^2, y - x1 - x2}
```

This resets the system option to its default value.

```
In[15]:= SetSystemOptions["GroebnerBasisOptions" → {"EliminateFromGroebnerBasis" → True}];
```

Arbitrary Complex Polynomial Systems

FindInstance

`FindInstance` can handle arbitrary complex polynomial systems giving instances of complex solutions, or an empty list for systems that have no solutions. If the number of instances requested is more than one, the instances are randomly generated from the full solution of the system given by `Reduce`. If one instance is requested, a faster algorithm that produces one instance is used. Here is a description of the algorithm used to find a single instance, or prove that a system has no solutions.

If the system contains general quantifiers (∀), the quantifier elimination algorithm is used to eliminate the innermost quantifiers until the system contains only existential quantifiers (∃) or is quantifier-free. Note that

$$\exists_{x_1} \exists_{x_2} \ldots \exists_{x_n} \Phi(x_1, \ldots, x_n, y_1, \ldots, y_m) \tag{1}$$

has solutions if and only if $\Phi(x_1, \ldots, x_n, y_1, \ldots, y_m)$ has solutions, and if $(a_1, \ldots, a_n, b_1, \ldots, b_m)$ is a solution of $\Phi(x_1, \ldots, x_n, y_1, \ldots, y_m)$, then $(b_1, \ldots, b_m)$ is a solution of (1). Hence to find instances of solutions of systems containing only existential quantifiers it is enough to be able to find instances of quantifier-free systems. Moreover, $(a_1, \ldots, a_n)$ is a solution of

$$\Phi_1(x_1, \ldots, x_n) \bigvee \ldots \bigvee \Phi_m(x_1, \ldots, x_n)$$

if and only if it is a solution of one of the $\Phi_i(x_1, \ldots, x_n)$, with $1 \le i \le m$, so it is enough to show how to find instances of solutions of

$$f_1(x_1, \ldots, x_n) == 0 \bigwedge \ldots \bigwedge f_k(x_1, \ldots, x_n) == 0 \bigwedge g(x_1, \ldots, x_n) \neq 0. \tag{2}$$

First compute the `GroebnerBasis` G of $\{f_1, \ldots, f_k, 1 - g\, z\}$ with `MonomialOrder -> EliminationOrder`, eliminating the polynomials that depend on z (if there is no inequation condition, G is the `GroebnerBasis` of $\{f_1, \ldots, f_k\}$ with `MonomialOrder -> DegreeReverseLexicographic`). If G contains 1, there are no solutions. Otherwise, compute a subset S of $\{x_1, \ldots, x_n\}$ of the highest cardinality among subsets strongly independent modulo the ideal generated by G with respect to the degree reverse lexicographic order ([1], Section 9.3). Reorder $\{x_1, \ldots, x_n\}$ so that $S == \{x_{n-d+1}, \ldots, x_n\}$, and compute the lexicographic order `GroebnerBasis` H of the ideal generated by G. To compute H, *Mathematica* uses the Gröbner walk algorithm.

For each of the variables x_i, $1 \le i \le n-d$, select the polynomial $h_i \in H$ with the smallest leading monomial among elements of H that depend on x_i and not on $\{x_1, \ldots, x_{i-1}\}$. Let c_i be the leading coefficient of h_i as a polynomial in x_i. If c_i depends on a variable that is not in S, replace H with the lexicographic order Gröbner basis of the ideal generated by H and c_i. The following shows that this operation keeps S strongly independent modulo the ideal generated by H. Hence, possibly after a finite (by the Noetherian property of polynomial rings) number of extensions of H, the leading coefficient c_i of h_i depends only on $\{x_{n-d+1}, \ldots, x_n\}$, for all $1 \le i \le n-d$. For the set of polynomials P, let $Z(P)$ be the set of common zeros of elements of P. Both $Z(G)$ and $Z(H)$ have dimension d, and $Z(H) \subset Z(G)$, hence any d-dimensional irreducible component of $Z(H)$ is also a component of $Z(G)$. Since g does not vanish on any irreducible component of $Z(G)$, it does not vanish on any d-dimensional irreducible component of $Z(H)$. Therefore, the Gröbner basis of H and g contains a polynomial t depending only on $\{x_{n-d+1}, \ldots, x_n\}$. Let $p = t\, c_1 \ldots c_{n-d}$. To find a solution of (2), pick its last d coordinates $\{a_{n-d+1}, \ldots, a_n\}$ so that $p(a_{n-d+1}, \ldots, a_n) \ne 0$. For all $1 \le i \le n-d$, $c_i(a_{n-d+1}, \ldots, a_n) \ne 0$, and so by Property 4 if a_i, for $i = n-d, \ldots, 1$, is chosen to be the first root of $h_i(x_i, a_{i+1}, \ldots, a_n)$, then $(a_1, \ldots, a_n) \in Z(H) \subset Z(G)$. Moreover, $g(a_1, \ldots, a_n) \ne 0$, because otherwise $(a_1, \ldots, a_n)$ would belong to $Z(H \bigcup \{g\})$, which would imply that $t(a_{n-d+1}, \ldots, a_n) == 0$, which is impossible since t divides p.

To prove the correctness of the aforementioned algorithm, it must be shown that extending H by c_i that depend on a variable not in S preserves strong independence of S modulo the ideal generated by H. Suppose for some $1 \le i \le n-d$, c_i depends on a variable, which is not in S. Let $I_{i+1} \subset \mathbb{C}[x_{i+1}, \ldots, x_n]$ denote the ideal generated by $H \bigcap \mathbb{C}[x_{i+1}, \ldots, x_n]$, and let $J_{i+1} \subset \mathbb{C}[x_{i+1}, \ldots, x_n]$ denote the ideal generated by I_{i+1} and c_i. Then J_{i+1} does not contain nonzero elements of $\mathbb{C}[x_{n-d+1}, \ldots, x_n]$. To prove this, suppose that $r == p\, c_i + q \in J_{i+1} \bigcap \mathbb{C}[x_{n-d+1}, \ldots, x_n] \backslash \{0\}$ where $p \in \mathbb{C}[x_{i+1}, \ldots, x_n]$ and $q \in I_{i+1}$. Then $h_i == c_i\, x_i^k + t$, with $\deg_{x_i}(t) < k$, and

$$p\, h_i == p\, c_i\, x_i^k + p\, t == r\, x_i^k - q\, x_i^k + p\, t$$

belongs to the ideal generated by H, and so does $g_i = r\,x_i^k + p\,t$. This contradicts the choice of h_i since the leading monomial of g_i depends on x_i and is strictly smaller than the leading monomial of h_i. Therefore, the projection of $Z(J_{i+1})$ on $A_d = \left(\mathbb{C}^d\right)_{\{x_{n-d+1},\ldots,x_n\}}$ is dense in A_d, and so, since $Z(I_{i+1})$ has dimension d, c_i must be zero on some irreducible component C_{i+1} of $Z(I_{i+1})$ whose projection on A_d is dense in A_d. Since $Z(I_{i+1})$ is the Zariski closure of the projection of the d-dimensional set $Z(H)$, C_{i+1} is contained in the Zariski closure of the projection of an irreducible component C of $Z(H)$. $Z(c_i) \bigcap C$ has dimension d, hence c_i is zero on C, and the projection of C on A_d is dense in A_d, which proves that S is strongly independent modulo the ideal generated by H and c_i.

Here is an example in which H needs to be extended. Here $S == \{x_3\}$, $h_1 == (x_2 - x_3)\,x_1$, $c_1 == x_2 - x_3$, and $I_2 == \ <(x_2 - x_3)^2\,(x_2 - 2\,x_3)>$. c_1 is zero on one of the two one-dimensional components of I_2.

```
In[16]:= GroebnerBasis[{(x2 - x3)^2 (x2 - 2 x3), (x2 - x3) x1, x1^2 - x1}, {x1, x2, x3}]
Out[16]= {x2^3 - 4 x2^2 x3 + 5 x2 x3^2 - 2 x3^3, x1 x2 - x1 x3, -x1 + x1^2}
```

Extending H by c_1 results in all c_i depending on x_3 only (in fact even constant) while preserving the strong independence of $\{x_3\}$.

```
In[17]:= GroebnerBasis[{x2^3 - 4 x2^2 x3 + 5 x2 x3^2 - 2 x3^3, x1 x2 - x1 x3, -x1 + x1^2, x2 - x3 }, {x1, x2, x3}]
Out[17]= {x2 - x3, -x1 + x1^2}
```

Reduce

`Reduce` can solve arbitrary complex polynomial systems. As the first step, `Reduce` uses the quantifier elimination algorithm to eliminate the quantifiers. If the obtained quantifier-free system is a disjunction, each term of the disjunction is solved separately, and the solution is given as a disjunction of the solutions of the terms. Thus, the problem is reduced to solving quantifier-free systems of the form

$$f_1(x_1,\,\ldots,\,x_n) == 0 \bigwedge \ldots \bigwedge f_k(x_1,\,\ldots,\,x_n) == 0 \bigwedge\ g(x_1,\,\ldots,\,x_n) \neq 0. \tag{3}$$

First compute the `GroebnerBasis` G of $\{f_1, \ldots, f_k, 1 - g\,z\}$ with variable order $\{z,\ x_n,\ \ldots,\ x_1\}$ and `MonomialOrder` $->$ `Lexicographic`, and select the polynomials that do not depend on z. Then the solution set of $G == 0 \bigwedge\ g(x_1,\,\ldots,\,x_n) \neq 0$ is equal to the solution set of (3) and g does not vanish on any component of the zero set $Z(G)$ of G. If G contains 1, (3) has no solutions. Otherwise for each $1 \le i \le n$, such that the set G_i of elements of G depending on x_i and not on any x_j with $j > i$ is not empty, select an element h_i of G_i with the lowest positive degree in x_i. If one of the leading coefficients c_i of h_i is zero on $Z(G)$, that is, it belongs to the radical of the ideal generated by G, replace G by the lexicographic Gröbner basis of the ideal generated by G and c_i. Now split the system into

$$\begin{gathered}(c_{i_1} == 0 \wedge G == 0 \wedge \ g \neq 0) \bigvee (c_{i_2} == 0 \wedge G == 0 \wedge \ c_{i_1}\, g \neq 0) \bigvee \\ \ldots \bigvee \left(c_{i_s} == 0 \bigwedge G == 0 \bigwedge c_{i_1} \ldots c_{i_{s-1}}\, g \neq 0\right) \bigvee \left(G == 0 \wedge c_{i_1} \ldots c_{i_s}\, g \neq 0\right).\end{gathered} \tag{4}$$

and call the solving procedure recursively on all but the last term of the disjunction (4). Note that the algebraic set $c_{i_j} == 0 \bigwedge G == 0$ is strictly contained in $G == 0$, so the recursion is finite. If the product of all the c_i and g belongs to the radical of the ideal generated by G, the last term has no solutions. Otherwise, by Property 4, the solution set of the last term is equal to

$$\begin{gathered}c_{i_1}\,(x_1,\,\ldots,\,x_{i_1-1}) \neq 0 \wedge \texttt{Roots}\,[h_{i_1} == 0,\, x_{i_1}] \wedge \ldots \wedge \\ c_{i_s}\left(x_1,\,\ldots,\,x_{i_s-1}\right) \neq 0 \wedge \texttt{Roots}\left[h_{i_s} == 0,\, x_{i_s}\right] \wedge g\,(x_1,\,\ldots,\,x_n) \neq 0.\end{gathered}$$

The conditions $c_{i_j} \neq 0$ guarantee that all the solutions (represented as radicals or `Root` objects) given by $\texttt{Roots}\left[h_{i_j} == 0, x_{i_j}\right]$ are well defined. `Reduce` performs several operations in order to simplify the inequation conditions returned, like removing multiple factors, removing factors common with earlier inequation conditions, reducing modulo the h_{i_j}, and removing factors that are nonzero on $Z(G)$.

Options

Options for Reduce, Resolve, and FindInstance

The *Mathematica* functions for solving complex polynomial systems have a number of options that control the way they operate. This section gives a summary of these options.

option name	*default value*	
`Backsubstitution`	`False`	whether the solutions given by `Reduce` and `Resolve` with specified variables should be unwound by backsubstitution
`Cubics`	`False`	whether the Cardano formulas should be used to express solutions of cubics
`Quartics`	`False`	whether the Cardano formulas should be used to express solutions of quartics

Options of `Reduce` and `Resolve` affecting the behavior of complex polynomial systems.

option name	*default value*	
`WorkingPrecision`	∞	the working precision to be used in computations, with the default settings of system options; the value of working precision affects only calls to `Roots`

Options of `Reduce`, `Resolve`, and `FindInstance` affecting the behavior of complex polynomial systems.

Backsubstitution

By default, `Reduce` may use variables appearing earlier in the variable list to express solutions for variables appearing later in the variable list.

In[18]:= $\textbf{Reduce}\left[x^2+y^2==1\;\&\&\;x^5-3x+7==0,\ \{x,y\}\right]$

Out[18]= $\left(x==\text{Root}\left[7-3\,\#1+\#1^5\,\&,\,1\right]\;||\;x==\text{Root}\left[7-3\,\#1+\#1^5\,\&,\,2\right]\;||\;x==\text{Root}\left[7-3\,\#1+\#1^5\,\&,\,3\right]\;||\;x==\text{Root}\left[7-3\,\#1+\#1^5\,\&,\,4\right]\;||\;x==\text{Root}\left[7-3\,\#1+\#1^5\,\&,\,5\right]\right)\;\&\&\;\left(y==-\sqrt{1-x^2}\;||\;y==\sqrt{1-x^2}\right)$

With `Backsubstitution -> True`, `Reduce` uses backsubstitution to eliminate variables from the right-hand sides of the equations.

In[19]:= `Reduce[x^2 + y^2 == 1 && x^5 - 3 x + 7 == 0, {x, y}, Backsubstitution → True]`

Out[19]=
$$\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 1\right] \&\& y = -\sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 1\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 1\right] \&\& y = \sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 1\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 2\right] \&\& y = -\sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 2\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 2\right] \&\& y = \sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 2\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 3\right] \&\& y = -\sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 3\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 3\right] \&\& y = \sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 3\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 4\right] \&\& y = -\sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 4\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 4\right] \&\& y = \sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 4\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 5\right] \&\& y = -\sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 5\right]^2}\right) \| \\
\left(x = \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 5\right] \&\& y = \sqrt{1 - \text{Root}\left[7 - 3 \#1 + \#1^5 \&, 5\right]^2}\right)$$

Cubics and Quartics

By default `Reduce` does not use the Cardano formulas for solving cubics or quartics.

In[20]:= `Reduce[x^3 - 3 x + 7 == 0, x]`

Out[20]=
$$x = \text{Root}\left[7 - 3 \#1 + \#1^3 \&, 1\right] \| x = \text{Root}\left[7 - 3 \#1 + \#1^3 \&, 2\right] \| x = \text{Root}\left[7 - 3 \#1 + \#1^3 \&, 3\right]$$

Setting the options `Cubics` and `Quartics` to `True` allows `Reduce` to use the Cardano formulas for solving cubics and quartics.

In[21]:= `Reduce[x^3 - 3 x + 7 == 0, x, Cubics → True]`

Out[21]=
$$x = -\left(\frac{2}{7 - 3\sqrt{5}}\right)^{1/3} - \left(\frac{1}{2}\left(7 - 3\sqrt{5}\right)\right)^{1/3} \| x = \frac{1}{2}\left(1 + i\sqrt{3}\right)\left(\frac{1}{2}\left(7 - 3\sqrt{5}\right)\right)^{1/3} + \frac{1 - i\sqrt{3}}{2^{2/3}\left(7 - 3\sqrt{5}\right)^{1/3}} \| \\
x = \frac{1}{2}\left(1 - i\sqrt{3}\right)\left(\frac{1}{2}\left(7 - 3\sqrt{5}\right)\right)^{1/3} + \frac{1 + i\sqrt{3}}{2^{2/3}\left(7 - 3\sqrt{5}\right)^{1/3}}$$

WorkingPrecision

With `WorkingPrecision` set to a finite number, `Reduce` uses numeric methods to find polynomial roots.

```
In[22]:= Reduce[x^3 - 3 x + 7 == 0, x, WorkingPrecision → 20]

Out[22]= x == -2.4259887573616221261 || x == 1.2129943786808110630 - 1.1891451081065508908 ⅈ ||
          x == 1.2129943786808110630 + 1.1891451081065508908 ⅈ
```

The ReduceOptions Group of System Options

Here are the system options from the `ReduceOptions` group that may affect the behavior of `Reduce`, `Resolve`, and `FindInstance` for complex polynomial systems. The options can be set with

`SetSystemOptions["ReduceOptions" -> {"`*option name*`" -> `*value*`}]`.

This sets the option `FinitePrecisionGB` to `True`.

```
In[23]:= SetSystemOptions["ReduceOptions" → {"FinitePrecisionGB" → True}];
```

This checks the value of `FinitePrecisionGB`.

```
In[24]:= "FinitePrecisionGB" /. ("ReduceOptions" /. SystemOptions[])

Out[24]= True
```

This sets the option `FinitePrecisionGB` back to the default value `False`.

```
In[25]:= SetSystemOptions["ReduceOptions" → {"FinitePrecisionGB" → False}];
```

option name	*default value*	
`"FinitePrecisionGB"`	`False`	whether finite values of working precision should be used in calls to `GroebnerBasis`
`"ReorderVariables"`	`False`	whether `Reduce` and `Resolve` are allowed to reorder the specified variables

`ReduceOptions` group options that affect the behavior of `Reduce`, `Resolve`, and `FindInstance` for complex polynomial systems.

FinitePrecisionGB

By default, `Reduce` uses `GroebnerBasis` with `CoefficientDomain -> Automatic`. This means that even with `WorkingPrecision` set to a finite number *prec*, if the input is exact `GroebnerBasis` uses exact computations.

```
In[26]:= SeedRandom[123];
f = Sum[RandomInteger[{-10^100, 10^100}] x^i y^j, {i, 0, 2}, {j, 0, 3}];
g = Sum[RandomInteger[{-10^100, 10^100}] x^i y^j, {i, 0, 3}, {j, 0, 2}];
Timing[a1 =
   Reduce[f == 0 && g == 0, {x, y}, WorkingPrecision → 100, Backsubstitution → True];]
Out[28]= {0.481, Null}
```

Setting the system option `"FinitePrecisionGB" -> True` makes `Reduce` use `GroebnerBasis` with `CoefficientDomain -> InexactNumbers[`*prec*`]`.

```
In[29]:= SetSystemOptions["ReduceOptions" → {"FinitePrecisionGB" → True}];
Timing[a2 =
   Reduce[f == 0 && g == 0, {x, y}, WorkingPrecision → 100, Backsubstitution → True];]
Out[30]= {0.25, Null}
```

Using finite precision may significantly improve the speed of `GroebnerBasis` computations. However, the numeric computations may fail due to loss of precision, or give incorrect answers. They usually give less precise results than exact `GroebnerBasis` computations followed by numeric root finding.

```
In[31]:= Precision /@ {a1, a2}
Out[31]= {90.7267, 48.2583}
```

This shows that the results are equal up to their precision.

```
In[32]:= Sort[{x, y} /. {ToRules[a1]}] - Sort[{x, y} /. {ToRules[a2]}]
```

Out[32]=
$$\{\{0.\times10^{-55}+0.\times10^{-55}\,i,\ 0.\times10^{-49}+0.\times10^{-49}\,i\},$$
$$\{0.\times10^{-55}+0.\times10^{-55}\,i,\ 0.\times10^{-49}+0.\times10^{-49}\,i\},\ \{0.\times10^{-54},\ 0.\times10^{-49}\},$$
$$\{0.\times10^{-57}+0.\times10^{-57}\,i,\ 0.\times10^{-55}+0.\times10^{-55}\,i\},\ \{0.\times10^{-57}+0.\times10^{-57}\,i,\ 0.\times10^{-55}+0.\times10^{-55}\,i\},$$
$$\{0.\times10^{-57}+0.\times10^{-57}\,i,\ 0.\times10^{-53}+0.\times10^{-53}\,i\},\ \{0.\times10^{-57}+0.\times10^{-57}\,i,\ 0.\times10^{-53}+0.\times10^{-53}\,i\},$$
$$\{0.\times10^{-57}+0.\times10^{-57}\,i,\ 0.\times10^{-53}+0.\times10^{-53}\,i\},\ \{0.\times10^{-57}+0.\times10^{-57}\,i,\ 0.\times10^{-53}+0.\times10^{-53}\,i\},$$
$$\{0.\times10^{-58}+0.\times10^{-58}\,i,\ 0.\times10^{-55}+0.\times10^{-55}\,i\},\ \{0.\times10^{-58}+0.\times10^{-58}\,i,\ 0.\times10^{-55}+0.\times10^{-55}\,i\},$$
$$\{0.\times10^{-57}+0.\times10^{-57}\,i,\ 0.\times10^{-52}+0.\times10^{-52}\,i\},\ \{0.\times10^{-57}+0.\times10^{-57}\,i,\ 0.\times10^{-52}+0.\times10^{-52}\,i\}\}$$

```
In[33]:= SetSystemOptions["ReduceOptions" → {"FinitePrecisionGB" → False}];
```

ReorderVariables

By default, Reduce is not allowed to reorder the specified variables. Variables appearing earlier in the variable list may be used to express solutions for variables appearing later in the variable list, but not vice versa.

In[34]:=
```
Reduce[z^3 + 3 z - 2 y + 1 == x && z^2 - 7 == y, {x, y, z}]
```

Out[34]=
$$\left(x == 21 \,\&\&\, y == -10 \,\&\&\, \left(z == -i\sqrt{3} \;||\; z == i\sqrt{3}\right)\right) \;||\; (x == 21 \,\&\&\, y == -3 \,\&\&\, z == 2) \;||$$
$$\left(\left(y == \text{Root}\left[699 + 2x - x^2 + (244 - 4x)\,\#1 + 23\,\#1^2 + \#1^3 \,\&,\, 1\right] \;||\right.\right.$$
$$y == \text{Root}\left[699 + 2x - x^2 + (244 - 4x)\,\#1 + 23\,\#1^2 + \#1^3 \,\&,\, 2\right] \;||$$
$$\left.\left. y == \text{Root}\left[699 + 2x - x^2 + (244 - 4x)\,\#1 + 23\,\#1^2 + \#1^3 \,\&,\, 3\right]\right) \,\&\&\, -21 + x \neq 0 \,\&\&\, z == \frac{72 - 2x + 13y + y^2}{-21 + x}\right)$$

Setting the system option "ReorderVariables" -> True allows Reduce to pick a variable order that makes the equations easier to solve.

In[35]:=
```
SetSystemOptions["ReduceOptions" → {"ReorderVariables" → True}];
Reduce[z^3 + 3 z - 2 y + 1 == x && z^2 - 7 == y, {x, y, z}]
```

Out[35]= $y == -7 + z^2 \,\&\&\, x == 15 + 3z - 2z^2 + z^3$

In[36]:=
```
SetSystemOptions["ReduceOptions" → {"ReorderVariables" → False}];
```

References

[1] Becker, T. and V. Weispfenning. *Gröbner Bases*. Springer-Verlag, 1993.

[2] Cox, D., J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms.* (2nd ed.) Springer-Verlag, 1997

[3] Łojasiewicz, S. *Introduction to Complex Analytic Geometry*. Birkhaüser, 1991.

Real Polynomial Systems

Introduction

A *real polynomial system* is an expression constructed with polynomial equations and inequalities

$$f(x_1, \ldots, x_n) == g(x_1, \ldots, x_n),\ f(x_1, \ldots, x_n) \neq g(x_1, \ldots, x_n),$$
$$f(x_1, \ldots, x_n) \geq g(x_1, \ldots, x_n),\ f(x_1, \ldots, x_n) > g(x_1, \ldots, x_n),$$
$$f(x_1, \ldots, x_n) \leq g(x_1, \ldots, x_n),\ f(x_1, \ldots, x_n) < g(x_1, \ldots, x_n)$$

combined using logical connectives and quantifiers

$$\Phi_1 \bigwedge \Phi_2,\ \Phi_1 \bigvee \Phi_2,\ \Phi_1 \Rightarrow \Phi_2,\ \neg\,\Phi,\ \forall_x \Phi,\ \text{ and }\ \exists_x \Phi.$$

An occurrence of a variable x inside $\forall_x \Phi$ or $\exists_x \Phi$ is called a bound occurrence; any other occurrence of x is called a free occurrence. A variable x is called a free variable of a real polynomial system if the system contains a free occurrence of x. A real polynomial system is quantifier free if it contains no quantifiers.

An example of a real polynomial system with free variables x, y, and z is the following

$$x^2 + y^2 \leq z^2 \bigwedge \exists_t \left(\forall_u\, t\, x > u\, y\, z + 7 \bigvee x^2\, t == 2\, z + 1\right). \tag{1}$$

Any real polynomial system can be transformed to the *prenex normal form*

$$Q_{1\, y_1}\, Q_{2\, y_2} \ldots \mathrm{Q}_{m\, y_m}\, \Phi(x_1, \ldots, x_n; y_1, \ldots, y_m), \tag{2}$$

where each Q_i is $\forall$ or $\exists$, and $\Phi(x_1, \ldots, x_n; y_1, \ldots, y_m)$ is a quantifier-free formula called the quantifier-free part of the system.

Any quantifier-free real polynomial system can be transformed to the disjunctive normal form

$$\left(\varphi_{1,1} \wedge \ldots \wedge \varphi_{1,n_1}\right) \vee \ldots \vee \left(\varphi_{m,1} \wedge \ldots \wedge \varphi_{m,n_m}\right), \tag{3}$$

where each $\varphi_{i,j}$ is a polynomial equation or inequality.

`Reduce`, `Resolve`, and `FindInstance` always put real polynomial systems in the prenex normal form, with quantifier-free parts in the disjunctive normal form, and subtract sides of equations and inequalities to put them in the form

$$f(x_1, \ldots, x_n) == (\text{or } \neq, \geq, >, \leq, <)\, 0.$$

In all of the real polynomial system solving tutorials, we will always assume the system has been transformed to this form.

`Reduce` can solve arbitrary real polynomial systems. For a system with free variables $x_1, \ldots, x_n$, the solution (possibly after expanding $\wedge$ with respect to $\vee$) is a disjunction of terms of the form

$$B(x_1;) \bigwedge B(x_2; x_1) \bigwedge B(x_3; x_1, x_2) \bigwedge \ldots \bigwedge B(x_n; x_1, \ldots, x_{n-1}), \tag{4}$$

where $B(x_k; x_1, \ldots, x_{k-1})$ is one of

$$\begin{array}{l} x_k == r_1(x_1, \ldots, x_{k-1}) \\ < (\text{or } \leq)\, x_k < (\text{or } \leq)\, r_2(x_1, \ldots, x_{k-1}) \\ < (\text{or } \leq)\, r_2(x_1, \ldots, x_{k-1}) \\ > (\text{or } \geq)\, r_1(x_1, \ldots, x_{k-1}) \\ \texttt{True} \end{array} \tag{5}$$

and r_1 and r_2 are algebraic functions (expressed using `Root` objects or radicals) such that for all $x_1, \ldots, x_{k-1}$ satisfying $B(x_1;) \bigwedge B(x_2; x_1) \bigwedge \ldots \bigwedge B(x_{k-1}; x_1, \ldots, x_{k-2})$, $r_1$and r_2 are well defined (that is, denominators and leading terms of `Root` objects are nonzero), real valued, continuous, and satisfy inequality $r_1 < r_2$.

The subset of $\mathbb{R}^n$ described by formula (4) is called a *cell*. The cells described by different terms of solution of a real polynomial system are disjoint.

This solves the system (1). The cells are represented in a nested form.

In[1]:= $$\textbf{sol} = \textbf{Reduce}\left[x^2 + y^2 \le z^2 \;\&\&\; \exists_t \left(\forall_u\, t\, x > u\, y\, z + 7 \;||\; x^2\, t == 2\, z + 1\right),\ \{x, y, z\},\ \textbf{Reals}\right]$$

Out[1]= $$\left(x < 0 \;\&\&\; \left(z \le -\sqrt{x^2+y^2} \;||\; z \ge \sqrt{x^2+y^2}\right)\right) \;||\; \left(x == 0 \;\&\&\; -\frac{1}{2} \le y \le \frac{1}{2} \;\&\&\; z == -\frac{1}{2}\right) \;||$$
$$\left(0 < x < \frac{1}{2} \;\&\&\; \left(\left(y < -\frac{1}{2}\sqrt{1-4x^2} \;\&\&\; \left(z \le -\sqrt{x^2+y^2} \;||\; z \ge \sqrt{x^2+y^2}\right)\right) \;||\right.\right.$$
$$\left(y == -\frac{1}{2}\sqrt{1-4x^2} \;\&\&\; \left(z \le -\frac{1}{2} \;||\; z \ge \sqrt{x^2+y^2}\right)\right) \;||$$
$$\left(-\frac{1}{2}\sqrt{1-4x^2} < y < \frac{1}{2}\sqrt{1-4x^2} \;\&\&\; \left(z \le -\sqrt{x^2+y^2} \;||\; z \ge \sqrt{x^2+y^2}\right)\right) \;||\; \left(y == \frac{1}{2}\sqrt{1-4x^2} \;\&\&\right.$$
$$\left.\left.\left.\left(z \le -\frac{1}{2} \;||\; z \ge \sqrt{x^2+y^2}\right)\right) \;||\; \left(y > \frac{1}{2}\sqrt{1-4x^2} \;\&\&\; \left(z \le -\sqrt{x^2+y^2} \;||\; z \ge \sqrt{x^2+y^2}\right)\right)\right)\right) \;||$$
$$\left(x == \frac{1}{2} \;\&\&\; \left(\left(y < 0 \;\&\&\; \left(z \le -\sqrt{\frac{1}{4}+y^2} \;||\; z \ge \sqrt{\frac{1}{4}+y^2}\right)\right) \;||\; \left(y == 0 \;\&\&\; \left(z \le -\frac{1}{2} \;||\; z \ge \frac{1}{2}\right)\right) \;||\right.\right.$$
$$\left.\left.\left(y > 0 \;\&\&\; \left(z \le -\sqrt{\frac{1}{4}+y^2} \;||\; z \ge \sqrt{\frac{1}{4}+y^2}\right)\right)\right)\right) \;||\; \left(x > \frac{1}{2} \;\&\&\; \left(z \le -\sqrt{x^2+y^2} \;||\; z \ge \sqrt{x^2+y^2}\right)\right)$$

This defines a function expanding $\land$ with respect to $\lor$.

In[2]:=
```
lexp[e_Or] := lexp /@ e
lexp[And[a___, b_Or, c___]] := lexp[And[a, #, c]] & /@ b
lexp[other_] := other
```

Here is the solution of the system (1) written explicitly as a union of disjoint cells.

In[5]:= **lexp[sol]**

Out[5]= $\left(x<0\ \&\&\ z\le -\sqrt{x^2+y^2}\right)\ ||\ \left(x<0\ \&\&\ z\ge \sqrt{x^2+y^2}\right)\ ||\ \left(x==0\ \&\&\ -\frac{1}{2}\le y\le \frac{1}{2}\ \&\&\ z==-\frac{1}{2}\right)\ ||$
$\left(0<x<\frac{1}{2}\ \&\&\ y<-\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\le -\sqrt{x^2+y^2}\right)\ ||\ \left(0<x<\frac{1}{2}\ \&\&\ y<-\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\ge \sqrt{x^2+y^2}\right)\ ||$
$\left(0<x<\frac{1}{2}\ \&\&\ y==-\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\le -\frac{1}{2}\right)\ ||\ \left(0<x<\frac{1}{2}\ \&\&\ y==-\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\ge \sqrt{x^2+y^2}\right)\ ||$
$\left(0<x<\frac{1}{2}\ \&\&\ -\frac{1}{2}\sqrt{1-4x^2}<y<\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\le -\sqrt{x^2+y^2}\right)\ ||$
$\left(0<x<\frac{1}{2}\ \&\&\ -\frac{1}{2}\sqrt{1-4x^2}<y<\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\ge \sqrt{x^2+y^2}\right)\ ||$
$\left(0<x<\frac{1}{2}\ \&\&\ y==\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\le -\frac{1}{2}\right)\ ||\ \left(0<x<\frac{1}{2}\ \&\&\ y==\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\ge \sqrt{x^2+y^2}\right)\ ||$
$\left(0<x<\frac{1}{2}\ \&\&\ y>\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\le -\sqrt{x^2+y^2}\right)\ ||\ \left(0<x<\frac{1}{2}\ \&\&\ y>\frac{1}{2}\sqrt{1-4x^2}\ \&\&\ z\ge \sqrt{x^2+y^2}\right)\ ||$
$\left(x==\frac{1}{2}\ \&\&\ y<0\ \&\&\ z\le -\sqrt{\frac{1}{4}+y^2}\right)\ ||\ \left(x==\frac{1}{2}\ \&\&\ y<0\ \&\&\ z\ge \sqrt{\frac{1}{4}+y^2}\right)\ ||$
$\left(x==\frac{1}{2}\ \&\&\ y==0\ \&\&\ z\le -\frac{1}{2}\right)\ ||\ \left(x==\frac{1}{2}\ \&\&\ y==0\ \&\&\ z\ge \frac{1}{2}\right)\ ||\ \left(x==\frac{1}{2}\ \&\&\ y>0\ \&\&\ z\le -\sqrt{\frac{1}{4}+y^2}\right)\ ||$
$\left(x==\frac{1}{2}\ \&\&\ y>0\ \&\&\ z\ge \sqrt{\frac{1}{4}+y^2}\right)\ ||\ \left(x>\frac{1}{2}\ \&\&\ z\le -\sqrt{x^2+y^2}\right)\ ||\ \left(x>\frac{1}{2}\ \&\&\ z\ge \sqrt{x^2+y^2}\right)$

`Resolve` can eliminate quantifiers from arbitrary real polynomial systems. If no variables are specified in the input and all input polynomials are at most linear in the bound variables, `Resolve` may be able to eliminate the quantifiers without solving the resulting system. Otherwise, `Resolve` uses the same algorithm and gives the same answer as `Reduce`.

This eliminates quantifiers from the system (1).

In[6]:= **Resolve**$\left[x^2+y^2\le z^2\ \&\&\ \exists_t\left(\forall_u\ t\,x>u\,y\,z+7\ ||\ x^2\,t==2z+1\right),\ \text{Reals}\right]$

Out[6]= $\left(\frac{1}{2}+z==0\ \&\&\ x^2+y^2-z^2\le 0\right)\ ||\ \left(x^2\ne 0\ \&\&\ x^2+y^2-z^2\le 0\right)\ ||$
$\left(-x<0\ \&\&\ y\,z==0\ \&\&\ x^2+y^2-z^2\le 0\right)\ ||\ \left(x<0\ \&\&\ y\,z==0\ \&\&\ x^2+y^2-z^2\le 0\right)$

`FindInstance` can handle arbitrary real polynomial systems, giving instances of real solutions or an empty list for systems that have no solutions. If the number of instances requested is more than one, the instances are randomly generated from the full solution of the system and therefore may depend on the value of the `RandomSeed` option. If one instance is requested and the system does not contain general (∀) quantifiers, a faster algorithm producing one instance is used and the instance returned is always the same.

This finds a solution of the system (1).

In[7]:= `FindInstance[x^2 + y^2 ≤ z^2 && ∃_t (∀_u t x > u y z + 7 || x^2 t == 2 z + 1), {x, y, z}, Reals]`

Out[7]= $\left\{\left\{x \to -18,\ y \to \frac{8}{5},\ z \to -115\right\}\right\}$

The main general tool used in solving real polynomial systems is the Cylindrical Algebraic Decomposition (CAD) algorithm (see, for example, [1]). CAD for quantifier-free systems is available in *Mathematica* directly as `CylindricalDecomposition`. There are also several other algorithms used to solve special case problems.

Cylindrical Algebraic Decomposition

Semi-Algebraic Sets and Cell Decomposition

A subset of $\mathbb{R}^n$ is *semi-algebraic* if it is a solution set of a quantifier-free real polynomial system. According to Tarski's theorem [2], solution sets of arbitrary (quantified) real polynomial systems are semi-algebraic.

Every semi-algebraic set can be represented as a finite union of disjoint cells [3] defined recursively as follows:

- A cell in $\mathbb{R}$ is a point or an open interval
- A cell in $\mathbb{R}^k$ has one of the two forms

$$\begin{array}{l} \{(a_1, \ldots, a_k, a_{k+1}) : (a_1, \ldots, a_k) \in C_k \wedge a_{k+1} = r(a_1, \ldots, a_k)\} \\ \{(a_1, \ldots, a_k, a_{k+1}) : (a_1, \ldots, a_k) \in C_k \wedge r_1\,(a_1, \ldots, a_k) < a_{k+1} < r_2(a_1, \ldots, a_k)\}, \end{array} \tag{6}$$

where C_k is a cell in $\mathbb{R}^k$, r is a continuous algebraic function, r_1 and r_2 are continuous algebraic functions or $-\infty$ or ∞, and $r_1 < r_2$ on C_k.

By an algebraic function we mean a function $r : C_k \longrightarrow \mathbb{R}$ for which there is a polynomial

$$f = c_0\, x_{k+1}{}^m + c_1\, x_{k+1}{}^{m-1} + \ldots \mathrm{c}_m \in \mathbb{R}[x_1, \ldots, x_k, x_{k+1}]$$

such that

$$c_0(a_1, \ldots, a_k) \neq 0 \wedge f\,(a_1, \ldots, a_k, r(a_1, \ldots, a_k)) = 0.$$

In *Mathematica* algebraic functions can be represented as `Root` objects or radicals.

The CAD algorithm, introduced by Collins [4], computes a cell decomposition of solution sets of arbitrary real polynomial systems. The objective of the original Collins algorithm was to eliminate quantifiers from a quantified real polynomial system and to produce an equivalent quantifier-free polynomial system. After finding a cell decomposition, the algorithm performed an additional step of finding an implicit representation of the semi-algebraic set in terms of polynomial equations and inequalities in the free variables. The objective of `Reduce` is somewhat different. Given a semi-algebraic set presented by a real polynomial system, quantified or not, `Reduce` finds a cell decomposition of the set, explicitly written in terms of algebraic functions.

While `Reduce` may use other methods to solve the system, `CylindricalDecomposition` gives a direct access to the CAD algorithm. For a quantifier-free real polynomial system, `CylindricalDecomposition` gives a nested formula representing disjunction of cells in the solved form (4). As in the output of `Reduce`, the cells are disjoint and additionally are always ordered lexicographically with respect to ranges of the subsequent variables.

This finds a cell decomposition of an annulus.

