# 1 Introduction

## *1.1 Goals of the Thesis*

For automated geometrical theorem proving, various approaches are known in the literature ([44], [10], [28], [30], [11], etc.) and various software systems have implemented some of these methods (some of the most recent systems: [11], [41], [15], [34]) including attractive graphical interfaces for specifying geometrical proof problems and for graphical illustration of geometrical proofs. *Theorema* is a mathematical software system (implemented in *Mathematica* and, hence, platform-independent) that aims at providing *one uniform* logical and software technological frame for automated theorem proving in all areas of mathematics or, in other words and more generally, for formal mathematics, i.e. proving, solving, and simplifying mathematical formulae relative to mathematical knowledge bases, see [7], [6]. The main goal of this thesis is

- to implement known algorithms for automated geometrical theorem proving within the uniform frame of *Theorema*,
- to improve some of these methods by adding and developing certain subalgorithms,
- to experiment and compare the various approaches on the basis of their implementation in one common frame,
- to provide an attractive graphical interface for geometric theorem proving in Theorema, and
- to obtain new algorithms for geometrical theorem proving by combining the known methods in various ways.

## *1.2 Achievements of the Thesis*

The main achievements of this thesis are:

- the implementation of the following known geometrical theorem proving methods in the frame of Theorema:
  - Wu's method,
  - the Gröbner bases method,
  - the area method,
- new subalgorithms and heuristics for improving the above methods:

  - finding, automatically, an appropriate coordinate system for the coordinate-based geometrical proving methods that simplifies the proof problem,
  - incremental version of Wu's method by which the polynomial conditions are added one at a time and the characteristic set is computed after each addition,

- new algorithm for geometrical theorem proving by combining the known methods:
  - usage of the area method for simplifying intermediate formulae in Collins' cylindrical algebraic decomposition method for theorem proving in the theory of real closed fields,

- implementation of an attractive graphical interface for geometrical theorem proving in Theorema:
  - drawing the graphical representation of geometrical theorem proving problems,
  - possibility for moving points in the graphical representation of geometrical theorem proving problems,
  - visualization of conditions described by inequalities in addition to visualization of conditions described by equalities,
  - possibility of checking conjectures for the configurations visualized by the graphical interface,

- a systematic exploration of geometric configurations:
  - systematic exploration using the area method (collinear points, congruent segments, parallel and perpendicular lines, triangles with the same area)
  - series of proofs reusing the already computed characteristic sets (incremental Wu's method)

# 2 Historical Overview

## *2.1 First Steps: Axiomatic Systems*

A first step towards formal proving of theorems in geometry was in 1899 Hilbert's *Grundlagen der Geometrie* (*The Foundations of Geometry* [24]). He proposed five groups of axioms, showed the consistency and independence of the sets of axioms and from them derived the various properties of Euclidean geometry. Hilbert's geometry consists of points, lines and planes as primitives and of relations between these for angle congruence and incidence amongst others. The relations between the primitives are completely determined by the axioms. Hilbert's insistence, that no geometrical intuition was needed to prove any results marked a clear departure from the geometry of Euclid. Ancient Greek geometry was meant as an axiomatization of concepts that were intuitively obvious, while Hilbert's abstracted geometry away from any concrete interpretation.

The role of Hilbert's *Grundlagen* in relation to automated geometry theorem proving is quite an important one. Indeed, the formalization given by Hilbert made clear for the first time the possibility of mechanizing elementary geometry: reasoning should be capable of being carried out according to purely mechanical rules, and for doing geometry it suffices to apply these rules to the axioms slavishly without knowing what they mean. This was realized very early by Poincaré, who, in his review of the Grundlagen (1902) [32] observed, that in this way "One could give the axioms to a logic machine, for example the logical piano of Stanley Jevons, and one would see all of geometry emerge from it."

In 1926 Alfred Tarski gave a new axiomatic development of elementary Euclidean geometry [39]. Around 1930 he proved, that his system admits elimination of quantifiers. Later on he drew several fundamental corollaries: the theory is complete, decidable and there is a constructive consistency proof for it [38]. During the period 1955-65 Tarski and his students simplified substantially the initial system, obtaining a rather elegant and simple set of axioms. In contrast with Hilbert's geometry, Tarski's universe contains only points with two primitive relations on them: betweenness and equidistance.

As far as actual machine geometry is concerned, the first step was Gelernter's *Geometry Machine* [21] based on logical deduction. It included specific knowledge about the geometry domain and had a backward chaining search strategy.

Gilmore [22], Nevins [31] and Elcock [20] used the same approach, with additions, such as forward chaining. The axiomatic or synthetic approaches used by Gelernter and the others above were, however not very successful in proving or discovering any non-trivial theorems. They tried different search strategies and heuristics. Still, the main problem, the low efficiency persisted. This is due, among others, to the huge search space of geometry rule applications.

## 2.2 Algebraic Techniques

Already in 1969, Cerutti and Davis used a symbolic manipulation system called FORMAC to prove theorems in elementary analytic geometry [8]. They used Descartes' method to prove Pappus' theorem. Unfortunately their techniques were not investigated further at that time. It took several more years before the potential of the algebraic approach was finally recognized. This changed a rather stagnant area into one of the leading fields in automated reasoning.

The algebraic techniques usually proceed through the introduction and use of some coordinate system. The geometry problem or statement is then translated into an algebraic form that can be dealt with using a number of powerful algorithms. In each of the next paragraphs we assume, that the first step of the algorithm is finding an appropriate coordinate system and transforming the geometrical statement into polynomial equations, inequations and inequalities of geometry quantities, representing the hypothesis and conclusion of the statement. We outline a few of the most successful algebraic techniques next.

### 2.2.1 Wu's Method

The work of Wu Wen-tsün laid the foundations for automatic theorem proving in geometry. Inspired by ancient Chinese mathematics and using the concept of characteristic sets introduced by J. F. Ritt [35][36], Wu published his method first in 1977 in China [44]. The method can be applied to problems of constructive type in which no order relation is involved, thus the algebraic transcription of the statement contains only polynomial equalities. The main steps of the algorithm are:

- Triangulation of the hypothesis polynomials using pseudo division. For certain problems decomposition or check of the irreducibility is needed in this step.
- Successive pseudo division of the conclusion polynomial to compute the finite remainder $R_0$. If $R_0 = 0$, then by the remainder formula we can infer the conclusion by adding the subsidiary conditions available after triangulation. If the final remainder is not zero, then the decomposition or check of irreducibility is needed to make further conclusions.
- Analysis of subsidiary conditions.

This simple algebraic procedure works well to prove a remarkable number of non-trivial theorems. Also several new theorems were discovered by this method. We have to mention the later contributions of Shang-Ching Chou [10] and Donming Wang [40] to the implementation and further efficientization of the algorithm.

### *2.2.2 Gröbner Bases Method*

The Gröbner bases method was introduced by Bruno Buchberger in 1965. It is a powerful algorithmic tool for solving many important problems in polynomial ideal theory [4]. It can be applied to the same class of theorems as Wu's method. There are two main approaches to use Gröbner basis in proving statements in geometry. As a first step both need the conversion of a geometry statement into the corresponding polynomial equations.

a. The first approach, mainly due to Kutzler and Stifter [28] and also mentioned by Chou [10] consists of:

- Computing the Gröbner bases $G$ of the set of hypotheses polynomials.
- Computing the normal form of the conclusion polynomial modulo $G$. The theorem is true, if this normal form equals to zero.

b. The other approach is due to Kapur [25] and involves deciding whether a finite set of polynomial equations does not have a solution in an algebraically closed field. This refutational approach is equivalent to checking whether 1 is in the ideal generated by the polynomials. Therefor the procedure consists of:

- Computing the Gröbner bases $G^*$ of the union of the set of hypotheses polynomials and $\{z\,c-1\}$, where $c$ is the conclusion polynomial and $z$ is a new variable. The theorem is true if $1 \in G^*$.

The above two methods are very efficient for problems whose algebraic formulation involves only polynomial equalities, but neither of them can deal with polynomial inequalities, thus no geometrical theorem that involves the notion of betweenness can be treated.

