Using jointly geometry and algebra to determine RC-constructibility

Pascal Schreck, Pascal Mathis

Using jointly geometry and algebra to determine RC-constructibility

Pascal Schreck

UFR de Mathématique et Informatique - ICube
7, rue René Descartes
67084, Strasbourg
France

Pascal Mathis

UFR de Mathématique et Informatique - ICube
7, rue René Descartes
67084, Strasbourg
France

Abstract

In most cases in geometry, applying analytic or algebraic tools on coordinates helps to solve some difficult problems. For instance, proving that a geometrical construction problem is solvable using ruler and compass is often impossible within a synthetic geometry framework. But in an analytic geometry framework, it is a direct application of Galois theory after performing triangularizations. However, these algebraic tools lead to a large amount of computation. Their implementation in modern Computer Algebra Systems (CAS) are still too time consuming to provide an answer in a reasonable time. In addition, they require a lot of memory space which can grow exponentially with the size of the problem. Fortunately, some geometrical properties can be used to setup the algebraic systems so that they can be more efficiently computed. These properties turn polynomials into new ones so as to reduce both the degrees and the number of monomials. The present paper promotes this approach by considering two corpora of geometric construction problems, namely Wernick's and Connely's lists. These lists contain about 280 problems. The purpose is to determine their status i.e. whether they are constructible or not with ruler and compass. Some of these problems had unknown status that will be settled in this paper. More generally, the status of all problems of these corpora are fully automatically given by an approach combining geometry and algebra.

Key words: Ruler and compass constructibility, triangle problems, geometric knowledge-based system, regular chains

1. Introduction

Geometric constructions are well-known by those who are interested in the epistemology of mathematics. Indeed, they played a key role in the definition and the understanding of numbers in Ancient Greece. Besides, they also have a practical aspect in several technical domains like architecture, mechanical design or topography. Ruler (or straightedge) and compass constructions, referred to as RC-constructions, are geometric constructions performed using only ruler and compass. They are famous since they allowed to precisely define a class of constructions studied by the Ancient Greeks. Through the ages, they have also provided generations of students in mathematics with a lot of problems in geometry. The RC-constructibility issue is also famous because of problems that are *not* solvable using only straightedge and compass, like, for instance, squaring the circle.

Significant advances in geometry came after the discovery of analytic geometry. And it was not before the nineteenth century that problems such as squaring the circle was determined as not RC-constructible through the algebraic notion of field extensions (Stewart, 2003). Currently, in the field of computer science, almost every geometry software implements geometric data structures and algorithms that handle coordinates and analytic geometry. This is the case in industrial CAD systems and also in the field of formal proof in geometry where powerful algebraic tools are used to prove difficult theorems (Wu, 1984). Even the domain of ruler and compass constructions, which is emblematic of synthetic geometry, had greatly benefited from the algebraic approach. However, geometric reasoning can sometimes facilitate the implementation of the algebraic approaches. For instance, Wu's method such as implemented by Chou (1988) highlights this aspect. The way of choosing variables and parameters as well as the choice of the order of the variables is crucial to solve the problem in a reasonable running time. The strategies for that choices are closely related to geometric constructions based on geometric reasoning. The first goal of this paper is to present a framework where going back and forth between geometry and algebra is essential to provide an answer to some problems of geometry.

RC-construction problems also have a recreational aspect in mathematical communities: they are not difficult to understand, some are easy to solve while others are difficult but usually the solution does not involve specialized theories. In the eighties, a list of 139 construction problems about triangles has been proposed by Wernick (1982), this corpus has been extended by Connelly (2009) by adding four characteristic points related to the nine-point circle. We shortly present these corpora in section 2. Some years after the publication of Wernick's list and after few updates, 15 problems were still open: nobody knew if they were RC-solvable or not. Connelly's list was even more difficult to address, and more than 30 problems were unsolved before the work described in this paper. Note that, the question of RC-constructions is *in theory* solved at least since the end of the XIXth century, see (Lebesgue, 1950). More recently Gao and Chou proposed a practical method to treat simple problems (Gao and Chou, 1998). But, in practice, even small problems among those in Wernick's or Connelly's lists are intractable with these methods.

Our first work on Wernick's list is described in (Schreck and Mathis, 2016). In that paper, we stress the fact that to prove RC-unsolvability of a generic problem, it is enough to

Email addresses: schreck@unistra.fr (Pascal Schreck), mathis@unistra.fr (Pascal Mathis).
URLs: URL 1 (Pascal Schreck), URL 2 (Pascal Mathis).

provide a special case which is not RC-solvable, we call counter-example such an instance of the problem. Then, we describe a method to produce and check such counter-examples. But to show that a problem is RC-solvable, generic equations must be considered where coordinates of given points are symbolic parameters. At this step, a Maple program has been designed to automatically check the *unconstructibility*. But as for the *constructibility* of problems, Maple was not powerful enough to treat the polynomial systems with symbolic parameters, these problems had to be dealt "by hand" by making up the generic problem and by choosing the way of treating the algebraic side. The status, i.e. RC-solvable or not, was discovered for a lot a problems. But some problems remained open especially in Connelly's list.

The second and main goal of this paper is to provide a method that overcomes this issue. Unlike our previous approach where the translation from geometry to algebra is direct and uniform —all the problems are translated in the same way— a knowledge-based system is used to translate each problem in a specific way. The idea behind the knowledge base comes from synthetic geometry and is quite simple: each rule allows to simplify the algebraic formulation of one constraint according to the context specified by the generic problem. The knowledge-based system is written in Prolog. The batch processing of a corpus is a two phases process: (a) from a file containing a list of problems, it computes a new file with the simplified problems, (b) this file is in turn treated by our Maple program to output a file with the nature of each problem—RC-constructible, RC-unconstructible or mis-constrained. Thus, our method is *fully automatic* for these corpora. This approach is not limited to Wernick or Connelly's lists and it can be adapted to other corpora where geometric knowledge is used to express relations between geometric elements.

The rest of the paper is organized as follows. In section 2, we recall the background about geometric constructions. In Section 3, we summarize the method we used to numerically check Wernick's list. Section 4 shows how geometry is used to simplify the algebraic system before their treatment.

2. Some basics about RC-constructions and algebra

This section provides the background about geometric relations between RC-constructibility and algebra. We do not go into details about foundations of geometry like in Boutry et al. (2016), we just give some classical definitions in both synthetic and analytic geometry without proofs (see for instance, Stewart (2003) for the proofs).

2.1. RC-constructibility

First, RC-constructibility can be classically defined as follows.

Definition 1. Given a finite set of points $\mathcal{B} = \{B_0, \ldots, B_m\}$ in the Euclidean plane, a point P is *RC-constructible from the set* $\mathcal{B}$ if there is a finite set of points $\{P_0, \ldots, P_n\}$ such that $P = P_n$, $P_0 \in \mathcal{B}$ and every point P_i $(1 \leq i \leq n)$ is either a point of $\mathcal{B}$ or is at the intersection either of two lines, or of a line and a circle, or of two circles, themselves obtained as follows:

- any considered line passes through two points from the set $\{P_0, \ldots, P_{i-1}\}$;
- any considered circle has its center in the set $\{P_0, \ldots, P_{i-1}\}$ and its radius is equal to the distance P_jP_k for some $j < i$ and $k < i$.

The sequence of these points with their basic construction in terms of intersection between lines and circles is called a RC-construction of point P. □

This definition can be extended to problems where $\mathcal{B}$ contains also lines or circles: it is then possible to consider specific points on these loci or even arbitrary points if it can be proved that the final result of the construction does not depend on this choice.

A construction problem consists in a specification of a figure made of geometric relations between given points, lines or circles and sought points, lines or circles. It is then asked to produce a RC-construction of the sought entities. If it is possible, the problem is called RC-solvable. For instance, the following problem is RC-solvable.

Example 2. Given three different points M_a, M_b and M_c, construct a triangle ABC such that M_a, M_b and M_c are respectively the midpoints of segments BC, CA and AB.

This problem is easily solved thanks to the midpoint theorem: it suffices to draw the three lines parallel to lines M_aM_b, M_bM_c and M_cM_a which are respectively the lines AB, BC and CA. The construction is then:

(1) draw line L_{ab} parallel to line M_aM_b and passing through M_c
(2) draw line L_{bc} parallel to line M_bM_c and passing through M_a
(3) draw line L_{ca} parallel to line M_cM_a and passing through M_b
(4) A is the intersection of lines L_{ab} and L_{ca}
(5) B is the intersection of lines L_{bc} and L_{ab}
(6) C is the intersection of lines L_{ca} and L_{bc}

This raises a few remarks. First, this construction does not fulfill the definition since the latter does not mention the construction of a line parallel to another one as a basic step. But it is well-known that this basic construction can be achieved by using only straightedge and compass: this kind of auxiliary construction is often used in RC-construction for the sake of clarity. Second, this construction does not take degenerate cases into account —here when points M_a, M_b and M_c are collinear. Actually, this is a simplified RC-construction: when all the possible cases are considered, the formal language used to express RC-construction must contain conditional structures (Marinković et al., 2014; Schreck et al., 2012). Detecting degenerate cases and tacking them into account has been accurately described in (Chou, 1988). Third, notice that in this example, degenerate cases correspond to a problem with no solution, RC-constructible or not.

Another issue of the definition lies in the status of the points which belong to set $\mathcal{B}$: they can be either real points in the plane, like point $O(0,0)$, or variable points. The latter are also called *free points* in dynamic geometry terminology or *parameters* in CAD parametric design: the coordinates of these points are symbolic variables. We will use the term of parameter in this paper. A *generic problem* is a construction problem where the coordinates of the given points are parameters. A generic problem is then RC-solvable, (i) if it admits numerical solutions for parameter values taken into some open set in the parameter space, and (ii) if all that solutions are RC-constructible. For instance, given a right angle defined by points $A(0,0)$, $B(1,0)$ and $C(0,1)$, it is possible to construct by straightedge and compass a point P such that $\angle BAP = \pi/6$. But, given any three points A, B and C such that $\angle BAC = \alpha$, it is not possible, *in general* to construct by straightedge and compass a point P such that $\angle BAP = \alpha/3$. For instance, this is not possible when $\alpha = \pi/3$. More generally, this is impossible for any angle α such that the polynomial $4X^3 - 3X - \cos(\alpha)$ is irreducible in $\mathbb{Q}(\cos(\alpha))[X]$: we will explain this below,

but first, let us consider an example using analytic geometry.