```
In[8]:= CylindricalDecomposition[1 ≤ x^2 + y^2 < 2, {x, y}]
```

Out[8]= $\left(-\sqrt{2} < x < -1 \,\&\&\, -\sqrt{2-x^2} < y < \sqrt{2-x^2}\right) \,||$
$\left(-1 \le x \le 1 \,\&\&\, \left(-\sqrt{2-x^2} < y \le -\sqrt{1-x^2} \,||\, \sqrt{1-x^2} \le y < \sqrt{2-x^2}\right)\right) \,||$
$\left(1 < x < \sqrt{2} \,\&\&\, -\sqrt{2-x^2} < y < \sqrt{2-x^2}\right)$

The Projection Phase of the CAD Algorithm

Finding a cell decomposition of a semi-algebraic set using the CAD algorithm consists of two phases, projection and lifting. In the projection phase, we start with the set A_{n+m} of factors of the polynomials present in the quantifier-free part $\Phi(x_1, \ldots, x_n; y_1, \ldots, y_m)$ of the system (2) and eliminate variables one by one using a projection operator P such that

$$P_{k+1} : \mathbb{R}[t_1, \ldots, t_k, t_{k+1}] \supset A_{k+1} \longrightarrow A_k \subset \mathbb{R}[t_1, \ldots, t_k].$$

Generally speaking, if all polynomials of A_k have constant signs on a cell $C \subset \mathbb{R}^k$, then all polynomials of A_{k+1} are delineable over C, that is, each has a fixed number of real roots on C as a polynomial in t_{k+1}, the roots are continuous functions on C, they have constant multiplicities, and two roots of two of the polynomials are equal either everywhere or nowhere in C. Variables are ordered so that

$$(t_1, \ldots, t_{n+m}) = (x_1, \ldots, x_n, y_1, \ldots, y_m).$$

This way the roots of polynomials of $A_1, \ldots, A_n$ are the algebraic functions needed in the construction of the cell decomposition of the semi-algebraic set.

Several improvements have reduced the size of the original Collins projection. The currently best projection operator applicable in all cases is due to Hong [5]; however, in most situations we can use a smaller projection operator given by McCallum [6, 7], with an improvement by Brown [8]. There are even smaller projection operators that can be applied in some special cases. When equational constraints are present, we can use the projection operator suggested by Collins [9], and developed and proven by McCallum [10, 11]. When there are no equations and only strict inequalities, and there are no free variables or we are interested only in the full-dimensional part of the semi-algebraic set, we can use an even smaller projection operator described in [12, 13]. For systems containing equational constraints that generate a zero-dimensional ideal, Gröbner bases are used to find projection polynomials.

Mathematica uses the smallest of the previously mentioned projections that is appropriate for the given example. Whenever applicable, we use the equational constraints; otherwise, we attempt to use McCallum's projection with Brown's improvement. When the system does not turn out to be well oriented, we compute Hong's projection.

The Lifting Phase of the CAD Algorithm

In the lifting phase, we find a cell decomposition of the semi-algebraic set. Generally speaking, although the actual details depend on the projection operator used, we start with cells in $\mathbb{R}^1$ consisting of all distinct roots of A_1 and the open intervals between the roots. We find a sample point in each of the cells and remove the cells whose sample points do not satisfy the system describing the semi-algebraic set (the system may contain conditions involving only t_1). Next we lift the cells to cells in $\mathbb{R}^n$, one dimension at a time. Suppose we have lifted the cells to $\mathbb{R}^k$. To lift a cell $C \subset \mathbb{R}^k$ to $\mathbb{R}^{k+1}$, we find the real roots of A_{k+1} with $t_1, \ldots, t_k$ replaced with the coordinates of the sample point c in C. Since the polynomials of A_{k+1} are delineable on C, each root r is a value of a continuous algebraic function at c, and the function can be represented as a p^{th} root of a polynomial $f \in A_{k+1}$ such that r is the p^{th} root of $f(c, t_{k+1})$. Now the lifting of the cell C to $\mathbb{R}^{k+1}$ will consist of graphs of these algebraic functions and of the slices of $C\times\mathbb{R}$ between the subsequent graphs. The sample points in each of the new cells will be obtained by adding the $k+1^{\text{st}}$ coordinate to c, equal to one of the roots, or to a number between two subsequent roots. As in the first step, we remove those lifted cells whose sample points do not satisfy the system describing the semi-algebraic set.

If $k \geq n$, $t_{k+1} = y_l$ is a quantifier variable and we may not need to construct all the lifted cells. All we need is to find the (necessarily constant) truth value of $Q_{l\,y_l}\, Q_{l+1\,y_{l+1}} \ldots Q_{m\,y_m}\, \Phi$ on C. If $Q_l == \exists$, we know that the value is `True` as soon as the truth value of $Q_{l+1\,y_{l+1}} \ldots Q_{m\,y_m}\, \Phi$ on one of the lifted cells is `True`. If $Q_l == \forall$, we know that the value is `False` as soon as the truth value of $Q_{l+1\,y_{l+1}} \ldots Q_{m\,y_m}\, \Phi$ on one of the lifted cells is `False`.

The coefficients of sample points computed this way are in general algebraic numbers. To save costly algebraic number computations, *Mathematica* uses arbitrary-precision floating-point number (*Mathematica* "bignum") approximations of the coefficients, whenever the results can be validated. Note that using approximate arithmetic may be enough to prove that two roots of a polynomial or a pair of polynomials are distinct, and to find a nonzero sign of a polynomial at a sample point. What we cannot prove with approximate arithmetic is that two roots of a polynomial or a pair of polynomials are equal, or that a polynomial is zero at a sample point. However, we can often use information about the origins of the cell to resolve these problems. For instance, if we know that the resultant of two polynomials vanishes on the cell, and these two polynomials have exactly one pair of complex roots that can be equal within the precision bounds, we can conclude that these roots are equal. Similarly, if the last coordinate of a sample point was a root of a factor of the given polynomial, we know that this polynomial is zero at the sample point. If we cannot resolve all the uncertainties using the collected information about the cell, we compute the exact algebraic number values of the coordinates. For more details, see [14, 24].

Decision Problems, FindInstance, and Assumptions

A *decision problem* is a system with all variables existentially quantified, that is, a system of the form

$$\exists_{x_1} \exists_{x_2} \ldots \exists_{x_n} \Phi(x_1, \ldots, x_n),$$

where $x_1, \ldots, x_n$ are all variables in Φ. Solving a decision problem means deciding whether it is equivalent to `True` or to `False`, that is, deciding whether the quantifier-free system of polynomial equations and inequalities $\Phi(x_1, \ldots, x_n)$ has solutions.

All algorithms used by *Mathematica* to solve real polynomial decision problems are capable of producing a point satisfying $\Phi(x_1, \ldots, x_n)$ if the system has solutions. Therefore the algorithms discussed in this section are used not only in `Reduce` and `Resolve` for decision problems, but also in `FindInstance`, whenever a single instance is requested and the system is quantifier free or contains only existential quantifiers. The algorithms discussed here are also used for inference testing by *Mathematica* functions using assumptions such as `Simplify`, `Refine`, `Integrate`, and so forth.

Solving this decision problem proves that the set $S = \{(x, y) \in \mathbb{R}^2 : x^4 + y^4 - 2\,x\,y \leq 1\}$ contains the disk of radius 4/5 centered at the origin.

In[9]:= $\textbf{Reduce}\left[\exists_{\{x,y\}} \left(x^2 + y^2 \leq \frac{16}{25} \text{ \&\& } x^4 + y^4 - 2\,x\,y > 1\right), \textbf{Reals}\right]$

Out[9]= `False`

This shows that S does not contain the unit disk and provides a counterexample: a point in the unit disk that does not belong to S.

In[10]:= `FindInstance[x^2 + y^2 ≤ 1 && x^4 + y^4 - 2 x y > 1, {x, y}, Reals]`

Out[10]= $\left\{\left\{x \to \frac{3}{4},\ y \to -\frac{1}{2}\right\}\right\}$

The primary method that allows *Mathematica* to solve arbitrary real polynomial decision problems is the Cylindrical Algebraic Decomposition (CAD) algorithm. There are, however, several other special case algorithms that provide much better performance in cases in which they are applicable.

When all polynomials are linear with rational number or floating-point number coefficients, *Mathematica* uses a method based on the Simplex linear programming method. For other linear systems, *Mathematica* uses a variant of the Loos-Weispfenning linear quantifier elimination algorithm [15]. When the system contains no equations and only strict inequalities, a faster "generic" version of CAD is used [12, 13]. For systems containing equational constraints that generate a zero-dimensional ideal, *Mathematica* uses Gröbner bases to find a solution. For nonlinear systems with floating-point number coefficients, an inexact coefficient version of CAD [16] is used.

There are also some special case methods that can be used as preprocessors to other decision methods. When the system contains an equational constraint linear with a constant coefficient in one of the variables, the constraint is used to eliminate the linear variable. If there is a variable that appears in the system only linearly with constant coefficients, the variable is eliminated using the Loos-Weispfenning linear quantifier elimination algorithm [15]. If there is a variable that appears in the system only quadratically, the quadratic case of Weispfenning's quantifier elimination by virtual substitution algorithm [22, 23] could be used to eliminate the variable. For some examples this gives a substantial speedup; however, quite often it results in a significant slowdown. By default, the algorithm is not used as a preprocessor. Setting the system option `QVSPreprocessor` in the `InequalitySolvingOptions` group to `True` makes *Mathematica* use it.

There are two other special cases of real decision algorithms available in *Mathematica*. An algorithm by Aubry, Rouillier, and Safey El Din [17] applies to systems containing only equations. There are examples for which the algorithm performs much better than CAD; however, for randomly chosen systems of equations, it seems to perform significantly worse; therefore, it is not used by default. Setting the system option `ARSDecision` in the `InequalitySolvingOptions` group to `True` causes *Mathematica* to use the algorithm. Another algorithm by G. X. Zeng and X. N. Zeng [18] applies to systems that consist of a single strict inequality. Again, the algorithm is faster than CAD for some examples, but slower in general; therefore, it is not used by default. Setting the system option `ZengDecision` in the `InequalitySolvingOptions` group to `True` causes *Mathematica* to use the algorithm.

Arbitrary Real Polynomial Systems

Solving Real Polynomial Systems

According to Tarski's theorem [2], the solution set of an arbitrary (quantified) real polynomial system is a semi-algebraic set. `Reduce` gives a description of this set in the solved form (4).

This shows for what $r>0$ the set $S=\{(x,y)\in\mathbb{R}^2 : x^4+y^4-2\,x\,y\le 1\}$ contains the disk of radius r centered at the origin.

In[11]:= **Reduce**$\left[\forall_{\{x,y\},r>0\,\&\&\,x^2+y^2\le r^2}\ \mathbf{x^4+y^4-2\,x\,y\le 1,\ r,\ Reals}\right]$

Out[11]= $r \le \mathrm{Root}\left[-2+2\,\#1^2+\#1^4\ \&,\ 2\right]$

This gives the projection of $x^2+y^2+z^2-x\,y\,z\le 1$ on the (x,y) plane along the z axis.

In[12]:= **Reduce**$\left[\exists_z\ \mathbf{x^2+y^2+z^2-x\,y\,z\le 1,\ \{x,\ y\}}\right]$

Out[12]= $$\left(x<-2\ \&\&\ \left(y\le-\sqrt{\frac{-4+4\,x^2}{-4+x^2}}\ ||\ y\ge\sqrt{\frac{-4+4\,x^2}{-4+x^2}}\right)\right)\ ||$$
$$(x==-1\ \&\&\ y==0)\ ||\ \left(-1<x<1\ \&\&\ -\sqrt{\frac{-4+4\,x^2}{-4+x^2}}\le y\le\sqrt{\frac{-4+4\,x^2}{-4+x^2}}\right)\ ||$$
$$(x==1\ \&\&\ y==0)\ ||\ \left(x>2\ \&\&\ \left(y\le-\sqrt{\frac{-4+4\,x^2}{-4+x^2}}\ ||\ y\ge\sqrt{\frac{-4+4\,x^2}{-4+x^2}}\right)\right)$$

This finds the projection of Whitney's umbrella $x^2-y^2\,z==0$ on the (y,z) plane along the x axis.

In[13]:= **Reduce**$\left[\exists_x\ \mathbf{x^2-y^2\,z==0,\ \{y,\ z\},\ Reals}\right]$

Out[13]= $(y<0\ \&\&\ z\ge 0)\ ||\ y==0\ ||\ (y>0\ \&\&\ z\ge 0)$

Here we find the interior of the previous projection set by directly using the definition.

In[14]:= Reduce[$\exists_{d,d>0}$ ($\forall_{\{v,w\},(v-y)^2+(w-z)^2\leq d}$ ($\exists_u$ $u^2 - v^2 w == 0$)), {y, z}, Reals]

Out[14]= $z > 0$

Quantifier Elimination

The objective of `Resolve` with no variables specified is to eliminate quantifiers and produce an equivalent quantifier-free formula. The formula may or may not be in a solved form, depending on the algorithm used.

Producing a fully solved quantifier-free formula here is difficult because of the complexity of polynomials in a, b, and c appearing in the input. However, since x appears in the input polynomials only linearly, the quantifier can be quickly eliminated using the Loos-Weispfenning linear quantifier elimination algorithm, which depends very little on the complexity of coefficients.

In[15]:= Resolve[$\exists_x$ (a x ≥ b^3 - 3 a c^2 - 5 a^3 b c + 9 && b c^2 x - 3 x ≤ 11 a^2 b - 3 c^3 + 4 a b^2 c + 9)]

Out[15]= (a | b | c) ∈ Reals &&
((-a < 0 && -27 - 9 a - 11 a^3 b - 3 b^3 + 15 a^3 b c - 4 a^2 b^2 c + 9 a c^2 + 9 b c^2 + b^4 c^2 + 3 a c^3 - 5 a^3 b^2 c^3 - 3 a b c^4 ≤ 0) || (a < 0 &&
27 + 9 a + 11 a^3 b + 3 b^3 - 15 a^3 b c + 4 a^2 b^2 c - 9 a c^2 - 9 b c^2 - b^4 c^2 - 3 a c^3 + 5 a^3 b^2 c^3 + 3 a b c^4 ≤ 0) ||
(3 - b c^2 < 0 && -27 - 9 a - 11 a^3 b - 3 b^3 + 15 a^3 b c - 4 a^2 b^2 c + 9 a c^2 + 9 b c^2 +
b^4 c^2 + 3 a c^3 - 5 a^3 b^2 c^3 - 3 a b c^4 ≤ 0) || (-3 + b c^2 < 0 &&
27 + 9 a + 11 a^3 b + 3 b^3 - 15 a^3 b c + 4 a^2 b^2 c - 9 a c^2 - 9 b c^2 - b^4 c^2 - 3 a c^3 + 5 a^3 b^2 c^3 + 3 a b c^4 ≤ 0) ||
(a == 0 && 9 + b^3 - 5 a^3 b c - 3 a c^2 ≤ 0 && -9 - 11 a^2 b - 4 a b^2 c + 3 c^3 ≤ 0) ||
(-3 + b c^2 == 0 && 9 + b^3 - 5 a^3 b c - 3 a c^2 ≤ 0 && -9 - 11 a^2 b - 4 a b^2 c + 3 c^3 ≤ 0))

Algorithms

The primary method used by *Mathematica* for solving real polynomial systems and real quantifier elimination is the CAD algorithm. There are, however, simpler methods applicable in special cases.

If the system contains an equational constraint in a variable from the innermost quantifier, the constraint is used to simplify the system using the identity

$$\exists_y\, a\, y == b \wedge \Phi(x_1, \ldots, x_n; y) \Longleftrightarrow a \neq 0 \wedge \Phi(x_1, \ldots, x_n; b/a) \vee \exists_y\, a == 0 \wedge b == 0 \wedge \Phi(x_1, \ldots, x_n; y).$$

Note that if a or b is a nonzero constant, this eliminates the variable y.

If all polynomials in the system are linear in a variable from the innermost quantifier, the variable is eliminated using the Loos-Weispfenning linear quantifier elimination algorithm [15].

If all polynomials in the system are at most quadratic in a variable from the innermost quantifier, the variable is eliminated using the quadratic case of Weispfenning's quantifier elimination by virtual substitution algorithm [22, 23]. With the default setting of the system option `QuadraticQE`, the algorithm is used for `Resolve` with no variables specified and with at least two parameters present, and for `Reduce` and `Resolve` with at least three variables as long as elimination of one variable at most doubles the `LeafCount` of the system.

The CAD algorithm is used when the previous three special case methods are no longer applicable, but there are still quantifiers left to eliminate or a solution is required.

For systems containing equational constraints that generate a zero-dimensional ideal, *Mathematica* uses Gröbner bases to find the solution set.

Options

The *Mathematica* functions for solving real polynomial systems have a number of options that control the way that they operate. This section gives a summary of these options.

option name	*default value*	
`Cubics`	`False`	whether the Cardano formulas should be used to express numeric solutions of cubics
`Quartics`	`False`	whether the Cardano formulas should be used to express numeric solutions of quartics
`WorkingPrecision`	∞	the working precision to be used in computations

`Reduce`, `Resolve`, and `FindInstance` options affecting the behavior for real polynomial systems.

Cubics and Quartics

By default, `Reduce` does not use the Cardano formulas for solving cubics or quartics over the reals.

```
In[16]:= Reduce[x^3 - 3 x + 7 == 0, x, Reals]

Out[16]= x == Root[7 - 3 #1 + #1^3 &, 1]
```

Setting options `Cubics` and `Quartics` to `True` makes `Reduce` use the Cardano formulas to represent numeric solutions of cubics and quartics.

```
In[17]:= Reduce[x^3 - 3 x + 7 == 0, x, Reals, Cubics → True]
```

$$\text{Out[17]=}\quad x == -\left(\frac{2}{7-3\sqrt{5}}\right)^{1/3} - \left(\frac{1}{2}\left(7-3\sqrt{5}\right)\right)^{1/3}$$

Solutions of cubics and quartics involving parameters will still be represented using `Root` objects.

```
In[18]:= Reduce[x^3 == a, x, Reals, Cubics → True]

Out[18]= x == Root[-a + #1^3 &, 1]
```

This is because the Cardano formulas do not separate real solutions from nonreal ones. For instance, in this case, for $a = -1$ the third radical solution is real, but for $a = 1$ the first radical solution is real.

```
In[19]:= sol = Reduce[x^3 == a, x, Cubics → True]
```

$$\text{Out[19]=}\quad x == a^{1/3} \;||\; x == -(-1)^{1/3} a^{1/3} \;||\; x == (-1)^{2/3} a^{1/3}$$

```
In[20]:= sol /. {{a → -1}, {a → 1}}
```

$$\text{Out[20]=}\quad \{x == (-1)^{1/3} \;||\; x == -(-1)^{2/3} \;||\; x == -1,\; x == 1 \;||\; x == -(-1)^{1/3} \;||\; x == (-1)^{2/3}\}$$

WorkingPrecision

The setting of `WorkingPrecision` affects the lifting phase of the CAD algorithm. With a finite working precision $prec$, sample points in the first variable lifted are represented as arbitrary-precision floating-point numbers with $prec$ digits of precision. When we compute sample points for subsequent variables, we find roots of polynomials whose coefficients depend on already computed sample point coordinates and therefore may be inexact. Hence coordinates of sample points will have precision $prec$ or lower. Determining the sign of polynomials at sample points is simply done by evaluating `Sign` of the floating-point number obtained after the substitution. Using a finite `WorkingPrecision` may allow getting the answer faster; however, the answer may be incorrect or the computation may fail due to loss of precision.

This problem is too hard for `Reduce` working in infinite `WorkingPrecision`, due to the high degrees of the algebraic numbers involved. Using sample points with 30 digits of precision gives a solution in under two seconds.

```
In[21]:= Reduce[∃{y,z} (x^4 + 2 y^4 + 3 z^4 ≤ 1 && x^3 - 9 y^3 + 7 z^3 ≥ 2),
           x, Reals, WorkingPrecision → 30] // Timing

Out[21]= {0.821, Root[-192 899 - 6912 #1^3 + 589 065 #1^4 + 5184 #1^6 - 589 065 #1^8 - 1728 #1^9 + 196 571 #1^12 &, 1] ≤ x ≤
           Root[-192 899 - 6912 #1^3 + 589 065 #1^4 + 5184 #1^6 - 589 065 #1^8 - 1728 #1^9 + 196 571 #1^12 &, 2]}
```

ReduceOptions Group of System Options

Here are the system options from the `ReduceOptions` group that may affect the behavior of `Reduce`, `Resolve`, and `FindInstance` for real polynomial systems. The options can be set with

`SetSystemOptions["ReduceOptions" -> {"`*option name*`" -> `*value*`}]`.

option name	*default value*	
`"FactorInequalities"`	`False`	whether inequalities should be factored at the input preprocessing stage
`"ReorderVariables"`	`False`	whether `Reduce` and `Resolve` are allowed to reorder the specified variables

`ReduceOptions` group options affecting the behavior of `Reduce`, `Resolve`, and `FindInstance` for real polynomial systems.

FactorInequalities

Using transformations

$$\begin{aligned} f\,g < 0 &\longrightarrow f < 0 \wedge g > 0 \vee f > 0 \wedge g < 0 \\ f\,g \le 0 &\longrightarrow f \le 0 \wedge g \ge 0 \vee f \ge 0 \wedge g \le 0 \end{aligned} \tag{7}$$

at the input preprocessing stage may speed up the computations in some cases. In general, however, it does not make the problem easier to solve, and, in some cases, it may make the problem significantly harder. By default, these transformations are not used.

Here Reduce does not use transformations (7).

```
In[22]:= t1 =
           Timing[Reduce[(x^3 - 5 x y^2 - 3 y^2 + 7 z^2 - 1) (x^2 - 3 x y + 5 y^2 + 3 y z - 2) (x^2 - 2 z + y - 3) ≤ 0,
               {x, y, z}, Reals]][[1]];
         t2 = Timing[Reduce[Product[(x - y i), {i, 1, 10}] ≤ 0, {x, y}, Reals]][[1]];
         t3 = Timing[Reduce[
               y^21 - x y^7 + z - 1 < 0 && y^14 + 3 x^2 y^7 - 11 z + 7 > 0 && y^7 ≥ 0, {x, y, z}, Reals]][[1]];
         {t1,
          t2,
          t3}

Out[25]= {8.152, 0.02, 0.04}
```

Using transformations (7) speeds up the first example; however, it makes the other two examples significantly slower. The second example suffers from exponential growth of the number of inequalities. By replacing $y^7 \geq 0$ with $y \geq 0$ in the third example, we get a degree-21 system in y instead of a degree-3 system in y^7.

```
In[26]:= SetSystemOptions["ReduceOptions" → "FactorInequalities" → True];
         t1 = Timing[Reduce[(x^3 - 5 x y^2 - 3 y^2 + 7 z^2 - 1)
                  (x^2 - 3 x y + 5 y^2 + 3 y z - 2) (x^2 - 2 z + y - 3) ≤ 0, {x, y, z}, Reals]][[1]];
           t2 = Timing[Reduce[Product[(x - y i), {i, 1, 10}] ≤ 0, {x, y}, Reals]][[1]];
           t3 = Timing[Reduce[y^21 - x y^7 + z - 1 < 0 && y^14 + 3 x^2 y^7 - 11 z + 7 > 0 && y^7 ≥ 0,
                {x, y, z}, Reals]][[1]];
         {t1,
          t2,
          t3}

Out[28]= {7.861, 8.833, 0.411}

In[29]:= SetSystemOptions["ReduceOptions" → "FactorInequalities" → False];
```

ReorderVariables

By default, Reduce is not allowed to reorder the specified variables. Variables appearing earlier in the variable list may be used to express solutions for variables appearing later in the variable list, but not vice versa.

```
In[30]:= Reduce[x > y^3 + 7 y - 1, {x, y}, Reals]

Out[30]= y < Root[-1 - x + 7 #1 + #1^3 &, 1]
```

Setting the system option ReorderVariables -> True allows Reduce to pick a variable order that makes the system easier to solve.

```
In[31]:= SetSystemOptions["ReduceOptions" → "ReorderVariables" → True];
         Reduce[x > y^3 + 7 y - 1, {x, y}, Reals]

Out[32]= x > -1 + 7 y + y^3
```

```
In[33]:= SetSystemOptions["ReduceOptions" → "ReorderVariables" → False];
```

InequalitySolvingOptions Group of System Options

Here are the system options from the `InequalitySolvingOptions` group that may affect the behavior of `Reduce`, `Resolve`, and `FindInstance` for real polynomial systems. The options can be set with

`SetSystemOptions["InequalitySolvingOptions" -> {"`*option name*`" -> `*value*`}]`.

option name	*default value*	
"ARSDecision"	False	whether to use the decision algorithm given in [17]
"BrownProjection"	True	whether the CAD algorithm should use the improved projection operator given in [8]
"CAD"	True	whether to use the CAD algorithm
"CADDefaultPrecision"	30.103	the precision to which nonrational roots are computed in the lifting phase of the CAD algorithm; if computation with approximate roots cannot be validated, the algorithm reverts to exact algebraic number computation
"CADSortVariables"	True	whether the CAD algorithm should use variable reordering heuristics for quantifier variables within a single quantifier or in decision problems
"CADZeroTest"	{0,∞}	determines the zero testing method used by the CAD algorithm for expressions obtained by evaluating polynomials at points with algebraic number coordinates
"ContinuedFractionRootIsolation"	True	whether the CAD algorithm should use a real root isolation method based on continued fractions rather than on interval bisection [19]

`"EquationalConstraintsCAD"`	`Automatic`	whether the projection phase of the CAD algorithm should use equational constraints; with the default `Automatic` setting the operator proven correct in [11] is used; if `True` the unproven projection operator using multiple equational constraints suggested in [4] is used
`"FGLMBasisConversion"`	`False`	whether the CAD algorithm should use a Gröbner basis conversion algorithm based on [20] to find univariate polynomials in zero-dimensional Gröbner bases; otherwise, `GroebnerWalk` is used
`"FGLMElimination"`	`Automatic`	whether the decision and quantifier elimination algorithms for systems with equational constraints forming a zero-dimensional ideal should use an algorithm based on [20] to look for linear equation constraints (with constant leading coefficients) in one of the variables to be used for elimination
`"GenericCAD"`	`True`	whether to use the variant of the CAD algorithm described in [13] for decision and optimization problems
`"GroebnerCAD"`	`True`	whether the CAD algorithm for systems with equational constraints forming a zero-dimensional ideal should use Gröbner bases as projection
`"LinearDecisionMethodCrossovers"`	`{0,30,20}`	determines methods used to find solutions of systems of linear equations and inequalities with rational number coefficients
`"LinearEquations"`	`True`	whether to use linear equation constraints (with constant leading coefficients) to eliminate variables in decision problems
`"LinearQE"`	`True`	whether to use the Loos-Weispfenning linear quantifier elimination algorithm [15] for quantifier elimination problems
`"LWDecision"`	`True`	whether to use the Loos-Weispfenning linear quantifier elimination algorithm [15] for decision problems with linear inequality systems

"LWPreprocessor"	`Automatic`	whether to use the Loos-Weispfenning linear quantifier elimination algorithm [15] as a preprocessor for the decision problems
"ProjectAlgebraic"	`Automatic`	whether the CAD algorithm should compute projections with respect to variables replacing algebraic number coefficients or use their minimal polynomials instead
"ProveMultiplicities"	`True`	determines the way in which the lifting phase of the CAD algorithm validates multiple roots and zero leading coefficients of projection polynomials
"QuadraticQE"	`Automatic`	whether to use the quadratic case of Weispfenning's quantifier elimination by virtual substitution algorithm in quantifier elimination
"QVSPreprocessor"	`False`	whether to use the quadratic case of Weispfenning's quantifier elimination by virtual substitution algorithm as a preprocessor for the decision problems
"ReducePowers"	`True`	whether to replace x^d with x in the input to the CAD, where d is the GCD of all exponents of x in the system
"RootReduced"	`False`	whether the coordinates of solutions of systems with equational constraints forming a zero-dimensional ideal should be reduced to single `Root` objects
"Simplex"	`True`	whether to use the Simplex algorithm in the decision algorithm for linear inequality systems
"ThreadOr"	`True`	whether to solve each case of disjunction separately in decision problems, optimization, and in quantifier elimination of existential quantifiers when the quantifier-free system does not need to be solved
"ZengDecision"	`False`	whether to use the decision algorithm given in [18]

`InequalitySolvingOptions` group options affecting the behavior of `Reduce`, `Resolve`, and `FindInstance` for real polynomial systems.

ARSDecision

The option `ARSDecision` specifies whether *Mathematica* should use the algorithm by Aubry, Rouillier, and Safey El Din [17]. The algorithm applies to decision problems containing only equations. There are examples for which the algorithm performs much better than the CAD algorithm; however, for randomly chosen systems of equations it seems to perform significantly worse. Therefore it is not used by default. Here is a decision problem (referred to as butcher8 in the literature), which is not done by CAD in 1000 seconds, but which can be done quite fast by the algorithm given in [17].

In[34]:=
```
SetSystemOptions["InequalitySolvingOptions" → "ARSDecision" → True];
```

$$\text{FindInstance}\Big[-a-b+b_1+b_2+b_3 == 0 \;\&\&\; -\frac{1}{2}-\frac{b}{2}+a\,b-b^2+b_2\,c_2+b_3\,c_3 == 0 \;\&\&\; \frac{4\,b}{3}+b^2+b^3-a\left(\frac{1}{3}+b^2\right)+b_2\,c_2^2+b_3\,c_3^2 == 0 \;\&\&\; \frac{2\,b}{3}+b^2+b^3-a\left(\frac{1}{6}+\frac{b}{2}+b^2\right)+b_3\,c_2\,a_{3,2} == 0 \;\&\&\; -\frac{1}{4}-\frac{b}{4}-\frac{5\,b^2}{2}-\frac{3\,b^3}{2}-b^4+a\left(b+b^3\right)+b_2\,c_2^3+b_3\,c_3^3 == 0 \;\&\&\; -\frac{1}{8}-\frac{3\,b}{8}-\frac{7\,b^2}{4}-\frac{3\,b^3}{2}-b^4+a\left(\frac{b}{2}+\frac{b^2}{2}+b^3\right)+b_3\,c_2\,c_3\,a_{3,2} == 0 \;\&\&\; -\frac{1}{12}-\frac{b}{12}-\frac{7\,b^2}{6}-\frac{3\,b^3}{2}-b^4+a\left(\frac{2\,b}{3}+b^2+b^3\right)+b_3\,c_2^2\,a_{3,2} == 0 \;\&\&\; \frac{1}{24}+\frac{7\,b}{24}+\frac{13\,b^2}{12}+\frac{3\,b^3}{2}+b^4-a\left(\frac{b}{3}+b^2+b^3\right) == 0,\; \{a, b, a_{3,2}, b_1, b_2, b_3, c_2, c_3\}, \text{Reals}\Big] \;//\; \text{Timing}$$

Out[34]=

$$\Big\{0.46, \Big\{\Big\{a \to \text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big],\; b \to -1+\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big],\; a_{3,2} \to \frac{1}{356}\Big(-93+630\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]-684\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]^2\Big),$$

$$b_1 \to \frac{1}{2916}\Big(-3959+3954\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]+2028\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]^2\Big),\; b_2 \to \frac{1}{2916}\Big(-1381+4542\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]-3144\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]^2\Big),$$

$$b_3 \to \frac{1}{243}\Big(202-222\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]+93\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]^2\Big),$$

$$c_2 \to \frac{1}{3}\Big(-4+17\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]-12\,\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]^2\Big),$$

$$c_3 \to 2-\text{Root}\big[-8+25\,\sharp1-30\,\sharp1^2+12\,\sharp1^3\,\&,\,1\big]\Big\}\Big\}\Big\}$$

In[35]:=
```
SetSystemOptions["InequalitySolvingOptions" → "ARSDecision" → False];
```

BrownProjection

By default, the *Mathematica* implementation of the CAD algorithm uses Brown's improved projection operator [8]. The improvement usually speeds up computations substantially. There are some cases where using Brown's projection operator results in a slight slowdown. The option `BrownProjection` specifies whether Brown's improvement should be used. In the first example [21], using Brown's improved projection operator results in a speedup by a factor of 3; in the second, it results in a 40% slowdown.

```
In[36]:= t1 = Timing[Reduce[∃{q1,q2},q1>1&&q2>0 (∀w,w∈Reals ((4 - q1^2) w^4 +
              (4 ((1 + q1)^2 - 2 q2) - (q2^2 + q1^2)) w^2 + 3 q2^2 ≥ 0 &&
            (4 - q1^2) w^4 + (4 ((-1 + q1)^2 - 2 q2) - (q2^2 + q1^2)) w^2 + 3 q2^2 ≥ 0))]][[1]];
         f = x^3 - 5 x y - 3 y^2 + 7;
         g = x^4 - 4 x^2 y - y^3 - 1;
         t2 = Timing[Reduce[f z^2 < f + g, {x, y, z}, Reals]][[1]];
         {t1, t2}
Out[40]= {0.301, 0.24}
```

```
In[41]:= SetSystemOptions["InequalitySolvingOptions" → "BrownProjection" → False];
```

```
In[42]:= t1 = Timing[Reduce[∃{q1,q2},q1>1&&q2>0 (∀w,w∈Reals ((4 - q1^2) w^4 +
              (4 ((1 + q1)^2 - 2 q2) - (q2^2 + q1^2)) w^2 + 3 q2^2 ≥ 0 &&
            (4 - q1^2) w^4 + (4 ((-1 + q1)^2 - 2 q2) - (q2^2 + q1^2)) w^2 + 3 q2^2 ≥ 0))]][[1]];
         t2 = Timing[Reduce[f z^2 < f + g, {x, y, z}, Reals]][[1]];
         {t1, t2}
Out[44]= {0.611, 0.17}
```

```
In[45]:= SetSystemOptions["InequalitySolvingOptions" → "BrownProjection" → True];
```

CAD

The option `CAD` specifies whether *Mathematica* is allowed to use the CAD algorithm. With `CAD` set to `False`, computations that require CAD will fail immediately instead of attempting the high complexity CAD computation. With CAD enabled, this computation is not done in 1000 seconds.

```
In[46]:= SetSystemOptions["InequalitySolvingOptions" → "CAD" → False];
         Reduce[x^12 + 2 x^7 y^5 z^3 - 21 z^4 t^2 y^7 + 19 ≤ 0 && t^7 - 24 x^5 y^4 z - 32 z^11 == 0,
           {x, y, z, t}, Reals] // Timing

     Reduce::nsmet :
       This system cannot be solved with the methods available to Reduce. ≫

Out[47]= {0.641, Reduce[19 + x^12 + 2 x^7 y^5 z^3 - 21 t^2 y^7 z^4 ≤ 0 && t^7 - 24 x^5 y^4 z - 32 z^11 == 0, {x, y, z, t}, Reals]}
```

```
In[48]:= SetSystemOptions["InequalitySolvingOptions" → "CAD" → True];
```

CADDefaultPrecision

By default, *Mathematica* uses validated numeric computations in the lifting phase of the CAD algorithm, reverting to exact algebraic number computations only if the numeric computations cannot be validated [14]. The option `CADDefaultPrecision` specifies the initial precision with which the sample point coordinates are computed. Choosing the value of `CADDefaultPrecision` is a trade-off between speed of numeric computations and the number of points where the algorithm reverts to exact computations due to precision loss. With the default value of 100 bits, the cases where the algorithm needs to revert to exact computations due to precision loss seem quite rare. Setting `CADDefaultPrecision` to `Infinity` causes *Mathematica* to use exact algebraic number computations in the lifting phase of CAD. Here is an example that runs fastest with the lowest `CADDefaultPrecision` setting. (Specifying values lower than 16.2556 (54 bits) results in `CADDefaultPrecision` being set to 16.2556.) With `CADDefaultPrecision -> Infinity`, the example did not finish in 1000 seconds.

