IMPLEMENTING THEOREM PROVING IN GEOGEBRA BY EXACT CHECK OF A STATEMENT IN A BOUNDED NUMBER OF TEST CASES

ZOLTÁN KOVÁCS, TOMÁS RECIO, AND SIMON WEITZHOFER

Abstract. A method for the automatic and exact verification of a geometry statement is presented, relying on its validity on a bounded number of instances. This is achieved by merely performing some arithmetic computations with integer coordinates. This approach seems particularly suitable for dynamic geometry software such as GeoGebra, that includes some symbolic computation features.

Introduction

GeoGebra [1] is a widely used dynamic mathematics software for teaching elementary geometry. Its intuitive, multilingual user interface is an important reason for many teachers who choose mathematics software for illustrating geometry courses, and who use it to help the students towards experimenting and discovering different statements. The use of GeoGebra to help gaining intuition concerning geometric facts has already collected a wide interest from the mathematics education community. Here we go one step further, presenting an approach to the automatic and exact verification of geometric properties, which could be considered particularly adapted to GeoGebra features.

In this context, let us recall that, in general, there are several ways to prove a statement. A very popular implementation is to do a statistical checking, by changing the free points of the construction to a random set of instances: several dynamic geometry packages use this approach. In general, this method can give very good and quick results, but may be inaccurate or even false in some special cases. For example, computing intersections of the altitudes in a triangle can yield to inaccuracies due to numerical calculations, so that the statement will fail on a "visibly large set" of the free points [2]. On the other hand, even with exact computations this method can also fail: it could be that our random choice of instances, unluckily, are all non representative for the truth/falsity of some given statement. For example, it could be that the random check for the statement "all triangles are isosceles" of the free points is carried over... isosceles triangles!

A recent, but yet unpublished approach is to create the configuration by setting up the test points with carefully chosen coordinates, and bound the number of test cases depending on the degree of the statement polynomial and the number of variables. The main idea behind this method was proposed by the second author in 2002 [3]. Although this approach cannot exactly tell the non-degeneracy conditions, it does not require any symbolic computations, but calculating with arbitrary precision rational numbers. This means the calculation is very fast.

Second author partially supported by grants MTM2008-M04699-C03-03 and MTM2011-25816-C02-02.

Since GeoGebra is designed to work on a wide range of personal computers, it is crucial to use a fast and efficient method for its forthcoming theorem proving engine. Our aim is to create a prototype of such a prover subsystem, which is capable of proving theorems in elementary geometry, by using Recio's method.

1. A short explanation of the method

A two dimensional geometric construction consists of the set of free points and the dependent points. All the dependent points, having two coordinates, can be described by the coordinates of the free points. The statement s to be proved can be formulated by a rational equation on the involved coordinates, assuming that we have used only algebraic construction steps (e.g. midpoints, bisectors, perpendiculars, parallels). Let us write this equation into the form $p/q = 0$. Then, testing if the statement is true amounts to testing whether p is the constant zero polynomial, and $q \neq 0$. Unfortunately, at this moment, we will not be able to describe the set where q is not to be 0, but the vanishing of p can be investigated in a feasible way.

Calculating, in an explicit way, the polynomial p by following the construction steps could be not fast enough, since symbolic calculation may be, in general, too slow. What we can do easily is checking whether the polynomial evaluates to zero at given coordinates, namely by checking if the statement s is fulfilled in the construction with this coordinates. Of course we cannot evaluate s at all possible coordinates. Bézout's theorem for multivariate polynomials ensures that only a well chosen set of free points must be evaluated, and if for all of them we get zero, then p will vanish everywhere. The number of necessary points to check a statement depends on the number of free variables and on fixing an upper bound for the possible degree of the polynomial p.

Let us consider with some detail, given this abstract's space limitations, just the two dimensional case (that is, the case where only two independent variables rule the given construction). Then, we will have a polynomial p in two variables from which we know an upper limit d for its degree, and we can test for each point if the polynomial evaluates to zero. Now we want to test whether this polynomial is identically zero or not. If it is not zero, then it describes a curve of dimension less than or equal to d.

From Bézout's theorem we get that a curve of degree d and a line can have not more than d intersection points or the line is contained in the curve. If $d+1$ collinear points evaluate to zero, then the line determined by these points is contained in the curve. Thus we can test if a line is contained in the curve. If a curve of degree d contains d pairwise distinct lines, then the curve is the product of these lines, since the product of d lines has degree d and thus if it is contained in the curve, it coincides with it. No points outside these d lines can be contained in the curve.

To test whether the polynomial describes a curve or is constantly zero we can proceed as follows. Test if d pairwise different lines are contained by trying if the polynomial evaluates to zero at $d+1$ points on each of the lines. If at one point the polynomial does not evaluate to zero, it is obvious that p is not constantly zero. If at all these points the polynomial evaluates to zero, then we test if the polynomial vanishes at one point which is at none of the lines. If it does, we know that p does not describe a curve but is identically zero.

So we have to test the statement at $d+1$ different points on d different lines plus one point outside these lines. The minimal number of points to test is therefore $\binom{d+2}{2}$. This number can be reached if we choose $d+2$ pairwise non-parallel lines and test on the intersection points, since then there are $d+1$ points on d lines and one which is at none of these d lines. Obviously, the points can be chosen with very simple, integer, coordinates.

For many constructions the two dimensional case will be enough to investigate. Many geometric problems can be reduced to drawing a triangle and dragging just one of the points. This means two points can be fixed, and only we have to deal with the coordinates of the dragged point. This allows us to work in two dimensions in most cases.