2.2. From geometry to algebra and back: an example

An interesting example that comes from Wernick's corpus illustrates the transformation of a problem from geometry to algebra and, then, the geometrical interpretation of algebraic result. This problem is RC-solvable: let us prove this by using coordinates.

Example 3. Given three points H, T_a and M_a, is it possible to construct three points A, B and C such that H is the orthocenter of triangle ABC, T_a is the foot of the inner-angle bisector of angle A and M_a is the midpoint of segment BC (See Fig. 1)?

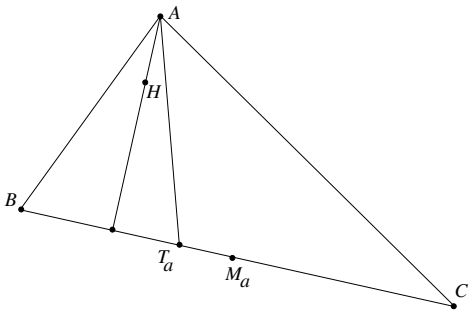

Fig. 1. RC-constructibility of triangle ABC knowing points H, T_a and M_a

First, a coordinate system is chosen in order to give coordinates to the points: T_a is, say, at location $(0, 0)$, point M_a at location $(1, 0)$ and point H at a parametric location (a, b). This can be done because the specification is invariant up to similarities. Then the unknowns for the coordinates of points A, B and C are respectively denoted by (x_A, y_A), (x_B, y_B) and (x_C, y_C) . This way, some constraints given in the statement lead directly to the equations:

- $y_B = y_C = 0$
- $x_B + x_C = 2$
- $x_A = a$.

It is then sufficient to find the values for x_C and y_A. Using the previous values to simplify the equations, the fact that T_a lies on the inner angle bisector from A gives simply:

$$-2y_A(a.x_C^2 + a^2 - 2a.x_C + y_A^2) = 0.$$

Then using the perpendicularity constraint $AH \perp BC$, we have:

$$x_C^2 - 2x_C - b.y_A - a^2 + 2a = 0.$$

Since $a \neq 0$ and $y_A \neq 0$, it comes $a.x_c^2 - 2a.x_C = ab.y_A + a^3 - 2a^2$ from the second equation, and then:

$$y_A^2 + ab.y_A + a^3 - a^2 = 0.$$

After solving, the result is:

$$y_A = \frac{a}{2}\left(-b \pm \sqrt{b^2 - 4a + 4}\right)$$
$$x_C = 1 \pm \sqrt{a^2 + b.y_A - 2.a + 1}$$
$$x_B = 2 - x_C.$$

Thus, we can solve the problem by doing some algebra. Does it mean that the problem is RC-solvable? The answer is yes because all the operations used in expressions for y_A, x_C and x_B can be computed by geometric means. The formulas yield then the RC-construction drawn on Fig. 2. Although it is not an appealing construction, this is a RC-construction.

The question of going back to synthetic geometry is not easy to answer. In this case, it is possible to have a pure geometric construction: a hint is that $b^2 - 4a + 4 = (2 - a)^2 + b^2 - a^2$ indicating that the point at location $(2 - a, -b)$ could be interesting. This point is the symmetric of H with respect to M_a, let us call it P. Actually, two properties are used for the construction:

- the inner angle bisector from A is also the inner angle bisector of the altitude AH and the ray AO where O is the circumcenter of triangle ABC.
- point P is also the symmetric of A with respect to O.

The construction is then reduced to the construction of the lines tangent to circle with center T_a and radius a, and passing through point P (see Fig. 3).

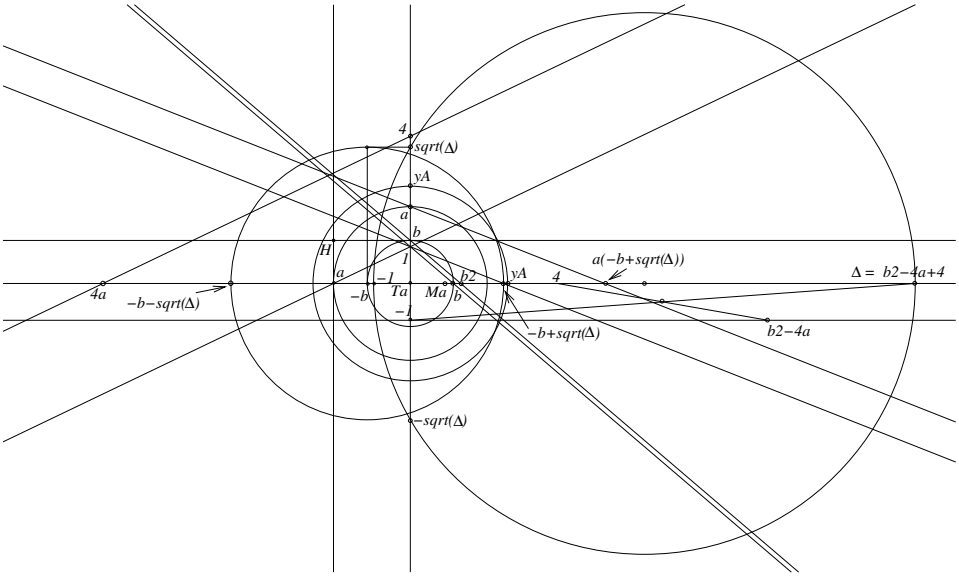

Fig. 2. A construction of $y_A = \frac{a}{2}(-b + \sqrt{b^2 - 4a + 4})$ of example 3 by constructing successively numbers b^2, $4a$, $b^2 - 4a$, $b^2 - 4a + 4$, $\sqrt{b^2 - 4a + 4}$, and so on.

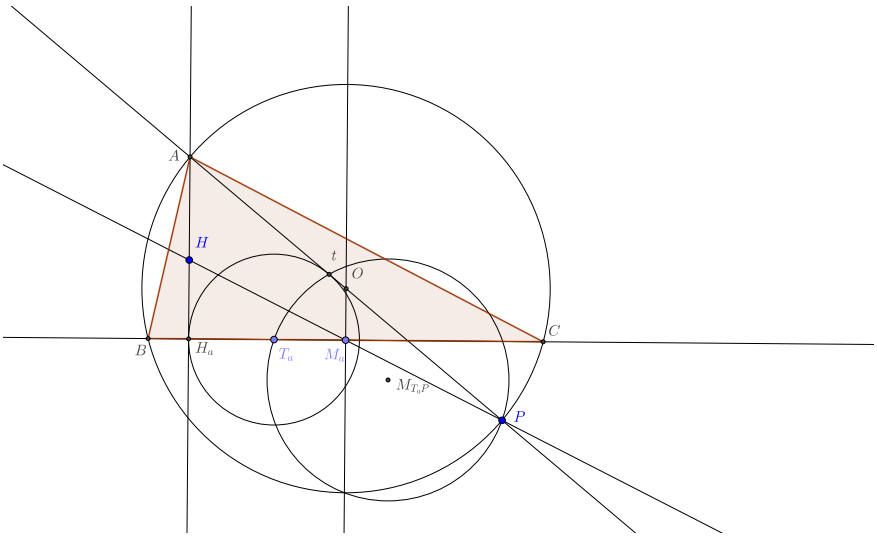

Fig. 3. A simple geometric construction for example 3, done by using point P.

2.3. Algebraic characterization of RC-constructibility

The translation from geometry to algebra allows us to use powerful tools as explained in this section.

Recall that a number is said RC-constructible from a point set $\mathcal{B}$ if and only if it is a coordinate of a point which is RC-constructible from $\mathcal{B}$. It is well-known that the set of the numbers that are constructible from $\mathcal{B}$ is an algebraic extension of field $F = \mathbb{Q}(a_1, ..., a_k)$ (and a subfield of $\mathbb{R}(a_1, ..., a_k)$) where the a_i stand for the coordinates of points in $\mathcal{B}$. Moreover, this field is closed under square root. It is easy to show this since the arithmetic operations can be geometrically performed using only straightedge and compass.

The famous result of Wantzel (1867) is based on this observation and it is often used in order to prove that a problem is not RC-solvable. This theorem states that if F is the field extension of $\mathbb{Q}$ containing the coordinates of point set $\mathcal{B}$, the sought points are RC-constructible in F if and only if their coordinates can be expressed by arithmetic expressions with radicals involving only numbers in F, arithmetic operations and square roots. Such numbers are algebraic in F, and their degrees over F are some powers of two. However, the converse is false. So if a number is the solution of an irreducible polynomial of degree three, it can be established that it is not RC-constructible. This result is sufficient to prove that the problems of angle trisection and doubling the cube are RC-insoluble since they are equivalent to solve some degree 3 equation, $4X^3 - 3X - \cos(\alpha)$ for the former and $X^3 - 2$ for the latter, which is generally irreducible. The problem of squaring the circle was proved RC-unconstructible when von Lidenman proved in 1882 that π is transcendental.

Conversely, if the degree of the irreducible polynomial P of a number α in F is four, or more generally a power of 2, it is not possible to conclude that the number is RC-constructible in F nor the other roots. This is the case, for instance for the polynomial $X^4 + X - 3$ which has two real roots both RC-unconstructible. This is why a stronger

result is generally needed. This result is a consequence of Galois theory: an algebraic number on F is constructible if and only if the splitting field of its minimal polynomial P, is an extension of degree 2^m for some m over F. The degree of this extension, 2^m, is also the cardinal of the Galois group of P (Stewart, 2003).

It is important to notice that the considered problems are often generic. And from the definitions, proving that a generic problem is RC-solvable consists in showing that whatever the values of the parameters leading to real non-degenerate triangles, the solutions are RC-constructible. If one wants to prove that a problem is not RC-solvable, it is enough to compute a counter-example, and if one wants to prove RC-constructibility, the generic problem has to be solved. Proving RC-constructibility leads in general to heavier computations.

2.4. Triangle problems

In the folklore of geometric constructions, most of the problems are about triangles. For instance, William Wernick proposed in 1982 to solve all the problems consisting in constructing a triangle ABC given three characteristic points among the points

- A, B, C, themselves and their circumcenter O;
- M_a, M_b, M_c, G: the side midpoints of the sides and the gravity center (or centroid);
- H_a, H_b, H_c, H: the three feet of altitudes and the orthocenter;
- T_a, T_b, T_c, I: the three feet of the internal angle bisectors, and the incenter.

More recently, Harold Connelly completed this framework by adding the possibility of considering 4 more points:

- E_a, E_b, E_c the midpoints of H and A, B and C respectively; and N the center of the nine-point circle i.e. the circle passing by E_a, E_b, E_c, M_a, M_b, M_c, H_a, H_b and H_c.