### *2.2.3 Cylindrical Algebraic Decomposition*

Starting from Tarski's complex decision procedure for elementary geometry [38], Collins developed an algorithm for quantifier elimination in real closed fields by cylindrical algebraic decomposition [18]. This algorithm is used by Kutzler and Stifter [30] to prove geometry theorems. Their method consists of:

- Finding the independent variables of the statement and verifying their independentness (existential quantifier elimination)
- Finding subsidiary conditions for the statement to be true (universal quantifier elimination)
- Verifying, if the found subsidiary conditions can be fulfilled under the hypothesis (existential quantifier elimination)

This method is important, because it can also deal with geometrical statements whose algebraic transcription involves inequalities. Furthermore, it can be used for proving "insufficiently specified" geometry theorems, obtaining the missing hypothesis. However, it is not really used for theorem proving because of the increased computing time of the existing implementations, not being yet possible the proof of nontrivial geometry theorems using just this method.

## *2.3 Coordinate-Free Techniques*

The main drawbacks of the algebraic methods are the long and hard to read proofs they usually generate. These proofs lack any geometrical intuition, they consist of polynomial manipulation only. We next describe a few of the so-called coordinate-free techniques that have been used successfully in automated geometry theorem proving, although they generally are not as powerful as those involving coordinates. Their main advantage is that the proofs generated can be interpreted geometrically.

### *2.3.1 The Area Method*

In this section and the next we present methods that are based on geometric invariants and high-level geometric lemmas about these invariants. This first powerful method was developed originally in China by J. Z. Zhang for geometry education (published in 1982).

The area method uses as geometric invariants signed areas and Pythagorean differences of triangles and can be used for theorems of constructive type involving line intersection, perpendicularity and certain types of circle intersections. The main steps of the algorithm (as presented by S. C. Chou et. al. in [11]) are:

- Definition of the construction steps of each point using admitted constructions and representation of the goal by means of signed areas, Pythagorean differences and ratios of segments involving the constructed points.
- Verification of the generated non-degenerate conditions.
- Elimination of the constructed points one after the other, in reverse order of their construction, using some elimination lemmas.
- Representation of the resulted expression in terms of independent area coordinates of the free points. The theorem is true, if the two sides of the obtained expression are literally equal.

### *2.3.2 The Full-Angle Method*

The notion of angle and its associated properties provide powerful tools that have been used traditionally in geometry theorem proving. In their work on producing human readable automated proofs, S. C. Chou et. al. [14] also propose a method based on the concept of full angles, that can be used to deal with classes of theorems that pose problems to the area method. A full-angle of an ordered pair of lines is the angle from the first line to the second measured anti-clockwise. It is essential, that the notion is defined by the help of lines rather than rays. This simplifies proofs by eliminating case-splits in certain cases.

The full-angle is used then to express other familiar properties, like perpendicularity and parallelity. In this method most of the elimination lemmas eliminate lines rather then points.

Although this method gives rather elegant proofs for many difficult theorems, it is not complete. Combined with the area method a powerful and complete method is obtained for the class of constructive geometry statements.

### *2.3.3 The Vector Method*

Chou, Gao and Zhang have also used vector calculations in automated geometry theorem proving [12]. They assert a set of basic rules about the operations that can be carried out on vectors. Replacement lemmas are then derived using the basic axioms of the theory. The algorithm is similar to that of area method: given a sequence of constructions and a conclusion as a vector expression, the points (i.e. vector variables) are eliminated one at a time from the later using the elimination lemmas, until only independent vector variables are left. The theorem is true, if the result is identically zero.

### *2.3.4 Clifford Algebra*

A geometric algebra is one which can be used as a suitable grammar for representing basic geometrical relations. Clifford Algebra has been proposed as a unifying framework for various fields encompassing physics, mathematics and computer science, since it admits a geometric interpretation [23].

## *2.4 Implementations*

Many software systems implemented the various algorithms, as geometry theorem proving has always been a challenging artificial intelligence problem. Some of the documented geometric software systems are [42]:

- Geometry Machine – based on the logical axiomatic approach (Gelernter [21])
- GEOM – based on the logical axiomatic approach, programmed in Prolog (Coelho, Pereira [17])
- GEO-prover – based on the algebraic methods, programmed in Lisp (Chou [9])
- ALGE-Prover/ALGE-Prover II/ GEOMETER – based on the algebraic methods, programmed in Lisp (Ko, Kapur [27] , [26] , [19])
- CPS/PS/DPMS1/DPMS2 – implementing Wu's method in Fortran (Gao, Wang)
- GPP – based on the algebraic methods (Kutzler, Stifter [29])
- EUCLID – implementing the area method (Chou, Gao, Zhang [11])
- GEOTHER – based on the algebraic methods (Wang [41])
- CINDARELLA – based on bracket algebra (Crapo, Richter-Gebert [33] , [34])
- Geometry Expert – implementing many essentially different methods (Chou, Gao, Zhang [15])
- Cabri Geometri – (Bellemain, Laborde [1])

# 3 Preliminaries

## *3.1 Mechanization of Geometry*

By elementary geometry we shall mean the one described in Hilbert's "Grundlagen der Geometrie", in which no differentiation is involved. It is well-known by the theorem of Tarski that the ordinary Euclidean geometry, as one of such elementary geometries, is decidable, i.e. there exists an algorithmic method, by which any meaningful statement in this geometry can be shown, in a finite number of steps, to be either true as a real theorem, or false. Any elementary geometry possessing such an algorithm is called mechanizable, and the theorem asserting that the geometry does poses such an algorithmic method is called the mechanization theorem. In the mechanizable case we may program according to the algorithm shown to exist and practice on a computer such that the proof (or disproof) of the theorem is carried out on the computer. This method is called automated geometry theorem proving. In this sense the above mentioned theorem of Tarski is the mechanization theorem of ordinary Euclidean geometry.

All these notions can be naturally extended to a certain class of meaningful statements in the geometry in question. Usually for a certain class of statements a more efficient algorithm can exist.

We shall deal with the class of constructive geometry statements not involving order relations, using different algorithms, that is different mechanization theorems. In addition we shall also consider a class of constructive geometry statements possibly involving inequalities.

## *3.2 Constructive Geometry Statements*

In this section we present the class of problems our prover can deal with: constructive statements in plane Euclidean geometry. We consider three kinds of geometric objects: points, lines and circles. To make sure that the constructed objects are well-defined, we need to assume some non-degenerate conditions (denoted by ndg conditions in what follows). These conditions are automatically generated by the prover.

A straight line can be defined either by two distinct points, or a point and its direction. So it can be given in one of the following forms:

line[*A, B*] is the line passing through two points *A* and *B*.

pline[*C, A, B*] is the line passing through point *C* and parallel to line[*A, B*].

tline[*C, A, B*] is the line passing through point *C* and perpendicular to line[*A, B*].

To make sure that all three kinds of lines are well-defined, we need to assume $A \neq B$.

circle[$O$, $A$] is the circle with point $O$ as its center and passing through point $A$. Again, $A \neq O$ has to be assumed.

**Definition 1**

*A construction is one of the following ways to introduce new points. For each construction we also give its ndg conditions.*

C1 point[$A_1$, …, $A_n$]. *Take arbitrary points* $A_1$, …, $A_n$ *in the plain. Each* $A_i$ *is a free point.*

C2 pon[$A$, $ln$]. *Take a point A on line ln. The ndg condition of* C2 *is the ndg condition of line ln.*

C*3* pon[$A$, circle[$O$, $U$]]. *Take a point A on* circle[$O$, $U$]. *The ndg condition is* $O \neq U$.

C4 inter[$A$, $ln1$, $ln2$]. *Point A is the intersection of line* $ln1$ *and line* $ln2$. *The ndg condition is* $ln1 \not\parallel ln2$.

C5 inter[$A$, $ln$, circle[$O$, $P$]]. *Point A is the intersection of line ln and* circle[$O$, $P$]. *The ndg condition is* $P \neq O$ *and line ln is not degenerate.*

C6 inter[$A$, circle[$O_1$, $P$], circle[$O_2$, $Q$]]. *Point A is the intersection of* circle[$O_1$, $P$] *and* circle[$O_2$, $Q$]. *The ndg condition is* $P \neq O_1$ *and* $Q \neq O_2$. *For the area method we consider* C6 *a valid construction only if the two circles are defined by the same point, that is* $P = Q$.