```
In[49]:= Reduce[∃{x,y} (931 + 576 y^3 + 626 x^2 y - 564 y z - 750 z^2 < 0 && -535 + 961 y + 578 z ≤ 0 &&
            -410 + 528 y^2 - 905 x ≤ 0 && z^2 - 71 x y + 4 y^2 - 81 ≤ 0), z, Reals] // Timing
```

```
Out[49]= {0.611, Root[
     -39 135 557 564 264 692 223 468 097 + 70 369 504 018 854 614 821 499 160 #1 - 7 499 740 633 203 604 239 774 740
       #1^2 - 91 567 784 348 961 473 370 737 040 #1^3 + 10 948 550 214 483 020 279 449 920 #1^4 -
       1 313 704 439 523 340 062 769 800 #1^5 - 47 035 179 704 857 006 865 939 040 #1^6 -
       18 217 590 707 582 813 495 520 #1^7 + 23 290 773 235 831 759 680 #1^8 + 9 309 551 043 209 472 #1^10 &, 1] < z ≤
   Root[-55 420 506 053 355 + 915 537 370 820 #1 - 18 135 837 359 975 #1^2 + 7 238 953 493 376 #1^3 &, 1]}
```

```
In[50]:= SetSystemOptions["InequalitySolvingOptions" → "CADDefaultPrecision" → 16];
         Reduce[∃{x,y} (931 + 576 y^3 + 626 x^2 y - 564 y z - 750 z^2 < 0 && -535 + 961 y + 578 z ≤ 0 &&
            -410 + 528 y^2 - 905 x ≤ 0 && z^2 - 71 x y + 4 y^2 - 81 ≤ 0), z, Reals] // Timing
```

```
Out[51]= {0.551, Root[
     -39 135 557 564 264 692 223 468 097 + 70 369 504 018 854 614 821 499 160 #1 - 7 499 740 633 203 604 239 774 740
       #1^2 - 91 567 784 348 961 473 370 737 040 #1^3 + 10 948 550 214 483 020 279 449 920 #1^4 -
       1 313 704 439 523 340 062 769 800 #1^5 - 47 035 179 704 857 006 865 939 040 #1^6 -
       18 217 590 707 582 813 495 520 #1^7 + 23 290 773 235 831 759 680 #1^8 + 9 309 551 043 209 472 #1^10 &, 1] < z ≤
   Root[-55 420 506 053 355 + 915 537 370 820 #1 - 18 135 837 359 975 #1^2 + 7 238 953 493 376 #1^3 &, 1]}
```

```
In[52]:= SetSystemOptions["InequalitySolvingOptions" → "CADDefaultPrecision" → 30.103];
```

CADSortVariables

The performance of the CAD algorithm often depends quite strongly on the order of variables used. Some aspects of the variable ordering are fixed by the problem we are solving: quantifier variables need to be projected before free variables, and variables from innermost quantifiers need to be projected first. Variables specified in `Reduce` and `Resolve` cannot be reordered unless `ReorderVariables` is set to `True`. This, however, still leaves some freedom in ordering of variables: variables from the same quantifier can be reordered, and so can be variables given to `FindInstance`. By default, *Mathematica* uses a variable ordering heuristic to determine the order of these variables. In most cases the heuristic improves the performance of CAD; in some examples, however, the heuristic does not pick the best ordering. Setting `CADSortVariables` to `False` disables the heuristic and the order of variables used is as given in the quantifier variable list or in the variable list argument to `FindInstance`. Here is an example [21] that without reordering of quantified variables does not finish in 1000 seconds.

In[53]:= **Timing[Reduce[**

$$\forall_{\{p_1,p_2,w_1,w_2\},16\le 20\,p_1\le 25\&\&16\le 20\,p_2\le 25\&\&0\le w_1\le 2}\left(p_2\,(1+p_1\,q_1)<0\;\&\&\;-24\,w_1^2+p_2^2\left((1+p_1\,q_1)^2-25\right)>0\;\&\&\;\left(400-q_1^2\right)w_2^2+p_2^2\left(400\,(1+p_1\,q_1)^2-q_1^2\right)>0\right),\;q_1,\;\text{Reals}]]$$

Out[53]= $\left\{0.521,\;-20\le q_1<\frac{5}{4}\left(-1-5\sqrt{7}\right)\right\}$

This shows the optimal variable ordering for the example.

In[54]:= **SetSystemOptions["InequalitySolvingOptions" → "CADSortVariables" → False];**

In[55]:= **Timing[Reduce[**

$$\forall_{\{w_1,w_2,p_2,p_1\},16\le 20\,p_1\le 25\&\&16\le 20\,p_2\le 25\&\&0\le w_1\le 2}\left(p_2\,(1+p_1\,q_1)<0\;\&\&\;-24\,w_1^2+p_2^2\left((1+p_1\,q_1)^2-25\right)>0\;\&\&\;\left(400-q_1^2\right)w_2^2+p_2^2\left(400\,(1+p_1\,q_1)^2-q_1^2\right)>0\right),\;q_1,\;\text{Reals}]]$$

Out[55]= $\left\{0.47,\;-20\le q_1<\frac{5}{4}\left(-1-5\sqrt{7}\right)\right\}$

In[56]:= **SetSystemOptions["InequalitySolvingOptions" → "CADSortVariables" → True];**

CADZeroTest

One of the most time-consuming operations in the lifting phase of the CAD algorithm is determining the sign of a polynomial evaluated at a sample point with algebraic number coordinates. We try to avoid the problem by using sample points with arbitrary-precision floating-point number coordinates and keeping track of the "genealogy" of projection polynomials and sample points in order to validate the results. However, if some of the results cannot be validated, we have to revert to computations with exact algebraic number coordinates. To determine the sign of a polynomial evaluated at a sample point with algebraic number coordinates, we first evaluate the polynomial at numeric approximations of the algebraic numbers. If the result is nonzero (that is, zero is not within the error bounds of the resulting bignum), we know the sign. Otherwise, we need to test whether a polynomial expression in algebraic numbers is zero. The value of the `CADZeroTest` option specifies what zero testing method should be used at this moment. The value should be a pair $\{t, acc\}$. With the default value $t == 0$, *Mathematica* computes an accuracy $eacc$ such that if the expression is zero up to this accuracy, it must be zero. If $eacc \leq acc$, the value of the expression is computed up to accuracy $eacc$ and its sign is checked. Otherwise, the expression is represented as a single `Root` object using `RootReduce` and the sign of the `Root` object is found. With the default value $acc == \infty$, we revert to `RootReduce` if $eacc >$ `$MaxPrecision`. If $t == 1$, `RootReduce` is always used. If $t == 2$, expressions that are zero up to accuracy acc are considered zero. This is the fastest method, but, unlike the other two, it may give incorrect results because expressions that are nonzero but close to zero may be treated as zero.

This example runs faster with the CAD algorithm using the 30 digits of accuracy numeric zero test. The result in this example is correct; however, this setting of `CADZeroTest` may lead to incorrect results.

```
In[57]:= t1 = Timing[Reduce[∃z (z^3 - a^2 z + b == 0 && z^3 - b^2 z + a == 0), {a, b}, Reals]][[1]];
         SetSystemOptions["InequalitySolvingOptions" → "CADZeroTest" → {2, 30}];
         t2 = Timing[Reduce[∃z (z^3 - a^2 z + b == 0 && z^3 - b^2 z + a == 0), {a, b}, Reals]][[1]];
         {t1, t2}
Out[60]= {0.271, 0.23}
```

```
In[61]:= SetSystemOptions["InequalitySolvingOptions" → "CADZeroTest" → {0, Infinity}];
```

ContinuedFractionRootIsolation

To isolate real roots of polynomials, *Mathematica* uses methods based on Descartes' rule of sign. There are two interval subdivision strategies implemented, one based on interval bisection and another based on continued fractions (see [19] for details). The variant based on continued fractions is generally faster and is used by default. Setting `ContinuedFractionRootIsolation` to `False` causes *Mathematica* to use the interval bisection variant.

Here is an example where the speed difference between the two root isolation methods affects `Reduce` timing. We need to clear the `Root` cache between the `Reduce` calls; otherwise, the second call would save time on factoring the 400th degree polynomial when `Root` objects are created.

```
In[62]:= SeedRandom[1234];
         f = Sum[RandomInteger[{-1000, 1000}] x^i, {i, 0, 399}] + x^400;
         t1 = Timing[Reduce[f ≤ 0, x, Reals]];
         ClearSystemCache["Root"];
         SetSystemOptions[
           "InequalitySolvingOptions" → "ContinuedFractionRootIsolation" → False];
         t2 = Timing[Reduce[f ≤ 0, x, Reals]];
         {t1⟦1⟧, t2⟦1⟧, t1⟦2⟧ === t2⟦2⟧}
Out[68]= {3.705, 4.607, True}
```

```
In[69]:= SetSystemOptions[
           "InequalitySolvingOptions" → "ContinuedFractionRootIsolation" → True];
```

EquationalConstraintsCAD

The `EquationalConstraintsCAD` option specifies whether the projection phase of the CAD algorithm should use equational constraints. With the default setting `Automatic`, *Mathematica* uses the projection operator proven correct in [11]. With `EquationalConstraintsCAD -> True`, the smaller but unproven projection operator suggested in [4] is used.

Here we find an instance satisfying the system using the CAD algorithm with `EquationalConstraintsCAD -> True`. Even though the method used to find the solution was based on an unproven conjecture, the solution is proven to be correct, that is, it satisfies the input system.

```
In[70]:= SetSystemOptions["InequalitySolvingOptions" → "EquationalConstraintsCAD" → True];
         FindInstance[-1 + a ≤ 0 && -1 - a < 0 && -3 - a + k^2 + a k^2 ≤ 0 && v1^2 == 2 &&
             1 + a - v2^2 == 0 && k + a k - v2 v3 ≤ 0 && -k - a k - v2 v3 ≤ 0 && 3 + a - v3^2 == 0 &&
             18 + 6 a + 6 a^2 + 2 a^3 - 21 k - 27 a k - 7 a^2 k - a^3 k + 6 k^2 + 10 a k^2 + 2 a^2 k^2 -
               2 a^3 k^2 + k^3 + 3 a k^3 + 3 a^2 k^3 + a^3 k^3 - 3 v1 v4 + 6 a v1 v4 + a^2 v1 v4 - 4 a k v1 v4 -
               4 a^2 k v1 v4 + k^2 v1 v4 + 2 a k^2 v1 v4 + a^2 k^2 v1 v4 == 0 && -3 - a + k^2 + a k^2 + v4^2 == 0 &&
             v1 > 0 && v2 ≥ 0 && v3 ≥ 0 && v4 ≥ 0, {v1, k, a, v2, v3, v4}, Reals] // Timing
```

Out[71]= $\left\{0.18, \left\{\left\{v_1 \to \sqrt{2}, k \to 1, a \to -\frac{7}{16}, v_2 \to \frac{3}{4}, v_3 \to \frac{\sqrt{41}}{4}, v_4 \to \sqrt{2}\right\}\right\}\right\}$

With the default setting `EquationalConstraintsCAD -> Automatic`, finding a solution of this system takes more than twice as long.

```
In[72]:= SetSystemOptions[
           "InequalitySolvingOptions" → "EquationalConstraintsCAD" → Automatic];
         FindInstance[-1 + a ≤ 0 && -1 - a < 0 && -3 - a + k^2 + a k^2 ≤ 0 && v1^2 == 2 &&
            1 + a - v2^2 == 0 && k + a k - v2 v3 ≤ 0 && -k - a k - v2 v3 ≤ 0 && 3 + a - v3^2 == 0 &&
            18 + 6 a + 6 a^2 + 2 a^3 - 21 k - 27 a k - 7 a^2 k - a^3 k + 6 k^2 + 10 a k^2 + 2 a^2 k^2 -
              2 a^3 k^2 + k^3 + 3 a k^3 + 3 a^2 k^3 + a^3 k^3 - 3 v1 v4 + 6 a v1 v4 + a^2 v1 v4 - 4 a k v1 v4 -
              4 a^2 k v1 v4 + k^2 v1 v4 + 2 a k^2 v1 v4 + a^2 k^2 v1 v4 == 0 && -3 - a + k^2 + a k^2 + v4^2 == 0 &&
            v1 > 0 && v2 ≥ 0 && v3 ≥ 0 && v4 ≥ 0, {v1, k, a, v2, v3, v4}, Reals] // Timing
```

Out[73]= $\left\{0.491, \left\{\left\{v_1 \to \sqrt{2}, k \to 1, a \to -\frac{7}{16}, v_2 \to \frac{3}{4}, v_3 \to \frac{\sqrt{41}}{4}, v_4 \to \sqrt{2}\right\}\right\}\right\}$

With `EquationalConstraintsCAD -> False`, finding a solution of this system again takes almost twice as long.

```
In[74]:= SetSystemOptions[
           "InequalitySolvingOptions" → "EquationalConstraintsCAD" → False];
         FindInstance[-1 + a ≤ 0 && -1 - a < 0 && -3 - a + k^2 + a k^2 ≤ 0 && v1^2 == 2 &&
            1 + a - v2^2 == 0 && k + a k - v2 v3 ≤ 0 && -k - a k - v2 v3 ≤ 0 && 3 + a - v3^2 == 0 &&
            18 + 6 a + 6 a^2 + 2 a^3 - 21 k - 27 a k - 7 a^2 k - a^3 k + 6 k^2 + 10 a k^2 + 2 a^2 k^2 -
              2 a^3 k^2 + k^3 + 3 a k^3 + 3 a^2 k^3 + a^3 k^3 - 3 v1 v4 + 6 a v1 v4 + a^2 v1 v4 - 4 a k v1 v4 -
              4 a^2 k v1 v4 + k^2 v1 v4 + 2 a k^2 v1 v4 + a^2 k^2 v1 v4 == 0 && -3 - a + k^2 + a k^2 + v4^2 == 0 &&
            v1 > 0 && v2 ≥ 0 && v3 ≥ 0 && v4 ≥ 0, {v1, k, a, v2, v3, v4}, Reals] // Timing
```

Out[75]= $\left\{0.921, \left\{\left\{v_1 \to \sqrt{2}, k \to 1, a \to -\frac{7}{16}, v_2 \to \frac{3}{4}, v_3 \to \frac{\sqrt{41}}{4}, v_4 \to \sqrt{2}\right\}\right\}\right\}$

Here `FindInstance` shows that the system has no solutions. Since it is using the CAD algorithm with `EquationalConstraintsCAD -> True`, the correctness of the answer depends on an unproven conjecture.

```
In[76]:= SetSystemOptions["InequalitySolvingOptions" → "EquationalConstraintsCAD" → True];
         FindInstance[k ≠ 1 && -1 + a ≤ 0 && -1 - a < 0 && -3 - a + k^2 + a k^2 ≤ 0 && v1^2 == 2 &&
            1 + a - v2^2 == 0 && k + a k - v2 v3 ≤ 0 && -k - a k - v2 v3 ≤ 0 && 3 + a - v3^2 == 0 &&
            18 + 6 a + 6 a^2 + 2 a^3 - 21 k - 27 a k - 7 a^2 k - a^3 k + 6 k^2 + 10 a k^2 + 2 a^2 k^2 -
              2 a^3 k^2 + k^3 + 3 a k^3 + 3 a^2 k^3 + a^3 k^3 - 3 v1 v4 + 6 a v1 v4 + a^2 v1 v4 - 4 a k v1 v4 -
              4 a^2 k v1 v4 + k^2 v1 v4 + 2 a k^2 v1 v4 + a^2 k^2 v1 v4 == 0 && -3 - a + k^2 + a k^2 + v4^2 == 0 &&
            v1 > 0 && v2 ≥ 0 && v3 ≥ 0 && v4 ≥ 0, {v1, k, a, v2, v3, v4}, Reals] // Timing
```

Out[77]= {0.301, {}}

With the default setting `EquationalConstraintsCAD -> Automatic`, proving that the system has no solutions takes longer, but the answer is known to be correct.

```
In[78]:= SetSystemOptions[
           "InequalitySolvingOptions" → "EquationalConstraintsCAD" → Automatic];
         FindInstance[k ≠ 1 && -1 + a ≤ 0 && -1 - a < 0 && -3 - a + k^2 + a k^2 ≤ 0 && v1^2 == 2 &&
             1 + a - v2^2 == 0 && k + a k - v2 v3 ≤ 0 && -k - a k - v2 v3 ≤ 0 && 3 + a - v3^2 == 0 &&
             18 + 6 a + 6 a^2 + 2 a^3 - 21 k - 27 a k - 7 a^2 k - a^3 k + 6 k^2 + 10 a k^2 + 2 a^2 k^2 -
               2 a^3 k^2 + k^3 + 3 a k^3 + 3 a^2 k^3 + a^3 k^3 - 3 v1 v4 + 6 a v1 v4 + a^2 v1 v4 - 4 a k v1 v4 -
               4 a^2 k v1 v4 + k^2 v1 v4 + 2 a k^2 v1 v4 + a^2 k^2 v1 v4 == 0 && -3 - a + k^2 + a k^2 + v4^2 == 0 &&
             v1 > 0 && v2 ≥ 0 && v3 ≥ 0 && v4 ≥ 0, {v1, k, a, v2, v3, v4}, Reals] // Timing
Out[79]= {0.911, {}}
```

FGLMBasisConversion

For systems with equational constraints generating a zero-dimensional ideal I, *Mathematica* uses a variant of the CAD algorithm that finds projection polynomials using Gröbner basis methods. If the lexicographic order Gröbner basis of I does not contain linear polynomials with constant coefficients in every variable but the last one, then for every variable x_i we find a univariate polynomial in x_i that belongs to I. *Mathematica* can do this in two ways. By default, it uses a method based on `GroebnerWalk` computations. Setting `FGLMBasisConversion` to `True` causes *Mathematica* to use a method based on [20].

The method based on [20] seems to be slightly slower in general.

```
In[80]:= t1 = Timing[Reduce[x^10 + 3 x^4 - 5 x^3 + 7 x^2 - 9 x == 11 &&
               y^3 - y^2 + x == 1 && z^3 + 2 z - 3 x == 4, {x, y, z}, Reals]];
         SetSystemOptions["InequalitySolvingOptions" → "FGLMBasisConversion" → True];
         t2 = Timing[Reduce[x^10 + 3 x^4 - 5 x^3 + 7 x^2 - 9 x == 11 &&
               y^3 - y^2 + x == 1 && z^3 + 2 z - 3 x == 4, {x, y, z}, Reals]];
         {t1[[1]], t2[[1]], t1[[2]] === t2[[2]]}
Out[83]= {0.15, 0.181, True}
```

```
In[84]:= SetSystemOptions["InequalitySolvingOptions" → "FGLMBasisConversion" → False];
```

FGLMElimination

The `FGLMElimination` option specifies whether *Mathematica* should use a special case heuristic applicable to systems with equational constraints generating a zero-dimensional ideal I. The heuristic uses a method based on [20] to find in I polynomials that are linear (with a constant coefficient) in one of the quantified variables and uses such polynomials for elimination. The method can be used both in the decision algorithm and in quantifier elimination. With the default `Automatic` setting, it is used only in `Resolve` with no "solve" variables specified and for systems with at least two free variables.

This by default uses the elimination method based on [20], and returns a quantifier-free system in an unsolved form.

In[85]:=
```
Resolve[
  ∃z (x^2 + 2 y^3 - 3 x y + 4 x z + 2 z^3 == 1 && y^3 - 2 x^2 z + 5 x - 7 z^3 == 2 && 3 x y + 4 z^3 - 5 y^3 == 0),
  Reals] // Timing
```

Out[85]=
```
{0.05,
 -387 703 943 456 010 + 836 307 322 497 954 x + 94 016 672 514 000 x^2 + 42 483 692 361 858 x^3 + 48 951 449 972 226
     x^4 + 592 457 191 920 x^5 + 6 111 106 822 080 x^6 - 682 099 934 085 412 y + 2 386 910 531 381 715 x y -
    33 458 557 021 065 x^2 y + 179 029 980 402 448 x^3 y + 23 352 969 127 806 x^4 y - 10 673 134 807 104 x^5 y +
    1 298 614 472 640 x^6 y - 6 165 373 996 350 y^2 + 1 787 681 183 046 234 x y^2 - 463 516 345 125 783 x^2 y^2 +
    1 221 461 511 750 x^3 y^2 + 7 275 931 779 870 x^4 y^2 - 3 220 021 226 880 x^5 y^2 + 453 968 712 000 x^6 y^2 -
    1 333 886 745 639 423 y^3 - 2 629 577 891 362 724 y^4 - 20 449 871 823 375 y^5 + 473 314 204 852 983 y^6 == 0 &&
  -567 795 134 + 1 059 962 112 x + 430 480 332 x^2 + 309 282 350 x^3 - 11 545 182 x^4 + 23 721 822 x^5 -
    7 231 680 x^6 + 2 099 520 x^7 + 320 591 520 y + 927 840 621 x y - 389 548 395 x^2 y - 209 188 980 x^3 y -
    29 695 086 x^4 y + 15 536 448 x^5 y - 3 779 136 x^6 y - 40 678 200 y^2 - 761 836 590 x y^2 +
    158 630 400 x^2 y^2 + 30 508 650 x^3 y^2 - 2 255 020 201 y^3 + 1 242 292 140 y^4 - 157 628 025 y^5 == 0 &&
  -394 500 + 962 118 x - 153 630 x^2 + 43 806 x^3 + 17 982 x^4 + 5760 x^6 - 578 624 y + 2 180 295 x y -
    352 890 x^2 y + 291 671 x^3 y + 12 492 x^4 y - 10 368 x^5 y + 168 480 y^2 + 1 168 968 x y^2 - 445 266 x^2 y^2 -
    50 220 x^3 y^2 - 1 370 121 y^3 - 2 271 328 y^4 + 652 860 y^5 + 445 266 y^6 - 112 995 y^7 == 0 &&
  -8 + 18 x + 2 x^3 + 21 x y - 9 x^2 y - 31 y^3 + 9 x y^3 == 0}
```

With `FGLMElimination` set to `False`, the example takes longer to compute and the answer is in a solved form. (We show `N` of the answer for better readability.)

In[86]:=
```
SetSystemOptions["InequalitySolvingOptions" → "FGLMElimination" → False];
Resolve[
   ∃z (x^2 + 2 y^3 - 3 x y + 4 x z + 2 z^3 == 1 && y^3 - 2 x^2 z + 5 x - 7 z^3 == 2 && 3 x y + 4 z^3 - 5 y^3 == 0),
   Reals] // Timing // N
```

Out[87]=
```
{0.11,
 (y == -0.616811 && x == -5.18103 - 137.347 y - 1010.78 y^2 - 2069.96 y^3 + 92.7062 y^4 + 7185.17 y^5 + 10 827. y^6 -
      17 208. y^7 - 25 441. y^8 + 59 919.3 y^9 + 5428.35 y^10 - 87 974.4 y^11 + 90 884.3 y^12 +
      9563.19 y^13 - 65 852.1 y^14 + 61 525.6 y^15 - 51 406.5 y^16 + 51 634.3 y^17 -
      27 621.2 y^18 + 1364.5 y^19 + 5842.54 y^20 - 1836.15 y^21 - 216.104 y^22 + 162.853 y^23) ||
  (y == -0.510025 && x == -5.18103 - 137.347 y - 1010.78 y^2 - 2069.96 y^3 + 92.7062 y^4 +
      7185.17 y^5 + 10 827. y^6 - 17 208. y^7 - 25 441. y^8 + 59 919.3 y^9 + 5428.35 y^10 - 87 974.4 y^11 +
      90 884.3 y^12 + 9563.19 y^13 - 65 852.1 y^14 + 61 525.6 y^15 - 51 406.5 y^16 + 51 634.3 y^17 -
      27 621.2 y^18 + 1364.5 y^19 + 5842.54 y^20 - 1836.15 y^21 - 216.104 y^22 + 162.853 y^23) ||
  (y == -0.0897985 && x == -5.18103 - 137.347 y - 1010.78 y^2 - 2069.96 y^3 + 92.7062 y^4 +
      7185.17 y^5 + 10 827. y^6 - 17 208. y^7 - 25 441. y^8 + 59 919.3 y^9 + 5428.35 y^10 - 87 974.4 y^11 +
      90 884.3 y^12 + 9563.19 y^13 - 65 852.1 y^14 + 61 525.6 y^15 - 51 406.5 y^16 + 51 634.3 y^17 -
      27 621.2 y^18 + 1364.5 y^19 + 5842.54 y^20 - 1836.15 y^21 - 216.104 y^22 + 162.853 y^23) ||
  (y == 0.664342 && x == -5.18103 - 137.347 y - 1010.78 y^2 - 2069.96 y^3 + 92.7062 y^4 + 7185.17 y^5 +
      10 827. y^6 - 17 208. y^7 - 25 441. y^8 + 59 919.3 y^9 + 5428.35 y^10 - 87 974.4 y^11 +
      90 884.3 y^12 + 9563.19 y^13 - 65 852.1 y^14 + 61 525.6 y^15 - 51 406.5 y^16 + 51 634.3 y^17 -
      27 621.2 y^18 + 1364.5 y^19 + 5842.54 y^20 - 1836.15 y^21 - 216.104 y^22 + 162.853 y^23)}
```

If there is only one free variable, `Resolve` by default does not use the elimination method based on [20]. (We show `N` of the answer for better readability.)

```
In[88]:= SetSystemOptions["InequalitySolvingOptions" → "FGLMElimination" → Automatic];
Resolve[∃{y,z} (x^2 + 2 y^3 - 3 x y + 4 x z + 2 z^3 == 1 &&
    y^3 - 2 x^2 z + 5 x - 7 z^3 == 2 && 3 x y + 4 z^3 - 5 y^3 == 0), Reals] // Timing // N

Out[89]= {0.13, x == -1.05088 || x == 0.452835 || x == 0.47114 || x == 0.534627}
```

With `FGLMElimination` set to `True`, the example takes longer to compute and the answer is given in an unsolved form.

```
In[90]:= SetSystemOptions["InequalitySolvingOptions" → "FGLMElimination" → True];
Resolve[∃{y,z} (x^2 + 2 y^3 - 3 x y + 4 x z + 2 z^3 == 1 &&
    y^3 - 2 x^2 z + 5 x - 7 z^3 == 2 && 3 x y + 4 z^3 - 5 y^3 == 0), Reals] // Timing

Out[91]= {0.2,
 -27 206 534 396 294 947 + 328 914 818 820 879 210 x - 1 654 010 622 073 883 961 x^2 + 4 186 250 649 401 504 955 x^3 -
   4 131 264 062 314 837 638 x^4 - 5 359 613 482 785 909 285 x^5 + 20 455 887 169 340 134 671 x^6 -
   18 111 422 036 067 816 735 x^7 - 14 851 799 572 578 604 767 x^8 + 46 025 930 760 201 888 392 x^9 -
   33 951 750 015 320 895 222 x^10 - 3 130 213 891 174 116 318 x^11 + 18 846 711 211 560 897 036 x^12 -
   13 729 694 750 794 525 104 x^13 + 8 758 251 556 584 250 005 x^14 - 4 917 731 156 959 045 278 x^15 +
   2 285 701 226 953 461 792 x^16 - 895 869 248 032 870 029 x^17 + 304 502 137 753 065 983 x^18 -
   88 547 080 320 192 096 x^19 + 21 286 381 859 013 600 x^20 - 4 017 686 252 055 552 x^21 +
   554 267 616 334 848 x^22 - 49 218 499 805 184 x^23 + 2 176 782 336 000 x^24 == 0}

In[92]:= SetSystemOptions["InequalitySolvingOptions" → "FGLMElimination" → Automatic];
```

GenericCAD

Mathematica uses a simplified version of the CAD algorithm described in [13] to solve decision problems or find solutions of real polynomial systems that do not contain equations. The method finds a solution or proves that there are no solutions if all inequalities in the system are strict (< or >). The method is also used for systems containing weak (<= or >=) inequalities. In this case, if it finds a solution of the strict inequality version of the system, it is also a solution of the original system. However, if it proves that the strict inequality version of the system has no solutions, the full version of the CAD algorithm is needed to decide whether the original system has solutions. The system option `GenericCAD` specifies whether *Mathematica* should use the method.

Here the `GenericCAD` method finds a solution of the strict inequality version of the system.

```
In[93]:= FindInstance[
  x^4 + y^4 + z^4 ≤ 12 && x^2 y^2 - 3 x^2 z^2 ≥ 1 && x y ≤ 3 z^3 + 4, {x, y, z}, Reals] // Timing

Out[93]= {0.191, {{x → 145/128, y → -113/64, z → -113/128}}}
```

Without `GenericCAD`, finding a solution of the system takes much longer.

```
In[94]:= SetSystemOptions["InequalitySolvingOptions" → "GenericCAD" → False];
FindInstance[
  x^4 + y^4 + z^4 ≤ 12 && x^2 y^2 - 3 x^2 z^2 ≥ 1 && x y ≤ 3 z^3 + 4, {x, y, z}, Reals] // Timing
```

Out[95]= $\left\{0.961, \left\{\left\{x \to \frac{309}{256}, y \to -\frac{223}{128}, z \to -\frac{1809}{2048}\right\}\right\}\right\}$

```
In[96]:= SetSystemOptions["InequalitySolvingOptions" → "GenericCAD" → True];
```

This system has no solutions and contains weak inequalities. After the `GenericCAD` method finds no solutions of the strict inequality version of the system, *Mathematica* needs to run the full CAD to prove that there are no solutions.

```
In[97]:= FindInstance[x^4 + y^4 + z^4 ≤ 12 && x^3 + y^3 - z^3 ≥ 9, {x, y, z}, Reals] // Timing
Out[97]= {1.122, {}}
```

Running the same example with `GenericCAD -> False` allows you to save the time previously used by the `GenericCAD` computation.

```
In[98]:= SetSystemOptions["InequalitySolvingOptions" → "GenericCAD" → False];
FindInstance[x^4 + y^4 + z^4 ≤ 12 && x^3 + y^3 - z^3 ≥ 9, {x, y, z}, Reals] // Timing
Out[99]= {0.611, {}}
```

```
In[100]:= SetSystemOptions["InequalitySolvingOptions" → "GenericCAD" → True];
```

This system contains only strict inequalities, so `GenericCAD` can prove that it has no solutions.

```
In[101]:= FindInstance[
  x^4 + y^4 + z^4 < 12 && x^2 y^2 - 3 x^2 z^2 > 7 && x y < 3 z^3 + 4, {x, y, z}, Reals] // Timing
Out[101]= {0.18, {}}
```

Without `GenericCAD`, it takes much longer to prove that the system has no solutions.

```
In[102]:= SetSystemOptions["InequalitySolvingOptions" → "GenericCAD" → False];
FindInstance[
  x^4 + y^4 + z^4 < 12 && x^2 y^2 - 3 x^2 z^2 > 7 && x y < 3 z^3 + 4, {x, y, z}, Reals] // Timing
Out[103]= {2.393, {}}
```

```
In[104]:= SetSystemOptions["InequalitySolvingOptions" → "GenericCAD" → True];
```

GroebnerCAD

For systems with equational constraints generating a zero-dimensional ideal I, *Mathematica* uses a variant of the CAD algorithm that finds projection polynomials using Gröbner basis methods. Setting `GroebnerCAD` to `False` causes *Mathematica* to use the standard CAD projection instead.

With `GroebnerCAD -> False`, this example runs three orders of magnitude slower.

```
In[105]:= a1 = Reduce[x^2 + y^2 + z^2 == 12 && x^2 y^2 - 3 x^2 z^2 == 1 && x y == 3 z^3 + 4, {x, y, z}, Reals]; //
            Timing
Out[105]= {0.03, Null}
```

```
In[106]:= SetSystemOptions["InequalitySolvingOptions" → "GroebnerCAD" → False];
```

```
In[107]:= a2 = Reduce[x^2 + y^2 + z^2 == 12 && x^2 y^2 - 3 x^2 z^2 == 1 && x y == 3 z^3 + 4, {x, y, z}, Reals]; //
            Timing
Out[107]= {2.043, Null}
```

This checks that the solutions are equivalent.

```
In[108]:= Chop[({x, y, z} //. N[{ToRules[a1]}, 30]) - ({x, y, z} //. N[{ToRules[a2]}, 30])]
Out[108]= {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}, {0, 0, 0}}
```

```
In[109]:= SetSystemOptions["InequalitySolvingOptions" → "GroebnerCAD" → True];
```

LinearDecisionMethodCrossovers, LWDecision, and Simplex

These three options specify methods used to solve decision problems or find solution instances for systems of linear equations and inequalities. The available methods are the Loos-Weispfenning algorithm [15], the Simplex algorithm, and the Revised Simplex algorithm. All three methods can handle systems with rational or floating-point number coefficients. For systems with exact numeric nonrational coefficients, only the Loos-Weispfenning algorithm is implemented. `LWDecision` specifies whether the Loos-Weispfenning algorithm is available. `Simplex` specifies whether the Simplex and Revised Simplex algorithms can be used. `LinearDecisionMethodCrossovers` determines which method is used if all are available and applicable. The value of the option should be a triple $\{m, n, p\}$. For linear systems with up to m variables, *Mathematica* uses the Loos-Weispfenning method [15]; for systems with $m+1$ to n variables, the Simplex algorithm; and for more than n variables, the Revised Simplex algorithm. If the Simplex algorithm is used, the slack variables are used if the number of inequalities is no more than p times the number of variables. The default values are $m=0$, $n=30$, and $p=20$.

By default, the Simplex algorithm is used to find a solution of a linear system with three variables.

```
In[110]:= FindInstance[
             x + 2 y + 3 z == 4 && 5 x + 6 y - 7 z ≤ 8 && 9 x - 10 y + 11 z > 12, {x, y, z}, Reals] // Timing
```

Out[110]= $\left\{0.,\left\{\left\{x\to\frac{199}{138},\ y\to\frac{149}{276},\ z\to\frac{34}{69}\right\}\right\}\right\}$

Here the Revised Simplex algorithm is used.

```
In[111]:= SetSystemOptions[
            "InequalitySolvingOptions" → "LinearDecisionMethodCrossovers" → {0, 0, 20}];
          FindInstance[x + 2 y + 3 z == 4 && 5 x + 6 y - 7 z ≤ 8 && 9 x - 10 y + 11 z > 12,
            {x, y, z}, Reals] // Timing
```

Out[112]= $\left\{0.081, \left\{\left\{x \to 0,\ y \to \frac{5}{52},\ z \to \frac{33}{26}\right\}\right\}\right\}$

Here the Loos-Weispfenning algorithm is used.

```
In[113]:= SetSystemOptions[
            "InequalitySolvingOptions" → "LinearDecisionMethodCrossovers" → {10, 0, 20}];
          FindInstance[x + 2 y + 3 z == 4 && 5 x + 6 y - 7 z ≤ 8 && 9 x - 10 y + 11 z > 12,
            {x, y, z}, Reals] // Timing
```

Out[114]= $\left\{3.17801\times 10^{-15}, \left\{\left\{x \to \frac{34}{23},\ y \to \frac{5}{46},\ z \to \frac{53}{69}\right\}\right\}\right\}$

```
In[115]:= SetSystemOptions[
            "InequalitySolvingOptions" → "LinearDecisionMethodCrossovers" → {0, 30, 20}];
```

Here the Loos-Weispfenning algorithm is used because the Simplex and Revised Simplex algorithms are not implemented for systems with exact nonrational coefficients.

In[116]:= $\textbf{FindInstance}\left[x + \pi\, y + e\, z > \text{Sin}[1]\ \&\&\ \text{Log}[2]\, x + \pi^{e}\, y - 7^{\pi}\, z == \frac{8}{e},\ \{x, y, z\},\ \text{Reals}\right]\ //\ \text{Timing}$

Out[116]= $\left\{0.01, \left\{\left\{x \to 0,\ y \to 2,\ z \to -\frac{8\ 7^{-\pi}}{e} + 2\ 7^{-\pi}\,\pi^{e}\right\}\right\}\right\}$

With `LWDecision` set to `False`, and Simplex and Revised Simplex not applicable, `FindInstance` has to use the CAD algorithm here.

```
In[117]:= SetSystemOptions["InequalitySolvingOptions" → "LWDecision" → False];
```

$\textbf{FindInstance}\left[x + \pi\, y + e\, z > \text{Sin}[1]\ \&\&\ \text{Log}[2]\, x + \pi^{e}\, y - 7^{\pi}\, z == \frac{8}{e},\ \{x, y, z\},\ \text{Reals}\right]\ //\ \text{Timing}$

Out[118]= $\left\{0.03, \left\{\left\{x \to \frac{33}{10},\ y \to 66,\ z \to \frac{7^{-\pi}\,(-80 + 660\, e\, \pi^{e} + 33\, e\, \text{Log}[2])}{10\, e}\right\}\right\}\right\}$

```
In[119]:= SetSystemOptions["InequalitySolvingOptions" → "LWDecision" → True];
```

LinearEquations

The `LinearEquations` option specifies whether linear equation constraints with constant leading coefficients should be used to eliminate variables. This generally improves the performance of the algorithm. The option is provided to allow experimentation with the “pure” CAD-based decision algorithm.

Here *Mathematica* uses the first equation to eliminate x before using CAD to find a solution of the resulting system with two variables.

