

How to get rid of quantifiers?

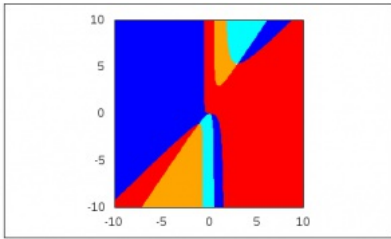
Reinhard Oldenburg, Michele Artigue

Angaben zur Veröffentlichung / Publication details:

Oldenburg, Reinhard, and Michele Artigue. 2014. "How to get rid of quantifiers?" Klein Project Blog, no. 13. Oktober 2014. <http://blog.kleinproject.org/?p=2466>.

How to get rid of quantifiers?

2014-10-13 14:10:36 Sarah Spoenemann



Originating authors are Reinhard Oldenburg and Michele Artigue.

How do computer packages do abstract algebraic problems such as proving statements “for all x ” or finding whether a Real Number x with certain conditions exists?

Recent advances draw on theorems in mathematical logic, as well as improvements in computing.

High school students learn how to solve problems such as the following: « For what values of the real number c , does the polynomial $P(x) = x^2 + cx + c$ have two distinct real roots ? », and algorithmically get the answer: $c < 0$ or $c > 4$. Doing so, they have in some sense, found a way of transforming the sentence expressing the problem ($\exists x_1 \exists x_2 (x_1 \neq x_2 \text{ and } P(x_1) = 0 \text{ and } P(x_2) = 0)$) involving the two existential quantifiers ‘there exists x_1 ’ and ‘there exists x_2 ’ into the sentence $c < 0$ or $c > 4$ which no longer includes quantifiers. For what kind of problems is this possible, theoretically but also practically with an effective computer program? In 1938, thanks to a theorem of elimination of quantifiers proved by the logician Alfred Tarski, a decisive step was achieved regarding these questions, but this was not at all the end of the story... READ MORE

Since the most ancient times, mathematicians have tried to invent algorithms allowing them to solve automatically classes of mathematical problems. For instance, systematic techniques for solving problems that we model by quadratic equations today, Euclid’s algorithm for deciding if two numbers are relatively primes, were already known more than thousand years ago. However, we also know that a general algorithm cannot be found for solving algebraic equations beyond degree 4, and since Gödel’s incompleteness theorem that it is impossible to have a procedure for deciding about the truth of all the formulas of any theory containing elementary number theory. The development of mathematical logic and its connection with computer sciences however produced new tools for approaching these questions and substantial advances. This vignette will tell you some pieces of this story.

Many problems in mathematics can be expressed in form of equations or inequalities universally or existentially quantified as is the case for the problem above, or involving more complex forms of quantifications. For instance, expressing that a particular function f is continuous at a given point x_0 requires the combination of universal (for every) and existential (there exists) quantifications:

$$\forall \epsilon > 0 \exists \delta > 0 \forall x : |x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon$$

What Tarski developed in the thirties was a method for quantifier elimination for a particular theory, that of real closed fields whose paradigmatic case is the field of real numbers. Doing so, he proved that the truth of any closed sentence of this theory can be decided, but also that sentences containing parameters (as in the introductory example in which c is a parameter) can be transformed into equivalent statements in the form of finite sets of conditions on these parameters. However Tarski’s method was very complex and could not lead to a practicable algorithm. More research was needed in order to reach the current stage in which effective algorithms are incorporated in symbolic packages such as Mathematica or Maxima, as shown in the examples below.