C7 pratio[A, W, U, V, r]. *Take a point A on the line passing through W and parallel to line UV such that* $\overline{WA} = r\,\overline{UV}$, *where r can be a rational number, a rational expression of some geometric quantities, or a variable. The ndg. condition is* $U \neq V$. ( $\overline{UV}$ *denotes the length of the oriented segment UV*)

C8 tratio[A, U, V, r]. *Take a point A on the line passing through U and perpendicular to line UV such that* $\overline{UA} = r\,\overline{UV}$, *where r can be a rational number, a rational expression of some geometric quantities, or a variable. The ndg. condition is* $U \neq V$.

*The point A in each construction is said to be introduced by that construction.*

The algebraic methods could be easily generalized for other second order curves (ellipses, parabolas, hyperbolas) just adding the corresponding translating functions from external into algebraic form. For the area method new elimination lemmas would be needed.

**Definition 2**

*A constructive geometry statement is a list* $S = (C_1, C_2, \ldots, C_k, G)$ *where*

1. $C_i$, $i = 1, \ldots, k$, *are constructions such that the point introduced by each* $C_i$ *must be different from points introduced by* $C_j$, $i = 1, \ldots, i-1$ *and other points occurring in* $C_i$ *must be introduced before; and*
2. *G, the conclusion, is a geometrical property about points introduced by* $C_i$ *not involving order relation.*

## *3.3 Constructive Geometry Statements Possibly Involving Inequalities*

As one of our provers deals also with constructive geometry statements in plane Euclidean geometry that involve order relation we have to define this class of problems.

**Definition 3**

*A constructive geometry statement possibly involving inequalities is a list* $S = (C, H, G)$, *where*

1. $C = \{C_1, C_2, \ldots, C_m\}$ *is a construction set. Each* $C_i$ *is a construction such that the point introduced by it must be different from points introduced by* $C_j$, $j = 1, \ldots, i-1$ *and other points occurring in* $C_i$ *must be introduced before;*
2. $H = \{H_1, H_2, \ldots, H_n\}$ *is a set of additional geometric properties whose algebraic representation may involve inequalities. All the points appearing in H have to be introduced by the constructions.*
3. $G = \{G_1, G_2, \ldots, G_k\}$, *the conclusion, is a set of geometric properties of the points introduced by the constructions* (*it may also contain inequalities*).

# 4 Basic Elements of the *Theorema* System

## *4.1 Theorema*

*Theorema* is a mathematical software system (implemented in *Mathematica* and, hence, platform-independent) that aims at providing one uniform logical and software technological frame for automated theorem proving in all areas of mathematics or, in other words and more generally, for formal mathematics, i.e. proving, solving, and simplifying mathematical formulae relative to mathematical knowledge bases, see [7], [6]. *Theorema* is being developed at the RISC Institute by the *Theorema* Group under the direction of Bruno Buchberger. The functionalities described in this thesis are available as *Theorema* functions that take *Theorema* formulae as input.

## *4.2 Theorema Constructs as Input*

To describe the geometric configuration we use three *Theorema* constructs, *Proposition, Assumption* and *KnowledgeBase*, as presented in detail in 5.2.1.

Examples:

**Proposition**["Pappus", any[A, B, A1, B1, C, C1, P, Q, S],
  point[A, B, A1, B1] ∧
    pon[C, line[A, B]] ∧
    pon[C1, line[A1, B1]] ∧
    inter[P, line[A, B1], line[A1, B]] ∧ inter[Q, line[A, C1], line[A1, C]] ∧
    inter[S, line[B, C1], line[B1, C]]
   ⇒ collinear[P, Q, S]]

**KnowledgeBase**["Pappus_g", any[A, B, A1, B1, C, C1],
  {{A, {10, 120}}, {B, {100, 150}}, {A1, {25, 60}},
    {B1, {120, 50}}, {C, {260, 200}}, {C1, {250, 45}}}]

**KnowledgeBase**["Ex", any[A, B, C, D, A1, A2, B1, r],
  point[A, B, C] ∧
    inter[D, pline[A, B, C], pline[C, A, B]] ∧
    divpoint[A1, C, D, r/(r − 1)] ∧ divpoint[B1, D, A, r/(r − 1)] ∧
    inter[A2, line[A, A1], line[B, B1]]]

**Assumption**["Trapezoid", any[A, B, C, D],
  triangle[A, B, C] ∧ pon[D, pline[C, A, B]] ∧ line[B, C] ∧ line[A, D]]

## 4.3 Simplify, Prove, Compute, Explore

To implement the main functionalities of our prover we use the *Theorema* functions *Simplify*, *Prove*, *Compute* and *Explore*, as presented in more detail in section 5.1. All these functions are general for any field of mathematics. The method they should apply is specified using the *by* option. This way only the domain specific part of the prover/simplifier/evaluator has to be written. Further specific options may also be given using *ProverOptions* (*SimplifierOptions*, *EvaluatorOptions*, *ExplorerOptions*)

Examples:

```
Simplify[Proposition["Pappus"], by → GraphicSimplifier,
  using → KnowledgeBase["KB_Pappus"]]

Prove[Proposition["Pappus"],
  by → GeometryProver, ProverOptions → {Method → "Area"}]

Compute[trarea[A, B, A2] / qarea[A, B, C, D] ,
  by → AreaCompute, using → KnowledgeBase["Ex"],
  EvaluatorOptions → {VerifyNdg → False, ShowSteps → False}]

Explore[Assumption["Trapezoid"], by → AreaExplore];
```

## 4.4 Theorema Notebooks as Output

*Theorema* notebooks provide an ingenious presentation of the proofs. The cell structure makes it possible to visualize the proof in all its detail or look only at the main steps. One can look at certain parts in detail while all other details are hidden. As one cannot take advantage of these possibilities in the printed form of the notebook we shall use "==== *closed cell* ====" to signal that the cell contains further information that is not visible. In the examples we also use "==== *explanation text* ====" to skip the printout of parts of the proofs that are similar to previous parts.

Examples:

========== Output Generated by the Program ==========

Prove:

(Proposition (Pappus))

$$\forall_{A,B,A1,B1,C,C1,P,Q,S} (\text{point}[A, B, A1, B1] \wedge \text{pon}[C, \text{line}[A, B]] \wedge \text{pon}[C1, \text{line}[A1, B1]] \wedge \text{inter}[P, \text{line}[A, B1], \text{line}[A1, B]] \wedge \text{inter}[Q, \text{line}[A, C1], \text{line}[A1, C]] \wedge \text{inter}[S, \text{line}[B, C1], \text{line}[B1, C]] \Rightarrow \text{collinear}[P, Q, S])$$

with no assumptions.

We shall use the area method.

First we have to transform the problem in internal form for the area method. We have to prove that constructions

{$A$, $B$, $A1$, $B1$} free points
$C\,A \parallel A\,B$ and $\frac{\overline{A\,C}}{\overline{A\,B}}=r_C$ with ndg. {$A \neq B$}
$C1\,A1 \parallel A1\,B1$ and $\frac{\overline{A1\,C1}}{\overline{A1\,B1}}=r_{C1}$ with ndg. {$A1 \neq B1$}
$P = A\,B1 \cap A1\,B$ with ndg. {$A \neq B1$, $A1 \neq B$, $A\,B1 \not\parallel A1\,B$}
$Q = A\,C1 \cap A1\,C$ with ndg. {$A \neq C1$, $A1 \neq C$, $A\,C1 \not\parallel A1\,C$}
$S = B\,C1 \cap B1\,C$ with ndg. {$B \neq C1$, $B1 \neq C$, $B\,C1 \not\parallel B1\,C$}

imply that points {$P$, $Q$, $S$} are collinear

Explanations ==== *closed cell* ====

First we have to verify the non-degenerate conditions for each construction, then to prove our statement.