Wernick and Connelly drew up the lists of all non trivial problems up to some symmetries giving 139 distinct problems for Wernick's corpus and 140 distinct problems for Connelly's corpus. Wernick's problems are presented in Table A.1 and Connelly's ones in Table A.2 where each problem is given with its status. The status is either **S**olvable, **U**nsolvable, **R**edundant or **L**ocus-restricted. Here, **S** means RC-solvable and **U** means that the problem is RC-unsolvable, not that the problem has no real solutions.

Problems with **R** or **L** status refer to over-constrained problems, that is problems where two or more constraints can be contradictory, leading to problems which have either no solutions (general case) or infinitely many solutions under some compatibility conditions. In Problems with **R** status, one of the three points is defined by the two others: it is redundant in the statement. For instance, in problem W3, ABM_c, which is the third problem in Wernick's list, point M_c which is the midpoint of AB, should be completely defined by the two other points A and B, so M_c is redundant. If M_c is chosen as the effective midpoint of the two others there is infinitely many solution (C can be anywhere in the plane), if not, there is no solution for C. In problem with **L** status, two points impose that the third point must lie on a certain locus. Problem W1 is such a problem: O must lie on the perpendicular bisector of segment AB. If it does not, there is no solution. If it does, the set of solutions is infinite: this is the circle with center O passing by points A and B minus these points. So, if one of the given points has zero degree of freedom, the problem is redundant, if it has one degree of freedom, the problem is locus restricted.

This list served as a benchmark for automated geometric construction in synthetic geometry (Marinković and Janičić, 2012). We developed an automatic method (Schreck

and Mathis, 2016, 2014) able to prove (by giving counter-examples arbitrarily chosen) the RC-unconstructibility of all problems in Wernick's corpus with status **U**. We also proved that problems W108 and W119 are RC-constructible by considering symbolic parameters as coordinates. Moreover, trying our naive numerical method on Connelly's corpus, we observe that it fails for six problems, marked with an asterisk on Table A.2, with a standard 2016 desktop computer with an Intel i5 processor and 16Gb of memory. But, as explained in the next sections, we manage to completely treat the corpus by pre-processing the problems using geometric knowledge. Notice that among all the problems in that list which were not solved by Connelly, problem C81 is the only one which is RC-constructible. Unfortunately, the resulting polynomials are too complicated to hope for a readable geometric construction.

3. A first straightforward algebraic method to prove RC-constructibility

This section summarizes our previous paper (Schreck and Mathis, 2016). We describe a simple pipeline for numerically checking RC-unconstructibility of all the problems in Wernick or Connelly's lists, and for proving the RC-construcbility of the **S** problems. Each statement in the list passes through the following pipeline:

(1) build a numerical figure fulfilling the statement (for looking for a counter-example)
or
choose a parametric coordinate system (for proving the constructibility);
(2) translate this statement into an algebraic system;
(3) triangularize that algebraic system and filter the resulting triangular systems to avoid degenerate cases;
(4) use Wantzel result or compute Galois order of the equations.

3.1. A systematic translation from geometry to algebra

As said in Example 3, the translation of a problem from geometry to algebra requires first to choose coordinates for the involved points and then to translate the geometric constraints into polynomial equations.

To setup a coordinate system, one of the three points has to be located at coordinates $(0, 0)$ and a second one must lies on the x-axis for instance. Actually, since the problems are invariant up to similarities, the second point can be put at coordinates $(v, 0)$ where v has some arbitrarily chosen nonzero value (for instance 1). For the third point, if we want to prove the RC-constructibility, we set it at coordinates (a, b) where a and b are symbolic parameters, and, on the contrary, if we want to check the RC-unconstructibility, the coordinates of the third point are integers more or less randomly chosen: we just verify that there is at least one solution with this choice of coordinates. Notice that we only consider integers as coordinates because, first, we want to perform exact computations and, second, in our case, we consider either $\mathbb{Q}$ (for checking RC-unconstructiblity) or $\mathbb{Q}(a, b)$ (for proving RC-constructibility) in order to compute the Galois groups of the produced polynomials.

For the second issue, a first idea is to exploit some static algebraic definitions of the characteristic points such as depicted on Table 1. This table shows equations for expressing the statement about points and a term associated to this equation. For the sake of simplicity, equations are represented by a term in the third column.

Point	Equation	Geometrical terms
A	$A = A$	
M_a	$x_{Ma} = \frac{x_B+x_C}{2}$ $y_{Ma} = \frac{y_B+y_C}{2}$	`Ma=midpoint(B,C)`
G	$x_G = \frac{x_A+x_B+x_C}{3}$ $y_G = \frac{y_A+y_B+y_C}{3}$	`G=bar(A,B,C)`
H_a	$\overrightarrow{AH_a}.\overrightarrow{BC} = 0$ $\det(\overrightarrow{BH_a}, \overrightarrow{BC}) = 0$	`perpend(A,Ha,B,C)` `collinear(B,Ha,C)`
H	$\overrightarrow{AH}.\overrightarrow{BC} = 0$ $\overrightarrow{BH}.\overrightarrow{AC} = 0$	`perpend(A,H,B,C)` `perpend(B,H,A,C)`
T_a	$\det(\overrightarrow{AB}, \overrightarrow{AT_a}).\|AC\| = det(\overrightarrow{AT_a}, \overrightarrow{AC}).\|AB\|$ $\det(\overrightarrow{BT_a}, \overrightarrow{BC}) = 0$	`onAngleBisector(Ta,A,B,C)` `collinear(B,Ta,C)`
I	$\det(\overrightarrow{AB}, \overrightarrow{AI}).\|AC\| = det(\overrightarrow{AI}, \overrightarrow{AC}).\|AB\|$ $\det(\overrightarrow{BC}, \overrightarrow{BI}).\|BA\| = det(\overrightarrow{BI}, \overrightarrow{BA}).\|BC\|$	`onAngleBisector(I,A,B,C)` `onAngleBisector(I,B,C,A)`
E_a	$x_{E_a} = \frac{x_A+x_H}{2}$ $y_{E_a} = \frac{y_A+y_H}{2}$	`Ea=midpoint(A,H)`
N	$NM_a = NM_b$ $NM_a = NM_c$	`onPerpendBisector(2*N,B+C,A+C)` `onPerpendBisector(2*N,B+C,A+B)`

Table 1. Usual definitions of some characteristic points. These formulas can be straightforwardly translated into polynomial equations.

The method is very sensible to the complexity of the produced algebraic system. A simple method to avoid too complicated systems consists in putting coordinates $(0, 0)$ for the characteristic point involving the more complex equations (see Table 2). This leads to order the points according to an estimation of this complexity: the simpler points are the vertices, then the midpoints M_x, then G and the more complex points are points I and then points T_x.

3.2. Algebraic treatment

Algebraic processes which determine RC-constructibility can be introduced by a simple example. Consider problem A, T_b, I which is numbered W61 in Table A.1. As explained above, the statement is expressed in a general manner even if one point, say I, is set to the origin, point T_b is set to $(1, 0)$ and coordinates of point A are symbolic parameters (a, b). The equations are given in the most general way possible, forgetting that point A is already given. So, coordinates of sought points A, B and C must satisfied six equations: For A:

- $x_A - a = 0$
- $y_A - b = 0$

For T_b:

- T_b is on angle bisector of $\angle ABC$:
  $x_B.y_B^2.y_C+(x_A-2).y_B^2.y_C-2.x_B.y_A.y_B.y_C+2.y_A.y_B.y_C+x_B^3.y_C+(-x_A-2).x_B^2.y_C+(2.x_A+1).x_B.y_C-x_A.y_C-x_C.y_B^3+(2-x_A).y_B^3+x_C.y_A.y_B^2+x_B.y_A.y_B^2-2.y_A.y_B^2-x_B^2.x_C.y_B+2.x_A.x_B.x_C.y_B+(1-2.x_A).x_C.y_B+(2-x_A).x_B^2.y_B-2.x_B.y_B+x_A.y_B-x_B^2.x_C.y_A+2.x_B.x_C.y_A-x_C.y_A+x_B^3.y_A-2.x_B^2.y_A+x_B.y_A=0$
- T_b, A and C are collinear:
  $(1-x_A).(y_C-y_A)-y_A.(x_A-x_C)=0$

For I:

- I is on angle bisector of $\angle ABC$:
  $-2.x_A.y_A.y_B.y_C+x_A.(y_A^2+x_A^2).y_C+x_B.(y_A-x_A).(y_A+x_A).y_C+x_A.(y_A^2+x_A^2).y_B+x_C.(y_A-x_A).(y_A+x_A).y_B-x_C.y_A.(y_A^2+x_A^2)-x_B.y_A.(y_A^2+x_A^2)+2.x_A.x_B.x_C.y_A=0$
- I is on angle bisector of $\angle BAC$:
  $x_B.y_B^2.y_C+x_A.y_B^2.y_C-2.x_B.y_A.y_B.y_C+x_B^3.y_C-x_A.x_B^2.y_C-x_C.y_B^3-x_A.y_B^3+x_C.y_A.y_B^2+x_B.y_A.y_B^2-x_B^2.x_C.y_B+2.x_A.x_B.x_C.y_B-x_A.x_B^2.y_B-x_B^2.x_C.y_A+x_B^3.y_A=0$

Information on RC-constructibility is given by the order of Galois groups for each variable. But Galois groups can only be computed for irreducible polynomials with respect to one variable. This is why, the system must be put into triangular form in order to compute Galois group for each equation with respect to the corresponding variable. Many methods can transform a polynomial system into a triangular form: resultants, Gröbner basis, regular chains. These methods are implemented in many CAS and we did not test them all. But, in our experimentations, we obtained good results in terms of running time by using regular chains with Maple (Aubry et al., 1999). But with this method, even if most problems are treated within seconds, the computation fails after hours for others. Another quality of regular chains is that the produced polynomials are irreducible which is required in the computation of the order of Galois group.

In Maple, the procedure `Triangularize` of the `RegularChains` package provides nine regular chains for this problem. The procedure was called with variables order $x_C, y_C, x_B, y_B, x_A, y_A, a, b$. This means that x_C is eliminated first, then y_C and so on. Problems in corpora are stated such that x_A and y_A appear more often than others points. As a rule of thumb, these two variables are eliminated last. Let us examine for instance the system corresponding to the first regular chain yielded by the procedure `Triangularize`:

$$\begin{cases} b.\boldsymbol{x_C}+(-a+1).\boldsymbol{y_C}-b = 0 \\ \boldsymbol{y_B}-b = 0 \\ \boldsymbol{x_B}-a = 0 \\ \boldsymbol{x_A}-a = 0 \\ \boldsymbol{y_A}-b = 0 \end{cases}$$

The triangular chain could be read from bottom to top: first y_a could be solved, then x_A and so on. When reading from bottom to top, newly introduced variables among x_i and y_i are written in bold.