In the first example, Mathematica is used for finding the values of the parameter c for which the function $f(x) = x^3 - cx^2 + cx + c$ has at least two distinct real roots. First the function is entered using the Mathematica syntax; then an expression is entered, expressing the problem in a formalized way, and finally the Resolve function that eliminates quantifiers is applied to this expression. The result displayed is $c \leq -1$ or $c \geq 27/5$.

```
In[1]:= f [x_] := x^3 - c * x^2 + c * x + c
```

```
In[2]:= expr = Exists[x0, Exists[x1, x0 != x1 && f[x0] == 0 && f[x1] == 0]]
```

```
Out[2]:=  $\exists x_0 \exists x_1 (x_0 \neq x_1 \ \&\& \ c + c x_0 - c x_0^2 + x_0^3 = 0 \ \&\& \ c + c x_1 - c x_1^2 + x_1^3 = 0)$ 
```

```
In[3]:= Resolve[expr, Reals]
```

```
Out[3]:=  $c \leq -1 \ || \ c \geq \frac{27}{5}$ 
```

In the second example, the same software is used for finding the values of c for which f admits a real root in which the curvature of f is positive, that is to say the second derivative of f is positive. This time the result is: $c \leq 0$ or $c \geq 27/5$.

```
In[4]:= Resolve[Exists[x0, f''[x0] > 0 && f[x0] == 0], Reals]
```

```
Out[4]:=  $c < 0 \ || \ c \geq \frac{27}{5}$ 
```

In the third example, the software Maxima is used for deciding about the continuity of the polynomial $p_1(x) = (x - 5)^2 + x + 1$ and the rational function $r_1(x) = x/(x - 3)$, first at point 2, then at a generic point a . Once again, the continuity of a function f at a given point x_0 is expressed in a formalized way that translates the definition given above into the Maxima syntax in which the universal quantifier is denoted by A and the existential quantifier by E . Note that the conditions expressed by absolute values in the usual definition are expressed here using inequalities between squares. Then the function for quantifier elimination is applied to this expression. In Maxima, it is called *qe*. It also applies to rational expressions as shown in the example. The answers provided tell us that the two functions are continuous for $x = 2$, but that the second one is only continuous in a for $a \neq 3$.

```

(%i26) continous(f,x0):= qe([[A,eps],[E,delta],[A,x]],
    (delta>0) %and ((eps>0) %implies (( (x-x0)^2<delta^2 ) %implies
        ((f(x)-f(x0))^2<eps^2) )))
(%o26) continous(f,x0):=
qe([[A,eps],[E,delta],[A,x]],delta>0 %and (eps>0 %implies ((x-x0)^2<delta^2 %implies (f(x)-f(x0))^2<eps^2)))

(%i27) p1(x):=(x-5)^2+x+1$ p2(x):=x^3-x$ r1(x):=x/(x-3)$
    r2(x):=1/((x-2)*(x-1))$ r3(x):=-1/(1+x^2)$

(%i32) continous(p1,2);
(%o32) true

(%i33) continous(r1,2);
(%o33) true

(%i22) continous(r1,a);
(%o22) a-3 # 0

```

How does it work?

What we have seen in these examples is the application of a method that proves advanced mathematical statements and is even guaranteed to be successful in a wide domain. So, how can computers do so such advanced “thinking”?

The language of the theory of real closed fields

The language we consider in the theory of real closed fields contains two operations $+$, $*$, an order relation $<$ and equality. It also contains two symbols for denoting the respective neutral elements of the two operations. Furthermore there are number constants, variables and quantifiers (by lopez). This is a first order language, that is to say that quantifiers only operate on variables, not on set of variables or function symbols. Even more restrictive is the condition that exponents can only be natural numbers (this excludes for example Fermat’s theorem from being proved in the context of the theory presented here).

The axioms of the theory are those ensuring that the structures satisfying these axioms are real closed fields, that is to say ordered fields in which every positive element has a square root and every odd degree polynomial has a root. Within this theory, new operations and relations can be defined, for instance subtraction, large order relationship, and new symbols introduced for denoting them, as well as for denoting particular terms, for instance integers, the inverse of a non zero element or the square root of a positive element. Most elementary sentences are of the form: $P = 0$, $P > 0$, $P < 0$, P being a polynomial expression in one or several variables, and their negations $P \neq 0$, $P \leq 0$, $P \geq 0$.

One first important point is that any sentence in this language can be transformed into an equivalent sentence:

- having all quantifiers in front, what logicians call a prenex formⁱⁱ
- and in which the remaining part is a disjunction of conjunctionsⁱⁱⁱ

Note that the sentences in the three examples above are already in prenex form but that only the two first ones are in a disjunctive-conjunctive form. We suggest the reader to produce such a form for the last example by using the two rewriting rules listed below, and then check the answer by comparing to the one provided in note^v.