2. Example

Let us prove that the altitudes of a triangle go through on a common point. Let ABC be the triangle, and D, E and F the feet of the altitudes going through A, B and C respectively. By using the Sage code below (also available at [4]) we can learn that by fixing A and B to $(-1,1)$ and $(2,3)$, respectively, and putting (c_1,c_2) into C, the statement $s = (AD \cap BE \cap CF \neq \emptyset)$ is—leaving aside some degenerate cases—equal to the vanishing of the determinant of the matrix whose entries are the coefficients of the following linear system

$$\begin{aligned} AD:\ -c_1+c_2-1+ \quad & (-c_1+2)x \quad +(-c_2+3)y \\ BE:\ 2c_1+3c_2-1+ \quad & (-c_1-1)x \quad +(-c_2+1)y \\ CF:\ 3c_1+2c_2+ \quad & (-3)x \quad +(-2)y \end{aligned}$$

Now s can be written as a polynomial p. Obviously, after expansion p will be the constant zero polynomial, but in theory it might be a degree up to three polynomial. So we want to avoid computing p; instead we would do exact tests for checking the vanishing of p at different positions of C, and we would have to do this in $\binom{3+2}{2} = 10$ cases. By using arbitrary precision arithmetic everywhere, if for each of the 10 (suitable arranged, as detailed in the previous section) points we get the result 0, then we can be sure if p vanishes on any $(c_1,c_2) \in \mathbf{R}^2$ point, so the statement s holds in general.

An interesting question is how to determine that p is at most of degree 3, without computing the polynomial p. Indeed, it is easy to observe that the coefficients of the objects in the construction have the following degrees in c_1 and c_2: $A \mapsto (0,0)$, $B \mapsto (0,0)$, $C \mapsto (1,1)$, $AD \mapsto (1,1,1)$, $BE \mapsto (1,1,1)$, $CF \mapsto (1,0,0)$, $s \mapsto 3$. Luckily, to compute these degrees we do not need the symbolic expressions for each object, but only the degrees of their parent objects and the way we create the new object from them. For example, CF is constructed by C, AB and perpendicularity, so in general we must get at most degree one for all coefficients of CF.

Sage code:

```
def line(A,B): return (B[0]-A[0])*(y-A[1])-(B[1]-A[1])*(x-A[0])
def cv(e): return e.substitute({x:0,y:0})
def xc(e): return e.substitute({y:0,x:1})-cv(e)
def yc(e): return e.substitute({x:0,y:1})-cv(e)
def perpendicular(A,e): return xc(e)*(y-A[1])-yc(e)*(x-A[0])
var('a1,a2,b1,c1,c2,b2,x,y'); A=(-1,1); B=(2,3); C=(c1,c2)
AB=line(A,B); BC=line(B,C); AC=line(A,C)
```

```
CF=perpendicular(C,AB); BE=perpendicular(B,AC); AD=perpendicular(A,BC)
M=matrix(SR,3,2,["AD",AD,"BE",BE,"CF",CF])
print "The equations for the altitudes:"; show(M)
N=matrix(SR,3,3,[cv(AD),xc(AD),yc(AD),cv(BE),\
 xc(BE),yc(BE),cv(CF),xc(CF),yc(CF)])
print "The matrix to check common points:"; show(N)
print "Check if this expression has degree <= 3:"; show(N.det())
print "By expanding the expression, we must get 0:"; show(expand(N.det()))
```

3. More variables

As an example of how to proceed in the general case, let us deal with the case of three variables. Then the polynomial describes a surface or it is identically zero. If a surface of degree d contains $d+1$ points in a line, it contains this line. If the surface contains d pairwise distinct lines in a plane and a point which is on the plane but not at one of the lines, it contains the plane. If it contains d pairwise distinct planes, it must coincide with the product of these planes and cannot contain a point outside these planes. If the polynomial vanishes at this point outside the planes, the polynomial must be the zero constant.

Inductively we can generalize this to higher dimensions. If a hypersurface in dimension n and degree d contains d pairwise distinct affine spaces of codimension two in a hyperplane, and a point on the hyperplane, but in none of those affine spaces, it must contain the whole hyperplane. If the hypersurface contains d pairwise different hyperplanes then it is the product of these hyperplanes and cannot contain a point outside the hyperplanes. If the polynomial of degree less or equal to d vanishes at this point, too, it cannot describe a hypersurface but has to be the zero constant and, thus, the underlying theorem is generally true.

References

[1] M. Hohenwarter. GeoGebra – ein Softwaresystem für dynamische Geometrie und Algebra der Ebene. Diplomarbeit, University of Salzburg, Austria (2002). See http://www.geogebra.org

[2] M. A. Abánades, F. Botana. An Automatic Deduction Environment for GeoGebra, Based on the Open Source Computer Algebra System Sage. 2nd International GeoGebra Conference. See http://www.geogebra.org/trac/attachment/ticket/1044/miguels_prover.ggb

[3] T. Recio. La mecánica de la demostración y la demostración mecánica. In: E. Palacián y J. Sancho (eds.): Actas X Jornadas para el Aprendizaje y la Enseñanza de las Matemáticas, Vol. I, Sociedad Aragonesa de Profesores de Matemáticas/ICE Universidad de Zaragoza, Zaragoza, pp. 189-212. 2002.

[4] Z. Kovács. Theorem proving with Recio's method. http://www.sagenb.org/home/pub/4299/

Johannes Kepler University of Linz
E-mail address: zoltan @ geogebra.org

University of Cantabria
E-mail address: tomas.recio @ unican.es

Johannes Kepler University of Linz
E-mail address: simon @ geogebra.org