Since this chain contains less than six polynomials, the system corresponds to a degenerate situation. As expected, solution for point A is (a, b), but the coordinates of point B are also (a, b). Then, points B and A coincide, which is indeed a degenerate situation.

Finally, point C can be anywhere on the line AT_b since $(1,0)$ and (a,b) are solution of the linear polynomial $b.x_C + (-a+1).y_C - b$.

To exclude unwanted chains, two simple filters are applied. First, chains with less than six polynomials are discarded. For a chain of six polynomials, it is checked whether degenerate situations occur. Here points A and B are equal if polynomials $x_A - x_B$ and $y_A - y_B$ belong to the ideal generated by the chain. This could be easily checked by pseudo-division procedure. Obviously, this can be generalized to every degenerate configuration which can be expressed by polynomials. For instance, `collinear(A,B,C)` is used to filter cases where A, B and C are collinear.

A third kind of degenerate situation appears in chains of six polynomials where a polynomial expresses an algebraic dependence between a and b. Among the nine chains provided by Maple, the last one is the following:

$$\begin{cases} b.\boldsymbol{x_C} + (-a+1).y_C - b & = 0 \\ (y_B^2 - 2.by_B - 2.x_B.a + x_B^2 + 2.a - 1).\boldsymbol{y_C} + x_B^2.b + y_B^2.b - 2x_B.b + b & = 0 \\ b.\boldsymbol{y_B} + \boldsymbol{x_B}.a & = 0 \\ \boldsymbol{x_A} - a & = 0 \\ \boldsymbol{y_A} - b & = 0 \\ a^2 + b^2 & = 0 \end{cases}$$

This chain is similar to the first chain which is an under-determined case where five polynomials give locations for the three points. Here point B is freely located on a line. But since a and b are parameters, they should be algebraically independent, then this situation corresponds to a degenerate case. It is also interesting to notice that the relation $a^2 + b^2$ is satisfied either if $a = b = 0$ when only real solutions are considered or $a = \pm i.b$ in the complex framework where regular chains live.

After filtering out the degenerate cases, two chains remain, corresponding to the systems:

$$\begin{cases} b\boldsymbol{x_C} + (1-a)y_C - b & = 0 \\ E\boldsymbol{y_C} + (ab+b)y_B^3 + (-x_Bb^2 - b^2)y_B^2 + ((ab+b)x_B^2 - 2x_Bab)y_B - x_B^3b^2 + x_B^2b^2 & = 0 \\ (a^3 + ab^2 - a^2 + b^2)\boldsymbol{y_B} + (-a^2b - b^3 + 2ab)x_B - a^2b - b^3 & = 0 \\ (a^6 - 2a^5 + (3b^2+1)a^4 - 4a^3b^2 + (3b^4+2b^2)a^2 - 2ab^4 + b^6 + b^4)\boldsymbol{x_B^2} + & \\ (-a^6 + 2a^5 - a^4 - 4a^3b^2 + (3b^4+2b^2)a^2 - 6ab^4 + 2b^6 - b^4)\boldsymbol{x_B} + a^4b^2 + 2a^2b^4 + b^6 & = 0 \\ \boldsymbol{x_A} - a & = 0 \\ \boldsymbol{y_A} - b & = 0 \end{cases}$$

where
$E = ((a-1)y_B^3 + (-2ab - x_Bb + b)y_B^2 + ((a-1)x_B^2 + (-2a^2 + 2b^2 + 2a)x_B)y_B - x_B^3b + (2ab - b)x_B^2)$;

and:

$$\begin{cases} b.\boldsymbol{x_C} + (1-a).y_C - b & = 0 \\ (-2.a + x_B + 1).\boldsymbol{y_C} + x_B.b - b & = 0 \\ \boldsymbol{y_B} & = 0 \\ (a^2 + b^2 - 2.a).\boldsymbol{x_B} + a^2 + b^2 & = 0 \\ \boldsymbol{x_A} - a & = 0 \\ \boldsymbol{y_A} - b & = 0. \end{cases}$$

Both regular chains contain polynomials of degree one and two. Thus, this problem is RC-constructible. The curious reader may wonder about the meaning of these two chains. The second chain represents the solution in which point I is at the intersection of internal angle bisectors. There can be only one numerical solution since all polynomials are of degree one. The first chain can produce multiple numerical solutions. It corresponds to situations where point I is at the intersection of one internal and two external bisectors.

The initial system was set with point I and T_b pinned down in the plane and point A free. Choosing other reference such as $A(0,0)$, $I(1,0)$ and $T_b(a,b)$ leads to different initial system and different regular chains. Actually for the latter reference, there are eight regular chains. After removing degenerate chains, two chains remain which correspond to RC-constructible solutions. It is worth to mention that degenerate situations are different according to the chosen reference. For instance, a solution of the above chain with $a^2 + b^2 = 0$ implies that point A and I coincide. Here, point A and I are different and this degenerate situation could not occur. In addition, the size and degree of the initial system depend on the chosen reference and influence the computational time of the regular chains. The relevance of the points to choose as references is discussed below.

The more general framework should consist in giving six parameters for the coordinates of the three points of the statement. But, on the one hand, this is not useful because of similarity invariance which we already mentioned and, on the other hand, most problems should be far beyond the capabilities of actual CAS.

3.3. Basic pipeline for proving RC-constructibility

In the previous example, RC-constructibility is easily stated since equations are only of degree one or two. It is well-known that the solutions of such polynomials are RC-constructible. But in a general manner, a statement is a RC-constructible problem if the order of Galois group of each equation is a power of two. To save some computational time, a simple test is performed. If at least one of the polynomials has a degree which is not a power of two, then the problem is not RC-constructible. On the contrary the order of Galois group of each equation considered as single variable equation must be computed. Such a procedure exists in Maple for polynomials over rationals. It is effective for polynomials with a maximum degree of nine which is fortunately the case for all of our problems. So, in practice, order of Galois groups is computed for a polynomial of degree four or eight.

With Maple, the general process without optimization follows these steps for three statement points $P1, P2$ and $P3$:

(1) set the coordinates of points $P1$, $P2$ and $P3$ to $(0,0)$, $(1,0)$ and (a,b) respectively;
(2) set up the six polynomials over $\mathbb{Q}$, each P_i gives rise to two equations;

(3) call `Triangularize` of the `RegularChain` package with the variable order $xC, yC, xB, yB, xA, yA, a, b$;
(4) remove chains with less than six equations involving at least one the variables xC, yC, xB, yB, xA, yA;
(5) determine by pseudo-division the chains that correspond to point equalities, i.e. $A = B$, $A = C$, $B = C$, but also $P_i \in \{A, B, C\}$ unless A or B is one of the P_i's and remove such chains;
(6) for each chain s and each polynomial p of s in the single variable x: the problem is not RC-constructible if degree of p in x is not a power of two or if the order of galois group is not a power of two;
(7) if one chain is not RC-constructible the problem is not RC-constructible otherwise it is.

3.4. Issues

Unfortunately, using such a static method to build the algebraic systems from the geometric statements sometimes leads to unnecessary complicated polynomials. For instance, in problem C81, one of the given characteristic points is the incenter I: it can be defined as the intersection of the two inner bisectors of angles $\angle ABC$ and $\angle BAC$, but since point H_b which is on line AC is given, it is better to define point I as the intersection of the two inner bisectors of respective angles $\angle BCH_b$ and $\angle BAH_b$ where lesser unknowns are involved. In the next section, a heuristic is proposed to systematize this trick.

Several factors determine the complexity of the triangularization process. We use regular chains as a *black box* tool provided by Maple. We assume then that the time complexity of the process is closely related to the complexity in size of the result which can be measured by:
- the number of triangular forms found before filtering
- the degree(s) of the polynomials
- the number of monomials
- the size of the coefficients

Different criteria could have an influence on that complexity:
- the choice of a coordinate system,
- the order of the variables for elimination,
- the choices made when making up the algebraic system to be solved.

As equivalent systems have by definition the same solutions, they also have the same status. The question is then "how can we build an equation system equivalent to the given one such that the result has an optimal complexity, that is, in practice, the minimal size within Maple data structures".

4. Geometric reasoning and algebra

We do not have a definitive answer for the question raised in the previous section, but we describe below some heuristics which give good results in our tests. The key idea is to provide good equivalent algebraic systems by using geometric reasoning guided by the complexity in terms of degree and number of monomials.

Term	Degree	Monos	Vars	(0,0)	(1,0)
`midpointx(M,P1,P2)`	linear	3	3	1(2)	1(3)
`onPerpendBisector(M,P1,P2)`	2 (2)	12	6	2 (4)	2 (6)
`isobarx(M,P1,P2,P3)`	linear	3	4	1(3)	1(4)
`perpend(M,P1,P2,P3)`	1 (2)	8	8	2 (4)	2 (6)
`collinear(M,P1,P2)`	1 (2)	8	8	2 (4)	2 (6)
`onAngleBisector(M,P1,P2,P3)`	deg(XM) = 2 deg(YM) = 2 deg(XP1) = 3 deg(YP1) = 3 deg(XP2) = 1 deg(YP2) = 1 deg(XP3) = 1 deg(YP3) = 1 (4)	56	8	4 (14)	4 (31)

Table 2. Some complexities: the "degree" column indicates the local degree and the global degree, the "monos" column indicates the number of monomials and the "vars" column indicates the number of variables. The last two columns show the degree and the number of monomials when point M is located at $(0, 0)$ and in $(1, 0)$.

4.1. Equivalent systems in Connelly's corpus

The easiest way to produce equivalent systems consists in permuting the given points. For instance, problem A, B, G is trivially equivalent to B, A, G but also to problem B, C, G. This kind of equivalence clearly corresponds to the variable order issue and, in a sense, it is treated by both Wernick and Connelly's corpora where only one problem of each class appears.