R1: Replace $A \Rightarrow B$ by $\neg A \vee B$
R2: Replace $\neg(A \wedge B)$ by $\neg A \vee \neg B$

When a sentence is in prenex form, multiple quantifiers can be eliminated in turn, starting from the innermost. Noticing that, using its negation, the treatment of a sentence of the form $\forall x A(x)$ can be replaced by the treatment of the sentence $\neg \exists x \neg A(x)$, it finally comes out that what needs to be understood is how an existential quantifier can be eliminated from a sentence of the form $\exists x (A_1 \vee A_2 \vee \dots \vee A_n)$ in which each A_i is a conjunction of polynomial equalities and inequalities depending on x and possibly other variables. In fact, such a sentence is equivalent to $(\exists x A_1) \vee (\exists x A_2) \vee \dots \vee (\exists x A_n)$, thus what needs to be understood is how an existential quantifier can be eliminated from a sentence of the form $\exists x (P_1(x) = 0 \wedge P_2(x) > 0 \wedge P_3(x) < 0 \dots)$, the P_i being polynomial expressions possibly depending on other variables than x .

In this vignette, we restrict ourselves to the real number field as the most important case of closed real fields and the most connected to secondary mathematics. Solving the problem of elimination for it means that an algorithm can be developed for reducing the examination of the values of polynomials P_i over the real line to a finite number of cases. Sturm’s theorem and cylindrical algebraic decomposition (CAD) provide a solution.

Sturm’s theorem

A polynomial in one variable $P(x) = a_n x^n + \dots + a_0$ divides the real number line into regions depending on the sign of its value at particular points. There are at most n real roots, so these delimitate at most $n + 1$ open intervals on which $P(x)$ is either positive or negative, the finite endpoints of these intervals being the roots of $P(x)$. While there is no general formula to express the zeros exactly as mentioned above, the number of zeros in every interval can be counted exactly by the method given in Sturm’s theorem. Thus we can locate the different roots in intervals of arbitrary precision and be sure not to miss any. This is enough for the procedure of quantifier elimination to work.

When more than one variable comes into play, we have still a finite number of regions in the variable space where the polynomial at hand has constant sign. For instance let us come back to the first example. The function $f(x) = x^3 - cx^2 + cx + c$ can be interpreted as a polynomial in the two variables x and c , and the image below, obtained with the software Maxima visualizes the three regions associated to it in the square $[-10, 10] \times [-10, 10]$ of the plane (x, c) . The red color is associated to positive values of f and the blue color to negative values of f .

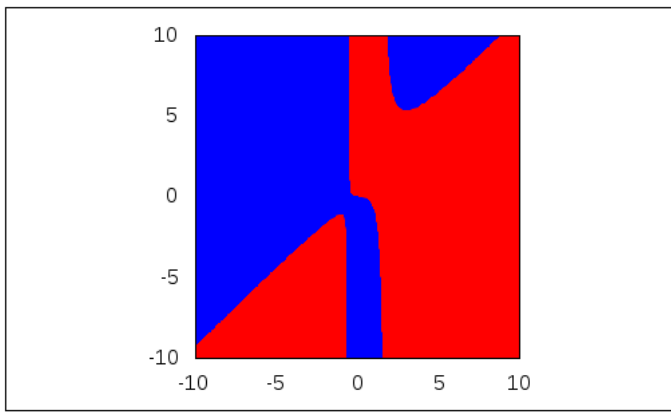


Figure 1: Regions associated to the polynomial $x^3 - cx^2 + cx + c$. generated by Maxima commands:

```
f(x):=x^3-c*x^2+c*x+c;
wxdraw2d(fill_color = red, region(f(x)>0 ,x,-10,10,c,-10,10), fill_color = blue, region(f(x)<0
,x,-10,10,c,-10,10));
```

[caption] Similar decompositions can be associated to sets of polynomials. Each region is then characterized by the fact that all polynomials in the set have constant sign in the whole region. The figure 2 below visualizes the regions associated to the previous polynomial and its second derivative on $[-10, 10] \times [-10, 10]$, also obtained with Maxima.