```
In[120]:= FindInstance[x + 2 y^2 + z^3 == 7 && 2 x^2 y^3 - 3 x z^3 + 5 x y z - x^3 + x + y - z ≥ 3 &&
             4 x y^3 + 5 y^4 z^2 + 11 y z^2 ≤ 3 z^3 + 7 && x^2 + y^2 + z^2 ≤ 4, {x, y, z}, Reals] // Timing
```

Out[120]= $\left\{0.15, \left\{\left\{x \to -\frac{17411}{55296}, y \to \frac{123}{64}, z \to -\frac{5}{12}\right\}\right\}\right\}$

Here *Mathematica* uses CAD to find a solution of the original system with three variables.

```
In[121]:= SetSystemOptions["InequalitySolvingOptions" → "LinearEquations" → False];
          FindInstance[x + 2 y^2 + z^3 == 7 && 2 x^2 y^3 - 3 x z^3 + 5 x y z - x^3 + x + y - z ≥ 3 &&
             4 x y^3 + 5 y^4 z^2 + 11 y z^2 ≤ 3 z^3 + 7 && x^2 + y^2 + z^2 ≤ 4, {x, y, z}, Reals] // Timing
```

Out[122]= $\left\{0.31, \left\{\left\{x \to -\frac{78015}{262144}, y \to \frac{491}{256}, z \to -\frac{25}{64}\right\}\right\}\right\}$

```
In[123]:= SetSystemOptions["InequalitySolvingOptions" → "LinearEquations" → True];
```

LinearQE

The `LinearQE` option specifies methods used to handle systems containing at least one innermost quantifier variable that appears at most linearly in all equations and inequalities in the system. The option setting does not affect solving of decision problems. With the default setting `True`, *Mathematica* uses the Loos-Weispfenning algorithm [15] to eliminate all quantifier variables that appear only linearly in the system, and then if there are any quantifiers left or the result needs to be solved for the free variables, the CAD algorithm is used. With `LinearQE -> Automatic`, the Loos-Weispfenning algorithm is used only for variables that appear in the system only linearly with constant coefficients. With `LinearQE -> False`, the Loos-Weispfenning algorithm is not used.

With the default setting `LinearQE -> True`, the Loos-Weispfenning algorithm is used to eliminate both x and y, and CAD is used to solve the remaining quantifier-free system with two variables.

```
In[124]:= SetSystemOptions["InequalitySolvingOptions" → "LinearQE" → True];
```

```
In[125]:= a1 = Reduce[∃{x,y} (2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 - x ≤ 3 &&
                 3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^3 + z^3 ≤ z^2 y - 3 y + 5 x), {z, t}]; // Timing
```

Out[125]= {10.205, Null}

```
In[126]:= SetSystemOptions["InequalitySolvingOptions" → "LinearQE" → Automatic];
```

With `LinearQE -> Automatic`, the Loos-Weispfenning algorithm is used only to eliminate x, and CAD is used to solve the remaining system with three variables. For this example, the default method is much faster.

```
In[127]:= SetSystemOptions["InequalitySolvingOptions" → "LinearQE" → Automatic];
a2 = Reduce[∃{x,y} (2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 - x ≤ 3 &&
        3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^3 + z^3 ≤ z^2 y - 3 y + 5 x), {z, t}]; // Timing

Out[128]= {48.81, Null}
```

With `LinearQE -> False`, the Loos-Weispfenning algorithm is not used. `Reduce` uses CAD to solve the original system with four variables, which for this example takes much longer.

```
In[129]:= SetSystemOptions["InequalitySolvingOptions" → "LinearQE" → False];
a3 = Reduce[∃{x,y} (2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 - x ≤ 3 &&
        3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^3 + z^3 ≤ z^2 y - 3 y + 5 x), {z, t}]; // Timing

Out[130]= {97.881, Null}
```

All three methods give the same answer.

```
In[131]:= a1 === a2 === a3

Out[131]= True
```

Here is an example where the default method is not the fastest. With the default setting `LinearQE -> True`, the Loos-Weispfenning algorithm is used to eliminate both x and y, and CAD is used to solve the remaining system with one quantified and one free variable.

```
In[132]:= SetSystemOptions["InequalitySolvingOptions" → "LinearQE" → True];
Reduce[∃{x,y,z} (2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
      3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 ≤ z y), t] // Timing

Out[133]= {0.421,
  t ≤ Root[55 696 + 611 712 #1 + 3 248 544 #1^2 + 13 500 064 #1^3 + 41 178 060 #1^4 + 72 638 592 #1^5 + 76 002 697 #1^6 +
      88 447 680 #1^7 + 181 305 153 #1^8 + 201 350 948 #1^9 + 88 499 331 #1^10 + 68 427 618 #1^11 +
      155 219 660 #1^12 + 20 594 160 #1^13 + 99 572 016 #1^14 + 167 324 192 #1^15 &, 1]}
```

With `LinearQE -> Automatic`, the Loos-Weispfenning algorithm is used only to eliminate x, and then CAD is used to solve the remaining system with two quantified variables and one free variable. This is the fastest method for this example.

```
In[134]:= SetSystemOptions["InequalitySolvingOptions" → "LinearQE" → Automatic];
Reduce[∃{x,y,z} (2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
      3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 ≤ z y), t] // Timing

Out[135]= {0.31,
  t ≤ Root[55 696 + 611 712 #1 + 3 248 544 #1^2 + 13 500 064 #1^3 + 41 178 060 #1^4 + 72 638 592 #1^5 + 76 002 697 #1^6 +
      88 447 680 #1^7 + 181 305 153 #1^8 + 201 350 948 #1^9 + 88 499 331 #1^10 + 68 427 618 #1^11 +
      155 219 660 #1^12 + 20 594 160 #1^13 + 99 572 016 #1^14 + 167 324 192 #1^15 &, 1]}
```

With `LinearQE -> False`, the CAD algorithm is used to solve the system.

In[136]:=
```
SetSystemOptions["InequalitySolvingOptions" → "LinearQE" → False];
Reduce[∃{x,y,z} (2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
    3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 ≤ z y), t] // Timing
```

Out[137]=
```
{0.401,
 t ≤ Root[55 696 + 611 712 #1 + 3 248 544 #1^2 + 13 500 064 #1^3 + 41 178 060 #1^4 + 72 638 592 #1^5 + 76 002 697 #1^6 +
     88 447 680 #1^7 + 181 305 153 #1^8 + 201 350 948 #1^9 + 88 499 331 #1^10 + 68 427 618 #1^11 +
     155 219 660 #1^12 + 20 594 160 #1^13 + 99 572 016 #1^14 + 167 324 192 #1^15 &, 1]}
```

The default setting `LinearQE -> True` is definitely advantageous for quantifier elimination problems where all quantified variables appear only linearly in the system and the quantifier-free version of the system does not need to be given in a solved form. This is because the complexity of the Loos-Weispfenning algorithm depends very little on the number of free variables, unlike the complexity of the CAD algorithm that is doubly exponential in the number of all variables. With `LinearQE -> False`, this example does not finish in 1000 seconds.

In[138]:=
```
SetSystemOptions["InequalitySolvingOptions" → "LinearQE" → True];
Resolve[∃{x,y} (2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 u + v^7 - 6 w^4 t ≤ 3 &&
    3 x - 5 t z^2 - 3 t^2 - y z t - 5 y w z ≥ 2), Reals] // Timing
```

Out[139]=
```
{0.01,
 (-t^3 < 0 && 5 t^3 + 30 t^5 + 27 z + 6 t z + 36 t^3 z + 8 t^4 z - 9 v^7 z - 2 t v^7 z + 30 w z + 40 t^3 w z - 10 v^7 w z + 54 t w^4 z +
     12 t^2 w^4 z + 60 t w^5 z + 110 t^4 z^2 - 36 t^2 u z^3 - 8 t^3 u z^3 - 40 t^2 u w z^3 - 63 z^5 - 14 t z^5 - 70 w z^5 ≤ 0) ||
  (t^3 < 0 && -5 t^3 - 30 t^5 - 27 z - 6 t z - 36 t^3 z - 8 t^4 z + 9 v^7 z + 2 t v^7 z - 30 w z - 40 t^3 w z + 10 v^7 w z - 54 t w^4
     z - 12 t^2 w^4 z - 60 t w^5 z - 110 t^4 z^2 + 36 t^2 u z^3 + 8 t^3 u z^3 + 40 t^2 u w z^3 + 63 z^5 + 14 t z^5 + 70 w z^5 ≤ 0) ||
  (-9 z - 2 t z - 10 w z < 0 && -5 t^3 - 30 t^5 - 27 z - 6 t z - 36 t^3 z - 8 t^4 z + 9 v^7 z + 2 t v^7 z -
     30 w z - 40 t^3 w z + 10 v^7 w z - 54 t w^4 z - 12 t^2 w^4 z - 60 t w^5 z - 110 t^4 z^2 + 36 t^2 u z^3 +
     8 t^3 u z^3 + 40 t^2 u w z^3 + 63 z^5 + 14 t z^5 + 70 w z^5 ≤ 0) || (9 z + 2 t z + 10 w z < 0 &&
   5 t^3 + 30 t^5 + 27 z + 6 t z + 36 t^3 z + 8 t^4 z - 9 v^7 z - 2 t v^7 z + 30 w z + 40 t^3 w z - 10 v^7 w z + 54 t w^4 z +
     12 t^2 w^4 z + 60 t w^5 z + 110 t^4 z^2 - 36 t^2 u z^3 - 8 t^3 u z^3 - 40 t^2 u w z^3 - 63 z^5 - 14 t z^5 - 70 w z^5 ≤ 0) ||
  (t^3 == 0 && 1/2 + 3 t^2 + 11 t z^2 ≤ 0 && -3 - 4 t^3 + v^7 - 6 t w^4 + 4 t^2 u z^2 + 7 z^4 ≤ 0) ||
  (9 z + 2 t z + 10 w z == 0 && 1/2 + 3 t^2 + 11 t z^2 ≤ 0 && -3 - 4 t^3 + v^7 - 6 t w^4 + 4 t^2 u z^2 + 7 z^4 ≤ 0)}
```

LWPreprocessor

The `LWPreprocessor` option setting affects solving decision problems and instance finding. The option specifies whether the Loos-Weispfenning algorithm [8] should be used to eliminate variables that appear at most linearly in all equations and inequalities before applying the CAD algorithm to the resulting system. With the default setting `Automatic`, *Mathematica* uses the Loos-Weispfenning algorithm to eliminate variables that appear only linearly with constant coefficients. With `LWPreprocessor -> True`, the Loos-Weispfenning algorithm is used for all variables that appear only linearly. With `LWPreprocessor -> False`, the Loos-Weispfenning algorithm is not used as a preprocessor to the CAD-based decision algorithm.

With the default setting `LWPreprocessor -> Automatic`, the Loos-Weispfenning algorithm is used only to eliminate x, and CAD is used to find a solution of the remaining system with three variables.

In[140]:=
```
FindInstance[2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
    3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 ≤ z y, {x, y, z, t}] // Timing
```

Out[140]= $\left\{0.06, \left\{\left\{x \to -\frac{44923}{48}, y \to 306, z \to \frac{3}{4}, t \to -15\right\}\right\}\right\}$

With `LWPreprocessor -> True`, the Loos-Weispfenning algorithm is used to eliminate both x and y, and CAD is used to find a solution of the remaining system with two variables. For this example, this method is slower than the default one.

In[141]:=
```
SetSystemOptions["InequalitySolvingOptions" → "LWPreprocessor" → True];
FindInstance[2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
    3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 ≤ z y, {x, y, z, t}] // Timing
```

Out[142]= $\left\{0.17, \left\{\left\{x \to -\frac{845057}{1109760}, y \to \frac{54532}{24565}, z \to 1, t \to -\frac{17}{16}\right\}\right\}\right\}$

With `LWPreprocessor -> False`, the CAD algorithm is used to find a solution of the original system with four variables. For this example, this method is as fast as the default.

In[143]:=
```
SetSystemOptions["InequalitySolvingOptions" → "LWPreprocessor" → False];
FindInstance[2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
    3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 ≤ z y, {x, y, z, t}] // Timing
```

Out[144]= $\left\{0.06, \left\{\left\{x \to -332, y \to 306, z \to \frac{3}{4}, t \to -15\right\}\right\}\right\}$

This example differs from the previous one only in that the last inequality was turned into an equation. With the default setting `LWPreprocessor -> Automatic`, the Loos-Weispfenning algorithm is only used to eliminate x, and CAD is used to find a solution of the remaining system with three variables.

In[145]:=
```
SetSystemOptions["InequalitySolvingOptions" → "LWPreprocessor" → Automatic];
FindInstance[2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
    3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 == z y, {x, y, z, t}] // Timing
```

Out[146]= $\left\{0.2, \left\{\left\{x \to \frac{1}{3}\left(\frac{3341}{256} - \frac{4117\sqrt{943}}{4096}\right), y \to 4, z \to \frac{23}{16}, t \to -\frac{\sqrt{943}}{16}\right\}\right\}\right\}$

With `LWPreprocessor -> True`, the Loos-Weispfenning algorithm is used to eliminate both x and y, and CAD is used to find a solution of the remaining system with two variables. For the revised example, this method is faster than the default one.

```
In[147]:= SetSystemOptions["InequalitySolvingOptions" → "LWPreprocessor" → True];
FindInstance[2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
    3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 == z y, {x, y, z, t}] // Timing
```

```
Out[148]= {0.08, {{x → 1/3 (2 + 5 Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1] + 3 Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1]^2 +
        (-4 - 4 Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1]^2 + 4 Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1]^3) /
         (5 Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1]^2)),
   y → (-4 - 4 Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1]^2 + 4 Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1]^3) /
     (5 Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1]^3), z → 1, t → Root[4 + 4 #1^2 + #1^3 + 5 #1^5 &, 1]}}}
```

With `LWPreprocessor -> False`, the CAD algorithm is used to find a solution of the original system with four variables. For the revised example, this is seven times slower than the default method.

```
In[149]:= SetSystemOptions["InequalitySolvingOptions" → "LWPreprocessor" → False];
FindInstance[2 x + 3 y z + 4 z^2 t ≤ 1 && 5 t^3 y + 7 z^4 - 4 t^3 + 4 z^2 t^2 ≤ 3 &&
    3 x - 5 t z^2 - 3 t^2 - y z t ≥ 2 && t^2 + z^2 == z y, {x, y, z, t}] // Timing
```

```
Out[150]= {1.432, {{x → 0, y → 11, z → 3, t → -2 Sqrt[6]}}}
```

```
In[151]:= SetSystemOptions["InequalitySolvingOptions" → "LWPreprocessor" → Automatic];
```

ProjectAlgebraic

The setting of the `ProjectAlgebraic` option affects handling of algebraic number coefficients in the CAD algorithm.

Algebraic numbers found in coefficients of the input system are replaced with new variables. The new variables are always put first in the variable ordering so that in the projection phase of the CAD algorithm they are eliminated last. When the current projection polynomials contain $k+1$ variables with at least k first variables replacing algebraic number coefficients, we have a choice of whether or not to continue the projection phase. If we do not continue the projection phase, we can start the lifting phase extending the zero-dimensional cell in the first k variables on which each of the variables is equal to the corresponding algebraic number coefficient. If we choose to compute the last k projections, we may find in the lifting phase that the algebraic number coefficient corresponding to a variable being lifted lies between the roots of the projection polynomials. Hence for this variable we will be extending a one-dimensional cell with a rational number sample point. Thus there is a trade-off between avoiding computation of the last k projections and avoiding algebraic number coordinates in sample points.

With `ProjectAlgebraic -> True`, the projection phase is continued for variables replacing algebraic number coefficients until there is one variable left. With `ProjectAlgebraic -> False`, the projection phase is stopped as soon as there is one variable left that does not replace an algebraic number coefficient. With the default setting `ProjectAlgebraic -> Automatic`, the projection phase is stopped if there is at most one variable left that does not replace an algebraic number coefficient and there are at least three projection polynomials, or there is a projection polynomial of degree more than two in the projection variable.

With few high-degree algebraic number coefficients, equations, and inequalities in the system, `ProjectAlgebraics -> True` tends to be a better choice. (`N` is applied to the output for better readability.)

```
In[152]:= SetSystemOptions["InequalitySolvingOptions" → "ProjectAlgebraic" → True];
          FindInstance[Root[#1^9 - 11 #1 + 7 &, 1] x^2 - Root[#1^7 - 5 #1 + 3 &, 1] y^2 - x y == 1,
            {x, y}, Reals] // Timing // N
Out[153]= {0.011, {{x → -1., y → -1.72698}}}
```

```
In[154]:= SetSystemOptions["InequalitySolvingOptions" → "ProjectAlgebraic" → False];
          FindInstance[Root[#1^9 - 11 #1 + 7 &, 1] x^2 - Root[#1^7 - 5 #1 + 3 &, 1] y^2 - x y == 1,
            {x, y}, Reals] // Timing // N
Out[155]= {0.39, {{x → -1., y → -1.72698}}}
```

With many low-degree algebraic number coefficients, equations, and inequalities in the system, `ProjectAlgebraics -> False` tends to be faster.

```
In[156]:= SetSystemOptions["InequalitySolvingOptions" → "ProjectAlgebraic" → True];
          FindInstance[x^2 + y^2 - Sqrt[2] x - Sqrt[3] y - Sqrt[5] < 0 && x < Sqrt[7] y^2, {x, y}, Reals] // Timing
Out[157]= {6.509, {{x → 3/4, y → -57/64}}}
```

```
In[158]:= SetSystemOptions["InequalitySolvingOptions" → "ProjectAlgebraic" → False];
          FindInstance[x^2 + y^2 - Sqrt[2] x - Sqrt[3] y - Sqrt[5] < 0 && x < Sqrt[7] y^2, {x, y}, Reals] // Timing
Out[159]= {0.01, {{x → 3/4, y → -57/64}}}
```

With `ProjectAlgebraics -> Automatic`, *Mathematica* picks the faster method in the second example, but fails to pick the faster method in the first example.

```
In[160]:= SetSystemOptions["InequalitySolvingOptions" → "ProjectAlgebraic" → Automatic];
```

```
In[161]:= FindInstance[Root[#1^9 - 11 #1 + 7 &, 1] x^2 - Root[#1^7 - 5 #1 + 3 &, 1] y^2 - x y == 1,
            {x, y}, Reals] // Timing // N
Out[161]= {0.291, {{x → -1., y → -1.72698}}}
```

```
In[162]:= FindInstance[x^2 + y^2 - √2 x - √3 y - √5 < 0 && x < √7 y^2, {x, y}, Reals] // Timing
```

$$\text{Out[162]= } \left\{0.01, \left\{\left\{x \to \frac{3}{4}, y \to -\frac{57}{64}\right\}\right\}\right\}$$

ProveMultiplicities

The setting of `ProveMultiplicities` determines the way in which the lifting phase of the CAD algorithm validates multiple roots and zero leading coefficients of projection polynomials obtained using arbitrary-precision floating-point number (*Mathematica* "bignum") computations (for more details, see [14, 24]). With the default setting `ProveMultiplicities -> True`, *Mathematica* uses information about the origins of the cell, if this is not sufficient computes exact values of cell coordinates and uses principal subresultant coefficients and exact zero testing, and only if this fails reverts to exact computations. With `ProveMultiplicities -> Automatic`, *Mathematica* uses information about the origins of the cell and, if this is not sufficient, reverts to exact computation. With `ProveMultiplicities -> False`, *Mathematica* reverts to exact computation each time bignum computations fail to separate all roots or prove that the leading coefficients of projection polynomials are nonzero.

Generally, using all available methods of validating results obtained with arbitrary-precision floating-point number computations leads to better performance.

```
In[163]:= SetSystemOptions["InequalitySolvingOptions" → "ProveMultiplicities" → True];
          Reduce[Exists[{y, z}, x^4 + y^4 + z^4 == 1 && 2 + x y + z ≤ x^2 + y^2 + z^2], x, Reals] //
           Timing
```

```
Out[164]= {0.17,
  Root[-5 915 760 + 39 370 017 #1^2 - 148 378 932 #1^4 + 577 876 048 #1^6 - 2 081 150 580 #1^8 + 5 343 033 030 #1^10 -
      9 257 957 588 #1^12 + 10 980 806 064 #1^14 - 9 088 500 912 #1^16 + 5 325 466 813 #1^18 - 2 232 144 792 #1^20 +
      671 693 097 #1^22 - 143 343 788 #1^24 + 20 981 862 #1^26 - 1 920 672 #1^28 + 88 209 #1^30 &, 1] ≤ x ≤
   Root[-5 915 760 + 39 370 017 #1^2 - 148 378 932 #1^4 + 577 876 048 #1^6 - 2 081 150 580 #1^8 + 5 343 033 030 #1^10 -
      9 257 957 588 #1^12 + 10 980 806 064 #1^14 - 9 088 500 912 #1^16 + 5 325 466 813 #1^18 - 2 232 144 792 #1^20 +
      671 693 097 #1^22 - 143 343 788 #1^24 + 20 981 862 #1^26 - 1 920 672 #1^28 + 88 209 #1^30 &, 2]}
```

```
In[165]:= SetSystemOptions["InequalitySolvingOptions" → "ProveMultiplicities" → Automatic];
          Reduce[Exists[{y, z}, x^4 + y^4 + z^4 == 1 && 2 + x y + z ≤ x^2 + y^2 + z^2], x, Reals] //
           Timing
```

```
Out[166]= {9.314,
  Root[-5 915 760 + 39 370 017 #1^2 - 148 378 932 #1^4 + 577 876 048 #1^6 - 2 081 150 580 #1^8 + 5 343 033 030 #1^10 -
      9 257 957 588 #1^12 + 10 980 806 064 #1^14 - 9 088 500 912 #1^16 + 5 325 466 813 #1^18 - 2 232 144 792 #1^20 +
      671 693 097 #1^22 - 143 343 788 #1^24 + 20 981 862 #1^26 - 1 920 672 #1^28 + 88 209 #1^30 &, 1] ≤ x ≤
   Root[-5 915 760 + 39 370 017 #1^2 - 148 378 932 #1^4 + 577 876 048 #1^6 - 2 081 150 580 #1^8 + 5 343 033 030 #1^10 -
      9 257 957 588 #1^12 + 10 980 806 064 #1^14 - 9 088 500 912 #1^16 + 5 325 466 813 #1^18 - 2 232 144 792 #1^20 +
      671 693 097 #1^22 - 143 343 788 #1^24 + 20 981 862 #1^26 - 1 920 672 #1^28 + 88 209 #1^30 &, 2]}
```

In[167]:=
```
SetSystemOptions["InequalitySolvingOptions" → "ProveMultiplicities" → False];
TimeConstrained[
 Reduce[Exists[{y, z}, x^4 + y^4 + z^4 == 1 && 2 + x y + z ≤ x^2 + y^2 + z^2], x, Reals] //
  Timing, 60]
```
Out[168]= $Aborted

In[169]:=
```
SetSystemOptions["InequalitySolvingOptions" → "ProveMultiplicities" → True];
```

QuadraticQE

The `QuadraticQE` option specifies whether the quadratic case of Weispfenning's quantifier elimination by virtual substitution algorithm [22, 23] should be used to eliminate quantified variables that appear at most quadratically in all equations and inequalities in the system. The complexity of Weispfenning's algorithm depends very little on the number of free variables, unlike the complexity of the CAD algorithm that is doubly exponential in the number of all variables. Hence, it is definitely advantageous to use it when all quantifiers can be eliminated using the algorithm, there are many free variables present, and the quantifier-free version of the system does not need to be given in a solved form. On the other hand, eliminating a variable using Weispfenning's algorithm often significantly increases the size of the formula. So if *Mathematica* needs to apply CAD to the result or if the system contains few free variables, using CAD on the original system may be faster. With the default setting `Automatic`, *Mathematica* uses the algorithm for `Resolve` with no variables specified and with at least two parameters present, and for `Reduce` and `Resolve` with at least three variables as long as elimination of one variable at most doubles the `LeafCount` of the system. This criterion seems to work reasonably well; however, for some examples it does not give the optimal choice of the algorithm. Changing the option value may allow problems to be solved which otherwise take a very long time. With `LinearQE -> True`, Weispfenning's algorithm is used whenever there is a quadratic variable to eliminate, with `LinearQE -> False`, Weispfenning's algorithm is not used.

`Resolve` with no variables specified and with at least two parameters present uses Weispfenning's algorithm to eliminate x. The result is not solved for the parameters a, b, and c.

In[170]:=
```
Resolve[∃x (a x^2 + b x + c) (c x^2 + b x + a) <= 0, Reals] // Timing
```
Out[170]=
```
{0.03, a c < 0 || (a == 0 && a b + b c > 0) || (c == 0 && a b + b c > 0) || (a ≠ 0 && -b^2 + 4 a c ≤ 0) ||
  (c ≠ 0 && -b^2 + 4 a c ≤ 0) || (a == 0 && b == 0 && a^2 + b^2 + c^2 ≤ 0) || (a == 0 && a + c == 0 && a^2 + b^2 + c^2 ≤ 0) ||
  (a == 0 && b ≠ 0 && a^2 b^2 c^2 - a b^2 c^3 + a c^5 ≤ 0) || (b == 0 && c == 0 && a^2 + b^2 + c^2 ≤ 0) ||
  (c == 0 && a + c == 0 && a^2 + b^2 + c^2 ≤ 0) || (b ≠ 0 && c == 0 && a^5 c - a^3 b^2 c + a^2 b^2 c^2 ≤ 0)}
```

`Reduce` by default uses CAD for this example. The result is solved for the parameters a, b, and c.

In[171]:=
```
Reduce[∃x (a x^2 + b x + c) (c x^2 + b x + a) <= 0, {a, b, c}, Reals] // Timing
```

Out[171]=
$$\left\{0.291, \left(a < 0 \&\& c \geq \frac{b^2}{4\,a}\right) \,||\, a == 0 \,||\, \left(a > 0 \&\& c \leq \frac{b^2}{4\,a}\right)\right\}$$

With `QuadraticQE -> True`, `Reduce` uses Weispfenning's algorithm to eliminate x and then CAD to solve the quantifier-free formula for the parameters a, b, and c. In this example this is faster than the default method of using CAD from the beginning.

In[172]:=
```
SetSystemOptions["InequalitySolvingOptions" → "QuadraticQE" → True];
Reduce[∃x (a x^2 + b x + c) (c x^2 + b x + a) <= 0, {a, b, c}, Reals] // Timing
```

Out[173]=
$$\left\{0.17, \left(a < 0 \&\& c \geq \frac{b^2}{4\,a}\right) \,||\, a == 0 \,||\, \left(a > 0 \&\& c \leq \frac{b^2}{4\,a}\right)\right\}$$

For this system with three free variables Weispfenning's algorithm works much better than CAD. With `QuadraticQE -> False`, `Resolve` does not finish in 1000 seconds.

In[174]:=
```
SetSystemOptions["InequalitySolvingOptions" → "QuadraticQE" → Automatic];
Resolve[
  ∃t (-602 + 528 z - 222 z^2 - 410 t - 685 z t + 427 t^2 + 422 x + 5 z x - 279 t x - 188 x^2 + 1000 y -
        704 z y + 879 t y - 179 x y - 689 y^2 == 0 &&
      -723 - 380 z + 323 z^2 - 964 t - 749 z t - 9 t^2 + 497 x - 191 z x + 147 t x +
        815 x^2 + 935 y - 536 z y - 558 t y - 152 x y + 400 y^2 ≥ 0), Reals] // Timing
```

Out[175]=
```
{0.02,
 (93 310 576 x + 312 563 284 x^2 + 619 260 202 y - 174 039 637 x y + 343 049 591 y^2 - 552 659 742 z - 119 285 170
      x z - 72 117 355 y z - 107 223 538 z^2 ≥ 438 555 086 && -491 996 x + 398 945 x^2 - 2 428 780 y -
    184 750 x y + 1 949 453 y^2 - 340 124 z + 373 690 x z - 1798 y z + 848 401 z^2 ≥ -1 196 316) ||
  (-491 996 x + 398 945 x^2 - 2 428 780 y - 184 750 x y + 1 949 453 y^2 - 340 124 z +
    373 690 x z - 1798 y z + 848 401 z^2 ≥ -1 196 316 &&
   86 243 585 140 x - 498 040 191 089 x^2 + 109 809 123 842 x^3 + 131 969 169 211 x^4 - 484 187 889 894 y +
    419 393 624 593 x y + 362 278 042 647 x^2 y - 133 070 811 401 x^3 y + 202 349 297 280 y^2 - 82 166 879 722 x y^2 +
    289 809 046 115 x^2 y^2 + 247 824 969 889 y^3 - 76 078 568 059 x y^3 + 19 522 904 791 y^4 + 300 933 814 382 z +
    137 426 763 740 x z - 651 512 554 048 x^2 z - 82 614 322 010 x^3 z - 106 739 496 711 y z -
    29 291 657 121 x y z - 82 755 933 843 x^2 y z - 840 794 940 583 y^2 z + 38 003 381 704 x y^2 z -
    469 158 313 975 y^3 z + 299 057 381 894 z^2 + 126 196 261 244 x z^2 - 114 619 700 688 x^2 z^2 +
    129 231 867 162 y z^2 + 56 929 552 463 x y z^2 - 439 149 714 263 y^2 z^2 - 102 928 178 270 z^3 +
    26 325 949 198 x z^3 - 153 241 487 043 y z^3 - 107 856 045 675 z^4 ≤ 19 224 638 243)}
```

For this system with only one free variable `Resolve` uses CAD.

In[176]:=
```
Resolve[∀{x,y} Implies[x > r && y > r, x^2 (1 + 2 y)^2 > y^2 (1 + 2 x^2)], Reals] // Timing
```

Out[176]=
$$\left\{0.06, r \geq \frac{1}{\sqrt{2}}\right\}$$

Weispfenning's algorithm is slower here and gives a more complicated result.

In[177]:=
```
SetSystemOptions["InequalitySolvingOptions" → "QuadraticQE" → True];
Resolve[∀{x,y} Implies[x > r && y > r, x^2 (1 + 2 y)^2 > y^2 (1 + 2 x^2)], Reals] // LeafCount //
 Timing
```
Out[177]=
```
{0.27, 2711}
```

In[178]:=
```
SetSystemOptions["InequalitySolvingOptions" → "QuadraticQE" → Automatic];
```

QVSPreprocessor

The `QVSPreprocessor` option setting affects solving decision problems and instance finding. The option specifies whether the quadratic case of Weispfenning's quantifier elimination by virtual substitution algorithm [22, 23] should be used to eliminate variables that appear at most quadratically in all equations and inequalities before applying the CAD algorithm to the resulting system. The default setting is `False` and the algorithm is not used. There are examples where using Weispfenning's algorithm as a preprocessor significantly helps the performance, and there are examples where using the preprocessor significantly hurts the performance. It seems that the preprocessor tends to help in examples with many variables and where instances exist. With `QVSPreprocessor -> True`, Weispfenning's algorithm is used each time there is a quadratic variable. With `QVSPreprocessor -> Automatic`, *Mathematica* uses the algorithm for systems with at least four variables.

Here *Mathematica* finds a solution using Weispfenning's algorithm as a preprocessor. Without the preprocessor this example takes 470 seconds.

In[179]:=
```
SetSystemOptions["InequalitySolvingOptions" → "QVSPreprocessor" → True];
FindInstance[-11 - 909 y z - 462 y^2 z + (657 - 471 y^3 + 501 z - 48 x z) t + t^2 == 0 &&
   258 + 223 x^2 y - 544 y^3 + 571 z^2 + 38 y z^2 + (-798 + 79 x^2 y + 214 y^2 - 828 x^2 z - 392 z^2) t + t^2 ≥
    0, {x, y, z, t}, Reals] // Timing
```
Out[180]=
$$\left\{0.07, \left\{\left\{x \to 0,\ y \to \frac{1}{308}\left(-303 + \sqrt{\frac{282\,203}{3}}\right),\ z \to -1,\ t \to 0\right\}\right\}\right\}$$

This uses CAD to show that there are no solutions. With `QVSPreprocessor -> True` this example does not finish in 1000 seconds, due to complexity of computing `LogicalExpand` for the generated large logical formulas.

In[181]:=
```
SetSystemOptions["InequalitySolvingOptions" → "QVSPreprocessor" → False];
f =.; FindInstance[! (a < 0 || b < 0 || c < 0 || d < 0 || e < 0 || f < 0 || a^2 + b^2 > e^2 ||
      c^2 + d^2 > f^2 || a c + b d ≤ e f), {a, b, c, d, e, f}, Reals] // Timing
```
Out[181]=
```
{0.071, {}}
```

ReducePowers

For any variable x in the input to the CAD algorithm, if all powers of x appearing in the system are integer multiples of an integer k, *Mathematica* replaces x^k in the input system with a new variable, runs the CAD on the new system, and then resolves the answer so that it is expressed in terms of the original variables. Setting `ReducePowers -> False` turns off this shortcut. With `ReducePowers -> False`, the algebraic functions appearing as cell bounds in the output of the CAD algorithm are always rational functions, quadratic radical expressions, or `Root` objects. With the default setting `ReducePowers -> True`, they may in addition be $e^{1/n}$ for any of the previous expressions e, or `Root[`$a\,\#^n - e$` &, 1]` for some integer a, and a rational function or a quadratic radical expression e.

With the default setting `ReducePowers -> True`, the CAD algorithm solves a quadratic equation in variables replacing x^7 and y^5, and then the result is represented in terms of x and y. The result contains `Root` objects with quadratic radical expressions inside.

In[182]:=
```
Reduce[x^14 + 3 x^7 y^5 - 5 y^10 == 1, {x, y}, Reals] // Timing
```

Out[182]=

$$\Big\{0.02,\ \Big(x < -\left(\tfrac{5}{29}\right)^{1/14} 2^{1/7}\ \&\&\ \big(y == \mathrm{Root}[-3\,x^7 + \sqrt{-20 + 29\,x^{14}} + 10\,\#1^5\ \&,\ 1]\ ||\ y == \mathrm{Root}[-3\,x^7 - \sqrt{-20 + 29\,x^{14}} + 10\,\#1^5\ \&,\ 1]\big)\Big)\ ||$$
$$\Big(x == -\left(\tfrac{5}{29}\right)^{1/14} 2^{1/7}\ \&\&\ y == \mathrm{Root}[-3\,x^7 + \sqrt{-20 + 29\,x^{14}} + 10\,\#1^5\ \&,\ 1]\Big)\ ||$$
$$\Big(x == \left(\tfrac{5}{29}\right)^{1/14} 2^{1/7}\ \&\&\ y == \mathrm{Root}[-3\,x^7 + \sqrt{-20 + 29\,x^{14}} + 10\,\#1^5\ \&,\ 1]\Big)\ ||\ \Big(x > \left(\tfrac{5}{29}\right)^{1/14} 2^{1/7}\ \&\&$$
$$\big(y == \mathrm{Root}[-3\,x^7 + \sqrt{-20 + 29\,x^{14}} + 10\,\#1^5\ \&,\ 1]\ ||\ y == \mathrm{Root}[-3\,x^7 - \sqrt{-20 + 29\,x^{14}} + 10\,\#1^5\ \&,\ 1]\big)\Big)\Big\}$$

With `ReducePowers -> True`, the CAD algorithm solves the original 14$^{\text{th}}$ degree equation that takes several times longer. The result contains only `Root` objects with polynomial expressions inside.

In[183]:=
```
SetSystemOptions["InequalitySolvingOptions" → "ReducePowers" → False];
Reduce[x^14 + 3 x^7 y^5 - 5 y^10 == 1, {x, y}, Reals] // Timing
```

Out[184]=

$$\{0.07,\ (x < \mathrm{Root}[-20 + 29\,\#1^{14}\ \&,\ 1]\ \&\&$$
$$(y == \mathrm{Root}[1 - x^{14} - 3\,x^7\,\#1^5 + 5\,\#1^{10}\ \&,\ 1]\ ||\ y == \mathrm{Root}[1 - x^{14} - 3\,x^7\,\#1^5 + 5\,\#1^{10}\ \&,\ 2]))\ ||$$
$$(x == \mathrm{Root}[-20 + 29\,\#1^{14}\ \&,\ 1]\ \&\&\ y == \mathrm{Root}[1 - x^{14} - 3\,x^7\,\#1^5 + 5\,\#1^{10}\ \&,\ 1])\ ||$$
$$(x == \mathrm{Root}[-20 + 29\,\#1^{14}\ \&,\ 2]\ \&\&\ y == \mathrm{Root}[1 - x^{14} - 3\,x^7\,\#1^5 + 5\,\#1^{10}\ \&,\ 1])\ ||$$
$$(\mathrm{Root}[-20 + 29\,\#1^{14}\ \&,\ 2] < x < 1\ \&\&$$
$$(y == \mathrm{Root}[1 - x^{14} - 3\,x^7\,\#1^5 + 5\,\#1^{10}\ \&,\ 1]\ ||\ y == \mathrm{Root}[1 - x^{14} - 3\,x^7\,\#1^5 + 5\,\#1^{10}\ \&,\ 2]))\ ||$$
$$(x == 1\ \&\&\ (y == 0\ ||\ y == \mathrm{Root}[-3 + 5\,\#1^5\ \&,\ 1]))\ ||$$
$$(x > 1\ \&\&\ (y == \mathrm{Root}[1 - x^{14} - 3\,x^7\,\#1^5 + 5\,\#1^{10}\ \&,\ 1]\ ||\ y == \mathrm{Root}[1 - x^{14} - 3\,x^7\,\#1^5 + 5\,\#1^{10}\ \&,\ 2])))\}$$

RootReduced

For systems with equational constraints generating a zero-dimensional ideal I, *Mathematica* uses a variant of the CAD algorithm that finds projection polynomials using Gröbner basis methods. If the lexicographic order Gröbner basis of I contains linear polynomials with constant coefficients in every variable but the last one (which is true "generically"), then all coordinates of solutions are easily represented as polynomials in the last coordinate. Setting `RootReduced` to `True` causes *Mathematica* to represent each coordinate as a single numeric `Root` object. Computing this reduced representation often takes much longer than solving the system.

By default, we get the value of y expressed in terms of x.

In[185]:= **Reduce[y^5 - 3 y^2 + 2 y + x^5 + 7 x + 4 == 0 && y^2 + y - x^5 - 3 x - 11 == 0, {x, y}, Reals] // Timing**

Out[185]= $\{0.011,$
$x == \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^2 + 32\,850\,\#1^3 + 4455\,\#1^4 + 72\,765\,\#1^5 + 80\,162\,\#1^6 + 32\,790\,\#1^7 + 5940\,\#1^8 + 405\,\#1^9 + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} + 270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]\ \&\&$
$y == \frac{106\,736\,486}{182\,019} + \frac{1\,296\,051\,x}{1462} + \frac{66\,665\,810\,x^2}{182\,019} - \frac{969\,563\,x^3}{8466} - \frac{24\,035\,081\,x^4}{364\,038} + \frac{96\,373\,723\,x^5}{364\,038} + \frac{54\,920\,533\,x^6}{182\,019} + \frac{25\,145\,123\,x^7}{364\,038} - \frac{10\,853\,975\,x^8}{182\,019} - \frac{1\,489\,646\,x^9}{182\,019} + \frac{8\,836\,411\,x^{10}}{182\,019} + \frac{4\,385\,547\,x^{11}}{121\,346} + \frac{93\,708\,x^{12}}{60\,673} - \frac{530\,532\,x^{13}}{60\,673} + \frac{69\,480\,x^{14}}{60\,673} + \frac{85\,501\,x^{15}}{21\,414} + \frac{305\,971\,x^{16}}{182\,019} - \frac{3130\,x^{17}}{10\,707} - \frac{5092\,x^{18}}{10\,707} + \frac{2072\,x^{19}}{10\,707} + \frac{43\,705\,x^{20}}{364\,038} + \frac{3719\,x^{21}}{182\,019} - \frac{2194\,x^{22}}{182\,019} - \frac{1492\,x^{23}}{182\,019} + \frac{1208\,x^{24}}{182\,019}\}$

With `Backsubstitution -> True`, we get a numeric value of y, but the representation of the value is large.

In[186]:= **Reduce[y^5 - 3 y^2 + 2 y + x^5 + 7 x + 4 == 0 && y^2 + y - x^5 - 3 x - 11 == 0, {x, y}, Reals, Backsubstitution → True] // Timing**

Out[186]= $\{0.02,$
$x == \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^2 + 32\,850\,\#1^3 + 4455\,\#1^4 + 72\,765\,\#1^5 + 80\,162\,\#1^6 + 32\,790\,\#1^7 + 5940\,\#1^8 + 405\,\#1^9 + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} + 270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]\ \&\&$
$y == \frac{1}{364\,038}\,(213\,472\,972 + 322\,716\,699\,\text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^2 + 32\,850\,\#1^3 + 4455\,\#1^4 + 72\,765\,\#1^5 + 80\,162\,\#1^6 + 32\,790\,\#1^7 + 5940\,\#1^8 + 405\,\#1^9 + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} + 270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1] +$
$133\,331\,620\,\text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^2 + 32\,850\,\#1^3 + 4455\,\#1^4 + 72\,765\,\#1^5 + 80\,162\,\#1^6 + 32\,790\,\#1^7 + 5940\,\#1^8 + 405\,\#1^9 + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} + 270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^2 -$
$41\,691\,209\,\text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^2 + 32\,850\,\#1^3 + 4455\,\#1^4 + 72\,765\,\#1^5 +$
$80\,162\,\#1^6 + 32\,790\,\#1^7 + 5940\,\#1^8 + 405\,\#1^9 + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{3} -$

$24\,035\,081\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{4} +$

$96\,373\,723\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{5} +$

$109\,841\,066\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{6} +$

$25\,145\,123\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{7} -$

$21\,707\,950\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{8} -$

$2\,979\,292\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{9} +$

$17\,672\,822\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{10} +$

$13\,156\,641\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{11} +$

$562\,248\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{12} -$

$3\,183\,192\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{13} +$

$416\,880\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{14} +$

$1\,453\,517\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{15} +$

$611\,942\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{16} -$

$106\,420\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{17} -$

$173\,128\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{18} +$

$70\,448\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

$80\,162\,\#1^{6} + 32\,790\,\#1^{7} + 5940\,\#1^{8} + 405\,\#1^{9} + 13\,281\,\#1^{10} + 10\,910\,\#1^{11} + 2970\,\#1^{12} +$

$270\,\#1^{13} + 1210\,\#1^{15} + 660\,\#1^{16} + 90\,\#1^{17} + 55\,\#1^{20} + 15\,\#1^{21} + \#1^{25}\,\&,\ 1]^{19} +$

$43\,705\ \text{Root}[156\,956 + 220\,462\,\#1 + 120\,941\,\#1^{2} + 32\,850\,\#1^{3} + 4455\,\#1^{4} + 72\,765\,\#1^{5} +$