Also there are equivalences between systems coming from theorems of Euclidean geometry such as Euler relation. For example, problem A, O, G (W13) is equivalent to problem A, O, H (W16). Indeed, it is known that points O, G, H are located on the Euler line and are linked by the relationship $HO = \frac{3}{2}HG$. There is thus a simple RC-construction to go from W13 to W16 and conversely. Thus, both problems share the same status. However, in terms of symbolic processing, problem W13 is much simpler than problem W16. In the latter, point H leads to quadratic equations each with up to 8 monomials (see Table. 2). For problem W13, point G is translated into linear equations with three monomials. Considering Connelly's corpus, we identify four geometric relations usable for computing equivalent systems. List of these relations is given below where x denotes any point among A, B or C. Actually these relations are directly extracted from the corpora since they correspond exactly to **R** problems.

- Relation 1. For each vertex x, $xG = 2/3xM_x$.
- Relation 2. N is the midpoint of E_x and M_x.
- Relation 3. If there are two points among G, H, O, N then any of these two points can be replaced by one of the two others.
- Relation 4. If there are two points among x, E_x, H then any of these two points can

be replaced by the third.

Of course, these relations are made to fit exactly Connelly's corpus and new relations must be considered for other corpora. To deal with a new corpus of geometric construction problems, our approach can be followed by identifying the relations between points, or more generally entities, characteristic of that corpus, and also see which points lead to the lowest complexity.

In Schreck et al. (2016), a method using a knowledge base is described in order to discover these relations in the case of Wernick's corpus and then to gather problems of this list in classes. Moreover, automatic methods for theorem discovery from a figure could also be used to find such relations, see for instance (Botana and Valcarce, 2006; Chen et al., 2015).

Relation 1 states the well-known property of centroid G. Relations 2 expresses the fact that for any x, segment E_xM_x is a diameter of the Euler circle. Relations 3 comes from the well-known metric relations among points G, H, O, N which all lie on the Euler line. Relation 4 translates the definition of E_x as the midpoint of x and H.

With problem C53: E_a, H, G, considering relation 3 leads to the 5 following problems:

E_a, H, O: C69 E_a, H, N: C68
E_a, G, O: C60 E_a, G, N: C59
E_a, N, O: C107.

Then, with relation 2, it comes:

$E_a, H, N \to M_a, H, N$: C122
$E_a, G, N \to M_a, G, N$: C117
$E_a, N, O \to M_a, N, O$: C136

Relation 4 gives:

$E_a, H, O \to A, H, O$: W16 $E_a, H, O \to A, E_a, O$: C13
$E_a, H, G \to A, H, G$: W49 $E_a, H, G \to A, E_a, G$: C5
$E_a, H, N \to A, H, N$: C32 $E_a, H, N \to A, E_a, N$: C12.

From these new triples, relations 2 and 3 apply again:

M_a, O, G: W63 M_a, O, H: W66 M_a, N, A: C36
M_a, G, H: W93 A, O, G: W13
A, O, N: C38 A, G, N: C31.

Finally, with relation 1, we get:

M_a, G, H: W24 A, E_a, M_a: C10
E_a, G, M_a: C58 A, O, M_a: W11.

Thereby, from problem C53, a set of 26 problems are identified as equivalent and problem A, O, M_a seems to be one of the easiest to solve since point A is already given. Indeed, a construction can be easily found:

(1) draw circumcircle $\mathcal{C}$ of center O and passing through A,
(2) draw line $\mathcal{L}$ perpendicular to line OM_a in point M_a,
(3) point B and C are the intersection points of $\mathcal{C}$ and $\mathcal{L}$

Consequently, all these 26 problems are constructible and their constructions easily derive from reduction relations.

With Maple 18 using an Intel©Core i5, we find that, under the same conditions, it takes 0.1s to treat the generic version of initial problem C53 while it takes 0.02s for problem A, O, M_a (W11), 0.04s for problem A, O, G (W13), 0.05s for A, H, G (W49) and 0.54s for problem E_a, H, O (C69). On another scale, trying to solve problem E_a, N, O (C107) directly leads to heavy computations stopped after waiting 15min and after using more than 4Go. Notice that computational time is measured by the function `time()` of Maple which is not very accurate. Also, these values may change with another coordinate system.

Problem $W49$ has a degree 4 and has 16 monomials: using the coordinate system $H(0,0), G(1,0), A(a,b)$ its triangularization takes more or less the same time than for $W13$ (just replace $H(0,0)$ by $O(0,0)$). But using the coordinate system $H(0,0), G(a,b)$, $A(1,0)$, its triangularization is faster than for $W11$ which has degree 4 and 14 monomials. This is seemingly because using these coordinates for H and A decreases the degree to 2, the number of monomials and the number of variables after the first steps of elimination leading to a simpler system to deal with.

Another example is problem O, G, T_a (W72) which is not RC-constructible. Relation 3 makes appear that this problem belongs to an equivalence class that contains the unconstructible problems:
O, H, T_a: W79
O, N, T_a: C139
G, H, T_a: W120
H, N, T_a: C124
G, N, T_a: C119.
It takes 0.146s to treat the initial problem O, G, T_a (W72) with a counter-example, 0.152s for problem G, H, T_a (W120) and 1.13 for problem H, N, T_a (C124). Again, time is measured using `time()` function and changing the coordinate system can greatly modify these values by changing the size of the coefficients.

4.2. Replacing equations

As stated in Table 1, points are defined by two specific formulas. Then, each point of the statement gives rise to two equations. For point H for example, we have:

```
perpend(H,A,B,C)
perpend(H,B,A,C).
```

Each algebraic equation has degree two and eight monomials. For point H_a, equations are:

```
perpend(Ha,A,B,C)
collinear(Ha,B,C).
```

Again, each of these two polynomials is of degree 2 and they each contain eight monomials. These are the *default* equations related to a characteristic point.

However, it is possible to get better results. If points H and H_a occur both in a statement, we could express that A, H and H_a are collinear. Since the coordinates of H and H_a are known, the equation for `collinear(A,Ha,H)` is of degree one and contains three terms. This relation could replace one equation among the four equations of degree two above.

Rather than expressing each point independently, a simpler system can be built by considering several points of the statement in the equations. Each of these equations replaces a default equation. However, the replacement should be done carefully. Consider

problem W112: I, T_a, M_a. The default system could be:
For I: `(eq1) onAngleBisector(I,A,B,C)`
`(eq2) onAngleBisector(I,B,A,C).`
For T_a: `(eq3) onAngleBisector(Ta,A,B,C)`
`(eq4) collinear(Ta,B,C).`
For M_a:`(eq5, eq6) Ma=midpoint(A,B).`

Specific equations for that problem are possible. Since points A, I and T_a are collinear and that points T_a, M_a and B (or C) are collinear as well, it comes:

```
(eq7)  collinear(Ta,I,A)
(eq8)  collinear(Ta,Ma,B).
```

Equations (eq7) and (eq8) could replace two equations in the default system. Since (eq7) involves I and T_a, it has to replace an equation among (eq1), (eq2), (eq3) or (eq4). If (eq4) is selected, the system becomes mis-constrained since (eq7) is a consequence of (eq1) and (eq3) but also of (eq2) and (eq3). So, (eq7) must replace (eq3). Detecting such dependences between equations can be done automatically by computing a maximum matching on the bipartite graph unknowns/equations: the system is well-constrained if the matching is perfect. In our example, the replacement is relevant because an equation of degree four with more than ten monomials is simplified into a polynomial of degree one with three monomials.

Several situations were detected where a collinearity relation could replace a more complex equation. We translate them into directed rules that can be applied to the statement (see appendix for the list of rules). Many patterns can lead to simplifications. We give here some examples (each of them can be generalized):

- if T_a and H_b are given, the default equation for T_a, `onAngleBisector(Ta, A, B, C)`, can be replaced by the simpler equation `onAngleBisector(Ta, A, B, Hc)`
- if N and H_a are given, the default equation `onPerpendBisector(2*N,A+B,A+C)` used for N, can be replaced by `onPerpendBisector(2*N,A+B,2*Ha)`
- if H_a and T_a are given, the default equations for H_a and T_a, `collinear(Y,Ha,Z)` and `collinear(Y,Ta,Z)` , can be replaced respectively by `collinear(Ha,Z,Ta)` and `collinear(Ha,Y,Ta)`.

We have more specific rules as described in appendix B.

4.3. A geometric knowledge-based system

We use Prolog to design a knowledge-based system that aims at simplifying the statements in both Wernick and Connelly's lists. It inputs a file with the statements and outputs a file with the modified statements which in turn will be treated by our Maple program. The modifications are made by using rules based on the relations described in the previous subsections.

The knowledge base contains two parts. A first part is made of general knowledge about the corpus, such as the rules given below. A second part includes particular knowledge which takes into account the current statement such as the given points and the corresponding terms. Recall that in Prolog, an identifier starting with an uppercase character refers to a variable. Therefore points A, B, C, G *etc.* are denoted by `a`, `b`, `c`, `g` *etc.*. The compound point names like H_a, M_b, T_c, *etc.* are denoted by `h(a)`, `m(b)`, `t(c)` *etc.*

Using the ability of Prolog to define syntactic sugar, we design a language to write rules, to query and to update the current statement. Classically, each rule takes the form

`if <list of facts> then <list of actions>`. For instance, in the following rule:

```
 if
    vertices(X,Y,Z) and
    h(X) and t(X) and
    collinear(Y,t(X),Z) as F1 and
    collinear(Y,h(X),Z) as F2
then [
       change F1 by collinear(h(X),Y,t(X)),
       change F2 by collinear(h(X),Z,t(X))
     ].
```

each fact, separated from the others by the keyword **and**, corresponds to a query of the knowledge base including the current statement:

- the query **vertices**`(X,Y,Z)` instantiates the variables such as `{X,Y,Z}={a,b,c}`
- the query `h(X)` searches for a value x for `X` such that H_x is given, then this value x is used in the query `t(X)` to verify that T_x is present in the current statement
- the query `collinear(Y,t(X),Z) as F1` searches if the term `collinear(Y,t(X),Z)` where the variables are instantiated, is present in the statement and if it succeeds, variable `F1` refers to it. The same goes for the last query.

When all the queries succeed, the variables are instantiated to some values with respect to the current statement and a list of action is launched in order to modify the statement.

The list of actions is a classical Prolog list: this is the imperative part of the rule. An action can be any call to a Prolog predicate but in our framework only the modifications of the current statement were useful so far. In our example, the first action consists in replacing the fact `collinear(Y,t(X),Z)`, a.k.a. `F1`, by the term `collinear(h(X),Y,t(X))`.

The engine behind this base of rules is quite simple: for each rule, check the facts and instantiate the variables, then do the modifications. Actually, this is decomposed into two stages: the first one treats the question of equivalence and the second one is dedicated to the translation into algebra. Let us describe this two-stage process in more details.