Step 1 ==== *closed cell* ====

Step 2 ==== *closed cell* ====

Step 3 ==== *closed cell* ====

Step 4

As all the non-degenerate conditions are verified now we have to prove

{$A$, $B$, $A1$, $B1$} free points
$C\,A \parallel A\,B$ and $\frac{\overline{A\,C}}{\overline{A\,B}}=r_C$ with ndg. {$A \neq B$}
$C1\,A1 \parallel A1\,B1$ and $\frac{\overline{A1\,C1}}{\overline{A1\,B1}}=r_{C1}$ with ndg. {$A1 \neq B1$}
$P = A\,B1 \cap A1\,B$ with ndg. {$A \neq B1$, $A1 \neq B$, $A\,B1 \not\parallel A1\,B$}
$Q = A\,C1 \cap A1\,C$ with ndg. {$A \neq C1$, $A1 \neq C$, $A\,C1 \not\parallel A1\,C$}
$S = B\,C1 \cap B1\,C$ with ndg. {$B \neq C1$, $B1 \neq C$, $B\,C1 \not\parallel B1\,C$}
$\Rightarrow$points {$P$, $Q$, $S$} are collinear

Eliminating point $S$: ==== *closed cell* ====

Eliminating point $Q$: ==== *closed cell* ====

Eliminating point $P$:

We eliminate point $P$ using construction:
$P = A\,B1 \cap A1\,B$ with ndg. {$A \neq B1$, $A1 \neq B$, $A\,B1 \not\parallel A1\,B$}
==== *closed cell* ====

After ordering the points we obtain: ==== *closed cell* ====

The simplified expression is:

$$(-\bullet S_{\{A,A1,C\}}\,\bullet S_{\{A,B,B1\}}\,\bullet S_{\{B,A1,C1\}}\,\bullet S_{\{B1,C,C1\}} + \bullet S_{\{A,B,A1\}}\,\bullet S_{\{A1,C,C1\}}\,(\bullet S_{\{A,B1,C1\}}\,\bullet S_{\{B,B1,C\}} + \bullet S_{\{A,B,B1\}}\,\bullet S_{\{B1,C,C1\}}))\,/\,((\bullet S_{\{A,A1,C\}} + -\bullet S_{\{A1,C,C1\}})\,(\bullet S_{\{A,B,A1\}} + -\bullet S_{\{B,A1,B1\}})\,(\bullet S_{\{B,B1,C\}} + -\bullet S_{\{B1,C,C1\}}))$$

Eliminating point $C1$: ==== *closed cell* ====

Eliminating point $C$: ==== *closed cell* ====

The initial problem was proved, as we obtained 0 == 0.

========== End of Generated Output ==========

# 5 Implementation of Known Proving Methods

## *5.1 Main Functionalities*

### *5.1.1 Proving with GeometryProver*

In order to fit in the *Theorema* system for all proofs we use the *Theorema* function *Prove*, with our user prover *GeometryProver* implemented in the *Geometry* package. This prover applies for the different proving methods the corresponding basic prover specified by the *ProverOption Method*. Possible values for this option are: "*Wu*", "*GroebnerProver*", "*Area*", "*AreaCAD*". For each method different further options can be specified.

The function call is:

`Prove[`*`proposition`*`, by→GeometryProver, ProverOptions→`*`options`*`]`

where

- *proposition* is the proposition that should be proved,
- *options* specify some options admitted by the given prover.

Examples for different prover calls for the same proposition:

```
Prove[Proposition["Pappus"],
  by → GeometryProver, ProverOptions → {Method → "Wu"}]

Prove[Proposition["Pappus"], by → GeometryProver,
  ProverOptions → {Method → "Wu", MyAxis → False}]

Prove[Proposition["Pappus"], by → GeometryProver,
  ProverOptions → {Method → "GroebnerProver", Refutation → True}]

Prove[Proposition["Pappus"],
  by → GeometryProver, ProverOptions → {Method → "Area"}]
```

### *5.1.2 Computation Problems*

We can use certain proving methods not only to prove certain propositions but also to compute various expressions that result from the given constructions. For this purpose we use the *Theorema* function *Compute* with different options for different methods.

The function call is:

`Compute[`*`expression`*`, by→AreaCompute, using→`*`knowledgebase`*`,`
`EvaluatorOptions→`*`options`*`]`

where

- *expression* specifies the expression that has to be computed,
- *knowledgebase* describes the given geometric configuration,
- *options* specify some options admitted by the given evaluator.

Example:

```
Compute[trarea[A, B, A2] / qarea[A, B, C, D] ,
  by → AreaCompute, using → KnowledgeBase["cgzex2_53"],
  EvaluatorOptions → {VerifyNdg → False, ShowSteps → False}]
```

### *5.1.3 Obtaining Different Forms of the Input Problem*

To obtain various forms of the problem, we use the *Theorema* function *Simplify*, with various simplifiers implemented in the *GeomSimplifiers* package. These simplifiers take the input form of the problem and display it in the form the different provers deal with or draw the graph of the problem. We shall present these simplifiers in more detail in the respective sections.

The function call is:

```
Simplify[object, by→simplifier, using→knowledgebase,
SimplifierOptions→options]
```

where

- *object* is a Proposition or a KnowledgeBase object,
- *simplifier* specifies the simplifier that should be used, that is the transformation that has to be applied to the given configuration. Its values may be: GraphicSimplifier, AlgebraicSimplifier or AreaSimplifier
- *knowledgebase* specifies a knowledgebase with additional information
- *options* specify some options admitted by the given simplifier.

Examples for various simplifier calls for the same proposition:

```
Simplify[Proposition["Pappus"], by → GraphicSimplifier]

Simplify[Proposition["Pappus"], by → AreaSimplifier];

Simplify[Proposition["Pappus"],
  by → AlgebraicSimplifier, SimplifierOptions → {MyAxis → False}]
```

### *5.1.4 Systematic Exploration*

As a *Theorema* function for systematic exploration of a theory didn't exist, we implemented the function *Explore* in a way that fits in the general layout of *Theorema*. As input the function takes a knowledgebase that resulted from a previous exploration step, that contains all relevant information about previous explored configuration and a list of assumptions that should be added to the previous configuration. The knowledge base is not used at the first exploration step. As output the function produces a new knowledge base having its label name and containing all relevant information about the new configuration.

The function call is:

```
Explore[as, by→AreaExplore, using→kb, producing→name,
ExplorerOptions→options]
```

where

- *as* specifies the new geometric knowledge that should be used in exploring ,

- *kb* is the knowledgebase obtained in a previous step of exploration,

- *name* represents the label of the knowledgebase that should be obtained in the exploration step

- *options* specify some options admitted by the explorer.

Example of successive calls of *Explore* for exploring more and more complicated geometric configurations:

**Assumptions**["Trapezoid1", any[A, B, C, D],
  triangle[A, B, C] ∧ pon[D, pline[C, A, B]] ∧ line[B, C] ∧ line[A, D]]

kb1 = Explore[Assumptions["Trapezoid1"], by → AreaExplore];

**Assumptions**["Trapezoid2", any[A, B, C, D, M, N],
  midpoint[M, A, D] ∧ midpoint[N, B, C] ∧ line[M, N]]

kb2 = Explore[Assumptions["Trapezoid2"],
    by → AreaExplore, using → kb1, producing → Tr2];

**Assumptions**["Trapezoid3",
  any[A, B, C, D, M, N, X, P, Q, S], inter[X, line[M, N], line[B, D]] ∧
    inter[P, line[A, C], line[B, D]] ∧ inter[Q, line[B, C], pline[P, A, B]] ∧
    inter[S, line[P, Q], line[A, D]] ∧ line[M, C] ∧ line[A, Q]]

kb3 = Explore[Assumptions["Trapezoid3"],
    by → AreaExplore, using → kb2, producing → Tr3];

## 5.2 Problem Input

### 5.2.1 Input Formats Supported

- The geometric configuration can be given using two kinds of *Theorema* constructs:
  - If a geometric configuration is given that should be used as hypothesis in computing an expression we use

    **KnowledgeBase**[*label*, any[*variable_list*], *constr_conj*]

    where

    - *label* is the identifier of the knowledge base,

    - *variable_list* enumerates all the points and other variables that appear in the constructions

    - *constr_conj* is a formal conjunction of geometric constructions admitted by the provers and other methods.

Example:

**KnowledgeBase**["example", any[A, B, C, D, A1, A2, B1, r],
point[A, B, C] ∧
inter[D, pline[A, B, C], pline[C, A, B]] ∧
divpoint[A1, C, D, r/(r − 1)] ∧
divpoint[B1, D, A, r/(r − 1)] ∧
inter[A2, line[A, A1], line[B, B1]]]

  - If a geometric configuration is given that should be explored we use

**Assumptions**[*label*, any[*variable_list*], *constr_conj*]

where

- *label* is the identifier of the knowledge base,

- *variable_list* enumerates all the points and other variables that appear in the constructions

- *constr_conj* is a formal conjunction of geometric constructions admitted by the provers and other methods.