[caption id="attachment_82" align="aligncenter" width="576"]

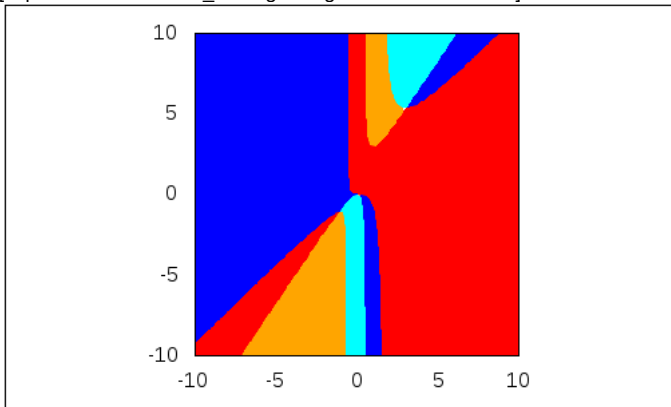


Figure 2:

Regions associated to the polynomials f and f''

The curves delimitating the different regions are pieces of the curves $c = \frac{x^3}{x^2 - x - 1}$ and $c = 3x$ respectively associated with the roots of f and of its second derivative. However, regions do not have an easy algebraic description even in such a simple case. To make the description simpler, the idea is to split up these regions into smaller cells that are easily described and make systematic computation straightforward. This leads to the notion of cylindrical algebraic decomposition.

Cylindrical algebraic decomposition

A cylindrical algebraic decomposition of R^n decomposes this space into a finite set of disjoint cells. Each point of R^n is thus included in exactly one cell. The cells have a special form defined recursively. For $n = 1$, a cell is either a single real number or an open interval, that is to say an interval of the form $]-\infty, a[$, $]a, b[$ or $]b, \infty[$. For $n = 2$, a cell is a cylinder over such a cell. It has one of the following forms: $\{(x, y) : x \in C \text{ and } y = f(x)\}$, $\{(x, y) : x \in C \text{ and } y > f(x)\}$, $\{(x, y) : x \in C \text{ and } y < f(x)\}$, $\{(x, y) : x \in C \text{ and } f(x) < y < g(x)\}$, with C a one dimension cell, f and g polynomials or rational functions. A cell in dimension n is defined in the same way, as a cylinder over a cell of dimension $n - 1$.

Remarkably, CAD can be performed algorithmically for all sets of polynomials in n variables resulting in a finite set of cells such that within each cell all polynomials have constant sign. The algorithm is complex and we can just try to give a rough idea of it. For a set P of polynomials in n variables x_1, \dots, x_n , the algorithm works by recursion on n : the procedure for n variables calls for the procedure for $n - 1$ variables. In fact, it includes the three following steps:

1. Projection: Construct a projected set P' of polynomials in the $n - 1$ variables x_1, \dots, x_{n-1} , from the set P .
2. Recursion step: Find a CAD D' for P' .
3. Extension: For each cell of D' , take a representative point and substitute its coordinates into the polynomials of P , yielding a set of polynomials in one variable x_n . Determine their real roots and use them to define cells over the cells of D' .

As explained above, for $n = 1$, thanks to Sturm's theorem, such a CAD can be obtained. If there is more than one polynomial, one just has to take the intersection of the cells calculated for each polynomial. A proof that this recursive procedure works is given in (Winkler, 1996).

Let us come back now to the particular examples mentioned above. For each set of polynomial conditions given as an entry, the CAD algorithm produces the cylindrical decomposition of the space associated to the polynomials involved, and returns the cells satisfying the given conditions. If we use for instance the CAD algorithm implemented in Mathematica with the two conditions $f > 0$ and $f'' > 0$, we obtain the following decomposition in dimension 2. We invite the reader to check that it fits the definition given above for such a decomposition (intervals with large inequalities can be seen as the union of cells with open intervals and of cells reduced to a single point) and that it algebraically describes the red regions of figure 2.