Equivalence. After reading the input file, the current statement is the three names of the characteristic points. Before translating it into equations, a first stage consists in finding a representative of the equivalence class of that current statement, as described in Section 4.1.

To this end, our knowledge-based system is used thanks to a special set of rules characterized by the keyword `equivalenceStep`. `equivalenceStep` is a fact put into the database during this phase and removed after that. This allows to distinguish two classes of rules. In this phase, the sole modification of the statement should be to replace a characteristic point by a simpler one. We use the following self-evident rules.

```
if
    equivalenceStep
    and g and m(X)
then
    [
     change m(X) by X
    ].

if
    equivalenceStep
    and h and e(X)
then
    [
     change e(X) by X
    ].

if
    equivalenceStep
    and n and e(X)
then
    [
    change e(X) by m(X)
    ].

if   equivalenceStep and
    [P1,P2] stated among [o, g, n, h]
    and [P1,P2]  are_not [o,g]
then  [
        change P1 by g,
        change P2 by o     ].
```

These rules correspond to the orientation of relations 1 to 4 by following an ordering of characteristic points based on an estimation of the complexity of their definition: $X < m(X) < e(X)$ and $G < O < H < N$.

Replacing equations. Once the equivalent system is found, it is translated into algebra according to the default equations. The second stage consists then in replacing default equations by simpler ones.

We already described a standard rule above and the whole list is given in Appendix B. Let us just mention one specific rule like this one:

```
if      vertices(X,Y,Z) and
        e(X) and o and
        perpend(X,2*e(X)-X,_,_) as F1 and
        [P] among [Y,Z] and
        perpend(P,2*e(X)-X,_,_) as F2
then [
        change F1 by egx(midpoint(Y+Z,2*X), midpoint(2*e(X),2*o)),
        change F2 by egy(midpoint(Y+Z,2*X), midpoint(2*e(X),2*o))  ].
```

expressing the fact that, with $X = A$, AE_aM_aO is a parallelogram and hence the midpoint of AM_a is also the midpoint of EaO. These two equalities can then be used instead of the default definitions, `F1` and `F2`, of E_a since they are linear if Ea and O are given.

For instance, using this rule, it takes about 3.5s to triangularize a generic instance of problem E_a, O, T_b (C111) while Maple has to be stopped after about 10 minutes without this trick. Note that it takes about 2s for a numerical instance without the trick against 0.16s using it.

4.4. Results

In previous section, it is seen that geometric preprocessing can greatly reduce the cost of the process in terms of computational time and memory space. This allows us to speed up the triangularization process. But it also simplifies the resulting regular chain.

This is obvious when using equivalence. For instance, considering a previous example where W11 and C38 are equivalent. But for W11, with the location $O(0,0), M_a(1,0), A(a,b)$,

we get after triangularization:

$$\begin{cases} x_C - 1 & = 0 \\ y_C + y_B & = 0 \\ x_B - 1 & = 0 \\ y_B^2 - a^2 - b^2 + 1 & = 0 \\ x_A - a & = 0 \\ y_A - b & = 0 \end{cases}$$

and C38, with a similar location $O(0,0), N(1,0), A(a,b)$:

$$\begin{cases} (a-2)x_C + y_C b + (-a+2)x_B - y_B b & = 0 \\ (bx_B + (-a+2)y_B - 2b)y_C - (a-2)x_B^2 + (a^2 - by_B - 4)x_B + y_B ab - 2a^2 + 4a & = 0 \\ ((4ab-8b)y_B - a^3 + 2a^2 + (-b^2+4)a + 2b^2 - 8)x_B + (-2a^2 + 2b^2 + 8a - 8)y_B^2 & \\ +(-a^2 b - b^3 + 4b)y_B + a^4 - 2a^3 - 4a^2 + (-2b^2+8)a - b^4 + 4b^2 & = 0 \\ (4a^2 + 4b^2 - 16a + 16)y_B^2 + (4a^2 b + 4b^3 - 16ab + 16b)y_B - 3a^4 + 8a^3 & \\ +(-2b^2+8)a^2 + (8b^2 - 32)a + b^4 - 8b^2 + 16 & = 0 \\ x_A - a & = 0 \\ y_A - b & = 0. \end{cases}$$

This is also the case when only the simplication of the algebraic system is used. For instance with problem C13, points A, O, E_a were set at location $(a,b), (0,0), (1,0)$ respectively. Without preprocessing, the following regular chain yields:

$$\begin{cases} (a-1)x_C + y_C b + (-a+1)x_B - y_B b & = 0 \\ (x_B b + (-a+1)y_B - b)y_C + (-by_B + 2a - 2)x_B + (a-1)y_B^2 + y_B b - 2(a^2 - a) & = 0 \\ E.x_B + (2ab - 2b)y_B^3 + (-1 - a^3 + a^2 + (b^2+1)a)y_B^2 + 2a^4 + (-b^2 - 4)a^3 & \\ +(-2a^3 b + a^2 b + (-2b^3 + b)a + 2b^3)y_B + (2b^2+2)a^2 + (-b^4 - 2b^2)a + b^2 & = 0 \\ (a^2 + b^2 - 2a + 1)y_B^2 + (2a^2 b + 2b^3 - 4ab + 2b)y_B & \\ -2a^3 + (b^2+5)a^2 + (-2b^2 - 4)a + b^4 + b^2 + 1 & = 0 \\ x_A - a & = 0 \\ y_A - b & = 0 \end{cases}$$

where $E = ((a^2 - b^2 - 2a + 1)y_B^2 + (2a^2 b - ab - b)y_B - 2a^3 + (b^2+4)a^2 + (-b^2 - 2)a + b^4)$;

and, after simplification, it comes:

$$\begin{cases} x_C + x_B + 2a - 2 & = 0 \\ y_C + y_B + 2b & = 0 \\ (a-1)x_B + a^2 + b^2 + y_B b - 2a + 1 & = 0 \\ (a^2+b^2-2a+1)y_B^2 + (2a^2b + 2b^3 - 4ab + 2b)y_B - 2a^3 & \\ +(b^2+5)a^2 + (-2b^2-4)a + b^4 + b^2 + 1 & = 0 \\ x_A - a & = 0 \\ y_A - b & = 0. \end{cases}$$

As stated in Table A.2, a direct method fails in checking the status of six problems in Connelly's list. This number is obviously much higher when considering the generic cases. We just briefly discuss the example of problem C81: E_a, H_b, I. Its status, **S**, has been discovered and proved by our program. We show below how the corresponding system is simplified.

Using the default equation with the locations $I(0,0), Ea(a,b), Hb(1,0)$, we get the following algebraic system:

$$\begin{cases} x_A^3 y_B + x_A^3 y_C - x_A^2 x_B y_A - x_A^2 x_B y_C - x_A^2 x_C y_A - x_A^2 x_C y_B + 2x_A x_B x_C y_A & \\ +x_A y_A^2 y_B + x_A y_A^2 y_C - 2x_A y_A y_B y_C - x_B y_A^3 + x_B y_A^2 y_C - x_C y_A^3 + x_C y_A^2 y_B & = 0 \\ -x_A x_B^2 y_B - x_A x_B^2 y_C + 2x_A x_B x_C y_B - x_A y_B^3 + x_A y_B^2 y_C + x_B^3 y_A + x_B^3 y_C & \\ -x_B^2 x_C y_A - x_B^2 x_C y_B + x_B y_A y_B^2 - 2x_B y_A y_B y_C + x_B y_B^2 y_C + x_C y_A y_B^2 - x_C y_B^3 & = 0 \\ (2x_A - 2a)(x_B - x_C) + (2y_A - 2b)(y_B - y_C) & = 0 \\ (x_B - 2a + x_A)(x_C - x_A) + (y_B - 2b + y_A)(y_C - y_A) & = 0 \\ (x_B - 1)(x_C - x_A) + y_B(y_C - y_A) & = 0 \\ (1 - x_A)(y_C - y_A) - y_A(-x_C + x_A) & = 0 \end{cases}$$

whose degree is 256 and which contains 58 monomials. The average number of terms per monomial is about 2.95.

After geometric reasoning using the fact that H_b is on side AC, the first two equations are considerably simplified from 14 to 9 monomials. The fourth equation is also simplified by using the fact that points H, B and H_b are collinear (and also that E_a is the midpoint of AH) leading to an equation with 2 variables instead of 3. There are minor modifications

for the other equations but they are not related to geometry. The system

$$\begin{cases} -x_B x_C^2 y_C - x_B y_C^3 + x_C^3 y_B + x_C y_B y_C^2 + 2x_B x_C y_C - x_C^2 y_B - x_C^2 y_C + y_B y_C^2 - y_C^3 = 0 \\ x_A^3 y_B - x_A^2 x_B y_A + x_A y_A^2 y_B - x_B y_A^3 - x_A^2 y_A - x_A^2 y_B + 2x_A x_B y_A - y_A^3 + y_A^2 y_B = 0 \\ (x_A - a)(x_C - x_B) + (y_A - b)(y_C - y_B) = 0 \\ (2a - x_A - 1)y_B + (2b - y_A)(1 - x_B) = 0 \\ (1 - x_B)(x_C - x_A) - y_B(y_C - y_A) = 0 \\ (x_A - 1)y_C + y_A(1 - x_C) = 0 \end{cases}$$

whose degree is also 256, but it contains only 43 monomials and its average number of terms per monomial is about 2.56. It seems to us that the global indicators are important, but the local simplifications can be also very important: here the fact that 3 equations have been greatly simplified has allowed to make the triangularization much more tractable and eventually manageable in a short time, about 10s.

Here are some statistics about our geometric preprocessing on Connelly's list.

- 114 out of 140 (81%) problems are simplified, 45 of them are replaced with an equivalent problem. Among these 45 problems, 12 still have been simplified further: these are problems C49, C63, C64, C76, C85, C97, C102 which are over-constrained, C26, C92, C108 which are constructible (for C92, preprocessing is needed), and C109 which is not constructible.
- There are six problems that are numerically intractable without our geometric preprocessing (see Table A.2). This means that either no result is produced after four hours of computation or the memory is filled up during the computation. Without preprocessing, five RC-constructible problems are also intractable by considering parameters: C50, C81, C107, C108, C126.
- Without preprocessing, among the 129 tractable problems, 111 of them are solved in less than three seconds with an average time of 0.6 seconds. The other 18 problems could take from six seconds to 1377 seconds (C132).
- With preprocessing, all problems are tractable in less that one hour. Three of them takes more than 15 minutes, these are RC-constructible problems that are solved considering the generic problems. 118 problems take less than one second with an average time of 0.2 second.