Example:

**Assumptions**["ex", any[O, A, B, C, P],
point[O, A] ∧
eqdist[O, B, O, A] ∧
eqdist[O, C, O, A] ∧
circle[O, A] ∧
inter[P, line[A, C], tline[B, A, C]]]

- If a geometry proposition is given that should be proved we use

**Proposition**[*label*, any[*variable_list*], *constr_conj*⇒ *concl_conj*]

where

- *label* is the identifier of the proposition,

- *variable_list* enumerates all the points and other variables that appear in the constructions

- *constr_conj* is a formal conjunction of geometric constructions admitted by the provers, which represent the hypothesis of the proposition

- *concl_conj* is a conclusion or a formal conjunction of conclusions admitted by the provers, which represents the conclusion of the proposition.

Example:

**Proposition**["Pappus", any[A, B, A1, B1, C, C1, P, Q, S],
point[A, B, A1, B1] ∧
pon[C, line[A, B]] ∧
pon[C1, line[A1, B1]] ∧
inter[P, line[A, B1], line[A1, B]] ∧
inter[Q, line[A, C1], line[A1, C]] ∧
inter[S, line[B, C1], line[B1, C]]
⇒ collinear[P, Q, S]]

- Further information is needed if we want to obtain the graph of a geometric configuration with a particular position of the free points. In this purpose we use the following construct:

**KnowledgeBase**[*label*, any[*variable-list*], *point-list*]

where

- *label* is the identifier of the knowledge base,

- *variable-list* enumerates all the points whose coordinates are given

- *point-list* is a list of pairs of identifiers and coordinates of points.

Example:

**KnowledgeBase**["Pappus−g", any[A, B, A1, B1, C, C1],
{{A, {10, 120}}, {B, {100, 150}}, {A1, {25, 60}},
{B1, {120, 50}}, {C, {260, 200}}, {C1, {250, 45}}}]

### *5.2.2 Supported Constructions and Conclusions*

We shall use the notion of angles in the sense of full-angles as defined in [13].

The following geometric elements are accepted in constructions, where $p_i$ are previously defined points:

- line[$p_1$,$p_2$] - line defined by points $p_1$,$p_2$
- pline[$p_1$,$p_2$,$p_3$] - line through $p_1$ parallel to line $p_2\,p_3$
- tline[$p_1$,$p_2$,$p_3$] - line through $p_1$ perpendicular to line $p_2\,p_3$
- aline[$p_1$,$p_2$,$p_3$,$p_4$,$p_5$]
  - line $l$ passing through point $p_1$ such that $\angle[p_1\,p_2,l] = \angle[p_3\,p_4,p_4\,p_5]$
- circle[$p_1$,$p_2$] - circle defined by points $p_1$, $p_2$ (having as center $p_1$)
- segment[$p_1$,$p_2$] - segment $p_1\ p_2$; it can be used only by methods that deal with inequalities

In the sequel we shall refer to these elements by $l$ or $l_i$.

The following geometric quantities are accepted in constructions and conclusions:

- seglengthsq[$p_1$,$p_2$] - $(\overline{p_1\ p_2})^2$
- segratio[$p_1$,$p_2$,$p_3$,$p_4$] - $\overline{p_1\ p_2}$ / $\overline{p_3\ p_4}$
- trarea[$p_1$,$p_2$,$p_3$] - oriented area of $\triangle p_1\,p_2\,p_3$
- qarea[$p_1$,$p_2$,$p_3$,$p_4$] - oriented area of quadrilateral $p_1\,p_2\,p_3\,p_4$
- tanof[$p_1$,$p_2$,$p_3$,$p_4$] - tan($\angle[p_3\,p_4,p_4\,p_5]$)

Expressions formed of these elements will be denoted by $e$.

The constructions treated are:

- point[$p_1$, …, $p_n$] - points $p_1$, …, $p_n$ are free points
- triangle[$p_1$,$p_2$,$p_3$] - same as point[$p_1$,$p_2$,$p_3$]
- pon[$p_1$,$l$] - point $p_1$ is on element $l$ (line or circle)
- midpoint[$p_1$,$p_2$,$p_3$] - point $p_1$ is midpoint of the segment $p_2\ p_3$
- divpoint[$p_1$,$p_2$,$p_3$,$r$] - point $p_1$ divides segment $p_2\ p_3$ such that $\overline{p_1\ p_2}$=r $\overline{p_1\ p_3}$
- inter[$p_1$,$l_1$,$l_2$] - point $p_1$ is an intersection point of elements $l_1$ and $l_2$ (line-line, line-circle or circle-circle)
- eqdist[$p_1$,$p_2$,$p_3$,$p_4$] - take point $p_1$ such that $\overline{p_1\ p_2}$=$\overline{p_3\ p_4}$
- foot[$p_1$,$p_2$,$p_3$,$p_4$] - point $p_1$ is the intersection point of line $p_3\ p_4$ and the perpendicular line through $p_2$ to line $p_3\ p_4$
- height[$p_1$,$p_2$,$p_3$,$p_4$,$p_5$] - take point $p_2$ such that $p_1\ p_2$ is height in $\triangle p_3\,p_4\,p_5$, $p_1 \in \{p_3,p_4,p_5\}$

- median[$p_1,p_2,p_3,p_4,p_5$] - take point $p_2$ such that $p_1\,p_2$ is median in $\triangle p_3\,p_4\,p_5$, $p_1 \in \{p_3,p_4,p_5\}$
- midperpendicular[$p_1,p_2,p_3,p_4$] - take points $p_1$ and $p_2$ such that $p_1\,p_2$ is perpendicular bisector of segment $p_3\,p_4$ (two constructions, $p_1$ is the midpoint of $p_3\,p_4$)
- bisector[$p_1,p_2,p_3,p_4,p_5$] - take point $p_2$ such that $p_1\,p_2$ is bisector in $\triangle p_3\,p_4\,p_5$, $p_1 \in \{p_3,p_4,p_5\}$ (this construction can be used only for the algebraic methods)
- bisect[$p_1,p_2,\{p_3,p_4,p_5\}$] -$p_1\,p_2$ is bisector in $\triangle p_3\,p_4\,p_5$, $p_1 \in \{p_3,p_4,p_5\}$ (this should be the first construction after free points have been constructed; the third vertex of a triangle is constructed using the bisector)
- incircle[$p_1,p_2,p_3,p_4,p_5$] - point $p_1$ is the center of the inscribed circle of $\triangle p_2\,p_3\,p_4$; $p_1\,p_5$ is the radius of this circle; $p_5 \in p_2\,p_3$ (this should be the first construction; this construction can be used only if the proving method used allows inequalities)
- innerpoint[$p_1,p_2,p_3,p_4$] - take point $p_1$ such that it is an interior point of $\triangle p_2\,p_3\,p_4$
- bisectint[$p_1,p_2,\{p_3,p_4,p_5\}$] -$p_1\,p_2$ is interior bisector in $\triangle p_3\,p_4\,p_5$, $p_1 \in \{p_3,p_4,p_5\}$(this should be the first construction after free points have been constructed; the third vertex of a triangle is constructed using the bisector; this construction can be used only if the proving method used allows inequalities)
- hypothesis [$e_1 \times e_2$] - condition given in the form of an inequality, where $\times \in \{\neq,<,\leq,>,\geq,=\}$(only for provers that admit inequalities)

The accepted conclusions are:

- collinear[$p_1,p_2,p_3$] - points $p_1,p_2,p_3$ are collinear
- parallel[$p_1,p_2,p_3,p_4$] - lines $p_1\,p_2$ and $p_3\,p_4$ are parallel
- perpendicular[$p_1,p_2,p_3,p_4$] - lines $p_1\,p_2$ and $p_3\,p_4$ are perpendicular
- distequal[$p_1,p_2,p_3,p_4$] - $\overline{p_1\,p_2}=\overline{p_3\,p_4}$
- distprodeq[$p_1,p_2,p_3,p_4,p_5,p_6,p_7,p_8$] - $\overline{p_1\,p_2}\cdot\overline{p_3\,p_4}=\overline{p_5\,p_6}\cdot\overline{p_7\,p_8}$
- cocircle[$p_1,p_2,p_3,p_4$] - points $p_1,p_2,p_3,p_4$ are on the same circle
- equalzero[$e$] - $e = 0$
- inequation[$e_1 \times e_2$] - $e_1 \times e_2$, where $\times \in \{\neq,<,\leq,>,\geq,\}$ (only for provers that admit inequalities)

Many constructions are complex ones and also define some auxiliary points that cannot be used in further constructions (as they are hidden from the user). Such a point would be the touching point of the circle to the edge of the triangle in *incircle*. We decided to let the user define this point explicitly because it is very often needed for further constructions. Thus construction *incircle* introduces two new points: a vertex of the triangle and this touching point.