In[9]:= `CylindricalDecomposition[f[x] > 0 && f''[x] > 0, {x, c}]`

Out[9]=
$$\left(x \leq \frac{1}{4} (3 - \sqrt{33}) \ \&\& \ c < 3x \right) \ ||$$

$$\left(\frac{1}{4} (3 - \sqrt{33}) < x < \frac{1}{2} (1 - \sqrt{5}) \ \&\& \ c < \frac{x^3}{-1 - x + x^2} \right) \ ||$$

$$\left(0 < x < \frac{1}{2} (1 + \sqrt{5}) \ \&\& \ \frac{x^3}{-1 - x + x^2} < c < 3x \right) \ ||$$

$$\left(\frac{1}{2} (1 + \sqrt{5}) \leq x \leq \frac{1}{4} (3 + \sqrt{33}) \ \&\& \ c < 3x \right) \ ||$$

$$\left(x > \frac{1}{4} (3 + \sqrt{33}) \ \&\& \ c < \frac{x^3}{-1 - x + x^2} \right)$$

If we change the set of conditions, looking for the values of c for which the polynomial f has a real root for which the curvature is positive, we obtain of course different cells.

`CylindricalDecomposition[f[x] == 0 && f''[x] > 0, {c, x}]`

$$\left(c < \frac{3}{4} (3 - \sqrt{33}) \ \&\& \ (x = \text{Root}[c + c \#1 - c \#1^2 + \#1^3 \ \&, 2]) \ || \right.$$

$$\left. x = \text{Root}[c + c \#1 - c \#1^2 + \#1^3 \ \&, 3] \right) \ ||$$

$$\left(\frac{3}{4} (3 - \sqrt{33}) \leq c \leq -1 \ \&\& \ x = \text{Root}[c + c \#1 - c \#1^2 + \#1^3 \ \&, 3] \right) \ ||$$

$$\left(-1 < c < 0 \ \&\& \ x = \text{Root}[c + c \#1 - c \#1^2 + \#1^3 \ \&, 1] \right) \ ||$$

$$\left(c = \frac{27}{5} \ \&\& \ x = 3 \right) \ ||$$

$$\left(\frac{27}{5} < c < \frac{3}{4} (3 + \sqrt{33}) \ \&\& \ (x = \text{Root}[c + c \#1 - c \#1^2 + \#1^3 \ \&, 2]) \ || \right.$$

$$\left. x = \text{Root}[c + c \#1 - c \#1^2 + \#1^3 \ \&, 3] \right) \ ||$$

$$\left(c \geq \frac{3}{4} (3 + \sqrt{33}) \ \&\& \ x = \text{Root}[c + c \#1 - c \#1^2 + \#1^3 \ \&, 3] \right)$$

Note that here the variable x is in second position in the command used because it is the variable to be eliminated in this problem. The condition $f(x) = 0$ makes that all cylinders are of the type $C \times \{\alpha_i\}$ where C is a one dimension cell and α_i is one of the three roots of the polynomial. This cylindrical structure makes it easy to read off the conditions on c for each cell and to combine them for obtaining the final answer: $c < 0$ or $c^3 = 27/5$.

Doing the calculations by hand is only possible with toy problems, as the example we detail now. The reader not interested in these details can skip this part.

Problem: Find the values of x such as the statement $\exists y(P_1(x, y) > 0 \wedge P_2(x, y) > 0)$ is true with $P_1(x, y) = y - x$ and $P_2(x, y) = 1 - x^2y$.

The projection step of the CAD algorithm amounts to the calculation of discriminants and resultants of polynomials. We do not detail this step here, just give its result, the set $P' = \{1, x^2, 1 - x^3\}$.

In the second step, we have to produce a CAD for the set P' . The real roots of the polynomials in P' are 0 and 1. Thus the CAD decomposition D' associated to P' is the decomposition of \mathbb{R} into five cells: $]-\infty, 0[$, $\{0\}$, $]0, 1[$, $\{1\}$ and $]1, \infty[$.