```
80 162 ♯1^6 + 32 790 ♯1^7 + 5940 ♯1^8 + 405 ♯1^9 + 13 281 ♯1^10 + 10 910 ♯1^11 + 2970 ♯1^12 +
    270 ♯1^13 + 1210 ♯1^15 + 660 ♯1^16 + 90 ♯1^17 + 55 ♯1^20 + 15 ♯1^21 + ♯1^25 &, 1]^20 +
7438 Root[156 956 + 220 462 ♯1 + 120 941 ♯1^2 + 32 850 ♯1^3 + 4455 ♯1^4 + 72 765 ♯1^5 +
    80 162 ♯1^6 + 32 790 ♯1^7 + 5940 ♯1^8 + 405 ♯1^9 + 13 281 ♯1^10 + 10 910 ♯1^11 + 2970 ♯1^12 +
    270 ♯1^13 + 1210 ♯1^15 + 660 ♯1^16 + 90 ♯1^17 + 55 ♯1^20 + 15 ♯1^21 + ♯1^25 &, 1]^21 -
4388 Root[156 956 + 220 462 ♯1 + 120 941 ♯1^2 + 32 850 ♯1^3 + 4455 ♯1^4 + 72 765 ♯1^5 +
    80 162 ♯1^6 + 32 790 ♯1^7 + 5940 ♯1^8 + 405 ♯1^9 + 13 281 ♯1^10 + 10 910 ♯1^11 + 2970 ♯1^12 +
    270 ♯1^13 + 1210 ♯1^15 + 660 ♯1^16 + 90 ♯1^17 + 55 ♯1^20 + 15 ♯1^21 + ♯1^25 &, 1]^22 -
2984 Root[156 956 + 220 462 ♯1 + 120 941 ♯1^2 + 32 850 ♯1^3 + 4455 ♯1^4 + 72 765 ♯1^5 +
    80 162 ♯1^6 + 32 790 ♯1^7 + 5940 ♯1^8 + 405 ♯1^9 + 13 281 ♯1^10 + 10 910 ♯1^11 + 2970 ♯1^12 +
    270 ♯1^13 + 1210 ♯1^15 + 660 ♯1^16 + 90 ♯1^17 + 55 ♯1^20 + 15 ♯1^21 + ♯1^25 &, 1]^23 +
2416 Root[156 956 + 220 462 ♯1 + 120 941 ♯1^2 + 32 850 ♯1^3 + 4455 ♯1^4 + 72 765 ♯1^5 +
    80 162 ♯1^6 + 32 790 ♯1^7 + 5940 ♯1^8 + 405 ♯1^9 + 13 281 ♯1^10 + 10 910 ♯1^11 + 2970 ♯1^12 +
    270 ♯1^13 + 1210 ♯1^15 + 660 ♯1^16 + 90 ♯1^17 + 55 ♯1^20 + 15 ♯1^21 + ♯1^25 &, 1]^24)}
```

Setting `RootReduced -> True` causes *Mathematica* to represent the value of y as a single `Root` object. However, the computation takes ten times longer.

```
In[187]:= SetSystemOptions["InequalitySolvingOptions" → "RootReduced" → True];
          Reduce[y^5 - 3 y^2 + 2 y + x^5 + 7 x + 4 == 0 && y^2 + y - x^5 - 3 x - 11 == 0, {x, y}, Reals] // Timing
```

```
Out[188]= {0.09,
  x == Root[156 956 + 220 462 ♯1 + 120 941 ♯1^2 + 32 850 ♯1^3 + 4455 ♯1^4 + 72 765 ♯1^5 + 80 162 ♯1^6 + 32 790 ♯1^7 +
      5940 ♯1^8 + 405 ♯1^9 + 13 281 ♯1^10 + 10 910 ♯1^11 + 2970 ♯1^12 + 270 ♯1^13 +
      1210 ♯1^15 + 660 ♯1^16 + 90 ♯1^17 + 55 ♯1^20 + 15 ♯1^21 + ♯1^25 &, 1] &&
   y == Root[-33 447 + 39 343 ♯1 - 55 392 ♯1^2 + 54 390 ♯1^3 - 43 015 ♯1^4 + 38 216 ♯1^5 - 32 870 ♯1^6 +
      31 390 ♯1^7 - 22 700 ♯1^8 + 14 085 ♯1^9 - 9582 ♯1^10 + 6610 ♯1^11 - 5310 ♯1^12 + 2870 ♯1^13 - 1380 ♯1^14 +
      850 ♯1^15 - 500 ♯1^16 + 370 ♯1^17 - 120 ♯1^18 + 40 ♯1^19 - 35 ♯1^20 + 15 ♯1^21 - 10 ♯1^22 + ♯1^25 &, 1]}
```

```
In[189]:= SetSystemOptions["InequalitySolvingOptions" → "RootReduced" → False];
```

ThreadOr

The `ThreadOr` option specifies how the identity

$$\exists_{x_1,\ldots,x_n}(\Phi_1 \vee \ldots \vee \Phi_k) \Longleftrightarrow \exists_{x_1,\ldots,x_n}\Phi_1 \vee \ldots \vee \exists_{x_1,\ldots,x_n}\Phi_k \tag{8}$$

should be used in the decision algorithm (`Reduce` and `Resolve` for systems containing no free variables or parameters), `FindInstance`, and quantifier elimination (`Resolve` with no variables specified). With the default setting `ThreadOr -> True`, the identity (8) is used before attempting any solution algorithms. With `ThreadOr -> False`, the identity (8) may be used by algorithms that require using it (for instance, the Simplex algorithm), but will not be used by algorithms that do not require using it (for instance, the CAD algorithm).

Here `Reduce` finds an instance satisfying the first simpler term of `Or`, and hence avoids dealing with the second, more complicated, term.

```
In[190]:= Reduce[∃{x,y,z} (x + y + z ≥ 0 || (x^5 - 3 x y^4 z + 17 x^3 z^2 - 11 y == 0 && x^2 + y^2 + z^2 ≤ 1)),
            Reals] // Timing
Out[190]= {2.17604×10^-14, True}
```

With `ThreadOr -> False`, `Reduce` needs to run a CAD-based decision algorithm on the whole system.

```
In[191]:= SetSystemOptions["InequalitySolvingOptions" → "ThreadOr" → False];
          Reduce[∃{x,y,z} (x + y + z ≥ 0 || (x^5 - 3 x y^4 z + 17 x^3 z^2 - 11 y == 0 && x^2 + y^2 + z^2 ≤ 1)),
            Reals] // Timing
Out[192]= {0.801, True}
```

This system has no solutions and so with `ThreadOr -> True` `Reduce` needs to run a CAD-based decision algorithm on each of the terms.

```
In[193]:= SetSystemOptions["InequalitySolvingOptions" → "ThreadOr" → True];
          Reduce[
            ∃{x,y,z} ((x^2 + y^2 + z^2 < 1 && (x - 2)^2 + y^2 + z^2 < 1 && x^2 + (y - 2)^2 + z^2 ≥ 1 && x^2 + y^2 + (z - 2)^2 ≥
                  1) || (x^2 + y^2 + z^2 < 1 && (x - 2)^2 + y^2 + z^2 ≥ 1 && x^2 + (y - 2)^2 + z^2 < 1 &&
                 x^2 + y^2 + (z - 2)^2 ≥ 1) || (x^2 + y^2 + z^2 ≥ 1 && (x - 2)^2 + y^2 + z^2 < 1 &&
                 x^2 + (y - 2)^2 + z^2 < 1 && x^2 + y^2 + (z - 2)^2 ≥ 1) || (x^2 + y^2 + z^2 < 1 &&
                 (x - 2)^2 + y^2 + z^2 ≥ 1 && x^2 + (y - 2)^2 + z^2 ≥ 1 && x^2 + y^2 + (z - 2)^2 < 1) ||
               (x^2 + y^2 + z^2 ≥ 1 && (x - 2)^2 + y^2 + z^2 < 1 && x^2 + (y - 2)^2 + z^2 ≥ 1 &&
                 x^2 + y^2 + (z - 2)^2 < 1) || (x^2 + y^2 + z^2 ≥ 1 && (x - 2)^2 + y^2 + z^2 ≥ 1 &&
                 x^2 + (y - 2)^2 + z^2 < 1 && x^2 + y^2 + (z - 2)^2 < 1)), Reals] // Timing
Out[194]= {1.512, False}
```

Since all six terms of `Or` involve exactly the same polynomials, running a CAD-based decision algorithm on the whole expression and running a CAD-based decision algorithm on one of the terms consist of very similar computations. In this case the computation with `ThreadOr -> False` is faster.

```
In[195]:= SetSystemOptions["InequalitySolvingOptions" → "ThreadOr" → False];
          Reduce[
            ∃{x,y,z} ((x^2 + y^2 + z^2 < 1 && (x - 2)^2 + y^2 + z^2 < 1 && x^2 + (y - 2)^2 + z^2 ≥ 1 && x^2 + y^2 + (z - 2)^2 ≥
                  1) || (x^2 + y^2 + z^2 < 1 && (x - 2)^2 + y^2 + z^2 ≥ 1 && x^2 + (y - 2)^2 + z^2 < 1 &&
                 x^2 + y^2 + (z - 2)^2 ≥ 1) || (x^2 + y^2 + z^2 ≥ 1 && (x - 2)^2 + y^2 + z^2 < 1 &&
                 x^2 + (y - 2)^2 + z^2 < 1 && x^2 + y^2 + (z - 2)^2 ≥ 1) || (x^2 + y^2 + z^2 < 1 &&
                 (x - 2)^2 + y^2 + z^2 ≥ 1 && x^2 + (y - 2)^2 + z^2 ≥ 1 && x^2 + y^2 + (z - 2)^2 < 1) ||
               (x^2 + y^2 + z^2 ≥ 1 && (x - 2)^2 + y^2 + z^2 < 1 && x^2 + (y - 2)^2 + z^2 ≥ 1 &&
                 x^2 + y^2 + (z - 2)^2 < 1) || (x^2 + y^2 + z^2 ≥ 1 && (x - 2)^2 + y^2 + z^2 ≥ 1 &&
                 x^2 + (y - 2)^2 + z^2 < 1 && x^2 + y^2 + (z - 2)^2 < 1)), Reals] // Timing
Out[196]= {0.341, False}
```

```
In[197]:= SetSystemOptions["InequalitySolvingOptions" → "ThreadOr" → True];
```

ZengDecision

The option `ZengDecision` specifies whether *Mathematica* should use the algorithm by G. X. Zeng and X. N. Zeng [18]. The algorithm applies to decision problems with systems that consist of a single strict inequality. There are examples for which the algorithm performs better than the strict inequality variant of the CAD algorithm described in [13]. However, for randomly chosen inequalities, it seems to perform worse; therefore, it is not used by default. Here is an example from [18] that runs faster with `ZengDecision -> True`.

```
In[198]:= FindInstance[x^4 + y^4 + z^4 + w^4 - 5 x y z w + x^2 + y^2 + z^2 + w^2 + 1 < 0,
             {x, y, z, w}, Reals] // Timing

Out[198]= {7.17, {{x → -5, y → -5, z → -6, w → -4}}}

In[199]:= SetSystemOptions["InequalitySolvingOptions" → "ZengDecision" → True];
          FindInstance[x^4 + y^4 + z^4 + w^4 - 5 x y z w + x^2 + y^2 + z^2 + w^2 + 1 < 0,
             {x, y, z, w}, Reals] // Timing

Out[200]= {0.43, {{x → -5, y → -5, z → -6, w → -4}}}

In[201]:= SetSystemOptions["InequalitySolvingOptions" → "ZengDecision" → False];
```

References

[1] Caviness, B. F. and J. R. Johnson, eds. *Quantifier Elimination and Cylindrical Algebraic Decomposition: Texts and Monographs in Symbolic Computation*. Springer-Verlag, 1998.

[2] Tarski, A. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.

[3] Łojasiewicz, S. *Ensembles Semi-Analytiques.* Inst. Hautes Études Sci., 1964.

[4] Collins, G. E. "Quantifier Elimination for the Elementary Theory of Real Closed Fields by Cylindrical Algebraic Decomposition." *Lecture Notes in Computer Science* **33** (1975): 134–183.

[5] Hong, H. "An Improvement of the Projection Operator in Cylindrical Algebraic Decomposition." In *Issac '90: Proceedings of the International Symposium on Symbolic and Algebraic Computation* (M. Nagata, ed.), 261–264, 1990.

[6] McCallum, S. "An Improved Projection for Cylindrical Algebraic Decomposition of Three Dimensional Space." *J. Symb. Comput.* 5, no. 1/2 (1988): 141–161.

[7] McCallum, S. "An Improved Projection for Cylindrical Algebraic Decomposition." In *Quantifier Elimination and Cylindrical Algebraic Decomposition: Texts and Monographs in Symbolic Computation* (B. F. Caviness and J. R. Johnson, eds.). Springer-Verlag, 1998.

[8] Brown, C. W. "Improved Projection for Cylindrical Algebraic Decomposition." *J. Symb. Comput.* 32, no. 5 (2001): 447-465.

[9] Collins, G. E. "Quantifier Elimination by Cylindrical Algebraic Decomposition—Twenty Years of Progress." In *Quantifier Elimination and Cylindrical Algebraic Decomposition: Texts and Monographs in Symbolic Computation* (B. F. Caviness and J. R. Johnson, eds.). Springer-Verlag, 1998.

[10] McCallum, S. "On Projection in CAD-Based Quantifier Elimination with Equational Constraint." In *Issac '99: Proceedings of the International Symposium on Symbolic and Algebraic Computation* (Sam Dooley, ed.), 1999.

[11] McCallum, S. "On Propagation of Equational Constraints in CAD-Based Quantifier Elimination." In *Issac 2001: Proceedings of the International Symposium on Symbolic and Algebraic Computation*, 2001.

[12] Strzebonski, A. "An Algorithm for Systems of Strong Polynomial Inequalities." *The Mathematica Journal* 4, no. 4 (1994): 74-77.

[13] Strzebonski, A. "Solving Systems of Strict Polynomial Inequalities." *J. Symb. Comput.* 29, no. 3 (2000): 471-480.

[14] Strzebonski, A. "Cylindrical Algebraic Decomposition Using Validated Numerics." Paper presented at the ACA 2002 Session on Symbolic-Numerical Methods in Computational Science, Volos, Greece, 2002. (Notebook with the conference talk available at members.wolfram.com/adams).

[15] Loos, R. and V. Weispfenning. "Applying Linear Quantifier Elimination." *Comput. J.* 36, no. 5 (1993): 450-461.

[16] Strzebonski, A. "A Real Polynomial Decision Algorithm Using Arbitrary-Precision Floating Point Arithmetic." *Reliable Comput.* 5, no. 3 (1999): 337-346; *Developments in Reliable Computing* (Tibor Csendes, ed.). Kluwer Academic Publishers (1999): 337-346.

[17] Aubry, P., F. Rouillier, and M. Safey El Din. "Real Solving for Positive Dimensional Systems." *J. Symb. Comput.* 34, no. 6 (2002): 543-560.

[18] Zeng, G. X. and X. N. Zeng. "An Effective Decision Method for Semidefinite Polynomials." *J. Symb. Comput.* 37, no. 1 (2004): 83-99.

[19] Akritas, A. G. and A. Strzebonski. "A Comparative Study of Two Real Root Isolation Methods." *Nonlinear Analysis: Modelling and Control* 10, no. 4 (2005): 297-304.

[20] Faugere, J. C., P. Gianni, D. Lazard, and T. Mora. "Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering." *J. Symb. Comput.* 16, no. 4 (1993): 329-344.

[21] Dorato, P., W. Yang, and C. Abdallah. "Robust Multi-Objective Feedback Design by Quantifier Elimination." *J. Symb. Comput.* 24, no. 2 (1997): 153-159.

[22] Weispfenning, V. "Quantifier Elimination for Real Algebra—The Cubic Case." In *Issac 1994: Proceedings of the International Symposium on Symbolic and Algebraic Computation*, 1994.

[23] Weispfenning, V. "Quantifier Elimination for Real Algebra—The Quadratic Case and Beyond." *Appl. Algebra Eng. Commun. Comput.* 8, no. 2 (1997): 85-101.

[24] Strzebonski, A. "Cylindrical Algebraic Decomposition Using Validated Numerics." *J. Symb. Comput.* 41, no. 9 (2006): 1021-1038.

Diophantine Polynomial Systems

Introduction

A *Diophantine polynomial system* is an expression constructed with polynomial equations and inequalities

$$f(x_1, \ldots, x_n) == g(x_1, \ldots, x_n),\ f(x_1, \ldots, x_n) \neq g(x_1, \ldots, x_n),$$
$$f(x_1, \ldots, x_n) \geq g(x_1, \ldots, x_n),\ f(x_1, \ldots, x_n) > g(x_1, \ldots, x_n),$$
$$f(x_1, \ldots, x_n) \leq g(x_1, \ldots, x_n),\ f(x_1, \ldots, x_n) < g(x_1, \ldots, x_n),$$

combined using logical connectives and quantifiers

$$\Phi_1 \bigwedge \Phi_2,\ \Phi_1 \bigvee \Phi_2,\ \Phi_1 \Rightarrow \Phi_2,\ \neg\,\Phi,\ \forall_x \Phi,\ \text{and}\ \exists_x \Phi,$$

where the variables represent integer quantities.

An occurrence of a variable x inside $\forall_x \Phi$ or $\exists_x \Phi$ is called a *bound occurrence*; any other occurrence of x is called a *free occurrence.* A variable x is called a *free variable* of a polynomial system if the system contains a free occurrence of x. A Diophantine polynomial system is *quantifier-free* if it contains no quantifiers. A *decision problem* is a system with all variables existentially quantified, that is, a system of the form

$$\exists_{x_1} \exists_{x_2} \ldots \exists_{x_n} \Phi(x_1, \ldots, x_n), \tag{1}$$

where $x_1, \ldots, x_n$ are all variables in Φ. The decision problem (1) is equivalent to `True` or `False`, depending on whether the quantifier-free system of polynomial equations and inequalities $\Phi(x_1, \ldots, x_n)$ has integer solutions.

An example of a Diophantine polynomial system is

$$\forall_{n, n\geq 2} \exists_{p, p>1} \exists_{q, q>1} \forall_{a, a>1} \forall_{b, b>1}\ a\, b \neq p \bigwedge a\, b \neq q \bigwedge p + q = 2\, n. \tag{2}$$

Goldbach's conjecture [1], formulated in 1742 and still unproven, asserts that system (2) is equivalent to `True`. This suggests that *Mathematica* may not be able to solve arbitrary Diophantine polynomial systems. In fact, Matiyasevich's solution of Hilbert's tenth problem [2] shows that no algorithm can be constructed that would solve arbitrary Diophantine polynomial systems, not even quantifier-free systems or decision problems. Nevertheless, *Mathematica* functions `Reduce`, `Resolve`, and `FindInstance` are able to solve several reasonably large classes of Diophantine systems. This tutorial describes these classes of systems and methods used by *Mathematica* to solve them. The methods are presented in the order in which they are used. The tutorial also covers options affecting the methods used and how they operate.

Linear Systems

Systems of Linear Equations

Conjunctions of linear Diophantine equations are solvable for an arbitrary number of variables. *Mathematica* uses a method based on the computation Hermite normal form of matrices, available in *Mathematica* directly as `HermiteDecomposition`. The result may contain new unrestricted integer parameters. If the equations are independent, the number of parameters is equal to the difference between the number of variables and the number of equations.

This system has four variables and two independent equations, hence the result is expressed in terms of two integer parameters.

```
In[1]:= Reduce[3 a + 4 b + 18 c + 24 d == 30 && 27 a + 16 b + 28 c + 24 d == 30, {a, b, c, d}, Integers]

Out[1]= (C[1] | C[2]) ∈ Integers && a == 2 + 8 C[1] &&
         b == 6 - 6 C[1] + 15 C[2] && c == -12 - 12 C[1] - 18 C[2] && d == 9 + 9 C[1] + 11 C[2]
```

Frobenius Equations

A *Frobenius equation* is an equation of the form

$$a_1 x_1 + \ldots + a_n x_n == m,$$

where $a_1, \ldots, a_n$ are positive integers, m is an integer, and the coordinates $x_1, \ldots, x_n$ of solutions are required to be non-negative integers.

For finding solution instances of Frobenius equations *Mathematica* uses a fast algorithm based on the computation of the critical tree in the Frobenius graph [11]. The algorithm applies when the smallest of $a_1, \ldots, a_n$ does not exceed the value of the `MaxFrobeniusGraph` system option (the default is 1,000,000). Otherwise the more general methods for solving bounded linear systems are used. Functions `FrobeniusSolve` and `FrobeniusNumber` provide specialized function ality for solving Frobenius equations and computing Frobenius numbers.

This finds a solution of a Frobenius equation.

```
In[2]:= FindInstance[
  123 456 x + 234 567 y + 345 678 z + 456 789 u + 567 890 v + 678 901 w + 789 012 r + 890 123 s +
     901 234 t == 123 456 789 && x ≥ 0 && y ≥ 0 && z ≥ 0 && u ≥ 0 && v ≥ 0 &&
    w ≥ 0 && r ≥ 0 && s ≥ 0 && t ≥ 0, {x, y, z, u, v, w, r, s, t}, Integers]
Out[2]= {{x → 5, y → 8, z → 12, u → 17, v → 24, w → 29, r → 29, s → 29, t → 30}}
```

Bounded Systems of Linear Equations and Inequalities

If a real solution set of a system of linear equations and inequalities is a bounded polyhedron, the system has finitely many integer solutions. To find the solutions, *Mathematica* uses the following procedure.

You may assume the system has the form $M_{\text{eq}}.x == b_{\text{eq}} \bigwedge M_{\text{ineq}}.x \geq b_{\text{ineq}}$, where M_{eq} is a $k \times n$ integer matrix, b_{eq} is a length k integer vector, M_{ineq} is an $l \times n$ integer matrix, and b_{ineq} is a length l integer vector. First, the method for solving systems of linear equations is used to find an integer vector s such that $M_{\text{eq}}.s == b_{\text{eq}}$ and a $p \times n$ integer matrix N whose rows generate the nullspace of $M_{\text{eq}}.x == 0$. The integer solution set of $M_{\text{eq}}.x == b_{\text{eq}}$ is equal to $\{s + i.N : i \in \mathbb{Z}^p\}$. Put $M_{\text{mult}} = M_{\text{ineq}}.N^T$ and $b_{\text{mult}} = b_{\text{ineq}} - M_{\text{ineq}}.s$. The integer solution set of $M_{\text{eq}}.x == b_{\text{eq}} \bigwedge M_{\text{ineq}}.x \geq b_{\text{ineq}}$ is equal to $\{s + i.N : i \in \mathcal{I}\}$, where $\mathcal{I}$ is the integer solution set of $M_{\text{mult}}.i \geq b_{\text{mult}}$. To improve efficiency of finding the set $\mathcal{I}$, *Mathematica* simplifies $M_{\text{mult}}{}^T$ using `LatticeReduce`, obtaining $M_{\text{red}}{}^T$. Note that if the columns of M_{mult} are linearly dependent, $M_{\text{mult}}.i \geq b_{\text{mult}}$ has no solutions (otherwise it would have infinitely many solutions, which contradicts the assumptions). Hence you may assume that M_{mult} has linearly independent columns and so M_{red} has p columns. Put $R = \left(M_{\text{mult}}{}^T.M_{\text{mult}}\right)^{-1}\left(M_{\text{mult}}{}^T M_{\text{red}}\right)$. Lattice reduction techniques are also used to find a small vector b_{red} in the lattice $b_{\text{mult}} + M_{\text{red}}.v$. Let v_0 be such that $b_{\text{red}} = b_{\text{mult}} + M_{\text{red}}.v_0$. The set $\mathcal{I}$ can be computed from the set $\mathcal{I}_{\text{red}}$ of all $i \in \mathbb{Z}^p$ such that $M_{\text{red}}.i \geq b_{\text{red}}$ using the formula $\mathcal{I} = \{R.(i - v_0) : i \in \mathcal{I}_{\text{red}}\}$.

To find the set $\mathcal{I}_{\text{red}}$ a simple recursive algorithm can be used. The algorithm finds the bounds on the first variable using `LinearProgramming` and, for each integer value a_1 between the bounds, calls itself recursively with the first variable set to a_1. This algorithm is used when the system option BranchLinearDiophantine is set to False. With the default setting True a hybrid algorithm combining the recursive algorithm and a branch-and-bound type algorithm is used. At each level of the recursion, the recursion is continued for the "middle" values of the first variable (defined as a projection of the set of points contained in the real solution set together with a unit cube) while the remaining parts of the real solution set are searched for integer solutions using the branch-and-bound type algorithm. FindInstance finds the single element of $\mathcal{I}_{\text{red}}$ it needs using a branch-and-bound type algorithm.

There are two system options, BranchLinearDiophantine and LatticeReduceDiophantine, that allow you to control the exact algorithm used. In some cases changing the values of these options may greatly improve the performance of Reduce.

This finds all integer points in a triangle.

```
In[3]:= Reduce[7 x + y ≥ 3 && 3 x - 33 y + 333 ≥ 0 && x < y, {x, y}, Integers]

Out[3]= (x == -1 && y == 10) || (x == 0 && y == 3) || (x == 0 && y == 4) || (x == 0 && y == 5) || (x == 0 && y == 6) ||
  (x == 0 && y == 7) || (x == 0 && y == 8) || (x == 0 && y == 9) || (x == 0 && y == 10) || (x == 1 && y == 2) ||
  (x == 1 && y == 3) || (x == 1 && y == 4) || (x == 1 && y == 5) || (x == 1 && y == 6) || (x == 1 && y == 7) ||
  (x == 1 && y == 8) || (x == 1 && y == 9) || (x == 1 && y == 10) || (x == 2 && y == 3) || (x == 2 && y == 4) ||
  (x == 2 && y == 5) || (x == 2 && y == 6) || (x == 2 && y == 7) || (x == 2 && y == 8) || (x == 2 && y == 9) ||
  (x == 2 && y == 10) || (x == 3 && y == 4) || (x == 3 && y == 5) || (x == 3 && y == 6) || (x == 3 && y == 7) ||
  (x == 3 && y == 8) || (x == 3 && y == 9) || (x == 3 && y == 10) || (x == 4 && y == 5) || (x == 4 && y == 6) ||
  (x == 4 && y == 7) || (x == 4 && y == 8) || (x == 4 && y == 9) || (x == 4 && y == 10) || (x == 5 && y == 6) ||
  (x == 5 && y == 7) || (x == 5 && y == 8) || (x == 5 && y == 9) || (x == 5 && y == 10) || (x == 6 && y == 7) ||
  (x == 6 && y == 8) || (x == 6 && y == 9) || (x == 6 && y == 10) || (x == 7 && y == 8) || (x == 7 && y == 9) ||
  (x == 7 && y == 10) || (x == 8 && y == 9) || (x == 8 && y == 10) || (x == 9 && y == 10) || (x == 10 && y == 11)
```

Mathematica enumerates the solutions explicitly only if the number of integer solutions of the system does not exceed the maximum of the p^{th} power of the value of the system option DiscreteSolutionBound, where p is the dimension of the solution lattice of the equations, and the second element of the value of the system option ExhaustiveSearchMaxPoints.

Here Reduce does not give explicit solutions because their number would exceed the default limit of 10000.

```
In[4]:= Reduce[x ≥ 0 && y ≥ 0 && x + y ≤ 200, {x, y}, Integers]

Out[4]= (x | y) ∈ Integers && ((0 ≤ x ≤ 199 && 0 ≤ y ≤ 200 - x) || (x == 200 && y == 0))
```

This increases the value of the system option DiscreteSolutionBound to 1000.

```
In[5]:= SetSystemOptions["ReduceOptions" → {"DiscreteSolutionBound" → 1000}];
```

Since there are two variables and no equations, the limit on the number of solutions is now 1000^2, and Reduce can enumerate the solutions explicitly.

```
In[6]:= Reduce[x ≥ 0 && y ≥ 0 && x + y ≤ 200, {x, y}, Integers] // Length
Out[6]= 20 301
```

This resets DiscreteSolutionBound to the default value.

```
In[7]:= SetSystemOptions["ReduceOptions" → {"DiscreteSolutionBound" → 10}];
```

Arbitrary Systems of Linear Equations and Inequalities

Quantifier-free systems of linear Diophantine equations and inequalities are solvable for an arbitrary number of variables. The system is written in the *disjunctive normal form*, that is, as a disjunction of conjunctions, and each conjunction is solved separately. If a conjunction contains only equations, the method for solving systems of linear equations is used. If the difference between the number of variables and the number of equations is at most one, *Mathematica* solves the equations using the method for solving systems of linear equations, and if the solution contains at most one free parameter (which is true in the generic case), back substitutes the solution into the inequalities to determine inequality restrictions for the parameter. For all other quantifier-free systems of linear Diophantine equations and inequalities *Mathematica* uses the algorithm described in [3], with some linear-programming-based improvements for handling bounded variables. The result may contain new non-negative integer parameters, and the number of new parameters may be larger than the number of variables.

This system has three variables; however, to express the solution set, you need eight non-negative integer parameters.

```
In[8]:= Reduce[a + 2 b - 3 c == 4 && 3 a - 2 b + c ≥ 1, {a, b, c}, Integers]
Out[8]= (C[1] | C[2] | C[3] | C[4] | C[5] | C[6] | C[7] | C[8]) ∈ Integers && C[1] ≥ 0 &&
         C[2] ≥ 0 && C[3] ≥ 0 && C[4] ≥ 0 && C[5] ≥ 0 && C[6] ≥ 0 && C[7] ≥ 0 && C[8] ≥ 0 &&
         ((a == 3 + 2 C[1] + C[2] + 3 C[3] + C[4] + 2 C[5] - 2 C[6] - C[7] && b == 5 + 5 C[1] + C[2] - 2 C[4] - C[5] -
              5 C[6] - 4 C[7] - 3 C[8] && c == 3 + 4 C[1] + C[2] + C[3] - C[4] - 4 C[6] - 3 C[7] - 2 C[8]) ||
           (a == 2 + 2 C[1] + C[2] + 3 C[3] + C[4] + 2 C[5] - 2 C[6] - C[7] && b == 1 + 5 C[1] + C[2] - 2 C[4] -
              C[5] - 5 C[6] - 4 C[7] - 3 C[8] && c == 4 C[1] + C[2] + C[3] - C[4] - 4 C[6] - 3 C[7] - 2 C[8]) ||
           (a == 4 + 2 C[1] + C[2] + 3 C[3] + C[4] + 2 C[5] - 2 C[6] - C[7] && b == 5 C[1] + C[2] - 2 C[4] -
              C[5] - 5 C[6] - 4 C[7] - 3 C[8] && c == 4 C[1] + C[2] + C[3] - C[4] - 4 C[6] - 3 C[7] - 2 C[8]) ||
           (a == 1 + 2 C[1] + C[2] + 3 C[3] + C[4] + 2 C[5] - 2 C[6] - C[7] && b == 5 C[1] + C[2] - 2 C[4] -
              C[5] - 5 C[6] - 4 C[7] - 3 C[8] && c == -1 + 4 C[1] + C[2] + C[3] - C[4] - 4 C[6] - 3 C[7] - 2 C[8]) ||
           (a == -1 + 2 C[1] + C[2] + 3 C[3] + C[4] + 2 C[5] - 2 C[6] - C[7] &&
             b == -5 + 5 C[1] + C[2] - 2 C[4] - C[5] - 5 C[6] - 4 C[7] - 3 C[8] &&
             c == -5 + 4 C[1] + C[2] + C[3] - C[4] - 4 C[6] - 3 C[7] - 2 C[8]) ||
           (a == 2 C[1] + C[2] + 3 C[3] + C[4] + 2 C[5] - 2 C[6] - C[7] &&
             b == -4 + 5 C[1] + C[2] - 2 C[4] - C[5] - 5 C[6] - 4 C[7] - 3 C[8] &&
             c == -4 + 4 C[1] + C[2] + C[3] - C[4] - 4 C[6] - 3 C[7] - 2 C[8]))
```

Univariate Systems

Univariate Equations

To find integer solutions of univariate equations *Mathematica* uses a variant of the algorithm given in [4] with improvements described in [5]. The algorithm can find integer solutions of polynomials of much higher degrees than can be handled by real root isolation algorithms and with much larger free terms than can be handled by integer factorization algorithms.

Here Reduce finds integer solutions of a sparse polynomial of degree 100,000.