### *5.2.3 Implementation*

The input of the geometric statement is done by the package *Geominput*. The functions of this package transform the problem from its external form, as presented in the above section, into an internal form recognized by all the provers stored in the global variable *geomobj*. It also calls the functions that prepare the graphic representation of the problem from the package *Geomgraphics*.

The main functions of this package are: *InitGeom, TransElements, Ppoint, Lline, Pline, Tline, Aline, Ccircle, Pon, MidPoint, Divpoint, Eqdist, Inter, Foot, Triangle, Height, Median, Midperpendicular, Bisect, Bisector, Segment, InCircle, InnerPoint, BisectInt, Hyp, Collinear, DistEqual, Parallel, Perpendicular, Cocircle, DistProdEq, EqualZero, SegRatio, SegLengthSq, TrArea ,QArea, TanOf, ComputeExpr, Inequation*.

Examples:

```
Lline[a_, b_] := (Grline[a, b]; {"l", a, b})

Ccircle[a_, b_] := (Grcircle[a, b]; {"c", a, b})

Pon[p_, {"l", a_, b_}] :=
  (GronLine[p, a, b]; geomobj = Append[geomobj, {p, "online", {a, b}}])

Pon[p_, {"c", a_, b_}] :=
  (GronCircle[p, a, b]; geomobj = Append[geomobj, {p, "oncircle", {a, b}}])

Inter[a_, l1_, l2_] := (Pon [a, l1];  Pon[a, l2])

MidPoint[a_, b_, c_] :=
  (Grmidpoint[a, b, c]; geomobj = Append[geomobj, {a, "divpoint", {b, c}, -1}])
```

The internal object obtained for the Pappus theorem presented above is:

```
{{A, point}, {B, point}, {A1, point},
 {B1, point}, {C, online, {A, B}},
 {C1, online, {A1, B1}}, {P, online, {A, B1}},
 {P, online, {A1, B}}, {Q, online, {A, C1}},
 {Q, online, {A1, C}}, {S, online, {B, C1}},
 {S, online, {B1, C}}, {, collinear, {P, Q, S}}}
```

### *5.2.4 GeomPalette*

To make the input of the problem and specifying the functions easier, a palette was defined. It contains all possible elements with their number of arguments specified.

========== Palette Notebook ==========

# ▽ Geometry Prover

## ▽ Problem Input

Copy the third line of each construct as many times as necessary!

**Proposition**["□", any[□],
□ ∧
□ ∧
□
⇒ □]

**Assumptions**["□", any[□],
□ ∧
□ ∧
□]

**KnowledgeBase**["□", any[□],
□ ∧
□ ∧
□]

**KnowledgeBase**["□", any[□],
{{□, {□, □}}, {□, {□, □}}}]

## ▽ Constructions and Conclusions

### ▽ *Constructions*

| | | |
|---|---|---|
| point[□] | pon[□, □] | triangle[□, □, □] |
| line[□, □] | inter[□, □, □] | height[□, □, □, □, □] |
| pline[□, □, □] | midpoint[□, □, □] | median[□, □, □, □, □] |
| tline[□, □, □] | divpoint[□, □, □, □] | midperpendicular[□, □, □, □] |
| aline[□, □, □, □, □] | foot[□, □, □, □] | bisect[□, □, □] |
| circle[□, □] | eqdist[□, □, □, □] | bisector[□, □, □, □, □] |

### ▽ *Geometric quantities*

| |
|---|
| seglengthsq[□, □] |
| segratio[□, □, □, □] |
| trarea[□, □, □] |
| qarea[□, □, □, □] |
| tanof[□, □, □, □] |

### ▽ *Conclusions*

| |
|---|
| collinear[□, □, □] |
| parallel[□, □, □, □] |
| perpendicular[□, □, □, □] |
| distequal[□, □, □, □] |

cocircle[□, □, □, □]
distprodeq[□, □, □, □, □, □, □, □]
equalzero[□]

▽ *Constructions and conclusions involving inequalities*

segment[□, □]
incircle[□, □, □, □, □]
innerpoint[□, □, □, □]
bisectint[□, □, □]
hypothesis[□, □]
hypothesis[□]
inequation[□]

▽ **Simplify**

▽ *Graphic*

Simplify[Proposition["□"], by −> GraphicSimplifier]

Simplify[Proposition["□"],
by −> GraphicSimplifier, using → KnowledgeBase["□"]]

Simplify[KnowledgeBase["□"], by −> GraphicSimplifier]

Simplify[KnowledgeBase["□"],
by −> GraphicSimplifier, using → KnowledgeBase["□"]]

possible option:

SimplifierOptions → {Verify → False}

▽ *Algebraic*

Simplify[Proposition["□"], by → AlgebraicSimplifier]

possible option:

SimplifierOptions → {MyAxis → False}

▽ *Area*

Simplify[Proposition["□"], by → AreaSimplifier]

▽ **Prove**

▽ *Area*

```
Prove[Proposition["□"], by → GeometryProver,
  ProverOptions → {Method → "Area"}]
```

Prover options:

```
VerifyNdg → False   OrtCoordSyst → True
ElimFactors → False   DetailedPrint → False
```

▽ *Groebner*

```
Prove[Proposition["□"], by → GeometryProver,
  ProverOptions → {Method → "GroebnerProver"}]
```

Prover options:

```
MyAxis → False   Refutation → True   ReverseVars → False
```

▽ *Wu*

```
Prove[Proposition["□"], by → GeometryProver,
  ProverOptions → {Method → "Wu"}]
```

Prover options:

```
MyAxis → False   Iterate → False   ElimFact → False
```

▽ *AreaCAD*

```
Prove[Proposition["□"], by → GeometryProver,
  ProverOptions → {Method → "AreaCAD"}]
```

## ▽ Compute

```
Compute[□ , by −> AreaCompute, using −> KnowledgeBase["□"]]
```

Prover options:

```
EvaluatorOptions → {VerifyNdg → False, ShowSteps → False}
```

## ▽ Explore

```
Explore[Assumptions["□"], by → AreaExplore, producing → □];
```

```
Explore[Assumptions["□"], by → AreaExplore, using → □, producing → □];
```

```
Explore[□, by → AreaExplore, producing → □];
```

```
Explore[□, by → AreaExplore, using → □, producing → □];
```

Explorer options:

```
ExplorerOptions →
  {GraphicPoints → KnowledgeBase["□"], RelatedPoints → {□}}
```

========== End of Palette Notebook ==========

## *5.3 Graphical Interface*

### *5.3.1 Graphical Representation of Geometric Configurations*

In order to understand better a geometry problem or to discover some mistakes resulted from the improper input of the problem we need a graphical representation of the given geometric configuration. In this purpose we use the *Theorema* function *Simplify* with option *by→GraphicSimplifier*. The general form of the function call is:

```
Simplify[object, by→GraphicSimplifier, using→kb,
SimplifierOptions→{opts}]
```

where

–*object* is a *Propsition* or *KnowledgeBase* object representing the geometry configuration;

–*kb* specifies the coordinates of the free points;

–*opts* are options for the simplifier (*Verify* specifies whether a numerical check is wanted).

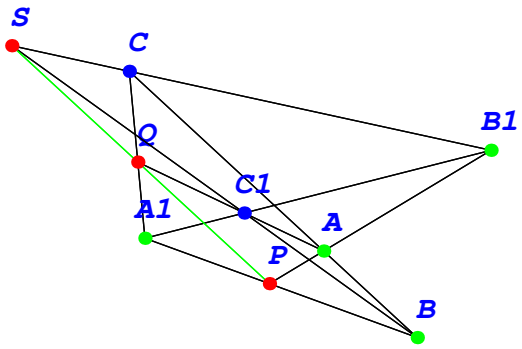
If the *using* option is omitted random values for the free coordinates are generated.