The third step is the extension step. Possible sampling points for D' are: $-1, 0, 0.5, 1, 2$. Substituting these sampling points for x in P_1 and P_2 , we obtain the following set of pairs of polynomials: $\{\{1 + y, 1 - y\}, \{1, y\}, \{y - 1/2, 1 - y/4\}, \{y - 1, 1 - y\}, \{y - 2, 1 - 4y\}\}$.

The roots of these polynomials are easy to determine in this particular example. For instance, for the first pair the roots are -1 and 1 . Thus over the x -cell $]-\infty, 0[$ we will have the y -cells $]-\infty, -1[$, $\{-1\}$, $]-1, 1[$, $\{1\}$ and $]1, \infty[$. Altogether \mathbb{R}^2 is decomposed into $5 + 3 + 5 + 5 + 5 = 23$ cells.

Sampling points can now be associated with each of these cells. For instance, for the first five cells corresponding to the x -sampling point -1 , one can choose the sampling points $(-1, -2)$, $(-1, -1)$, $(-1, 0)$, $(-1, 1)$ and $(-1, 2)$. Plugging these sampling points in the polynomials P_1 and P_2 , one can sort out the cases in which the two polynomials are positive. Only three cases are found and they come from the first three x -cells $]-\infty, 0[$, $\{0\}$, $]0, 1[$. Thus the quantifier-free formula equivalent to $\exists y(P_1(x, y) > 0 \wedge P_2(x, y) > 0)$ is simply $x < 1$.

From algebra and analysis to geometry

Quantifier elimination and the CAD algorithm can be used for solving a diversity of problems involving polynomial functions, as many problems can be expressed in the language of closed real fields. These can be algebraic problems but also calculus problems regarding continuity as in the third example we have used, limits, extrema or variations. Moreover, some pre-processing also allows us to handle rational functions as $A/B = C$ is equivalent to $A = B \cdot C$ and $B \neq 0$, and square roots can be replaced in the obvious way: $\sqrt{A} = B$ is equivalent to $A = B^2$ and $B \geq 0$.

Moreover, as was already stressed by Tarski in his seminal work, a decision method for elementary algebra gives a decision method for Euclidian geometry as well. For instance, we can prove Thales' theorem saying that if a point C lies on the circle with diameter $[A, B]$, then the lines (AC) and (BC) are perpendicular, as shown below. Points A and B are respectively $(0, 0)$ and $(b, 0)$, which does not limit the generality of the proof, and C is (x, y) . The perpendicularity is expressed through the vanishing of the scalar product of vectors AC and CB . (Note that we could have used universal quantification for all four variables. The fact that even without them the formula reduces to true shows that the implication holds without restrictions on the variables.)

$$\left[\begin{array}{l} (\%i29) \text{qe}([], b=2*r \ \&\ \& \ (x-r)^2 + y^2 = r^2 \ \&\ \& \ \text{implies } x*(b-x) + y*(-y) = 0); \\ (\%o29) \text{true} \end{array} \right.$$

Note that geometrical proofs can be carried out using Groebner bases. These are tools from complex algebraic geometry and allow among other things the systematic solution of systems of polynomial equations – but not inequalities, as the complex number field is not ordered. Thus, not all statements of geometry can be translated in a form suitable for Groebner basis methods and even if they can, the method may say that a proposition is true even if some of the coordinates

need to be imaginary numbers and thus the statement does not hold in real geometry. The other way round, a statement may be true in real geometry but there may be counterexamples involving complex coordinates.

Issues of complexity

The complexity of the algorithms used for quantifier elimination has limited their availability and the computational complexity of CAD has limited its practical usefulness for many years. CAD algorithm is indeed doubly exponential in the number of variables of the polynomials involved. However, due to theoretical (better algorithms) as well as practical (faster computers) advances, the situation has improved, and is still moving, the field being an active field of research. Today, as pointed out in this vignette, CAD algorithms are implemented in many symbolic packages.