```
In[9]:= poly = x^100000 + 1234 x^77777 - 2121 x^12345 + 7890 x^999 - x^11;
        freeterm = poly /. x → 1234567;
        Timing[Reduce[poly - freeterm == 0, x, Integers]]
Out[11]= {5.698, x == 1234567}
```

The free term of this polynomial has 609,152 digits and cannot be easily factored.

```
In[12]:= N[freeterm]
Out[12]= 2.926904998127343 × 10^609151
```

```
In[13]:= TimeConstrained[FactorInteger[freeterm] // Timing, 1000]
Out[13]= $Aborted
```

Systems of Univariate Equations and Inequalities

Systems of univariate Diophantine equations and inequalities are written in the disjunctive normal form, and each conjunction is solved separately. If a conjunction contains an equation, the method for solving univariate equations is used, and the solutions satisfying the remaining equations and inequalities are selected.

Here Reduce finds integer solutions of $x^4 - 25\,x^2 = -144$ and selects the ones that satisfy the inequality $x^{100001} - 27\,x + 5 \geq 0$.

```
In[14]:= Reduce[x^4 - 25 x^2 == -144 && x^100001 - 27 x + 5 ≥ 0, x, Integers]
Out[14]= x == 3 || x == 4
```

Conjunctions containing only inequalities are solved over the reals. Integer solutions in the resulting real intervals are given explicitly if their number in the given interval does not exceed the value of the system option `DiscreteSolutionBound`. The default value of the option is 10. For intervals containing more integer solutions, the solutions are represented implicitly.

Bivariate Systems

Quadratic Equations

Mathematica can solve arbitrary quadratic Diophantine equations in two variables. The general form of such an equation is

$$\Phi(x, y) = a\,x^2 + b\,x\,y + c\,y^2 + d\,x + e\,y + f == 0. \tag{1}$$

If $\Phi(x, y) = \Phi_1(x, y)\,\Phi_2(x, y)$, where $\Phi_1(x, y)$ and $\Phi_2(x, y)$ are linear polynomials, the equation (1) is equivalent to $\Phi_1(x, y) = 0 \bigvee \Phi_2(x, y) = 0$, and methods for solving linear Diophantine equations are used. For irreducible polynomials $\Phi(x, y)$, the algorithms used and the form of the result depend on the determinant $\Delta = b^2 - 4\,a\,c$ of the quadratic form. The algorithms may use integer factorization and hence the correctness of the results depends on the correctness of the probabilistic primality test used by `PrimeQ`.

Hyperbolic Type Equations with Square Determinants

If $\Delta > 0$ and $\sqrt{\Delta}$ is an integer, then $\Delta\,\Phi(x, y) - g = \Phi_1(x, y)\,\Phi_2(x, y)$, where $\Phi_1(x, y)$ and $\Phi_2(x, y)$ are linear polynomials and $g = c\,d^2 + a\,e^2 + b^2\,f - b\,d\,e - 4\,a\,c\,f$. In this case, the equation (1) is equivalent to the disjunction of linear systems $\Phi_1(x, y) = \delta \bigwedge \Phi_2(x, y) = -g/\delta$, for all divisors δ of g. Each of the linear systems has one solution over the rationals, hence the equation (1) has a finite number of integer solutions.

Here is a binary quadratic equation with $\Delta = 9$.

```
In[15]:= Reduce[1 + 12 x + 2 x^2 + 7 y + 5 x y + 2 y^2 == 0, {x, y}, Integers]
Out[15]= (x == -4 && y == -1) || (x == 2 && y == -3) || (x == 4 && y == -9)
```

Hyperbolic Type Equations with Nonsquare Determinants

If $\Delta > 0$ and $\sqrt{\Delta}$ is not an integer, then the equation (1) is a Pell-type equation. Methods for solving such equations have been developed since the 18th century and can be constructed based on [6] and [7] (though these books do not contain a complete description of the algorithm). The solution set is empty or infinite, parametrized by an integer parameter appearing in the exponent.

A Pell equation is an equation of the form $x^2 - D\,y^2 == 1$, where D is not a square. Solutions to Pell equations with small coefficients can be quite complicated.

In[16]:= **Reduce[x^2 - 61 y^2 == 1, {x, y}, Integers]**

Out[16]=

$$\left(C[1] \in \text{Integers}\ \&\&\ C[1] \geq 0\ \&\&\ x == \frac{1}{2}\left(-\left(1\,766\,319\,049 - 226\,153\,980\sqrt{61}\right)^{C[1]} - \left(1\,766\,319\,049 + 226\,153\,980\sqrt{61}\right)^{C[1]}\right)\ \&\&\ y == -\frac{1}{2\sqrt{61}}\left(\left(1\,766\,319\,049 - 226\,153\,980\sqrt{61}\right)^{C[1]} - \left(1\,766\,319\,049 + 226\,153\,980\sqrt{61}\right)^{C[1]}\right)\right)\ ||$$

$$\left(C[1] \in \text{Integers}\ \&\&\ C[1] \geq 0\ \&\&\ x == \frac{1}{2}\left(-\left(1\,766\,319\,049 - 226\,153\,980\sqrt{61}\right)^{C[1]} - \left(1\,766\,319\,049 + 226\,153\,980\sqrt{61}\right)^{C[1]}\right)\ \&\&\ y == \frac{1}{2\sqrt{61}}\left(\left(1\,766\,319\,049 - 226\,153\,980\sqrt{61}\right)^{C[1]} - \left(1\,766\,319\,049 + 226\,153\,980\sqrt{61}\right)^{C[1]}\right)\right)\ ||$$

$$\left(C[1] \in \text{Integers}\ \&\&\ C[1] \geq 0\ \&\&\ x == \frac{1}{2}\left(\left(1\,766\,319\,049 - 226\,153\,980\sqrt{61}\right)^{C[1]} + \left(1\,766\,319\,049 + 226\,153\,980\sqrt{61}\right)^{C[1]}\right)\ \&\&\ y == -\frac{1}{2\sqrt{61}}\left(\left(1\,766\,319\,049 - 226\,153\,980\sqrt{61}\right)^{C[1]} - \left(1\,766\,319\,049 + 226\,153\,980\sqrt{61}\right)^{C[1]}\right)\right)\ ||$$

$$\left(C[1] \in \text{Integers}\ \&\&\ C[1] \geq 0\ \&\&\ x == \frac{1}{2}\left(\left(1\,766\,319\,049 - 226\,153\,980\sqrt{61}\right)^{C[1]} + \left(1\,766\,319\,049 + 226\,153\,980\sqrt{61}\right)^{C[1]}\right)\ \&\&\ y == \frac{1}{2\sqrt{61}}\left(\left(1\,766\,319\,049 - 226\,153\,980\sqrt{61}\right)^{C[1]} - \left(1\,766\,319\,049 + 226\,153\,980\sqrt{61}\right)^{C[1]}\right)\right)$$

Here is the solution of a Pell-type equation with $\Delta = 5$.

In[17]:= **sol = Reduce[7 + 5 x + x^2 + 7 y + 3 x y + y^2 == 0, {x, y}, Integers]**

Out[17]=

$$\Bigg(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x == \frac{1}{10}\left(5\left(-5 - \frac{2\left(\left(9 - 4\sqrt{5}\right)^{1+2\,C[1]} - \left(9 + 4\sqrt{5}\right)^{1+2\,C[1]}\right)}{\sqrt{5}}\right) + \right.$$

$$\left. 3\left(1 - 2\left(\left(9 - 4\sqrt{5}\right)^{1+2\,C[1]} + \left(9 + 4\sqrt{5}\right)^{1+2\,C[1]}\right)\right)\right) \;\&\&$$

$$y == \frac{1}{5}\left(-1 + 2\left(\left(9 - 4\sqrt{5}\right)^{1+2\,C[1]} + \left(9 + 4\sqrt{5}\right)^{1+2\,C[1]}\right)\right)\Bigg) \;||$$

$$\Bigg(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x == \frac{1}{10}\left(5\left(-5 + \frac{2\left(\left(9 - 4\sqrt{5}\right)^{1+2\,C[1]} - \left(9 + 4\sqrt{5}\right)^{1+2\,C[1]}\right)}{\sqrt{5}}\right) + \right.$$

$$\left. 3\left(1 - 2\left(\left(9 - 4\sqrt{5}\right)^{1+2\,C[1]} + \left(9 + 4\sqrt{5}\right)^{1+2\,C[1]}\right)\right)\right) \;\&\&$$

$$y == \frac{1}{5}\left(-1 + 2\left(\left(9 - 4\sqrt{5}\right)^{1+2\,C[1]} + \left(9 + 4\sqrt{5}\right)^{1+2\,C[1]}\right)\right)\Bigg) \;||\; \Bigg(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10}\left(5\left(-5 - \frac{2\left(\left(9 - 4\sqrt{5}\right)^{2\,C[1]} - \left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)}{\sqrt{5}}\right) + 3\left(1 + 2\left(\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + \left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)\right)\right) \;\&\&$$

$$y == \frac{1}{5}\left(-1 - 2\left(\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + \left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)\right)\Bigg) \;||\; \Bigg(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10}\left(5\left(-5 + \frac{2\left(\left(9 - 4\sqrt{5}\right)^{2\,C[1]} - \left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)}{\sqrt{5}}\right) + 3\left(1 + 2\left(\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + \left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)\right)\right) \;\&\&$$

$$y == \frac{1}{5}\left(-1 - 2\left(\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + \left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)\right)\Bigg) \;||\; \Bigg(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10}\left(3\left(1 - 3\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + \sqrt{5}\left(9 - 4\sqrt{5}\right)^{2\,C[1]} - 3\left(9 + 4\sqrt{5}\right)^{2\,C[1]} - \sqrt{5}\left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right) + 5\left(-5 + \right.\right.$$

$$\left.\left.\frac{1}{5}\left(-5\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + 3\sqrt{5}\left(9 - 4\sqrt{5}\right)^{2\,C[1]} - 5\left(9 + 4\sqrt{5}\right)^{2\,C[1]} - 3\sqrt{5}\left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)\right)\right) \;\&\&$$

$$y == \frac{1}{5}\left(-1 + 3\left(9 - 4\sqrt{5}\right)^{2\,C[1]} - \sqrt{5}\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + 3\left(9 + 4\sqrt{5}\right)^{2\,C[1]} + \sqrt{5}\left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)\Bigg) \;||$$

$$\Bigg(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10}\left(3\left(1 - 3\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + \sqrt{5}\left(9 - 4\sqrt{5}\right)^{2\,C[1]} - 3\left(9 + 4\sqrt{5}\right)^{2\,C[1]} - \sqrt{5}\left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right) + 5\left(-5 + \right.\right.$$

$$\left.\left.\frac{1}{5}\left(5\left(9 - 4\sqrt{5}\right)^{2\,C[1]} - 3\sqrt{5}\left(9 - 4\sqrt{5}\right)^{2\,C[1]} + 5\left(9 + 4\sqrt{5}\right)^{2\,C[1]} + 3\sqrt{5}\left(9 + 4\sqrt{5}\right)^{2\,C[1]}\right)\right)\right) \;\&\&$$

$$y == \frac{1}{5} \left(-1 + 3 \left(9 - 4 \sqrt{5}\right)^{2 C[1]} - \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{2 C[1]} + 3 \left(9 + 4 \sqrt{5}\right)^{2 C[1]} + \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{2 C[1]}\right)\right) \; ||$$

$$\Big(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10} \Big(3 \left(1 + 3 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} + \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} + 3 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} - \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\right) +$$

$$5 \Big(-5 + \frac{1}{5} \Big(5 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} + 3 \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} +$$

$$5 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} - 3 \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\Big)\Big)\Big) \;\&\&$$

$$y == \frac{1}{5} \left(-1 - 3 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} - \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} - 3 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} + \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\right)\Big) \; ||$$

$$\Big(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10} \Big(3 \left(1 + 3 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} + \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} + 3 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} - \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\right) +$$

$$5 \Big(-5 + \frac{1}{5} \Big(-5 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} - 3 \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} -$$

$$5 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} + 3 \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\Big)\Big)\Big) \;\&\&$$

$$y == \frac{1}{5} \left(-1 - 3 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} - \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} - 3 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} + \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\right)\Big) \; ||$$

$$\Big(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10} \Big(3 \left(1 - 3 \left(9 - 4 \sqrt{5}\right)^{2 C[1]} - \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{2 C[1]} - 3 \left(9 + 4 \sqrt{5}\right)^{2 C[1]} + \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{2 C[1]}\right) + 5 \Big(-5 +$$

$$\frac{1}{5} \left(5 \left(9 - 4 \sqrt{5}\right)^{2 C[1]} + 3 \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{2 C[1]} + 5 \left(9 + 4 \sqrt{5}\right)^{2 C[1]} - 3 \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{2 C[1]}\right)\Big)\Big) \;\&\&$$

$$y == \frac{1}{5} \left(-1 + 3 \left(9 - 4 \sqrt{5}\right)^{2 C[1]} + \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{2 C[1]} + 3 \left(9 + 4 \sqrt{5}\right)^{2 C[1]} - \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{2 C[1]}\right)\Big) \; ||$$

$$\Big(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10} \Big(3 \left(1 - 3 \left(9 - 4 \sqrt{5}\right)^{2 C[1]} - \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{2 C[1]} - 3 \left(9 + 4 \sqrt{5}\right)^{2 C[1]} + \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{2 C[1]}\right) + 5 \Big(-5 +$$

$$\frac{1}{5} \left(-5 \left(9 - 4 \sqrt{5}\right)^{2 C[1]} - 3 \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{2 C[1]} - 5 \left(9 + 4 \sqrt{5}\right)^{2 C[1]} + 3 \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{2 C[1]}\right)\Big)\Big) \;\&\&$$

$$y == \frac{1}{5} \left(-1 + 3 \left(9 - 4 \sqrt{5}\right)^{2 C[1]} + \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{2 C[1]} + 3 \left(9 + 4 \sqrt{5}\right)^{2 C[1]} - \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{2 C[1]}\right)\Big) \; ||$$

$$\Big(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10} \Big(3 \left(1 + 3 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} - \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} + 3 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} + \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\right) +$$

$$5 \Big(-5 + \frac{1}{5} \Big(-5 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} + 3 \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} -$$

$$5 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} - 3 \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\Big)\Big)\Big) \;\&\&$$

$$y == \frac{1}{5} \left(-1 - 3 \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} + \sqrt{5} \left(9 - 4 \sqrt{5}\right)^{1+2 C[1]} - 3 \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]} - \sqrt{5} \left(9 + 4 \sqrt{5}\right)^{1+2 C[1]}\right)\Big) \; ||$$

$$\Big(C[1] \in \text{Integers} \;\&\&\; C[1] \geq 0 \;\&\&\; x ==$$

$$\frac{1}{10}\left(3\left(1+3\left(9-4\sqrt{5}\right)^{1+2\,C[1]}-\sqrt{5}\left(9-4\sqrt{5}\right)^{1+2\,C[1]}+3\left(9+4\sqrt{5}\right)^{1+2\,C[1]}+\sqrt{5}\left(9+4\sqrt{5}\right)^{1+2\,C[1]}\right)+5\left(-5+\frac{1}{5}\left(5\left(9-4\sqrt{5}\right)^{1+2\,C[1]}-3\sqrt{5}\left(9-4\sqrt{5}\right)^{1+2\,C[1]}+5\left(9+4\sqrt{5}\right)^{1+2\,C[1]}+3\sqrt{5}\left(9+4\sqrt{5}\right)^{1+2\,C[1]}\right)\right)\right)\ \&\&$$

$$y == \frac{1}{5}\left(-1-3\left(9-4\sqrt{5}\right)^{1+2\,C[1]}+\sqrt{5}\left(9-4\sqrt{5}\right)^{1+2\,C[1]}-3\left(9+4\sqrt{5}\right)^{1+2\,C[1]}-\sqrt{5}\left(9+4\sqrt{5}\right)^{1+2\,C[1]}\right)\right)$$

Even though the solutions are expressed using nonrational numbers, they are in fact integers, as they should be.

In[18]:= **Simplify[sol /. C[1] → 7]**

```
Out[18]= (y == -143 556 140 002 351 233 && (x == 375 834 853 819 893 935 || x == 54 833 566 187 159 759)) ||
  (y == 375 834 853 819 893 937 && (x == -983 948 421 457 330 581 || x == -143 556 140 002 351 235)) ||
  (y == 54 833 566 187 159 761 && (x == -143 556 140 002 351 235 || x == -20 944 558 559 128 053)) ||
  (y == 2 576 010 410 552 097 799 && (x == -983 948 421 457 330 581 || x == -6 744 082 810 198 962 821)) ||
  (y == -6 744 082 810 198 962 819 && (x == 2 576 010 410 552 097 797 || x == 17 656 238 020 044 790 655)) ||
  (y == -983 948 421 457 330 579 && (x == 375 834 853 819 893 935 || x == 2 576 010 410 552 097 797))
```

`Reduce` can solve systems consisting of a Pell-type equation and inequalities giving simple bounds on variables.

In[19]:= **Reduce[x^2 - 3 y^2 == 22 && 0 ≤ y ≤ 1 000 000, {x, y}, Integers]**

```
Out[19]= (x == -856 487 && y == 494 493) || (x == -472 765 && y == 272 951) || (x == -229 495 && y == 132 499) ||
  (x == -126 677 && y == 73 137) || (x == -61 493 && y == 35 503) || (x == -33 943 && y == 19 597) ||
  (x == -16 477 && y == 9513) || (x == -9095 && y == 5251) || (x == -4415 && y == 2549) ||
  (x == -2437 && y == 1407) || (x == -1183 && y == 683) || (x == -653 && y == 377) || (x == -317 && y == 183) ||
  (x == -175 && y == 101) || (x == -85 && y == 49) || (x == -47 && y == 27) || (x == -23 && y == 13) ||
  (x == -13 && y == 7) || (x == -7 && y == 3) || (x == -5 && y == 1) || (x == 5 && y == 1) || (x == 7 && y == 3) ||
  (x == 13 && y == 7) || (x == 23 && y == 13) || (x == 47 && y == 27) || (x == 85 && y == 49) ||
  (x == 175 && y == 101) || (x == 317 && y == 183) || (x == 653 && y == 377) || (x == 1183 && y == 683) ||
  (x == 2437 && y == 1407) || (x == 4415 && y == 2549) || (x == 9095 && y == 5251) || (x == 16 477 && y == 9513) ||
  (x == 33 943 && y == 19 597) || (x == 61 493 && y == 35 503) || (x == 126 677 && y == 73 137) ||
  (x == 229 495 && y == 132 499) || (x == 472 765 && y == 272 951) || (x == 856 487 && y == 494 493)
```

Parabolic Type Equations

If $\Delta = 0$, set $g = sign(a)\, gcd(a, c)$, $a_1 = \sqrt{a/g}$, and $c_1 = sign(b/g)\sqrt{c/g}$. Since $b^2 = 4\, g^2 (a/g)(c/g)$, a_1 and c_1 are nonzero integers, and $b = 2\, g\, a_1\, c_1$. Then

$$\Phi(x, y) = g(a_1\, x + c_1\, y)^2 + d\, x + e\, y + f.$$

Set $m = c_1\, d - a_1\, e$ and $t = a_1\, x + c_1\, y$. Then the equation (1) is equivalent to

$$a_1\, \Phi(x, y) = a_1\, g(a_1\, x + c_1\, y)^2 + d(a_1\, x + c_1\, y) - m\, y + a_1\, f = a_1\, g\, t^2 + d\, t - m\, y + a_1\, f == 0. \tag{2}$$

Suppose $m = 0$. If the equation (1) had integer solutions, $a_1\, g\, t^2 + d\, t + a_1\, f = 0$ would have integer solutions in t, and so $\Phi(x, y)$ would be a product of two linear polynomials. Since here $\Phi(x, y)$ is irreducible, the equation (1) has no integer solutions.

If $m \neq 0$, then the equation (2) implies

$$a_1 g t^2 + d t + a_1 f \equiv 0 \ (mod \ |m|). \tag{3}$$

If the modular equation (3) has no solutions in t, the equation (1) has no integer solutions. (If $|m| = 1$, the modular equation (3) has one solution, $t = 0$.) Otherwise $t = u + k m$, for some solution $0 \leq u < |m|$ of the modular equation (3). Replacing $t \to u + k m$ in the equation (2) and solving the resulting linear equation for y gives

$$y == a_1 g m k^2 + (d + 2 a_1 g u) k + \left(a_1 g u^2 + d u + a_1 f\right) \big/ m. \tag{4}$$

Note that since u satisfies the modular equation (3), the division in the last term of (4) gives an integer result. Since $t = a_1 x + c_1 y$ and $t = u + k m$, $x = (u + k m - c_1 y)/a_1$. Taking the equation (4) and the fact that $m = c_1 d - a_1 e$ into account gives

$$x == -c_1 g m k^2 - (e + 2 c_1 g u) k - \left(c_1 g u^2 + e u + c_1 f\right) \big/ m. \tag{5}$$

Therefore, the full solution of the equation (1) of parabolic type consists of one-parameter solution families given by equations (4) and (5) for each solution u of the modular equation (3), for which $\left(c_1 g u^2 + e u + c_1 f\right) \big/ m$ is an integer.

Here `Reduce` finds integer solutions of a quadratic equation of the parabolic type.

```
In[20]:= Reduce[x^2 - 2 x y + y^2 + 5 x - 7 y == 22, {x, y}, Integers]
Out[20]= (C[1] ∈ Integers && x == -11 + 7 C[1] + 2 C[1]^2 && y == -11 + 5 C[1] + 2 C[1]^2) ||
         (C[1] ∈ Integers && x == -7 + 9 C[1] + 2 C[1]^2 && y == -8 + 7 C[1] + 2 C[1]^2)
```

Elliptic Type Equations

If $\Delta < 0$, the solutions of equation (1) are integer points on an ellipse. Since an ellipse is a bounded set, the number of solutions must be finite. An obvious algorithm for finding integer points would be to compute the solutions for y for each of the finite number of possible integer values of x. This, however, would be prohibitively slow for larger ellipses. *Mathematica* uses a faster algorithm described in [8].

Here `Reduce` finds positive integer solutions of a quadratic equation of the elliptic type. There are more than 8×10^{18} possible positive integer values of x, so the obvious algorithm would not be practical for this ellipse.

```
In[21]:= Reduce[23 x^2 + 17 y^2 == 1 693 339 429 465 935 072 912 802 926 367 922 572 800 && x > 0 && y > 0,
           {x, y}, Integers]
Out[21]= (x == 1 234 567 890 987 654 321 && y == 9 876 543 210 123 456 789) ||
          (x == 2 394 388 915 976 549 628 && y == 9 583 927 013 507 483 052) ||
          (x == 3 587 688 774 846 621 081 && y == 9 066 079 904 941 225 629) ||
          (x == 4 628 858 032 573 225 308 && y == 8 403 549 375 095 756 172) ||
          (x == 4 730 542 803 202 013 073 && y == 8 326 586 121 887 736 693) ||
          (x == 6 448 688 263 945 408 950 && y == 6 583 719 143 723 572 530) ||
          (x == 6 563 464 511 756 847 993 && y == 6 428 433 631 978 684 413) ||
          (x == 7 787 179 624 084 878 150 && y == 4 191 136 305 399 154 530)
```

Thue Equations

A *Thue* equation is a Diophantine equation of the form

$$\Phi(x, y) = m,$$

where $\Phi(x, y)$ is an irreducible homogenous form of degree ≥ 3.

The number of solutions of Thue equations is always finite. *Mathematica* can in principle solve arbitrary Thue equations, though the time necessary to find the solutions lengthens very fast with degree and coefficient size. The hardest part of the algorithm is computing a bound on the size of solutions. *Mathematica* uses an algorithm based on the Baker-Wustholz theorem to find the bound [9]. If the input contains inequalities that provide a reasonable size bound on solutions, *Mathematica* can compute the solutions much faster.

This finds integer solutions of a cubic Thue equation.

```
In[22]:= Reduce[x^3 - 4 x y^2 + y^3 == 1, {x, y}, Integers] // Timing
Out[22]= {0.621, (x == -2 && y == 1) || (x == 0 && y == 1) ||
            (x == 1 && y == 0) || (x == 1 && y == 4) || (x == 2 && y == 1) || (x == 508 && y == 273)}
```

If we give `Reduce` a bound on the size of solutions, it can solve the equation much faster.

```
In[23]:= Reduce[x^3 - 4 x y^2 + y^3 == 1 && -10^10 < x < 10^10, {x, y}, Integers] // Timing
Out[23]= {0.05, (x == -2 && y == 1) || (x == 0 && y == 1) ||
            (x == 1 && y == 0) || (x == 1 && y == 4) || (x == 2 && y == 1) || (x == 508 && y == 273)}
```

Here `Reduce` finds the only solution of a degree 15 Thue equation with at most a 100-digit x coordinate. Without the bound on the solution size, `Reduce` did not finish in 1000 seconds.

```
In[24]:= Reduce[
           x^15 - 4 x^12 y^3 + 7 x^7 y^8 - 2 y^15 == 23 058 325 506 004 605 670 097 246 320 963 935 108 919 550 &&
            -10^100 < x < 10^100, {x, y}, Integers] // Timing
Out[24]= {12.36, x == 777 && y == -121}
```

Multivariate Nonlinear Systems

Systems Solvable with the Modular Sieve Method

Mathematica uses a variant of the modular sieve method (see e.g. [9]). The method may prove that a system has no solutions in integers modulo an integer m, and therefore, it has no integer solutions. Otherwise, it may find a solution with small integer coordinates or prove that the system has no integer solutions with all coordinates between $-b$ and b. The number of candidate solution points that the sieve method is allowed to test is controlled by the system option `SieveMaxPoints`.

This equation has no solutions modulo 2.

```
In[25]:= Reduce[-2 x^3 y^9 + 6 x^5 y^5 z^2 + 6 x^8 y^2 z^5 + 4 x^7 y^6 z^7 == 7, {x, y, z}, Integers]
Out[25]= False
```

Here `FindInstance` finds a small solution using the modular sieve.

```
In[26]:= FindInstance[9 x^6 y^8 z - 81 x^2 y^9 z - 5 x^9 y^5 z^5 + 2 x^6 y^2 z^9 == 1080, {x, y, z}, Integers]
Out[26]= {{x → 1, y → 2, z → 3}}
```

Systems with More Than One Equation

If a Diophantine polynomial system contains more than one equation, *Mathematica* uses `GroebnerBasis` in an attempt to reduce the problem to a sequence of simpler problems.

Systems Solvable by Recursion over Finitely Many Partial Solutions

Mathematica attempts to solve an element of the Gröbner basis that depends on the minimal number of the initial variables. If the number of solutions is finite, then for each solution *Mathematica* substitutes the computed values and attempts to solve the obtained system for the remaining variables.

Here the first equation has four integer solutions for x and y. For each of the solutions, the second equation becomes a univariate equation in z. The four univariate equations have a total of two integer solutions.

```
In[27]:= Reduce[x^4 - x y + y^2 == 7 && z^4 - x z + y^2 + y == 2400, {x, y, z}, Integers]
Out[27]= (x == -1 && y == -3 && z == -7) || (x == -1 && y == 2 && z == -7)
```

Here the first equation is a Thue equation with one solution. After replacing x and y with the values computed from the first equation, the second equation becomes a Pell equation.

```
In[28]:= Reduce[x^3 - 2 y^3 == 11 && z^2 - x y w^2 == 1 && z > 0 && w > 0, {x, y, z, w}, Integers]
Out[28]= x == 3 && y == 2 && C[1] ∈ Integers && C[1] ≥ 1 &&
```

$$z == \frac{1}{2}\left(\left(5-2\sqrt{6}\right)^{C[1]}+\left(5+2\sqrt{6}\right)^{C[1]}\right) \;\&\&\; w == -\frac{\left(5-2\sqrt{6}\right)^{C[1]}-\left(5+2\sqrt{6}\right)^{C[1]}}{2\sqrt{6}}$$

Systems with Linear-Triangular Gröbner Bases

This heuristic applies to systems with Gröbner bases of the form

$$\{c_1\,x_1 - f_1(Y),\ \ldots,\ c_k\,x_k - f_k(Y),\ g(Y)\}.$$

In this case, *Mathematica* solves the system of congruences

$$f_1(Y) \equiv 0 \bmod c_1 \wedge \ldots \wedge f_k(Y) \equiv 0 \bmod c_k. \tag{1}$$

The solutions are represented using new integer parameters. These are substituted into the equation $g(Y) == 0$ and the inequalities present in the original system, and *Mathematica* attempts to solve the so-obtained systems for the new parameters.

This system reduces to solving a congruence and a Pell equation.

In[29]:=
```
Reduce[x^2 - 7 y^2 == 1 && 2 z == x^3 - 1 && t - 4 z^2 + y == 7 && x > 0 && y > 0,
  {x, y, z, t}, Integers]
```

Out[29]=

$$C[1] \in \text{Integers} \,\&\&\, C[1] \geq 1 \,\&\&\, x == 1 + \frac{1}{4}\left(-4 + 2\left(\left(127 - 48\sqrt{7}\right)^{C[1]} + \left(127 + 48\sqrt{7}\right)^{C[1]}\right)\right) \,\&\&$$

$$y == -\frac{\left(127 - 48\sqrt{7}\right)^{C[1]} - \left(127 + 48\sqrt{7}\right)^{C[1]}}{2\sqrt{7}} \,\&\&$$

$$z == \frac{1}{2}\left(-1 + \left(1 + \frac{1}{4}\left(-4 + 2\left(\left(127 - 48\sqrt{7}\right)^{C[1]} + \left(127 + 48\sqrt{7}\right)^{C[1]}\right)\right)\right)^3\right) \,\&\&\, t == 8 - 2x^3 + x^6 - y$$

This system reduces to solving a system of two congruences and a quadratic Diophantine equation of the parabolic type for each family of congruence solutions.

In[30]:=
```
Reduce[3 z == x^2 - 2 x y && 2 t == x^3 + 96 z^2 - 1 && (x - 2 y)^2 - 3 x == 18, {x, y, z, t}, Integers]
```

Out[30]=

$$\Big(C[1] \in \text{Integers} \,\&\&\, x == 3 + 6\left(-1 + 4C[1] + 8C[1]^2\right) \,\&\&\, y == 3 + 6\left(-1 + C[1] + 4C[1]^2\right) \,\&\&$$

$$z == \frac{1}{3}\left(-2\left(3 + 6\left(-1 + C[1] + 4C[1]^2\right)\right)\left(3 + 6\left(-1 + 4C[1] + 8C[1]^2\right)\right) + \left(3 + 6\left(-1 + 4C[1] + 8C[1]^2\right)\right)^2\right) \,\&\&$$

$$t == \frac{1}{2}\left(-1 + 192\left(3 + 6\left(-1 + 4C[1] + 8C[1]^2\right)\right)^2 + 33\left(3 + 6\left(-1 + 4C[1] + 8C[1]^2\right)\right)^3\right)\Big) \,||$$

$$\Big(C[1] \in \text{Integers} \,\&\&\, x == 3 + 6\left(3 + 12C[1] + 8C[1]^2\right) \,\&\&\, y == 6\left(1 + 5C[1] + 4C[1]^2\right) \,\&\&$$

$$z == \frac{1}{3}\left(-12\left(1 + 5C[1] + 4C[1]^2\right)\left(3 + 6\left(3 + 12C[1] + 8C[1]^2\right)\right) + \left(3 + 6\left(3 + 12C[1] + 8C[1]^2\right)\right)^2\right) \,\&\&$$

$$t == \frac{1}{2}\left(-1 + 192\left(3 + 6\left(3 + 12C[1] + 8C[1]^2\right)\right)^2 + 33\left(3 + 6\left(3 + 12C[1] + 8C[1]^2\right)\right)^3\right)\Big)$$

Sums of Squares

Mathematica can solve equations of the form

$$x_1^{\,2} + x_2^{\,2} + \ldots + x_n^{\,2} == m \qquad (2)$$

using the algorithm described in [10]. For multivariate quadratic equations, *Mathematica* attempts to find an affine transformation that puts the equation in the form (2). A heuristic method based on `CholeskyDecomposition` is used for this purpose. However, the method may fail for some equations that can be represented in the form (2).

This solves a sum-of-squares equation in three variables.

```
In[31]:= Reduce[(x - 2 y + 3 z)^2 + (4 y + 5 z)^2 + z^2 == 14, {x, y, z}, Integers]
Out[31]= (x == -19 && y == -4 && z == 3) || (x == -15 && y == -4 && z == 3) ||
  (x == -9 && y == -2 && z == 1) || (x == -5 && y == -2 && z == 1) || (x == 5 && y == 2 && z == -1) ||
  (x == 9 && y == 2 && z == -1) || (x == 15 && y == 4 && z == -3) || (x == 19 && y == 4 && z == -3)
```

To find a single solution of (2) `FindInstance` uses an algorithm based on [12].

This finds a decomposition of a 10000-digit integer into a sum of seven squares. `N` is applied to make the printed result smaller.

```
In[32]:= SeedRandom[10]; a = RandomInteger[{0, 10^10000}];
         N[s7 = FindInstance[x^2 + y^2 + z^2 + t^2 + u^2 + v^2 + w^2 == a, {x, y, z, t, u, v, w}, Integers]] //
          Timing
Out[32]= {6.529, {{x → 4.654783993889879×10^4999, y → 2.728258415849877×10^2499,
    z → 3.456850125804598×10^1249, t → 4.532687928125587×10^624, u → 3.523387016717428×10^624,
    v → 3.170130382788626×10^624, w → 1.713114815166737×10^624}}}
```

This proves that the decomposition found is correct.

```
In[33]:= x^2 + y^2 + z^2 + t^2 + u^2 + v^2 + w^2 - a /. s7
Out[33]= {0}
```

Pythagorean Equation

Mathematica knows the solution to the Pythagorean equation

$$x^2 + y^2 == z^2.$$

This gives the general solution of the Pythagorean equation.

```
In[34]:= Reduce[x^2 + y^2 == z^2, {x, y, z}, Integers]
Out[34]= (C[1] | C[2] | C[3]) ∈ Integers && C[3] ≥ 0 &&
  ((x == C[1] (C[2]^2 - C[3]^2) && y == 2 C[1] C[2] C[3] && z == C[1] (C[2]^2 + C[3]^2)) ||
    (x == 2 C[1] C[2] C[3] && y == C[1] (C[2]^2 - C[3]^2) && z == C[1] (C[2]^2 + C[3]^2)))
```

For quadratic equations in three variables, *Mathematica* attempts to find a transformation of the form

$$x_1 = x + a\,y + b\,z + c,$$
$$y_1 = y + d\,z + e,$$
$$z_1 = z + f,$$

transforming the equation to the Pythagorean equation.

This equation can be transformed to the Pythagorean equation.

```
In[35]:= Reduce[-4 x + 5 x^2 - 2 y + 4 x y + y^2 + 28 x z + 6 y z + 72 z^2 == 8, {x, y, z}, Integers]
Out[35]= (C[1] | C[2] | C[3]) ∈ Integers && C[3] ≥ 0 &&
  ((x == 2 C[1] C[2] C[3] - 8 (3 + C[1] (C[2]^2 + C[3]^2)) && y == 1 + C[1] (C[2]^2 - C[3]^2) -
       3 (3 + C[1] (C[2]^2 + C[3]^2)) - 2 (2 C[1] C[2] C[3] - 8 (3 + C[1] (C[2]^2 + C[3]^2))) &&
     z == 3 + C[1] (C[2]^2 + C[3]^2)) || (x == C[1] (C[2]^2 - C[3]^2) - 8 (3 + C[1] (C[2]^2 + C[3]^2)) &&
     y == 1 + 2 C[1] C[2] C[3] - 3 (3 + C[1] (C[2]^2 + C[3]^2)) -
       2 (C[1] (C[2]^2 - C[3]^2) - 8 (3 + C[1] (C[2]^2 + C[3]^2))) && z == 3 + C[1] (C[2]^2 + C[3]^2)))
```

Equations with Reducible Nonconstant Parts

If the sum of nonconstant terms in an equation factors, *Mathematica* uses the formula

$$f\, g = c \Longleftrightarrow \bigvee_{d|c} f = d \bigwedge g = c/d$$

to reduce the equation to a disjunction of pairs of equations with lower degrees. Note that this reduction depends on the ability to find all divisors of c, hence the correctness of the results depends on the correctness of the probabilistic primality test used by `PrimeQ`.

This cubic equation reduces to 12 pairs of quadratic and linear equations.