In the original screen shots there are three colors for representing points: green for free points, blue for semi-bound points and red for bound points. Lines that are involved in the conclusion are green. In printout points appear in three tones: black for bound points, dark grey for semi-bound points and light grey for the free ones. The grey lines are involved in the conclusion.

Example 1. Pappus theorem with random coordinates

```
Proposition["Pappus", any[A, B, A1, B1, C, C1, P, Q, S],
  point[A, B, A1, B1] ∧ pon[C, line[A, B]] ∧ pon[C1, line[A1, B1]] ∧
      inter[P, line[A, B1], line[A1, B]] ∧ inter[Q, line[A, C1], line[A1, C]] ∧
      inter[S, line[B, C1], line[B1, C]] ⇒ collinear[P, Q, S]]

Simplify[Proposition["Pappus"],
  by → GraphicSimplifier, SimplifierOptions → {Verify → False}]
```

========== Output Generated by the Program ==========

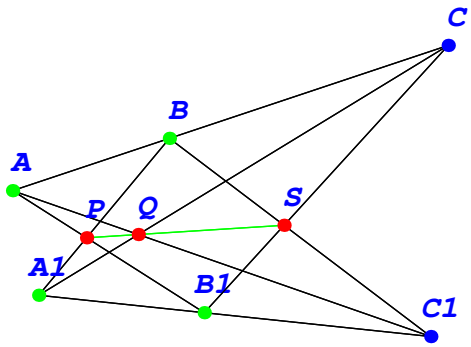



========== End of Generated Output ==========

Example 2. Pappus theorem with coordinates of free points specified by the user:

```
KnowledgeBase["Pappus_g", any[A, B, A1, B1, C, C1],
  {{A, {10, 120}}, {B, {100, 150}}, {A1, {25, 60}},
    {B1, {120, 50}}, {C, {260, 200}}, {C1, {250, 45}}}]

Simplify[Proposition["Pappus"], by → GraphicSimplifier,
  using → KnowledgeBase["Pappus_g"], SimplifierOptions → {Verify → False}]
```

========== Output Generated by the Program ==========




========== End of Generated Output ==========

If the given geometric configuration consists of more than one essentially different configurations (because of the segment orientation on perpendicular or parallel lines) all these configurations are presented.

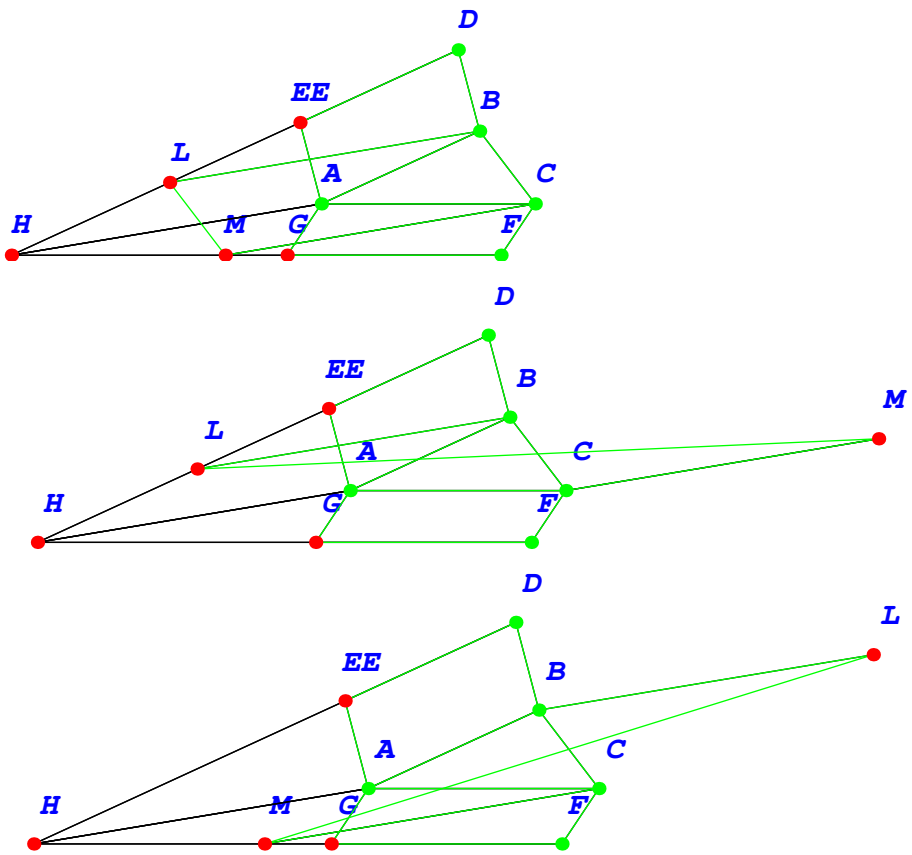
Example 3. Proposition with more than one variants

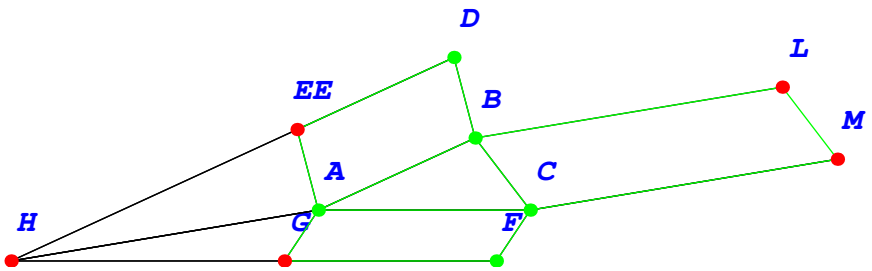
**Proposition**["Chou22", any[A, B, C, D, EE, F, G, H, L, M],
  triangle[A, B, C] ∧
    point[D, F] ∧
    inter[EE, pline[A, D, B], pline[D, B, A]] ∧
    inter[G, pline[A, F, C], pline[F, C, A]] ∧
    inter[H, line[F, G], line[D, EE]] ∧
    pon[L, pline[B, H, A]] ∧
    eqdist[B, L, A, H] ∧ pon[M, pline[C, H, A]] ∧
    eqdist[C, M, A, H]
   ⇒ equalzero[
      qarea[B, C, M, L]^2 − (qarea[A, B, D, EE] + qarea[C, A, G, F])^2]]

**KnowledgeBase**["C22", any[A, B, D, F, C],
  {{A, {10, 20}}, {B, {195, 105}}, {D, {170, 200}}, {F, {220, −40}}, {C, {260, 20}}}]

Simplify[Proposition["Chou22"], by → GraphicSimplifier,
  using → KnowledgeBase["C22"], SimplifierOptions → {Verify → False}]

========== Output Generated by the Program ==========





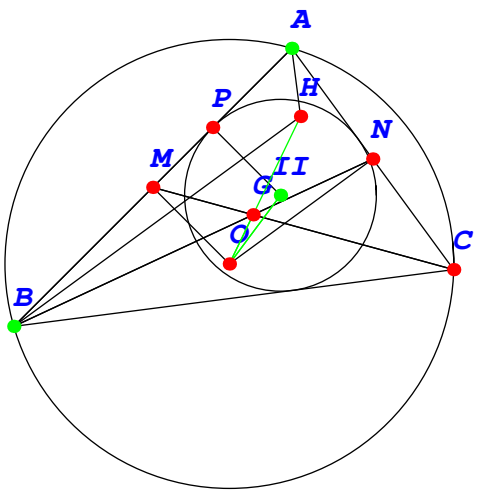

### *5.3.2 Visualization of Conditions Described by Inequalities*

The graphic representation of geometric configurations whose definitions contain inequalities is also possible. So the notions of a point on a segment, the center of the inscribed circle, inner bisector, etc. are treated.