Different lessons can be drawn from this story. It shows that mathematical logic is not just a meta-mathematical domain dealing with the foundations of mathematics but a domain whose fundamental results have substantial applications in a diversity of mathematical domains. It illustrates the strong connections existing between mathematical logic and computer sciences, and the role played by decision problems and algorithmics in these. It shows the distance which may separate the theoretical proof of the existence of an algorithm from the construction of effective algorithms, the importance of the research work carried out for improving existing algorithms and extending them. It also shows the complexity of the algorithms which can be encapsulated in a given simple command. Secondary students are rarely exposed to the automatic proving of algebra or geometric results. However formalizing a mathematical problem to make it manageable by a computer program is not an exercise without interest even at secondary level. It requires a careful management of the mathematical language, including of quantifications which remain often implicit in the usual mathematical discourse. Moreover, interpreting the results provided by the computer program is not necessarily a trivial task as shown in the examples of this vignette.

References

Qepmax : <https://github.com/YasuhaiHonda/qepmax>: Maxima-qepcad-interface by Y. Honda with contributions by R. Oldenburg

A. Tarski (1948): *A decision method for elementary algebra and geometry*.

Rand Corporation Publication.

F. Winkler (1996): *Algorithms for Polynomials*. Springer.

St. Wolfram : *Mathematica – A system for doing mathematics by Computer*.

ⁱ http://en.wikipedia.org/wiki/Gödel%27s_incompleteness_theorems

ⁱⁱ This is achieved using properties of first order predicate calculus such as the followings (using the usual symbolic notations for universal and existential quantifiers (\forall and \exists), for the negation (\neg), and for the and (\wedge) and or (\vee) logical connectors): $A \wedge \exists x B(x) \Leftrightarrow \exists x(A \wedge B(x))$ is x does not appear in A and similar formulas with the universal quantifier and the connector \vee ; $\exists x A(x) \vee \exists x B(x) \Leftrightarrow \exists x(A(x) \vee B(x))$; $\exists x A(x) \wedge \exists x B(x) \Leftrightarrow \exists x \exists y(A(x) \wedge B(y))$, y being a variable not already present in A or B ; $\neg(\exists x A(x)) \Leftrightarrow \forall x \neg A(x)$.

ⁱⁱⁱ This is achieved by using the properties of logical connectors, such as those underlying the rules R1 to R5 below.

^{iv} $\forall \epsilon \exists \delta \forall x (\delta \leq 0 \vee \epsilon \leq 0 \vee (x - x_0)^2 \geq \delta^2 \vee (f(x) - f(x_0))^2 < \epsilon^2)$

^v Sturm's theorem expresses that the number of distinct real roots of a polynomial in a given interval is

equal to the difference in the number of changes of signs of the values of the Sturm's sequence at the two bounds of the interval. The Sturm's sequence of a polynomial P is the sequence obtained by applying Euclidean algorithm to P and its derivative P' . $P_0 = P$, $P_1 = P'$, P_2 is the opposite of the remainder in the Euclidean division of P_0 by P_1 , and so on. As the successive P_i are of decreasing degrees, the sequence is finite, and the final non zero polynomial of the sequence is the greatest common divisor of P and P' .

^{vi} For constructing this projection, one takes all leading coefficients of the polynomials of P when seen as polynomials in the variable x_n , as well as all discriminants and resultants that can be calculated from P .

^{vii} Note that, in this decomposition, the values $\frac{1}{2}(1 + \sqrt{5})$ and $\frac{1}{2}(1 - \sqrt{5})$ are the roots of the polynomial

$P(x) = x^2 - x - 1$, and the values $\frac{1}{4}(3 - \sqrt{33})$ and $\frac{1}{4}(3 + \sqrt{33})$ are the abscissa of the intersecting points of the curves $c = \frac{x^3}{x^2 - x - 1}$ and $c = 3x$

respectively associated with the roots of f and of its second derivative.

Share this:

- [Email](#)
- [Print](#)
- [Facebook](#)
- [Twitter](#)
-

This post is also available in: [Arabic](#)