```
In[36]:= Reduce[(x - 2 y + 3 z) x^2 - (x - 2 y + 3 z) y z == 18, {x, y, z}, Integers]
Out[36]= (x == -71 && y == -112 && z == -45) || (x == -55 && y == -82 && z == -37) ||
  (x == -53 && y == -80 && z == -35) || (x == -11 && y == 8 && z == 15) ||
  (x == -9 && y == -12 && z == -7) || (x == -9 && y == 8 && z == 9) || (x == -9 && y == 10 && z == 9) ||
  (x == -3 && y == -6 && z == -1) || (x == -1 && y == -2 && z == 1) || (x == -1 && y == 4 && z == 1) ||
  (x == 6 && y == -6 && z == -7) || (x == 6 && y == -6 && z == -5) || (x == 13 && y == -10 && z == -17) ||
  (x == 34 && y == 50 && z == 23) || (x == 38 && y == 58 && z == 25) || (x == 83 && y == 130 && z == 53)
```

Equations with a Linear Variable

Mathematica attempts to solve Diophantine systems of the form

$$f(x_1, \ldots, x_n)\, y + g(x_1, \ldots, x_n) == 0 \bigwedge \Phi(x_1, \ldots, x_n, y),$$

where $\Phi(x_1, \ldots, x_n, y)$ is a conjunction of inequalities or `True`, by reducing them to

$$\begin{gathered} f(x_1, \ldots, x_n) == 0 \bigwedge g(x_1, \ldots, x_n) == 0 \bigwedge \Phi(x_1, \ldots, x_n, y) \bigvee \\ y == -g(x_1, \ldots, x_n)/f(x_1, \ldots, x_n) \bigwedge \Phi(x_1, \ldots, x_n, -g(x_1, \ldots, x_n)/f(x_1, \ldots, x_n)). \end{gathered} \tag{3}$$

The first part of the system (3) is solved using the method for solving systems with more than one equation. *Mathematica* recognizes three cases when the second part of the system (3) is solvable. If $f(x_1, \ldots, x_n) \equiv 1$, the solution is given by $y == -g(x_1, \ldots, x_n)$ and by the restrictions on $x_1, \ldots, x_n$ obtained by solving the inequalities $\Phi(x_1, \ldots, x_n, -g(x_1, \ldots, x_n))$. Nonlinear systems of inequalities are solved using `CylindricalDecomposition`. If $f(x_1, \ldots, x_n) \equiv m$ for an integer constant $m \geq 2$, the solution of the second part of the system (3) is given by $y == -g(x_1, \ldots, x_n)/m$ and by the restrictions on $x_1, \ldots, x_n$ obtained by solving the congruence $g(x_1, \ldots, x_n) \equiv 0 \bmod m$ and then solving the inequalities $\Phi(x_1, \ldots, x_n, -g(x_1, \ldots, x_n)/m)$ for each solution of the congruence. If $f(x_1, \ldots, x_n)$ is nonconstant, *Mathematica* can solve the second part of the system (3) if $n = 1$. Since *Mathematica* factors all equations at the preprocessing stage, $f(x_1)$ and $g(x_1)$ can be assumed to be relatively prime. Then

$$d\, g(x_1) = q(x_1)\, f(x_1) + r(x_1)$$

for an integer d and polynomials $q(x_1)$ and $r(x_1)$ with integer coefficients and $\deg(r) < \deg(f)$. If $-g(x_1)/f(x_1)$ is an integer, then $r(x_1)/f(x_1)$ is an integer, and so $r(x_1) == 0$ or $|r(x_1)| \geq |f(x_1)|$. Since $\deg(r) < \deg(f)$, the last condition is satisfied only by a finite number of integers x_1. Hence the solutions of the second part of the system (3) can be selected from a finite number of solution candidates.

Additionally, *Mathematica* uses the following heuristic to detect cases when the system (3) has no solutions. If there is an integer $m \geq 2$, such that $f(x_1, \ldots, x_n)$ is always divisible by m, and $g(x_1, \ldots, x_n)$ is never divisible by m, then the system (3) has no solutions. Candidates for m are found by computing the `GCD` of the values of f at several points.

The last two methods use exhaustive search over finite sets of points. The allowed number of search points is controlled by the system option `SieveMaxPoints`.

This reduces to (3) with $f(x_1, \ldots, x_n) \equiv 1$.

```
In[37]:= Reduce[x^3 - 7 x y + 5 y^4 - z == 3 && 2 x - y > 1, {x, y, z}, Integers]
Out[37]= (C[1] | C[2] | C[3] | C[4] | C[5]) ∈ Integers && C[1] ≥ 0 && C[2] ≥ 0 && C[3] ≥ 0 && C[4] ≥ 0 &&
 C[5] ≥ 0 && ((x == 1 + C[1] + C[2] + C[3] - C[4] && y == C[1] + 2 C[2] - 2 C[4] - C[5]) ||
    (x == C[1] + C[2] + C[3] - C[4] && y == -2 + C[1] + 2 C[2] - 2 C[4] - C[5])) && z == -3 + x^3 - 7 x y + 5 y^4
```

This reduces to (3) with $f(x_1, \ldots, x_n) \equiv 3$.

```
In[38]:= Reduce[x^3 - 2 x^2 y + 9 x y^4 - 3 z == 8, {x, y, z}, Integers]
Out[38]= (C[1] | C[2]) ∈ Integers &&
  ((x == 1 + 3 C[1] && y == 1 + 3 C[2]) || (x == 2 + 3 C[1] && y == 3 C[2])) && z == 1/3 (-8 + x^3 - 2 x^2 y + 9 x y^4)
```

This reduces to the $n == 1$ case of system (3).

```
In[39]:= Reduce[x^5 + 7 x - 271 + y (x^3 + 21 x^2 - 17) == 0, {x, y}, Integers]
Out[39]= (x == -1 && y == 93) || (x == 2 && y == 3)
```

Here `Reduce` detects that the equation has no solutions, because $9 x^6 y^3 z^4 - 9 x^2 y^3 z^8 - 5 y^8 z^9 - 10$ is always divisible by 5, and $7 - 5 x^4 y z^4 + 7 x^8 y^2 z^4 - 9 z^8 - 4 x^6 y z^8$ is never divisible by 5.

```
In[40]:= Reduce[(9 x^6 y^3 z^4 - 9 x^2 y^3 z^8 - 5 y^8 z^9 - 10) t + 7 - 5 x^4 y z^4 + 7 x^8 y^2 z^4 - 9 z^8 - 4 x^6 y z^8 == 0,
  {x, y, z, t}, Integers]
Out[40]= False
```

Systems with Empty or Bounded Real Solution Sets

If a Diophantine polynomial system is not solved by any other methods, *Mathematica* solves the system over the reals using the Cylindrical Algebraic Decomposition (CAD) algorithm. If the system has no real solutions, then clearly it has no integer solutions. If the real solution set is bounded, then the number of integer solutions is finite. In principle, all the integer solutions can be found in this case from a cylindrical decomposition. Namely, for each cylinder, you enumerate all possible integer values of the first coordinate, then for each value of the first coordinate, you enumerate all possible integer values of the second coordinate, and so on. However, for large bounded solution sets this method could lead to a huge number of points to try. Therefore, *Mathematica* has a bound on the number of explicitly enumerated integer solutions in a real interval. By default this bound is equal to 10. It can be changed using the system option `DiscreteSolutionBound`. For systems for which the real solution set is unbounded or bounded but large, the solution is represented implicitly by returning the CAD and a condition that all variables are integers. Note that for multivariate systems such an implicit representation may not even be enough to tell whether integer solutions exist. This should be expected, given Matiyasevich's solution of Hilbert's tenth problem [2].

Here the real solution set is bounded, but `Reduce` gives some cylinders in an implicit form. This is because some of the intervals bounding y contain more than 10 integers.

```
In[41]:= Reduce[x^5 + y^2 + z^3 - x y z == 8 && x^2 + y^2 ≤ 30, {x, y, z}, Integers]

Out[41]= (y | z) ∈ Integers && ((x == -2 && -5 ≤ y ≤ 5 && z == Root[-40 + y^2 + 2 y #1 + #1^3 &, 1]) ||
    (x == -1 && ((y == -3 && z == 0) || (y == 3 && z == 0))) || (x == 0 && -5 ≤ y ≤ 5 && z == Root[-8 + y^2 + #1^3 &, 1]) ||
    (x == 1 && ((y == -5 && z == -2) || (y == -2 && z == 1) || (y == 1 && z == 2) || (y == 3 && (z == -2 || z == 1)))) ||
    (x == 2 && -5 ≤ y ≤ 5 && z == Root[24 + y^2 - 2 y #1 + #1^3 &, 1]))
```

Increasing the value of the system option `DiscreteSolutionBound` allows `Reduce` to find all integer solutions explicitly.

```
In[42]:= SetSystemOptions["ReduceOptions" → {"DiscreteSolutionBound" → 11}];
         Reduce[x^5 + y^2 + z^3 - x y z == 8 && x^2 + y^2 ≤ 30, {x, y, z}, Integers]

Out[43]= (x == -1 && ((y == -3 && z == 0) || (y == 3 && z == 0))) || (x == 0 &&
    ((y == -4 && z == -2) || (y == -3 && z == -1) || (y == 0 && z == 2) || (y == 3 && z == -1) || (y == 4 && z == -2))) ||
  (x == 1 && ((y == -5 && z == -2) || (y == -2 && z == 1) || (y == 1 && z == 2) || (y == 3 && (z == -2 || z == 1)))) ||
  (x == -2 && y == 4 && z == 2)
```

This resets `DiscreteSolutionBound` to the default value.

```
In[44]:= SetSystemOptions["ReduceOptions" → {"DiscreteSolutionBound" → 10}];
```

Here the modular sieve method shows that there are no solutions in $(-15, 15]^3$. After adding inequalities to eliminate this cube, `Reduce` then recognizes that this equation has no solutions anywhere.

```
In[45]:= Reduce[9 x^2 y^2 + 7 x^2 z^2 + 5 y^2 z^2 == x y z + 10, {x, y, z}, Integers]

Out[45]= False
```

Equations of the Form $x\,g(x, y, z_1, \ldots, z_n) + y == c$

Mathematica attempts to solve Diophantine systems of the form

$$x\,g(x, y, z_1, \ldots, z_n) + y == c \bigwedge \Phi(x, y, z_1, \ldots, z_n),$$

where $\Phi(x, y, z_1, \ldots, z_n)$ is a conjunction of inequalities or `True`, by transforming them to

$$x == 0 \bigwedge y == c \bigwedge \Phi(0, c, z_1, \ldots, z_n) \bigvee y == c + t\,x \bigwedge g(x, c + t\,x, z_1, \ldots, z_n) + t == 0 \bigwedge \Phi(x, c + t\,x, z_1, \ldots, z_n). \quad (4)$$

The resulting system (4) may, or may not, be easier to solve. Systems exist for which this transformation could be applied recursively arbitrarily many times; therefore, *Mathematica* uses a recursion bound to ensure the heuristic terminates.

This transforms to a system (4) with no real solutions.

```
In[46]:= Reduce[-x^2 + y + x y + x z == 0 && x > 0 && y > 0 && z > 0, {x, y, z}, Integers]
Out[46]= False
```

Here the system (4) obtained after three recursive transformations has a reducible nonconstant part.

```
In[47]:= Reduce[x^3 - 2 x y^2 + 20 x y + y == 5, {x, y}, Integers]
Out[47]= (x == -7 && y == 12) || (x == 0 && y == 5) || (x == 7 && y == -2)
```

Systems Solvable by Exhaustive Search

For systems containing explicit lower and upper bounds on all variables, *Mathematica* uses exhaustive search to find solutions. The bounds of the search are specified by the value of the system option `ExhaustiveSearchMaxPoints`. The option value should be a pair of integers (the default is $\{1000, 10\,000\}$). If the number of integer points within the bounds does not exceed the first integer, the exhaustive search is used instead of any solution methods other than univariate polynomial solving. Otherwise, if the number of integer points within the bounds does not exceed the second integer, the exhaustive search is performed after all other methods fail.

This transcendental Diophantine equation with bounded variable values is solved by exhaustive search.

```
In[48]:= Reduce[Sin[π x y/2]^2 == Gamma[21 x - 37 y] && 0 < x < 100 && 1 ≤ y ≤ 100, x, Integers]
Out[48]= (y == 13 && x == 23) || (y == 55 && x == 97)
```

Options

The *Mathematica* functions for solving Diophantine polynomial systems have a number of options that control the way they operate. This tutorial gives a summary of these options.

option name	*default value*	
`GeneratedParameters`	`C`	specifies how the new parameters generated to represent solutions should be named

`Reduce` options affecting the behavior for Diophantine polynomial systems.

GeneratedParameters

To represent infinite solutions of some Diophantine systems, `Reduce` needs to introduce new integer parameters. The names of the new parameters are specified by the option `GeneratedParameters`. With `GeneratedParameters -> f`, the new parameters are named `f[1], f[2], ...`.

By default, the new parameters generated by `Reduce` are named `C[1], C[2], ...`.

```
In[49]:= Reduce[x + y + z == 2 && x > y + 1, {x, y, z}, Integers]

Out[49]= (C[1] | C[2] | C[3] | C[4] | C[5]) ∈ Integers && C[1] ≥ 0 && C[2] ≥ 0 &&
          C[3] ≥ 0 && C[4] ≥ 0 && C[5] ≥ 0 && ((x == 1 + C[1] + C[2] + C[3] - C[4] &&
              y == -1 + C[1] - C[3] - C[4] - C[5] && z == 2 - 2 C[1] - C[2] + 2 C[4] + C[5]) ||
            (x == 2 + C[1] + C[2] + C[3] - C[4] && y == C[1] - C[3] - C[4] - C[5] && z == -2 C[1] - C[2] + 2 C[4] + C[5]) ||
            (x == C[1] + C[2] + C[3] - C[4] &&
              y == -2 + C[1] - C[3] - C[4] - C[5] && z == 4 - 2 C[1] - C[2] + 2 C[4] + C[5]))
```

The option `GeneratedParameters` allows users to customize the parameter names.

```
In[50]:= Reduce[x + y + z == 2 && x > y + 1, {x, y, z},
          Integers, GeneratedParameters → (Subscript[k, #] &)]
```

Out[50]= $(k_1 \mid k_2 \mid k_3 \mid k_4 \mid k_5) \in \text{Integers} \,\&\&\, k_1 \geq 0 \,\&\&\, k_2 \geq 0 \,\&\&\, k_3 \geq 0 \,\&\&\, k_4 \geq 0 \,\&\&\, k_5 \geq 0 \,\&\&$
$((x == 1 + k_1 + k_2 + k_3 - k_4 \,\&\&\, y == -1 + k_1 - k_3 - k_4 - k_5 \,\&\&\, z == 2 - 2 k_1 - k_2 + 2 k_4 + k_5) \mid\mid$
$(x == 2 + k_1 + k_2 + k_3 - k_4 \,\&\&\, y == k_1 - k_3 - k_4 - k_5 \,\&\&\, z == -2 k_1 - k_2 + 2 k_4 + k_5) \mid\mid$
$(x == k_1 + k_2 + k_3 - k_4 \,\&\&\, y == -2 + k_1 - k_3 - k_4 - k_5 \,\&\&\, z == 4 - 2 k_1 - k_2 + 2 k_4 + k_5))$

ReduceOptions Group of System Options

Here are the system options from the `ReduceOptions` group that may affect the behavior of `Reduce`, `Resolve`, and `FindInstance` for Diophantine polynomial systems. The options can be set with

`SetSystemOptions["ReduceOptions" -> {"`*option name*`" -> `*value*`}].`

option name	default value	
"BranchLinearDiophantine"	True	whether Reduce should use a branch-and-bound type algorithm to compute solutions of bounded systems of linear Diophantine inequalities
"DiscreteSolutionBound"	10	the bound on the number of explicitly enumerated integer solutions in a real interval
"ExhaustiveSearchMaxPoints"	{1000,10 000}	the maximal number of integer points within variable bounds for which the exhaustive search is used before and after all other solution methods
"LatticeReduceDiophantine"	True	whether LatticeReduce should be used to preprocess bounded systems of linear Diophantine inequalities
"MaxFrobeniusGraph"	1 000 000	the maximal size of the smallest coefficient in a Frobenius equation for which FindInstance computes the critical tree in the Frobenius graph
"SieveMaxPoints"	10 000	the maximal number of points at which the modular sieve method evaluates the system

ReduceOptions group options affecting the behavior of Reduce, Resolve, and FindInstance for Diophantine polynomial systems.

BranchLinearDiophantine

The value of the system option BranchLinearDiophantine specifies which variant of the algorithm should be used in the final stage of solving bounded linear systems. Neither variant seems to be clearly better. For some examples the hybrid method combining a branch-and-bound type algorithm and a simple recursive enumeration is faster; for other examples the simple recursive enumeration alone is faster. The hybrid method seems to be more robust for badly conditioned problems, hence it is the default method.

This finds integer points in a long, narrow four-dimensional simplex using the default hybrid method.

```
In[51]:= a = 10 000;
         Reduce[a x + a y + a z - 3 (a - 1) t ≤ 3 a && a x + a y + a t - 3 (a - 1) z ≤ 3 a &&
              a x + a z + a t - 3 (a - 1) y ≤ 3 a && a y + a z + a t - 3 (a - 1) x ≤ 3 a &&
              x + y + z + t ≥ 1 && x < y && z < t, {x, y, z, t}, Integers] // Length // Timing
Out[52]= {0.671, 3336}
```

This sets the value of the system option `BranchLinearDiophantine` to `False`.

```
In[53]:= SetSystemOptions["ReduceOptions" → {"BranchLinearDiophantine" → False}];
```

Here the simple recursive enumeration method is used, and for this badly conditioned problem it is several times slower.

```
In[54]:= Reduce[a x + a y + a z - 3 (a - 1) t ≤ 3 a && a x + a y + a t - 3 (a - 1) z ≤ 3 a &&
             a x + a z + a t - 3 (a - 1) y ≤ 3 a && a y + a z + a t - 3 (a - 1) x ≤ 3 a &&
             x + y + z + t ≥ 1 && x < y && z < t, {x, y, z, t}, Integers] // Length // Timing

Out[54]= {4.447, 3336}
```

This resets the value of the system option `BranchLinearDiophantine` to the default value.

```
In[55]:= SetSystemOptions["ReduceOptions" → {"BranchLinearDiophantine" → True}];
```

Here are solutions of a system of two randomly generated equations *eqns* and three randomly generated inequalities *ineqs* in seven variables inside a simplex bounded by *bds*.

```
In[56]:= SeedRandom[1];
         A = Table[RandomInteger[{-1000, 1000}], {2}, {7}];
         a = Table[RandomInteger[{-1000, 1000}], {2}];
         B = Table[RandomInteger[{-1000, 1000}], {3}, {7}];
         b = Table[RandomInteger[{-1000, 1000}], {3}];
         X = x /@ Range[7];
         eqns = And @@ Thread[A.X == a];
         ineqs = And @@ Thread[B.X ≥ b];
         bds = And @@ Thread[X ≥ 0] && Total[X] ≤ 100;
         Reduce[eqns && ineqs && bds, X, Integers] // Timing

Out[64]= {7.32, (x[1] == 9 && x[2] == 8 && x[3] == 2 && x[4] == 14 && x[5] == 20 && x[6] == 22 && x[7] == 13) ||
           (x[1] == 12 && x[2] == 15 && x[3] == 0 && x[4] == 12 && x[5] == 11 && x[6] == 22 && x[7] == 18)}
```

For this system the nondefault simple recursion method is faster.

```
In[65]:= SetSystemOptions["ReduceOptions" → {"BranchLinearDiophantine" → False}];
         Reduce[eqns && ineqs && bds, X, Integers] // Timing

Out[66]= {1.643, (x[1] == 9 && x[2] == 8 && x[3] == 2 && x[4] == 14 && x[5] == 20 && x[6] == 22 && x[7] == 13) ||
           (x[1] == 12 && x[2] == 15 && x[3] == 0 && x[4] == 12 && x[5] == 11 && x[6] == 22 && x[7] == 18)}
```

Here is a random system very similar to the previous one, except that it contains one more variable and the right-hand side of the last of *bds* is changed from `100` to `200`. However, for this system the default method is faster.

```
In[67]:= SetSystemOptions["ReduceOptions" → {"BranchLinearDiophantine" → True}];
         SeedRandom[1];
         A = Table[RandomInteger[{-1000, 1000}], {2}, {8}];
         a = Table[RandomInteger[{-1000, 1000}], {2}];
         B = Table[RandomInteger[{-1000, 1000}], {3}, {8}];
         b = Table[RandomInteger[{-1000, 1000}], {3}];
         X = x /@ Range[8];
         eqns = And @@ Thread[A.X == a];
         ineqs = And @@ Thread[B.X ≥ b];
         bds = And @@ Thread[X ≥ 0] && Total[X] ≤ 200;
         Reduce[eqns && ineqs && bds, X, Integers] // Timing

Out[76]= {16.093, (x[1] == 2 && x[2] == 6 && x[3] == 27 && x[4] == 35 && x[5] == 0 && x[6] == 6 && x[7] == 0 && x[8] == 38) ||
           (x[1] == 10 && x[2] == 1 && x[3] == 48 && x[4] == 54 && x[5] == 1 && x[6] == 1 && x[7] == 1 && x[8] == 55)}
```

The nondefault method is slower for this system.

```
In[77]:= SetSystemOptions["ReduceOptions" → {"BranchLinearDiophantine" → False}];
         Reduce[eqns && ineqs && bds, X, Integers] // Timing
Out[78]= {47.789, (x[1] == 2 && x[2] == 6 && x[3] == 27 && x[4] == 35 && x[5] == 0 && x[6] == 6 && x[7] == 0 && x[8] == 38) ||
           (x[1] == 10 && x[2] == 1 && x[3] == 48 && x[4] == 54 && x[5] == 1 && x[6] == 1 && x[7] == 1 && x[8] == 55)}
```

This resets the value of the system option `BranchLinearDiophantine` to the default value.

```
In[79]:= SetSystemOptions["ReduceOptions" → {"BranchLinearDiophantine" → True}];
```

DiscreteSolutionBound

The value of the system option `DiscreteSolutionBound` specifies whether integer solutions in a real interval $a \leq x \leq b$ should be enumerated explicitly or represented implicitly as $x \in \mathbb{Z} \wedge a \leq x \leq b$. With `DiscreteSolutionBound -> n`, the integer solutions in the given real interval are enumerated explicitly if their number does not exceed n. The default value of the option is 10.

There are 10 integers in the real interval $0 \leq x < 10$. `Reduce` writes them out explicitly.

```
In[80]:= Reduce[0 ≤ x^3 < 1000, Integers]
Out[80]= x == 0 || x == 1 || x == 2 || x == 3 || x == 4 || x == 5 || x == 6 || x == 7 || x == 8 || x == 9
```

There are 11 integers in the real interval $0 \leq x < 1001^{1/3}$. `Reduce` represents them implicitly.

```
In[81]:= Reduce[0 ≤ x^3 < 1001, Integers]
Out[81]= x ∈ Integers && 0 ≤ x ≤ 10
```

This increases the `DiscreteSolutionBound` to 11.

```
In[82]:= SetSystemOptions["ReduceOptions" → {"DiscreteSolutionBound" → 11}];
```

Now `Reduce` represents the solutions explicitly.

```
In[83]:= Reduce[0 ≤ x^3 < 1001, Integers]
Out[83]= x == 0 || x == 1 || x == 2 || x == 3 || x == 4 || x == 5 || x == 6 || x == 7 || x == 8 || x == 9 || x == 10
```

This resets `DiscreteSolutionBound` to the default value.

```
In[84]:= SetSystemOptions["ReduceOptions" → {"DiscreteSolutionBound" → 10}];
```

The value of `DiscreteSolutionBound` also affects the solving of bounded linear systems.

ExhaustiveSearchMaxPoints

The system option `ExhaustiveSearchMaxPoints` specifies the maximal number of search points used by the exhaustive search method. The option value should be a pair of integers (the default is $\{1000, 10\,000\}$). If the number of integer points within the bounds does not exceed the first integer, the exhaustive search is used instead of any solution methods other than univariate polynomial solving. Otherwise, if the number of integer points within the bounds does not exceed the second integer, the exhaustive search is performed after all other methods fail.

With the default setting of `ExhaustiveSearchMaxPoints`, `Reduce` is unable to solve this equation.

```
In[85]:= Reduce[Binomial[x, y] == Gamma[x + y] && 1 ≤ x ≤ 200 && 1 ≤ y ≤ 200, {x, y}, Integers]
```

Reduce::nsmet :
This system cannot be solved with the methods available to Reduce. ≫

```
Out[85]= Reduce[Binomial[x, y] == Gamma[x + y] && 1 ≤ x ≤ 200 && 1 ≤ y ≤ 200, {x, y}, Integers]
```

This increases the value of the second element of `ExhaustiveSearchMaxPoints` to $100\,000$.

```
In[86]:= SetSystemOptions[
           "ReduceOptions" → {"ExhaustiveSearchMaxPoints" → {1000, 100 000}}];
```

Now `Reduce` can solve the equation.

```
In[87]:= Reduce[Binomial[x, y] == Gamma[x + y] && 1 ≤ x ≤ 200 && 1 ≤ y ≤ 200, {x, y}, Integers]

Out[87]= (x == 1 && y == 1) || (x == 2 && y == 1)
```

With the default setting of `ExhaustiveSearchMaxPoints`, `Reduce` solves this equation using the method for solving Pell equations.

```
In[88]:= SetSystemOptions["ReduceOptions" → {"ExhaustiveSearchMaxPoints" → {1000, 10 000}}];
         Reduce[x^2 - 2 y^2 == 1 && 1 ≤ x ≤ 1000 && 1 ≤ y ≤ 1000, {x, y}, Integers] // Timing

Out[88]= {0.06, (x == 3 && y == 2) || (x == 17 && y == 12) || (x == 99 && y == 70) || (x == 577 && y == 408)}
```

Increasing the first element of `ExhaustiveSearchMaxPoints` to 10^6 makes `Reduce` use the exhaustive search first. In this example the search is much slower than the Pell equation solver.

```
In[89]:= SetSystemOptions["ReduceOptions" → {"ExhaustiveSearchMaxPoints" → {10^6, 10^6}}];
         Reduce[x^2 - 2 y^2 == 1 && 1 ≤ x ≤ 1000 && 1 ≤ y ≤ 1000, {x, y}, Integers] // Timing

Out[90]= {5.538, (x == 3 && y == 2) || (x == 17 && y == 12) || (x == 99 && y == 70) || (x == 577 && y == 408)}
```

For this equation the Pell equation solver is slower than the exhaustive search.

In[91]:=
```
SetSystemOptions[
  "ReduceOptions" → {"ExhaustiveSearchMaxPoints" → {1000, 10 000}}];
Reduce[x^2 - 21 y^2 == 2004 && 1 ≤ x ≤ 100 && 1 ≤ y ≤ 100, {x, y}, Integers] // Timing
```
Out[92]=
```
{0.381, x == 45 && y == 1}
```

The exhaustive search is faster here.

In[93]:=
```
SetSystemOptions[
 "ReduceOptions" → {"ExhaustiveSearchMaxPoints" → {10 000, 10 000}}];
Reduce[x^2 - 21 y^2 == 2004 && 1 ≤ x ≤ 100 && 1 ≤ y ≤ 100, {x, y}, Integers] // Timing
```
Out[93]=
```
{0.12, x == 45 && y == 1}
```

This resets `ExhaustiveSearchMaxPoints` to the default value.

In[94]:=
```
SetSystemOptions["ReduceOptions" → {"ExhaustiveSearchMaxPoints" → {1000, 10 000}}];
```

LatticeReduceDiophantine

The value of the system option `LatticeReduceDiophantine` specifies whether `LatticeReduce` should be used to preprocess systems of bounded linear inequalities. The use of `LatticeReduce` is important for systems of inequalities describing polyhedra whose projections on some nonaxial lines are much smaller than their projections on the axes. However, there are systems for which `LatticeReduce`, instead of simplifying the problem, makes it significantly harder.

This finds the only two integer points in a triangle whose projections on both axes have sizes greater than a but whose projection on the line $x + 5000\, y == 0$ has size one.

In[95]:=
```
a = 10^4;
Reduce[a x ≤ (a + 1) y && (a + 1) x ≥ (a + 2) y && 0 ≤ x ≤ a + 1, {x, y}, Integers] // Timing
```
Out[96]=
```
{0., (x == 0 && y == 0) || (x == 10 001 && y == 10 000)}
```

This sets the value of the system option `LatticeReduceDiophantine` to `False`.

In[97]:=
```
SetSystemOptions["ReduceOptions" → {"LatticeReduceDiophantine" → False}];
```

The nondefault method is much slower for this system, and the speed difference grows with a.

In[98]:=
```
Reduce[a x ≤ (a + 1) y && (a + 1) x ≥ (a + 2) y && 0 ≤ x ≤ a + 1, {x, y}, Integers] // Timing
```
Out[98]=
```
{3.875, (x == 0 && y == 0) || (x == 10 001 && y == 10 000)}
```

Here is a system that contains a set of simple inequalities *bds*, which bound solutions to a reasonably small size polyhedron, combined with a set of relatively complicated inequalities *ineqs*. For such systems, using LatticeReduce tends to increase the timing.

```
In[99]:= SetSystemOptions["ReduceOptions" → {"LatticeReduceDiophantine" → True}];
SeedRandom[1];
B = Table[RandomInteger[{-1000, 1000}], {3}, {5}];
b = Table[RandomInteger[{-1000, 1000}], {3}];
X = x /@ Range[5];
ineqs = And @@ Thread[B.X ≥ b];
bds = And @@ Thread[X ≥ 0] && Total[X] ≤ 10;
Reduce[ineqs && bds, X, Integers] // Length // Timing
Out[105]= {1.773, 35}
```

The nondefault method is faster for this system.

```
In[106]:= SetSystemOptions["ReduceOptions" → {"LatticeReduceDiophantine" → False}];
Reduce[ineqs && bds, X, Integers] // Length // Timing
Out[107]= {0.09, 35}
```

This resets LatticeReduceDiophantine to the default value.

```
In[108]:= SetSystemOptions["ReduceOptions" → {"LatticeReduceDiophantine" → True}];
```

MaxFrobeniusGraph

The system option MaxFrobeniusGraph specifies the maximal size of the smallest coefficient in a Frobenius equation for which FindInstance uses an algorithm based on the computation of the critical tree in the Frobenius graph [11]. Otherwise, the more general methods for solving bounded linear systems are used. Unlike the general method for solving bounded linear systems, the method based on the computation of the Frobenius graph depends very little on the number of variables, hence it is the faster choice for equations with many variables. On the other hand, the method requires storing a graph of the size of the smallest coefficient, so for large coefficients it may run out of memory.

To find a solution of a Frobenius equation with the smallest coefficient larger than 10^6, FindInstance by default uses the general method for solving bounded linear systems. For this example the method is relatively slow but uses little memory. The kernel has been restarted to show the memory usage by the current example.

```
In[1]:= SeedRandom[1];
A = Table[RandomInteger[{5 10^6, 10^7}], {25}];
X = x /@ Range[25];
FindInstance[A.X == 123 456 789 && And @@ Thread[X ≥ 0], X, Integers] // Timing
Out[4]= {42.952, {{x[1] → 1, x[2] → 0, x[3] → 0, x[4] → 0, x[5] → 2, x[6] → 3, x[7] → 0, x[8] → 0, x[9] → 0,
   x[10] → 7, x[11] → 0, x[12] → 0, x[13] → 0, x[14] → 0, x[15] → 0, x[16] → 2, x[17] → 1,
   x[18] → 0, x[19] → 0, x[20] → 3, x[21] → 0, x[22] → 0, x[23] → 0, x[24] → 0, x[25] → 0}}}
```

In[5]:=
```
MaxMemoryUsed[]
```

Out[5]=
```
10 288 400
```

This increases the value of `MaxFrobeniusGraph` to 10^7.

In[6]:=
```
SetSystemOptions["ReduceOptions" → {"MaxFrobeniusGraph" → 10^7}];
```

Now `FindInstance` uses the method based on the computation of the Frobenius graph. It finds the solution faster, but it uses more memory.

In[7]:=
```
FindInstance[A.X == 123 456 789 && And @@ Thread[X ≥ 0], X, Integers] // Timing
```

Out[7]=
```
{2.213, {{x[1] → 0, x[2] → 14, x[3] → 0, x[4] → 0, x[5] → 1, x[6] → 0, x[7] → 0, x[8] → 0, x[9] → 0,
   x[10] → 0, x[11] → 0, x[12] → 0, x[13] → 0, x[14] → 0, x[15] → 0, x[16] → 2, x[17] → 1,
   x[18] → 1, x[19] → 1, x[20] → 0, x[21] → 0, x[22] → 1, x[23] → 1, x[24] → 0, x[25] → 0}}}
```

In[8]:=
```
MaxMemoryUsed[]
```

Out[8]=
```
77 722 760
```

This resets `MaxFrobeniusGraph` to the default value.

In[9]:=
```
SetSystemOptions["ReduceOptions" → {"MaxFrobeniusGraph" → 10^6}];
```

SieveMaxPoints

The system option `SieveMaxPoints` specifies the maximal number of search points used by the modular sieve method and by searches used in solving equations with a linear variable. The default value of the option is 10,000.

With the default setting of `SieveMaxPoints`, `FindInstance` is unable to find a solution for this equation.

In[10]:=
```
FindInstance[x^2 + 21 y^3 - 17 z^4 == 401, {x, y, z}, Integers]
```

FindInstance::nsmet : The methods available to FindInstance are insufficient to find the requested instances or prove they do not exist. ≫

Out[10]=
```
FindInstance[x^2 + 21 y^3 - 17 z^4 == 401, {x, y, z}, Integers]
```

Increasing the number of `SieveMaxPoints` to one million allows `FindInstance` to find a solution.

In[11]:=
```
SetSystemOptions["ReduceOptions" → {"SieveMaxPoints" → 1 000 000}];
FindInstance[x^2 + 21 y^3 - 17 z^4 == 401, {x, y, z}, Integers]
```

Out[12]=
```
{{x → -29, y → -2, z → -2}}
```

This resets `SieveMaxPoints` to the default value.

In[13]:=
```
SetSystemOptions["ReduceOptions" → {"SieveMaxPoints" → 10 000}];
```

References

[1] Goldbach, C. Letter to L. Euler, June 7, 1742. http://mathworld.wolfram.com/GoldbachConjecture.html.

[2] Matiyasevich, Yu. V. "The Diophantineness of Enumerable Sets (Russian)" *Dokl. Akad. Nauk SSSR* 191 (1970): 279-282. English translation: *Soviet Math. Dokl.* 11 (1970): 354-358.

[3] Contejean, E. and H. Devie. "An Efficient Incremental Algorithm for Solving Systems of Linear Diophantine Equations." *Information and Computation* 113 (1994): 143-172.

[4] Cucker, F., P. Koiran, and S. Smale. "A Polynomial Time Algorithm for Diophantine Equations in One Variable." *Journal of Symbolic Computation* 27, no. 1 (1999): 21-30.

[5] Strzebonski, A. "An Improved Algorithm for Diophantine Equations in One Variable." Paper presented at the ACA 2002 Session on Symbolic-Numerical Methods in Computational Science, Volos, Greece. Notebook with the conference talk available at members.wolfram.com/adams

[6] Dickson, L. E. *History of the Theory of Numbers*. Chelsea, 1952.

[7] Nagell, T. *Introduction to Number Theory.* Wiley, 1951.

[8] Hardy, K., J. B. Muskat and K. S. Williams. "A Deterministic Algorithm for Solving $n = f\, u^2 + g\, v^2$ in Coprime Integers u and v." *Mathematics of Computation* 55 (1990): 327-343.

[9] Smart, N. *The Algorithmic Resolution of Diophantine Equations*. Cambridge University Press, 1998.

[10] Bressoud, D. M. and S. Wagon. *A Course in Computational Number Theory*. Key College Publishing, 2000.

[11] Beihoffer, D. E., J. Hendry, A. Nijenhuis, and S. Wagon. "Faster Algorithms for Frobenius Numbers." to appear in *The Electronic Journal of Combinatorics.*

[12] Rabin, M. O. and J. O. Shallit. "Randomized Algorithms in Number Theory." *Communications on Pure and Applied Mathematics* 39 (1986): 239-256.

Algebraic Number Fields

Mathematica provides representation of algebraic numbers as `Root` objects. A `Root` object contains the minimal polynomial of the algebraic number and the root number—an integer indicating which of the roots of the minimal polynomial the `Root` object represents. This allows for unique representation of arbitrary complex algebraic numbers. A disadvantage is that performing arithmetic operations in this representation is quite costly. That is why *Mathematica* requires the use of an additional function, `RootReduce`, in order to simplify arithmetic expressions. Restricting computations to be within a fixed finite algebraic extension of the rationals, $\mathbb{Q}[\theta]$, allows a more convenient representation of its elements as polynomials in θ.

`AlgebraicNumber[`θ`,{`c_0`,`c_1`,…,`c_n`}]`	represent the algebraic number $c_0 + c_1\,\theta + \ldots + c_n\,\theta^n$ in $\mathbb{Q}[\theta]$

Representation of algebraic numbers as elements of a finite extension of rationals.

If θ is an algebraic integer with a `MinimalPolynomial` of degree l, and $\{c_0, \ldots, c_l\}$ are rational numbers, then `AlgebraicNumber[`θ`, {`c_0`, …, `c_l`}]` is an inert numeric object.

```
In[1]:= a = AlgebraicNumber[Root[#^3 - # + 1 &, 1], {1, 2, 3}]
Out[1]= AlgebraicNumber[Root[1 - #1 + #1^3 &, 1], {1, 2, 3}]
```

`N` can be used to find a numeric approximation of an `AlgebraicNumber` object.