5. Conclusion

This paper presents an approach mixing synthetic and analytic geometry in order to determine the status in terms of RC-constructibility of both Wernick and Connelly's corpora. Analytic geometry is mostly needed to prove unconstructibility but it also allows to automatize the treatment of the whole corpora by batch processing. However, even with 2016 computers and modern CAS, algebraic tools are not powerful enough in practice because computations are often too heavy and fail by lack of memory. This is why we designed a geometric knowledge-based system able to smartly translate geometric statements into simpler algebraic systems. Then, all the problems from Wernick and Connelly's corpora were automatically treated in a very reasonable time.

The examples found in these corpora could be considered as toy examples and de-

signed as puzzles for amateur mathematicians. However, the main interest of this paper does not lie in the results about Wernick and Connelly's corpora but much more in the approach consisting in conjointly using synthetic geometry and analytic geometry to solve problems. The method described here belongs to a set of works of our team aiming at adding some synthetic geometry reasoning in geometric constraint solving but also in automated or assisted proof in geometry .

This work can be continued in several directions. For the puzzle amateur, two RC-constructible problems remain without construction: W119 and C81. It could be challenging to discover such constructions and even more (1) to formally solve the corresponding algebraic systems, and (2) interpret the formulas by nice synthetic geometric constructions. More generally, it seems very interesting to study how a formula with square roots can be translated into a readable construction: analyzing quantities under the radicals as intersection of circles and lines could help.

There are also several interesting questions about the links between geometric complexity and algebraic complexity. In this paper, we used an implicit heuristic: simplifying the geometric problem makes the algebraic system easier to solve. Can this formula be expressed more precisely? Is there a link between the complexity of a geometric construction and the complexity of the corresponding triangularized system?

Another obvious direction is to extend the corpora described here or, even better, to design a more general geometric knowledge-based system which is able to simplify open problems out of corpora. This could be very interesting in Computer Aided Education, even if no construction is provided by the system. Indeed, it is very difficult to guess if a problem is RC-constructible or not. And, in education, RC-constructible problems are the more interesting, such a system could indicate if a problem is worth being investigated or not.

References

Aubry, P., Lazard, D., Maza, M. M., 1999. On the theories of triangular sets. Journal of Symbolic Computation 28 (2), 105–124.

Botana, F., Valcarce, J. L., 2006. Automated discovery in elementary extrema problems. In: Alexandrov, V. N., van Albada, G. D., Sloot, P. M. A., Dongarra, J. (Eds.), International Conference on Computational Science - ICCS 2006, Part II. Vol. 3992 of Lecture Notes in Computer Science. Springer, pp. 470–477.

Boutry, P., Braun, G., Narboux, J., 2016. From Tarski to Descartes: Formalization of the arithmetization of euclidean geometry. In: International Symposium on Symbolic Computation in Software (SCSS 2016). Vol. 39 of EasyChair Proceedings in Computing. James H. Davenport and Fadoua Ghourabi.

Chen, X., Song, D., Wang, D., 2015. Automated generation of geometric theorems from images of diagrams. Annals of Mathematics and Artificial Intelligence 74 (3-4), 333–358.

Chou, S., 1988. Mechanical geometry theorem proving. Mathematics and its Applications. D. Reidel, Dordrecht.

Connelly, H., 2009. An extension of triangle constructions from located points. Forum Geometricorum 9, 109–112.

Gao, X.-S., Chou, S.-C., 1998. Solving geometric constraint systems. II. A symbolic approach and decision of Rc-constructibility. Computer Aided Design 30 (2), 115–122.

Hulpke, A., 1999. Techniques for the computation of galois groups. In: Matzat, B., Greuel, G.-M., Hiss, G. (Eds.), Algorithmic Algebra and Number Theory. Springer Berlin Heidelberg, pp. 65–77.

Imbach, R., Mathis, P., Schreck, P., 2017. A robust and efficient method for solving point distance problems by homotopy. Mathematical Programming 163 (1-2), 115–144.

Imbach, R., Schreck, P., Mathis, P., 2014. Leading a continuation method by geometry for solving geometric constraints. Computer Aided Design 46, 138–147.

Lebesgue, H., 1950. Leçons sur les constructions géométriques. Gauthier-Villars, Paris, in French, re-edition by Editions Jacques Gabay, France.

Marinković, V., Janičić, P., 2012. Towards understanding triangle construction problems. In: Jeuring et al., J. (Ed.), Intelligent Computer Mathematics - CICM 2012. Vol. 7362 of Lecture Notes in Computer Science. Springer.

Marinković, V., Janičić, P., Schreck, P., 2014. Computer theorem proving for verifiable solving of geometric construction problems. In: Botana, F., Quaresma, P. (Eds.), Automated Deduction in Geometry - 10th International Workshop, ADG 2014, Coimbra, Portugal, July 9-11, 2014, Revised Selected Papers. Vol. 9201 of Lecture Notes in Computer Science. Springer, pp. 72–93.

Schreck, P., Marinković, V., Janičić, P., 2016. Constructibility classes for triangle location problems. Mathematics in Computer Science 10 (1), 27–39.

Schreck, P., Mathis, P., 2014. Rc-constructibility of problems in wernick's list. In: Botana, F., Quaresma, P. (Eds.), Proceedings of the 10th Int. Workshop on Automated Deduction in Geometry. Vol. TR 2014/01. University of Coimbra, pp. 85–104.

Schreck, P., Mathis, P., 2016. Automatic constructibility checking of a corpus of geometric construction problems. Mathematics in Computer Science 10 (1), 41–56.

Schreck, P., Mathis, P., Narboux, J., 2012. Geometric construction problem solving in computer-aided learning. In: International Conference on Tools with Artificial Intelligence - ICTAI. IEEE, pp. 1139–1144, core B - short paper.

Stewart, I., 2003. Galois Theory (third edition). Chapman Hall.

Wernick, W., 1982. Triangle constructions vith three located points. Mathematics Magazine 55 (4), 227–230.

Wu, W.-T., 1984. Basic principles of mechanical theorem proving in elementary geometries. Journal of Symbolic Computation 4, 207–235.

A. Wernick and Connelly's corpora

The following tables, Tables A.1 and A.2, give the statuses of all the problems of the two corpora. All the problems have been automatically treated by a batch procedure taking in arguments the list of the problems.

B. The list of geometric rules

We give here *in extenso* the list of geometric rules that we use in our preprocessing.

```
 if
      vertices(X,Y,Z) and
      h and h(X) and
      perpend(X,h(X),Y, Z) as F
then
    [
```

1.	A, B, O	L	36.	A, M_b, T_c	S	71.	O, G, H	R	106.	M_a, H_b, T_c	U
2.	A, B, M_a	S	37.	A, M_b, I	S	72.	O, G, T_a	U	107.	M_a, H_b, I	U
3.	A, B, M_c	R	38.	A, G, H_a	L	73.	O, G, I	U	108.	M_a, H, T_a	**S**
4.	A, B, G	S	39.	A, G, H_b	S	74.	O, H_a, H_b	U	109.	M_a, H, T_b	U
5.	A, B, H_a	L	40.	A, G, H	S	75.	O, H_a, H	S	110.	M_a, H, I	U
6.	A, B, H_c	L	41.	A, G, T_a	S	76.	O, H_a, T_a	S	111.	M_a, T_a, T_b	U
7.	A, B, H	S	42.	A, G, T_b	U	77.	O, H_a, T_b	**U**	112.	M_a, T_a, I	S
8.	A, B, T_a	S	43.	A, G, I	S	78.	O, H_a, I	**U**	113.	M_a, T_b, T_c	**U**
9.	A, B, T_c	L	44.	A, H_a, H_b	S	79.	O, H, T_a	U	114.	M_a, T_b, I	U
10.	A, B, I	S	45.	A, H_a, H	L	80.	O, H, I	U	115.	G, H_a, H_b	U
11.	A, O, M_a	S	46.	A, H_a, T_a	L	81.	O, T_a, T_b	**U**	116.	G, H_a, H	S
12.	A, O, M_b	L	47.	A, H_a, T_b	S	82.	O, T_a, I	S	117.	G, H_a, T_a	S
13.	A, O, G	S	48.	A, H_a, I	S	83.	M_a, M_b, M_c	S	118.	G, H_a, T_b	**U**
14.	A, O, H_a	S	49.	A, H_b, H_c	S	84.	M_a, M_b, G	S	119.	G, H_a, I	**S**
15.	A, O, H_b	S	50.	A, H_b, H	L	85.	M_a, M_b, H_a	S	120.	G, H, T_a	U
16.	A, O, H	S	51.	A, H_b, T_a	S	86.	M_a, M_b, H_c	S	121.	G, H, I	U
17.	A, O, T_a	S	52.	A, H_b, T_b	L	87.	M_a, M_b, H	S	122.	G, T_a, T_b	**U**
18.	A, O, T_b	S	53.	A, H_b, T_c	S	88.	M_a, M_b, T_a	U	123.	G, T_a, I	**U**
19.	A, O, I	S	54.	A, H_b, I	S	89.	M_a, M_b, T_c	U	124.	H_a, H_b, H_c	S
20.	A, M_a, M_b	S	55.	A, H, T_a	S	90.	M_a, M_b, I	U	125.	H_a, H_b, H	S
21.	A, M_a, G	R	56.	A, H, T_b	U	91.	M_a, G, H_a	L	126.	H_a, H_b, T_a	S
22.	A, M_a, H_a	L	57.	A, H, I	S	92.	M_a, G, H_b	S	127.	H_a, H_b, T_c	**U**
23.	A, M_a, H_b	S	58.	A, T_a, T_b	S	93.	M_a, G, H	S	128.	H_a, H_b, I	**U**
24.	A, M_a, H	S	59.	A, T_a, I	L	94.	M_a, G, T_a	S	129.	H_a, H, T_a	L
25.	A, M_a, T_a	S	60.	A, T_b, T_c	S	95.	M_a, G, T_b	U	130.	H_a, H, T_b	U
26.	A, M_a, T_b	U	61.	A, T_b, I	S	96.	M_a, G, I	S	131.	H_a, H, I	S
27.	A, M_a, I	S	62.	O, M_a, M_b	S	97.	M_a, H_a, H_b	S	132.	H_a, T_a, T_b	**U**
28.	A, M_b, M_c	S	63.	O, M_a, G	S	98.	M_a, H_a, H	L	133.	H_a, T_a, I	S
29.	A, M_b, G	S	64.	O, M_a, H_a	L	99.	M_a, H_a, T_a	L	134.	H_a, T_b, T_c	**U**
30.	A, M_b, H_a	L	65.	O, M_a, H_b	S	100.	M_a, H_a, T_b	U	135.	H_a, T_b, I	**U**
31.	A, M_b, H_b	L	66.	O, M_a, H	S	101.	M_a, H_a, I	S	136.	H, T_a, T_b	**U**
32.	A, M_b, H_c	L	67.	O, M_a, T_a	L	102.	M_a, H_b, H_c	L	137.	H, T_a, I	**U**
33.	A, M_b, H	S	68.	O, M_a, T_b	U	103.	M_a, H_b, H	S	138.	T_a, T_b, T_c	U
34.	A, M_b, T_a	S	69.	O, M_a, I	S	104.	M_a, H_b, T_a	S	139.	T_a, T_b, I	S
35.	A, M_b, T_b	L	70.	O, G, H_a	S	105.	M_a, H_b, T_b	S			

Table A.1. Wernick's problems represented by their three characteristic points and their status by a letter in {L, R, S, U}. Problems written in boldface were proven by Schreck and Mathis (2014) using algebraic tools.