```
In[2]:= N[a, 20]
Out[2]= 3.6151970842505862282
```

For any algebraic number θ and any list of rational numbers $\{c_0, \ldots, c_l\}$, `AlgebraicNumber[`θ`, {`c_0`, …, `c_l`}]` evaluates to `AlgebraicNumber[`ξ`, {`d_0`, …, `d_m`}]`, such that $\xi = d\theta$, d is a factor of the leading coefficient of `MinimalPolynomial` of θ, such that ξ is an algebraic integer, m is the degree of `MinimalPolynomial` of θ, and

$$c_0 + c_1\,\theta + \ldots + c_l\,\theta^l == d_0 + d_1\,\xi + \ldots + d_m\,\xi^m.$$

`AlgebraicNumber` automatically makes the generator of the extension an algebraic integer and the coefficient list equal in length to the degree of the extension.

```
In[3]:= AlgebraicNumber[Root[2 #^4 - 3 # + 2 &, 1], {1, 2, 3, 4, 5, 6}]

Out[3]= AlgebraicNumber[Root[16 - 12 #1 + #1^4 &, 1], {-4, 7/4, 3, 1/2}]
```

`AlgebraicNumber` objects representing rational numbers reduce automatically to numbers.

```
In[4]:= AlgebraicNumber[Root[#^5 - 7 # + 1 &, 1], {0, 7, 0, 0, 0, -1}]
Out[4]= 1
```

Adding or multiplying `AlgebraicNumber` objects that explicitly belong to the same field (i.e., have the same first elements), adding or multiplying a rational number and an `AlgebraicNumber` object, or raising an `AlgebraicNumber` object to an integer power yields an `AlgebraicNumber` object.

```
In[5]:= a = AlgebraicNumber[Root[#^4 + 7 # - 21 &, 1], {1, 2, 3, 4}];
        b = AlgebraicNumber[Root[#^4 + 7 # - 21 &, 1], {9, 8, 7, 5}];
        (2 a^2 - 3 a b + 5 b^5 - 3)/(a^8 - b^4 + 1/2) + 9

Out[5]= AlgebraicNumber[Root[-21 + 7 #1 + #1^4 &, 1],
         {41 286 695 899 369 558 776 723 710 439 212 189 982 056 327 290 172 063/
          4 586 375 026 009 762 651 263 976 115 838 375 027 468 985 058 462 049,
          6 520 802 026 300 441 952 691 134 470 541 521 717 177 617 572 114/
          13 759 125 078 029 287 953 791 928 347 515 125 082 406 955 175 386 147,
          3 688 721 281 596 550 115 065 494 536 738 395 724 701 865 336 152/
          13 759 125 078 029 287 953 791 928 347 515 125 082 406 955 175 386 147,
          2 274 021 184 276 897 634 212 701 763 901 059 483 341 282 983 762/
          13 759 125 078 029 287 953 791 928 347 515 125 082 406 955 175 386 147}]
```

`RootReduce` transforms `AlgebraicNumber` objects to `Root` objects.

```
In[6]:= RootReduce[a]

Out[6]= Root[-3 062 597 - 82 303 #1 + 1182 #1^2 + 80 #1^3 + #1^4 &, 1]
```

`ToNumberField[`a`,`θ`]`	express the algebraic number a in the number field generated by θ
`ToNumberField[{`a_1`,`a_2`,`…`},`θ`]`	express the a_i in the field generated by θ
`ToNumberField[{`a_1`,`a_2`,`…`}]`	express the a_i in a common extension field generated by a single algebraic number

Representing arbitrary algebraic numbers as elements of algebraic number fields.

`ToNumberField` can be used to find a common finite extension of rationals containing the given algebraic numbers.

In[7]:=
```
ToNumberField[{√2, √3}]
```

Out[7]=
$$\left\{\text{AlgebraicNumber}\left[\text{Root}\left[1 - 10\,\sharp 1^2 + \sharp 1^4\,\&,\, 4\right],\, \left\{0,\, -\frac{9}{2},\, 0,\, \frac{1}{2}\right\}\right],\ \text{AlgebraicNumber}\left[\text{Root}\left[1 - 10\,\sharp 1^2 + \sharp 1^4\,\&,\, 4\right],\, \left\{0,\, \frac{11}{2},\, 0,\, -\frac{1}{2}\right\}\right]\right\}$$

This represents $\sqrt{6}$ as an element of the field generated by `Root[1 - 10 #1^2 + #1^4 &, 4]`.

In[8]:=
```
ToNumberField[√6, Root[1 - 10 #^2 + #^4 &, 4]]
```

Out[8]=
$$\text{AlgebraicNumber}\left[\text{Root}\left[1 - 10\,\sharp 1^2 + \sharp 1^4\,\&,\, 4\right],\, \left\{-\frac{5}{2},\, 0,\, \frac{1}{2},\, 0\right\}\right]$$

Arithmetic within a fixed finite extension of rationals is much faster than arithmetic within the field of all complex algebraic numbers.

Suppose you need to find the value of rational function f with $\{x, y, z\}$ replaced by algebraic numbers $\{a, b, c\}$.

In[9]:=
```
{a, b, c} = {ⅈ, √2, Root[#^3 - 2 # + 3 &, 1]};
```
$$\mathbf{f} = \frac{-2\,y\,z\,\left(7 + x - y + z^2\right) + \left(6 + x^2 + 2\,y\right)\,\left(-11 + x\,y + z^2\right)}{2\,y\,z\,(-4 - x + 3\,y\,z) - \left(6 + x^2 + 2\,y\right)\,\left(2 - 2\,x + z^3\right)};$$

A direct computation of the value of f at $\{a, b, c\}$ using `RootReduce` takes a rather long time.

In[10]:=
```
RootReduce[f /. {x → a, y → b, z → c}] // Timing
```

Out[10]=
$$\{34.3301,\ \text{Root}[127\,463\,137\,729\,603\,858\,692 + 15\,069\,520\,316\,552\,576\,640\,\sharp 1 + 3\,151\,085\,417\,830\,482\,145\,156\,\sharp 1^2 - 10\,938\,243\,534\,840\,099\,267\,928\,\sharp 1^3 + 14\,492\,589\,303\,525\,156\,688\,533\,\sharp 1^4 - 7\,171\,605\,298\,335\,082\,808\,820\,\sharp 1^5 - 947\,445\,370\,794\,828\,405\,814\,\sharp 1^6 + 2\,510\,661\,531\,113\,587\,622\,448\,\sharp 1^7 - 606\,316\,032\,776\,880\,635\,517\,\sharp 1^8 - 100\,899\,537\,810\,316\,084\,288\,\sharp 1^9 + 74\,049\,398\,920\,051\,042\,942\,\sharp 1^{10} - 12\,985\,018\,306\,589\,245\,140\,\sharp 1^{11} + 879\,298\,673\,075\,259\,913\,\sharp 1^{12}\,\&,\ 4]\}$$

A faster alternative is to do the computation in a common algebraic number field containing $\{a, b, c\}$.

In[11]:=
```
({aa, bb, cc} = ToNumberField[{a, b, c}]) // Timing
```

Out[11]=

$$\begin{aligned}&\{0.048003,\\ &\{\text{AlgebraicNumber}[\text{Root}[648+2592\,\#1+3492\,\#1^2+1524\,\#1^3+217\,\#1^4-1152\,\#1^5-14\,\#1^6-72\,\#1^7+87\,\#1^8+\\ &\quad 12\,\#1^9-14\,\#1^{10}+\#1^{12}\,\&,4],\Big\{\frac{244\,141}{94\,827},\frac{12\,086\,198}{1\,991\,367},\frac{7\,515\,071}{3\,982\,734},\frac{42\,845\,617}{35\,844\,606},-\frac{26\,501\,665}{11\,948\,202},\\ &\quad \frac{1\,373\,087}{17\,922\,303},-\frac{718\,309}{3\,982\,734},\frac{890\,062}{5\,974\,101},\frac{8969}{284\,481},-\frac{926\,321}{35\,844\,606},-\frac{3503}{5\,974\,101},\frac{34\,196}{17\,922\,303}\Big\}],\\ &\text{AlgebraicNumber}[\text{Root}[648+2592\,\#1+3492\,\#1^2+1524\,\#1^3+217\,\#1^4-1152\,\#1^5-14\,\#1^6-72\,\#1^7+\\ &\quad 87\,\#1^8+12\,\#1^9-14\,\#1^{10}+\#1^{12}\,\&,4],\Big\{-\frac{196\,718}{94\,827},-\frac{688\,153}{284\,481},-\frac{1\,293\,697}{568\,962},\frac{3\,857\,569}{5\,120\,658},\\ &\quad \frac{3\,032\,287}{5\,120\,658},\frac{1\,444\,985}{7\,680\,987},\frac{4897}{1\,706\,886},-\frac{224\,722}{2\,560\,329},\frac{2477}{853\,443},\frac{55\,031}{5\,120\,658},-\frac{2143}{2\,560\,329},-\frac{5212}{7\,680\,987}\Big\}],\\ &\text{AlgebraicNumber}[\text{Root}[648+2592\,\#1+3492\,\#1^2+1524\,\#1^3+217\,\#1^4-\\ &\quad 1152\,\#1^5-14\,\#1^6-72\,\#1^7+87\,\#1^8+12\,\#1^9-14\,\#1^{10}+\#1^{12}\,\&,4],\\ &\quad \Big\{-\frac{47\,423}{94\,827},-\frac{5\,277\,760}{1\,991\,367},\frac{770\,404}{1\,991\,367},-\frac{34\,924\,300}{17\,922\,303},\frac{29\,139\,493}{17\,922\,303},-\frac{14\,234\,156}{53\,766\,909},\frac{1\,060\,324}{5\,974\,101},\\ &\quad -\frac{1\,097\,132}{17\,922\,303},-\frac{29\,384}{853\,443},\frac{90\,184}{5\,974\,101},\frac{25\,510}{17\,922\,303},-\frac{66\,104}{53\,766\,909}\Big\}]\}\}\end{aligned}$$

Arithmetic within the common number field is much faster.

In[12]:=

$$\left(\mathbf{d}=\frac{-2\,y\,z\,(7+x-y+z^2)+(6+x^2+2\,y)\,(-11+x\,y+z^2)}{2\,y\,z\,(-4-x+3\,y\,z)-(6+x^2+2\,y)\,(2-2\,x+z^3)}\;/.\;\{x\to aa,\,y\to bb,\,z\to cc\}\right)\;//\;\mathbf{Timing}$$

Out[12]=

$$\begin{aligned}&\{0.036002,\text{AlgebraicNumber}[\text{Root}[\\ &\quad 648+2592\,\#1+3492\,\#1^2+1524\,\#1^3+217\,\#1^4-1152\,\#1^5-14\,\#1^6-72\,\#1^7+87\,\#1^8+12\,\#1^9-14\,\#1^{10}+\#1^{12}\,\&,\\ &\quad 4],\Big\{-\frac{3\,860\,776\,239\,867\,194\,137\,278}{3\,970\,535\,965\,319\,412\,941\,431},-\frac{53\,260\,812\,035\,714\,120\,989\,033}{11\,911\,607\,895\,958\,238\,824\,293},-\frac{109\,038\,458\,622\,656\,664\,030\,115}{71\,469\,647\,375\,749\,432\,945\,758},\\ &\quad -\frac{192\,381\,933\,793\,750\,243\,587\,991}{107\,204\,471\,063\,624\,149\,418\,637},\frac{70\,556\,676\,211\,663\,475\,835\,676}{35\,734\,823\,687\,874\,716\,472\,879},-\frac{106\,803\,727\,028\,468\,004\,471\,691}{964\,840\,239\,572\,617\,344\,767\,733},\\ &\quad \frac{7\,665\,080\,170\,226\,573\,969\,564}{35\,734\,823\,687\,874\,716\,472\,879},-\frac{36\,535\,823\,424\,460\,554\,318\,055}{321\,613\,413\,190\,872\,448\,255\,911},-\frac{124\,880\,404\,825\,784\,359\,957}{3\,403\,316\,541\,702\,353\,949\,798},\\ &\quad \frac{7\,067\,040\,798\,332\,263\,363\,508}{321\,613\,413\,190\,872\,448\,255\,911},\frac{7\,357\,016\,108\,927\,986\,451}{5\,104\,974\,812\,553\,530\,924\,697},-\frac{1\,522\,619\,721\,558\,874\,444\,783}{964\,840\,239\,572\,617\,344\,767\,733}\Big\}]\}\end{aligned}$$

Converting the resulting `AlgebraicNumber` object to a `Root` object is fast as well.

In[13]:=
```
RootReduce[d] // Timing
```

Out[13]=

$$\begin{aligned}&\{0.044003,\text{Root}[127\,463\,137\,729\,603\,858\,692+15\,069\,520\,316\,552\,576\,640\,\#1+\\ &\quad 3\,151\,085\,417\,830\,482\,145\,156\,\#1^2-10\,938\,243\,534\,840\,099\,267\,928\,\#1^3+\\ &\quad 14\,492\,589\,303\,525\,156\,688\,533\,\#1^4-7\,171\,605\,298\,335\,082\,808\,820\,\#1^5-947\,445\,370\,794\,828\,405\,814\,\#1^6+\\ &\quad 2\,510\,661\,531\,113\,587\,622\,448\,\#1^7-606\,316\,032\,776\,880\,635\,517\,\#1^8-100\,899\,537\,810\,316\,084\,288\,\#1^9+\\ &\quad 74\,049\,398\,920\,051\,042\,942\,\#1^{10}-12\,985\,018\,306\,589\,245\,140\,\#1^{11}+879\,298\,673\,075\,259\,913\,\#1^{12}\,\&,4]\}\end{aligned}$$

`ToNumberField[{`a_1`, `a_2`, …}]` is equivalent to `ToNumberField[{`a_1`, `a_2`, …}, Automatic]`, and does not necessarily use the smallest common field extension. `ToNumberField[{`a_1`, `a_2`, …}, All]` always uses the smallest common field extension.

Here the first `AlgebraicNumber` object is equal to $\sqrt{2}$ so it does not generate the 4th-degree field $\mathbb{Q}$ (`Root[1 - 10 #1^2 + #1^4 &, 4]`) it is represented in. However, the common field found by `ToNumberField` contains the whole field $\mathbb{Q}$ (`Root[1 - 10 #1^2 + #1^4 &, 4]`).

In[14]:=
```
ToNumberField[{AlgebraicNumber[Root[1 - 10 #^2 + #^4 &, 4], {0, -9/2, 0, 1/2}], Sqrt[5]}]
```

Out[14]=
```
{AlgebraicNumber[Root[576 - 960 #1^2 + 352 #1^4 - 40 #1^6 + #1^8 &, 8], {0, 5/3, 0, -7/72, 0, -7/144, 0, 1/576}],
 AlgebraicNumber[Root[576 - 960 #1^2 + 352 #1^4 - 40 #1^6 + #1^8 &, 8], {0, -53/12, 0, 95/36, 0, -97/288, 0, 5/576}]}
```

Specifying the second argument `All` makes `ToNumberField` find the smallest field possible.

In[15]:=
```
ToNumberField[{AlgebraicNumber[Root[1 - 10 #^2 + #^4 &, 4], {0, -9/2, 0, 1/2}], Sqrt[5]}, All]
```

Out[15]=
```
{AlgebraicNumber[Root[9 - 14 #1^2 + #1^4 &, 4], {0, -11/6, 0, 1/6}],
 AlgebraicNumber[Root[9 - 14 #1^2 + #1^4 &, 4], {0, 17/6, 0, -1/6}]}
```

`MinimalPolynomial[`a`]`	give a pure function representation of the minimal polynomial over the integers of the algebraic number a
`MinimalPolynomial[`a`,`x`]`	give the minimal polynomial of the algebraic number a as a polynomial in x
`AlgebraicIntegerQ[`a`]`	give `True` if the algebraic number a is an algebraic integer and `False` otherwise
`AlgebraicNumberDenominator[`a`]`	give the smallest positive integer n such that na is an algebraic integer
`AlgebraicNumberTrace[`a`]`	give the trace of the algebraic number a
`AlgebraicNumberNorm[`a`]`	give the norm of the algebraic number a
`AlgebraicUnitQ[`a`]`	give `True` if the algebraic number a is an algebraic unit and `False` otherwise
`RootOfUnityQ[`a`]`	give `True` if the algebraic number a is a root of unity and `False` otherwise

Functions for computing algebraic number properties.

The minimal polynomial of an algebraic number a is the lowest-degree polynomial f with integer coefficients and the smallest positive leading coefficient, such that $f(a) == 0$.

This gives the minimal polynomial of $\sqrt{2}+\sqrt{3}$ expressed as a pure function.

In[16]:=
```
MinimalPolynomial[√2 + √3]
```
Out[16]=
```
1 - 10 #1^2 + #1^4 &
```

This gives the minimal polynomial of `Root[#1^5 - 2 #1 + 7 &, 1]^2 + 1` expressed as a polynomial in x.

In[17]:=
```
MinimalPolynomial[Root[#^5 - 2 # + 7 &, 1]^2 + 1, x]
```
Out[17]=
```
-50 - 3 x + 2 x^2 + 6 x^3 - 5 x^4 + x^5
```

An algebraic number is an algebraic integer if and only if its `MinimalPolynomial` is monic.

This shows that $\frac{1}{2}\left(1+\sqrt{5}\right)$ is an algebraic integer.

In[18]:=
```
AlgebraicIntegerQ[1/2 (1 + √5)]
```
Out[18]=
```
True
```

This shows that $\frac{1}{4}\left(1+\sqrt{5}\right)$ is not an algebraic integer.

In[19]:=
```
AlgebraicIntegerQ[1/4 (1 + √5)]
```
Out[19]=
```
False
```

This gives the smallest positive integer n for which $n\left(1+\sqrt{5}\right)/4$ is an algebraic integer.

In[20]:=
```
AlgebraicNumberDenominator[1/4 (1 + √5)]
```
Out[20]=
```
2
```

The trace of an algebraic number a is the sum of all roots of `MinimalPolynomial[`a`]`.

This gives the trace of $(-1)^{1/7}$.

In[21]:=
```
AlgebraicNumberTrace[(-1)^(1/7)]
```
Out[21]=
```
1
```

The norm of an algebraic number a is the product of all roots of `MinimalPolynomial[`a`]`.

This gives the norm of $\sqrt{3}+\sqrt{5}$.

In[22]:= **AlgebraicNumberNorm**$\left[\sqrt{3}+\sqrt{5}\right]$

Out[22]= 4

An algebraic number a is an algebraic unit if and only if both a and $1/a$ are algebraic integers, or equivalently, if and only if `AlgebraicNumberNorm[`a`]` is 1 or -1.

This shows that `GoldenRatio` is an algebraic unit.

In[23]:= **AlgebraicUnitQ[GoldenRatio]**

Out[23]= True

This shows that `AlgebraicNumber[Root[`$\#1^3-4\#1+17$` &, 1], {1, 2, 3}]` is not an algebraic unit.

In[24]:= **AlgebraicUnitQ**$\left[\right.$**AlgebraicNumber**$\left[\right.$**Root**$\left[\#^3-4\#+17\right.$ **&, 1**$\left.\right]$**, {1, 2, 3}**$\left.\left.\right]\right]$

Out[24]= False

An algebraic number a is a root of unity if and only if $a^n == 1$ for some integer n.

This shows that $\left(\sqrt{2+\sqrt{2}}+i\sqrt{2-\sqrt{2}}\right)\Big/2$ is a root of unity.

In[25]:= **RootOfUnityQ**$\left[\frac{1}{2}\left(\sqrt{2+\sqrt{2}}+i\sqrt{2-\sqrt{2}}\right)\right]$

Out[25]= True

`MinimalPolynomial[`s`,`x`,Extension->`a`]`	give the characteristic polynomial of the algebraic number s over the field $\mathbb{Q}[a]$
`MinimalPolynomial[`s`,`x`,Extension->Automatic]`	give the characteristic polynomial of the `AlgebraicNumber` object s over the number field generated by its first argument
`AlgebraicNumberTrace[`a`,Extension->`θ`]`	give the trace of the algebraic number a over the field $\mathbb{Q}[\theta]$
`AlgebraicNumberTrace[`a`,Extension->Automatic]`	give the trace of the `AlgebraicNumber` object a over the number field generated by its first argument
`AlgebraicNumberNorm[`a`,Extension->`θ`]`	give the norm of the algebraic number a over the field $\mathbb{Q}[\theta]$
`AlgebraicNumberNorm[`a`,Extension->Automatic]`	give the norm of the `AlgebraicNumber` object a over the number field generated by its first argument

Functions for computing properties of elements of algebraic number fields.

If a is `AlgebraicNumber[`θ`, `*coeffs*`]`, then `MinimalPolynomial[`a`, `x`, Extension -> Automatic]` is equal to `MinimalPolynomial[`a`, `x`]`^{d}, where d is the extension degree of $\mathbb{Q}(\theta)/\mathbb{Q}(a)$.

The characteristic polynomial of $\sqrt{2}$, represented as an element of an extension of rationals of degree 4, is the square of `MinimalPolynomial` of $\sqrt{2}$.

```
In[26]:= a = AlgebraicNumber[Root[1 - 10 #^2 + #^4 &, 4], {0, -9/2, 0, 1/2}];
         MinimalPolynomial[a, x]
         MinimalPolynomial[a, x, Extension → Automatic] // Factor
```

Out[26]= $-2 + x^2$

Out[26]= $\left(-2 + x^2\right)^2$

The trace of an algebraic number is the sum of all roots of its characteristic polynomial. If a is `AlgebraicNumber[`θ`, `*coeffs*`]`, then `AlgebraicNumberTrace[`a`, Extension -> Automatic]` is equal to d `AlgebraicNumberTrace[`a`]`, where d is the extension degree of $\mathbb{Q}(\theta)/\mathbb{Q}(a)$.

The trace of $\sqrt{2}+1$, represented as an element of an extension of rationals of degree 4, is twice the AlgebraicNumberTrace of $\sqrt{2}+1$.

In[27]:=
```
a = AlgebraicNumber[Root[1 - 10 #^2 + #^4 &, 4], {1, -9/2, 0, 1/2}];
AlgebraicNumberTrace[a]
AlgebraicNumberTrace[a, Extension → Automatic]
```

Out[27]= 2

Out[27]= 4

The norm of an algebraic number is the product of all roots of its characteristic polynomial. If a is AlgebraicNumber[θ, *coeffs*], then AlgebraicNumberNorm[a, Extension -> Automatic] is equal to AlgebraicNumberNorm[a]^d, where d is the extension degree of $\mathbb{Q}(\theta)/\mathbb{Q}(a)$.

The norm of $\sqrt{2}+5$, represented as an element of an extension of rationals of degree 4, is the square of AlgebraicNumberNorm of $\sqrt{2}+5$.

In[28]:=
```
a = AlgebraicNumber[Root[1 - 10 #^2 + #^4 &, 4], {5, -9/2, 0, 1/2}];
AlgebraicNumberNorm[a]
AlgebraicNumberNorm[a, Extension → Automatic]
```

Out[28]= 23

Out[28]= 529

`NumberFieldIntegralBasis[`a`]`	give an integral basis for the field $\mathbb{Q}[a]$ generated by the algebraic number a
`NumberFieldRootsOfUnity[`a`]`	give the roots of unity for the field $\mathbb{Q}[a]$ generated by the algebraic number a
`NumberFieldFundamentalUnits[`a`]`	give a list of fundamental units for the field $\mathbb{Q}[a]$ generated by the algebraic number a
`NumberFieldNormRepresentatives[`a`,`m`]`	give a list of representatives of classes of algebraic integers of norm $\pm m$ in the field $\mathbb{Q}[a]$ generated by the algebraic number a
`NumberFieldSignature[`a`]`	give the signature of the field $\mathbb{Q}[a]$ generated by the algebraic number a
`NumberFieldDiscriminant[`a`]`	give the discriminant of the field $\mathbb{Q}[a]$ generated by the algebraic number a
`NumberFieldRegulator[`a`]`	give the regulator of the field $\mathbb{Q}[a]$ generated by the algebraic number a
`NumberFieldClassNumber[`a`]`	gives the class number of a number field $\mathbb{Q}[a]$ generated by an algebraic number a

Functions of computing properties of algebraic number fields.

An integral basis of an algebraic number field K is a list of algebraic numbers forming a basis of the $\mathbb{Z}$-module of the algebraic integers of K. The set $\{a_1, \ldots, a_n\}$ is an integral basis of an algebraic number field K if and only if $a_i \in K$ are algebraic integers, and every algebraic integer $z \in K$ can be uniquely represented as

$$z = k_1\, a_1 + \ldots + k_n\, a_n$$

with integer coefficients k_i.

Here is an integral basis of $\mathbb{Q}(18^{1/3})$.

```
In[29]:= NumberFieldIntegralBasis[18^(1/3)]
Out[29]= {1, AlgebraicNumber[Root[-18 + #1^3 &, 1], {0, 1, 0}],
          AlgebraicNumber[Root[-18 + #1^3 &, 1], {0, 0, 1/3}]}
```

This gives an integral basis of the field generated by the first root of `533 + 429 #1 + 18 #1^2 + #1^3 &`.

```
In[30]:= NumberFieldIntegralBasis[Root[533 + 429 # + 18 #^2 + #^3 &, 1]]

Out[30]= {1, AlgebraicNumber[Root[533 + 429 #1 + 18 #1^2 + #1^3 &, 1], {0, 1, 0}],
 AlgebraicNumber[Root[533 + 429 #1 + 18 #1^2 + #1^3 &, 1], {742/759, 94/759, 1/759}]}
```

`NumberFieldIntegralBasis` allows specifying the number field by giving a polynomial and a root number.

```
In[31]:= NumberFieldIntegralBasis[533 + 429 # + 18 #^2 + #^3 &, 1]

Out[31]= {1, AlgebraicNumber[Root[533 + 429 #1 + 18 #1^2 + #1^3 &, 1], {0, 1, 0}],
 AlgebraicNumber[Root[533 + 429 #1 + 18 #1^2 + #1^3 &, 1], {742/759, 94/759, 1/759}]}
```

This gives the roots of unity in the field generated by `Root[9 - 2 #^2 + #^4 &, 4]`.

```
In[32]:= NumberFieldRootsOfUnity[Root[9 - 2 #^2 + #^4 &, 4]]

Out[32]= {-1, 1, AlgebraicNumber[Root[9 - 2 #1^2 + #1^4 &, 4], {-1/4, -5/12, 1/4, 1/12}],
 AlgebraicNumber[Root[9 - 2 #1^2 + #1^4 &, 4], {-1/4, 5/12, 1/4, -1/12}],
 AlgebraicNumber[Root[9 - 2 #1^2 + #1^4 &, 4], {0, -1/6, 0, -1/6}],
 AlgebraicNumber[Root[9 - 2 #1^2 + #1^4 &, 4], {0, 1/6, 0, 1/6}],
 AlgebraicNumber[Root[9 - 2 #1^2 + #1^4 &, 4], {1/4, -5/12, -1/4, 1/12}],
 AlgebraicNumber[Root[9 - 2 #1^2 + #1^4 &, 4], {1/4, 5/12, -1/4, -1/12}]}
```

Here are all roots of unity in the field $\mathbb{Q}\left(1 + i\sqrt{3}\right)$.

```
In[33]:= NumberFieldRootsOfUnity[1 + i Sqrt[3]]

Out[33]= {-1, 1, AlgebraicNumber[1 + i Sqrt[3], {-1, 1/2}], AlgebraicNumber[1 + i Sqrt[3], {0, -1/2}],
 AlgebraicNumber[1 + i Sqrt[3], {0, 1/2}], AlgebraicNumber[1 + i Sqrt[3], {1, -1/2}]}
```

$\{u_1, \ldots, u_n\}$ is a list of fundamental units of an algebraic number field K if and only if $u_i \in K$ are algebraic units, and every algebraic unit $u \in K$ can be uniquely represented as

$$u = \xi\, u_1^{n_1} \cdots u_t^{n_t}$$

with a root of unity ξ and integer exponents n_i.

Here is a set of fundamental units of the field generated by the third root of $\sharp 1^4 - 10\ \sharp 1^2 + 1$ &.

```
In[34]:= NumberFieldFundamentalUnits[Root[#^4 - 10 #^2 + 1 &, 3]]
```

```
Out[34]= {AlgebraicNumber[Root[1 - 10 #1^2 + #1^4 &, 3], {5/4, 9/4, -1/4, -1/4}],
 AlgebraicNumber[Root[1 - 10 #1^2 + #1^4 &, 3], {-1, 9/2, 0, -1/2}],
 AlgebraicNumber[Root[1 - 10 #1^2 + #1^4 &, 3], {0, 1, 0, 0}]}
```

This gives a fundamental unit of the quadratic field $\mathbb{Q}\left(\sqrt{21}\right)$.

```
In[35]:= NumberFieldFundamentalUnits[Sqrt[21]]
```

```
Out[35]= {AlgebraicNumber[Sqrt[21], {5/2, 1/2}]}
```

This gives a set of representatives of classes of elements of norm 9 in the field generated by the first root of $\sharp 1^2 - 7$ &.

```
In[36]:= NumberFieldNormRepresentatives[Root[#^2 - 7 &, 1], 9]
```

```
Out[36]= {3, AlgebraicNumber[-Sqrt[7], {-4, -1}], AlgebraicNumber[-Sqrt[7], {-4, 1}]}
```

Here is a set of representatives of classes of elements of norm 2 in the field $\mathbb{Q}\left(\sqrt{2} + \sqrt{3}\right)$.

```
In[37]:= NumberFieldNormRepresentatives[Sqrt[2] + Sqrt[3], 2]
```

```
Out[37]= {AlgebraicNumber[Root[1 - 10 #1^2 + #1^4 &, 4], {-9/4, 9/4, 1/4, -1/4}]}
```

This shows that the polynomial $\sharp^5 + \sharp^4 + \sharp^3 + \sharp^2 + 1$ & has 1 real root and 2 conjugate pairs of complex roots.

```
In[38]:= NumberFieldSignature[Root[#^5 + #^4 + #^3 + #^2 + 1 &, 1]]
Out[38]= {1, 2}
```

This shows that the field $\mathbb{Q}[a]$ has 12 real embeddings and 6 conjugate pairs of complex embeddings.

```
In[39]:= a = Sqrt[2] + Root[#^3 - 11 # - 2 &, 1] + AlgebraicNumber[Root[#^4 - 3 # + 1 &, 2], {1, 2, 3}];
         NumberFieldSignature[a]
Out[39]= {12, 6}
```

The discriminant of a number field K is the discriminant of an integral basis $\{a_1, \ldots, a_n\}$ of K (i.e., the determinant of the matrix with elements `AlgebraicNumberTrace[`$a_i\, a_j$`, Extension -> Automatic]`). The value of the determinant does not depend on the choice of integral basis.

Here is the discriminant of $\mathbb{Q}\left(2 - \sqrt{3} + 5^{1/4}\right)$.

```
In[40]:= NumberFieldDiscriminant[2 - √3 + 5^(1/4)]
Out[40]= 5 184 000 000
```

This gives the discriminant of the field generated by a root of the polynomial `#^5 + #^4 + #^3 + #^2 + 1 &`. The value of the discriminant does not depend on the choice of the root; hence, `NumberFieldDiscriminant` allows specifying just the polynomial.

```
In[41]:= NumberFieldDiscriminant[#^5 + #^4 + #^3 + #^2 + 1 &]
Out[41]= 2297
```

The regulator of a number field K is the lattice volume of the image of the group of units of K under the logarithmic embedding

$$K \setminus \{0\} \ni x \longrightarrow \{\mathrm{Log}[\mathrm{Abs}[\sigma_1(x)]], \ldots, \mathrm{Log}[\mathrm{Abs}[\sigma_s(x)]], 2\,\mathrm{Log}[\mathrm{Abs}[\sigma_{s+1}(x)]], \ldots, 2\,\mathrm{Log}[\mathrm{Abs}[\sigma_{s+t}(x)]]\} \in \mathbb{R}^{s+t},$$

where $\sigma_1, \ldots, \sigma_s$ are the real embeddings of K in $\mathbb{C}$, and $\sigma_{s+1}, \ldots, \sigma_{s+t}$ are one of each conjugate pair of the complex embeddings of K in $\mathbb{C}$.

Here is the regulator of $\mathbb{Q}\left(\sqrt{61}\right)$.

```
In[42]:= NumberFieldRegulator[√61]
Out[42]= Log[1/2 (39 + 5 √61)]
```

This gives the regulator of the field generated by a root of the polynomial `#1^3 - 3 #1^2 + 1 &`. The value of the regulator does not depend on the choice of the root; hence, `NumberFieldRegulator` allows specifying just the polynomial.

```
In[43]:= NumberFieldRegulator[#^3 - 3 #^2 + 1 &]
Out[43]= -Log[AlgebraicNumber[Root[1 - 3 #1^2 + #1^3 &, 1], {-1, -2, 1}]]
           Log[AlgebraicNumber[Root[1 - 3 #1^2 + #1^3 &, 2], {0, 3, -1}]] +
          Log[AlgebraicNumber[Root[1 - 3 #1^2 + #1^3 &, 1], {0, -3, 1}]]
           Log[AlgebraicNumber[Root[1 - 3 #1^2 + #1^3 &, 2], {1, 2, -1}]]
```

This gives the class number of $\mathbb{Q}\left(\sqrt{-71}\right)$.

In[44]:=
```
NumberFieldClassNumber[Sqrt[-71]]
```
Out[44]= 7

Solving Frobenius Equations and Computing Frobenius Numbers

A *Frobenius equation* is an equation of the form

$$a_1 x_1 + \ldots + a_n x_n == m,$$

where $a_1, \ldots, a_n$ are positive integers, m is an integer, and the coordinates $x_1, \ldots, x_n$ of solutions are required to be non-negative integers.

The *Frobenius number* of $a_1, \ldots, a_n$ is the largest integer m for which the Frobenius equation $a_1 x_1 + \ldots + a_n x_n == m$ has no solutions.

`FrobeniusSolve[{`a_1`,…,`a_n`},`b`]`	give a list of all solutions of the Frobenius equation $a_1 x_1 + \ldots + a_n x_n = b$
`FrobeniusSolve[{`a_1`,…,`a_n`},`b`,`m`]`	give m solutions of the Frobenius equation $a_1 x_1 + \ldots + a_n x_n = b$; if less than m solutions exist, give all solutions
`FrobeniusNumber[{`a_1`,…,`a_n`}]`	give the Frobenius number of $a_1, \ldots, a_n$

Functions for solving Frobenius equations and computing Frobenius numbers.

This gives all solutions of the Frobenius equation $12 x + 16 y + 20 z + 27 t == 123$.

```
In[1]:= FrobeniusSolve[{12, 16, 20, 27}, 123]

Out[1]= {{0, 1, 4, 1}, {0, 6, 0, 1}, {1, 4, 1, 1},
         {2, 2, 2, 1}, {3, 0, 3, 1}, {4, 3, 0, 1}, {5, 1, 1, 1}, {8, 0, 0, 1}}
```

This gives one solution of the Frobenius equation $12 x + 16 y + 20 z + 27 t == 123$.

```
In[2]:= FrobeniusSolve[{12, 16, 20, 27}, 123, 1]

Out[2]= {{8, 0, 0, 1}}
```

Here is the Frobenius number of $\{12, 16, 20, 27\}$, that is, the largest m for which the Frobenius equation $12 x + 16 y + 20 z + 27 t == m$ has no solutions.

```
In[3]:= FrobeniusNumber[{12, 16, 20, 27}]

Out[3]= 89
```

This shows that indeed, the Frobenius equation $12\,x + 16\,y + 20\,z + 27\,t == 89$ has no solutions.

In[4]:=
```
FrobeniusSolve[{12, 16, 20, 27}, 89, 1]
```
Out[4]=
```
{}
```

Here are all the ways of making 42 cents change using 1, 5, 10, and 25 cent coins.

In[5]:=
```
FrobeniusSolve[{1, 5, 10, 25}, 42]
```
Out[5]=
```
{{2, 0, 4, 0}, {2, 1, 1, 1}, {2, 2, 3, 0}, {2, 3, 0, 1}, {2, 4, 2, 0}, {2, 6, 1, 0}, {2, 8, 0, 0},
 {7, 0, 1, 1}, {7, 1, 3, 0}, {7, 2, 0, 1}, {7, 3, 2, 0}, {7, 5, 1, 0}, {7, 7, 0, 0},
 {12, 0, 3, 0}, {12, 1, 0, 1}, {12, 2, 2, 0}, {12, 4, 1, 0}, {12, 6, 0, 0}, {17, 0, 0, 1},
 {17, 1, 2, 0}, {17, 3, 1, 0}, {17, 5, 0, 0}, {22, 0, 2, 0}, {22, 2, 1, 0}, {22, 4, 0, 0},
 {27, 1, 1, 0}, {27, 3, 0, 0}, {32, 0, 1, 0}, {32, 2, 0, 0}, {37, 1, 0, 0}, {42, 0, 0, 0}}
```

Using 24, 29, 31, 34, 37, and 39 cent stamps, you can pay arbitrary postage of more than 88 cents.

In[6]:=
```
FrobeniusNumber[{24, 29, 31, 34, 37, 39}]
```
Out[6]=
```
88
```