1.	A, B, E_a	S	36.	A, M_a, N	S	71.	E_a, H, T_b	U	106.	E_a, M_b, T_c	**U**
2.	A, B, E_c	S	37.	A, M_b, N	S	72.	E_a, H_a, H_b	S	107.	E_a, N, O	S
3.	A, B, N	S	38.	A, N, O	S	73.	E_a, H_a, I	S	108.	E_a, N, T_a	S
4.	A, E_a, E_b	S	39.	A, N, T_a	**U**	74.	E_a, H_a, M_a	L	109.	E_a, N, T_b	**U**
5.	A, E_a, G	S	40.	A, N, T_b	**U**	75.	E_a, H_a, M_b	S	110.	E_a, O, T_a	**U**
6.	A, E_a, H	R	41.	E_a, E_b, E_c	S	76.	E_a, H_a, N	L	111.	E_a, O, T_b	**U**
7.	A, E_a, H_a	L	42.	E_a, E_b, G	S	77.	E_a, H_a, O	S	112.	E_a, T_a, T_b	**U**
8.	A, E_a, H_b	L	43.	E_a, E_b, H	S	78.	E_a, H_a, T_a	L	113.	E_a, T_b, T_c	**U**
9.	A, E_a, I	S	44.	E_a, E_b, H_a	S	79.	E_a, H_a, T_b	**U**	114.	G, H, N	R
10.	A, E_a, M_a	S	45.	E_a, E_b, H_c	S	80.	E_a, H_b, H_c	L	115.	G, H_a, N	S
11.	A, E_a, M_b	S	46.	E_a, E_b, I	U	81.	E_a, H_b, I	**S** *	116	G, I, N	U
12.	A, E_a, N	S	47	E_a, E_b, M_a	L	82.	E_a, H_b, M_a	L	117.	G, M_a, N	S
13.	A, E_a, O	S	48.	E_a, E_b, M_c	S	83.	E_a, H_b, M_b	S	118.	G, N, O	R
14.	A, E_a, T_a	S	49.	E_a, E_b, N	L	84.	E_a, H_b, M_c	S	119.	G, N, T_a	U
15.	A, E_a, T_b	U	50.	E_a, E_b, O	S	85.	E_a, H_b, N	L	120.	H, H_a, N	S
16.	A, E_b, E_c	S	51.	E_a, E_b, T_a	**U**	86.	E_a, H_b, O	S	121.	H, I, N	U *
17.	A, E_b, G	S	52.	E_a, E_b, T_c	U	87.	E_a, H_b, T_a	**U**	122.	H, M_a, N	S
18.	A, E_b, H	S	53.	E_a, G, H	S	88.	E_a, H_b, T_b	U	123.	H, N, O	R
19.	A, E_b, H_a	S	54.	E_a, G, H_a	S	89.	E_a, H_b, T_c	**U**	124.	H, N, T_a	U
20.	A, E_b, H_b	L	55.	E_a, G, H_b	S	90.	E_a, I, M_a	S	125.	H_a, H_b, N	L
21.	A, E_b, H_c	S	56.	E_a, G, I	**U**	91.	E_a, I, M_b	**U**	126.	H_a, I, N	S
22.	A, E_b, I	**U**	57.	E_a, G, M_a	S	92.	E_a, I, N	S *	127.	H_a, M_a, N	L
23.	A, E_b, M_a	S	58.	E_a, G, M_a	S	93.	E_a, I, O	**U** *	128.	H_a, M_b, N	L
24.	A, E_b, M_b	S	59.	E_a, G, N	S	94.	E_a, I, T_a	**U**	129.	H_a, N, O	S
25.	A, E_b, M_c	S	60.	E_a, G, O	S	95.	E_a, I, T_b	**U**	130.	H_a, N, T_a	S
26.	A, E_b, N	S	61.	E_a, G, T_a	**U**	96.	E_a, M_a, M_b	L	131.	H_a, N, T_b	**U**
27.	A, E_b, O	S	62.	E_a, G, T_b	**U**	97.	E_a, M_a, N	R	132.	I, M_a, N	S
28.	A, E_b, T_a	**U**	63.	E_a, H, H_a	L	98.	E_a, M_a, O	S	133.	I, N, O	U *
29.	A, E_b, T_b	**U**	64.	E_a, H, H_b	L	99.	E_a, M_a, T_a	S	134.	I, N, T_a	**U** *
30.	A, E_b, T_c	**U**	65.	E_a, H, I	S	100.	E_a, M_a, T_b	**U**	135.	M_a, M_b, N	L
31.	A, G, N	S	66.	E_a, H, M_a	S	101.	E_a, M_b, M_c	S	136.	M_a, N, O	S
32.	A, H, N	S	67.	E_a, H, M_b	S	102.	E_a, M_b, N	L	137.	M_a, N, T_a	S
33.	A, H_a, N	S	68.	E_a, H, N	S	103.	E_a, M_b, O	S	138.	M_a, N, T_b	**U**
34.	A, H_b, N	S	69.	E_a, H, O	S	104.	E_a, M_b, T_a	**U**	139.	N, O, T_a	U
35.	A, I, N	**U**	70.	E_a, H, T_a	S	105.	E_a, M_b, T_b	**U**	140.	N, T_a, T_b	**U**

Table A.2. Connelly's corpus Connelly (2009). A status written in boldface indicates that the result was not known. An asterisk means that a geometric preprocessing was needed for the numerical checking. Much more problems needed preprocessing when considering generic problems,that is, with symbolic parameters instead of numerical values.

```
      change F by collinear(X, h, h(X))
    ].

if
      vertices(X,Y,Z) and
      h(X) and e(X) and
      perpend(X,h(X),Y, Z) as F
then [
      change F by collinear(X, h(X), e(X))
      ].

if
      vertices(X,Y,Z) and
      i and t(X) and
      onAngleBisector(t(X), X, Y, Z) as F
then [
      change F by collinear(X, i, t(X))
      ].

if
      vertices(X,Y,Z) and
      t(X) and t(Y) and
      onAngleBisector(t(X), X, Y, Z) as F
then [
      change F by onAngleBisector(t(X), X, Y, t(Y))
      ].

if
      vertices(X,Y,Z) and
      t(X) and t(Z) and
      onAngleBisector(t(X), X, Y, Z) as F
then [
      change F by onAngleBisector(t(X), X, t(Z), Z)
      ].

if
      vertices(X,Y,Z) and
      t(X) and m(Y) and
      onAngleBisector(t(X), X, Y, Z) as F
then [
      change F by onAngleBisector(t(X), X, Y, m(Y))
      ].

if
      vertices(X,Y,Z) and
      t(X) and m(Z) and
      onAngleBisector(t(X), X, Y, Z) as F
then [
      change F by onAngleBisector(t(X), X, m(Z), Z)
      ].

if
      vertices(X,Y,Z) and
      t(X) and h(Y) and
      onAngleBisector(t(X), X, Y, Z) as F
then [
      change F by onAngleBisector(t(X), X, Y, h(Y))
      ].
```

```
if
      vertices(X,Y,Z) and
      t(X) and h(Z) and
      onAngleBisector(t(X), X, Y, Z) as F
then [
      change F by onAngleBisector(t(X), X, h(Z), Z)
      ].

if
n and h(X) and
onPerpendBisector(2*n,a+b,b+c) as F
then
    [
change F by onPerpendBisector(2*n,a+b,2*h(X))
    ].

if
vertices(X,Y,Z) and
e(X) and h(Y)
and [P] among [Y,Z] and
        perpend(P,2*e(X)-X,_,_) as F2
then
    [
change F2 by collinear(2*e(X)-X, h(Y), Y)
    ].


if
vertices(X,Y,Z) and
e(X) and o and
perpend(X,2*e(X)-X,_,_) as F1 and
[P] among [Y,Z] and
        perpend(P,2*e(X)-X,_,_) as F2
then
    [
change F1 by egx(midpoint(Y+Z,2*X), midpoint(2*e(X),2*o)),
change F2 by egy(midpoint(Y+Z,2*X), midpoint(2*e(X),2*o))
    ].

if
    vertices(X,Y,Z) and
    h(X) and t(X) and
    collinear(Y,t(X),Z) as F1 and
    collinear(Y,h(X),Z) as F2
then [
       change F1 by collinear(h(X),Y,t(X)),
       change F2 by collinear(h(X),Z,t(X))
     ].

 if
    vertices(X,Y,Z) and
    h(X) and m(X) and
    collinear(Y,h(X),Z) as F1
then [
       change F1 by collinear(h(X),Y,m(X))
     ].

 if
    vertices(X,Y,Z) and
```

```
    t(X) and m(X) and
    collinear(Y,t(X),Z) as F1
then [
       change F1 by collinear(t(X),Y,m(X))
     ].

if
    [P] stated among [m(X), t(X), h(X)] and
    vertices(X,Y,Z) and
    onAngleBisector(i, a, b, c) as F1 and
    onAngleBisector(i, b, c, a) as F2
then [
       change F1 by onAngleBisector(i, Y, X, P),
       change F2 by onAngleBisector(i, Z, X, P)
      ].

 if
    e(X) and e(Y) and
    perpend(Y,2*e(X)-X,Z,X) as F1
then [
       change F1 by collinear(2*e(X)-X,e(Y),Y)
      ].

 if
    e(X) and e(Y) and
    perpend(Z,2*e(X)-X,Y,X) as F1
then [
       change F1 by collinear(2*e(X)-X,e(Y),Y)
      ].
```