Example 4:

```
Proposition["Cad_points", any[II, A, B, C, P, H, M, N, O, G],
  point[A, B, II] ∧
    incircle[II, A, B, C, P] ∧
    inter[H, tline[A, B, C], tline[B, A, C]] ∧
    median[C, M, A, B, C] ∧
    median[B, N, A, B, C] ∧
    inter[O, tline[M, A, B], tline[N, A, C]] ∧
    circle[O, A] ∧
    inter[G, line[C, M], line[B, N]]
   ⇒ inequation[seglengthsq[O, H] − seglengthsq[O, II] >= 0] ∧
       inequation[seglengthsq[O, II] − seglengthsq[O, G] >= 0]]

KnowledgeBase["Cad_points_g", any[A, B, II],
  {{A, {0, 0}}, {B, {−123, −123}}, {II, {−5, −65}}}]

Simplify[Proposition["Cad_points"],
  by → GraphicSimplifier, using → KnowledgeBase["Cad_points_g"],
  SimplifierOptions → {Verify → False}]
```

========== Output Generated by the Program ==========



### *5.3.3 Checking Conjectures by Examples*

When trying to prove a theorem using a computer it is of real help if one can select the false statements without using a formal proving method. This can be done by checking conjectures by examples. Numerically checking the validity of a statement for sufficiently many different configurations of the graph associated with the problem can state whether the statement is false in the general case, it is false for some configurations and true for others, or it seems to be generically true.

Trying different configurations by introducing the coordinates of the points is very tedious. The solution is having an interactive graph of the problem. This way one can change the graph associated with the statement very simply, just using the mouse to specify the new coordinates of a point. Unfortunately *Mathematica* 4.0 does not permit changing a figure dragging the mouse, so no continuous movement is possible for the moment.

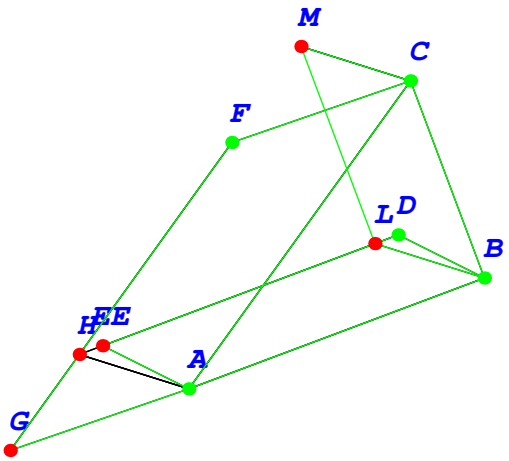
Thus we can try out different point configurations without having to introduce the coordinates only by letting the system chose random coordinates for the free points. This is not a systematic check of possible essentially different configurations, but for a great enough number of different configurations one can conclude, that the conjecture is probably generically true or not. And what is most important, the errors resulting from the input of the problem can be very easily detected, not trying to prove a theorem that cannot be true.

The option *Verify→True* (implicit value for this option) specifies that we want to check the validity of the proposition.

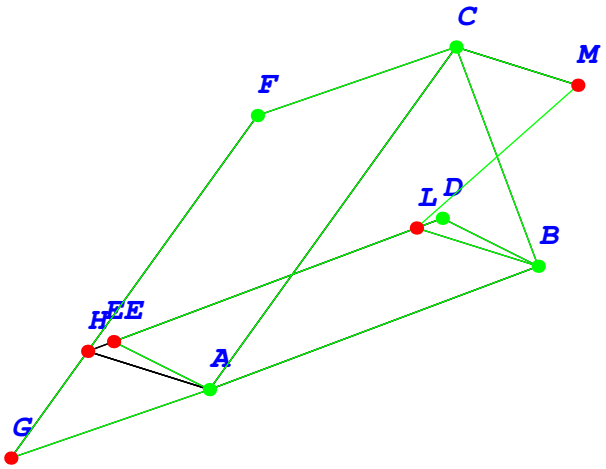
Example 5. Proposition with more than one variants, the proposition is generically true for 2 variants. The proposition is the same as in example 3 of section 5.3.1 with different position of the free points.

```
Proposition["Chou22", any[A, B, C, D, EE, F, G, H, L, M],
  triangle[A, B, C] ∧
    point[D, F] ∧
    inter[EE, pline[A, D, B], pline[D, B, A]] ∧
    inter[G, pline[A, F, C], pline[F, C, A]] ∧
    inter[H, line[F, G], line[D, EE]] ∧
    pon[L, pline[B, H, A]] ∧
    eqdist[B, L, A, H] ∧
    pon[M, pline[C, H, A]] ∧
    eqdist[C, M, A, H]
   ⇒ equalzero[
       qarea[B, C, M, L]^2 − (qarea[A, B, D, EE] + qarea[C, A, G, F])^2]]

KnowledgeBase["C22", any[A, B, D, F, C],
  {{A, {20, 10}}, {B, {260, 100}}, {D, {190, 135}}, {F, {55, 210}}, {C, {200, 260}}}]

Simplify[Proposition["Chou22"], by → GraphicSimplifier,
  using → KnowledgeBase["C22"], SimplifierOptions → {Verify → True}]
```

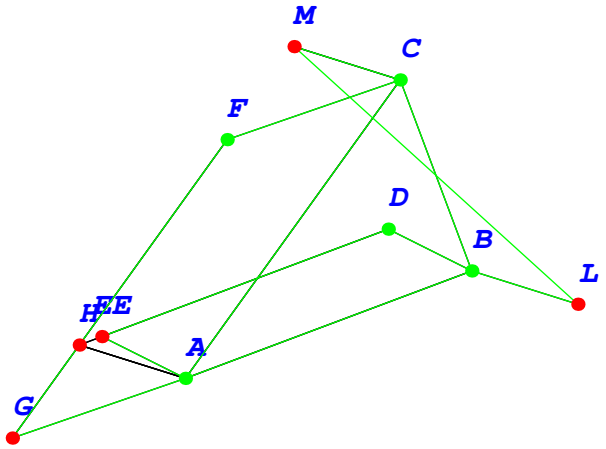
========== Output Generated by the Program ==========



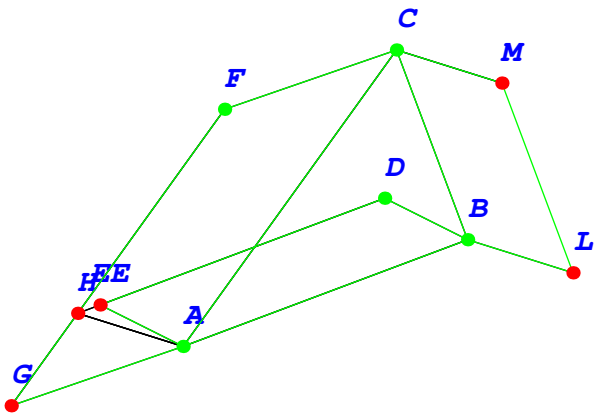

$(\text{Area}[B, C, M, L]^2 - (\text{Area}[A, B, D, EE] + \text{Area}[C, A, G, F])^2) == 0$ for this configuration of the points




Conclusion is not True: $-6.3001 \times 10^8 \neq 0$




Conclusion is not True: $-6.3001 \times 10^8 \neq 0$




(Area[*B*, *C*, *M*, *L*]$^2$ − (Area[*A*, *B*, *D*, *EE*] + Area[*C*, *A*, *G*, *F*])$^2$) == 0 for this configuration of the points

============== End of Generated Output ==============

### *5.3.4 Implementation*

The main function *GraphicSimplifier* is implemented in the *GeomSimplifiers* package.

```
Options[GraphicSimplifier] = {Verify → True};

GraphicSimplifier[•prop[l_, ___, •flist[•lf["", constr_]]],
    •kb[•kb[___, •flist[•lf["", pointcoord___]]], ___], ___, opts___?OptionQ] :=
  Module[{cc = constr, i, pc = pointcoord, ver},
    {ver} = {Verify} /. {opts} /. Options[GraphicSimplifier];
    cc = ReplacePart[cc, List, Position[cc, ™And, Heads → True]];
    cc = Flatten[ReplacePart[cc, List, Position[cc, ™Implies, Heads → True]]];
    InitGeom; Last[cc //. TransElements[True]];
    Do[If[NumberQ[pc[[i, 2]]], Grratio[pc[[i, 1]], pc[[i, 2]]],
        NewCoord[pc[[i, 1]], {pc[[i, 2, 1]] /. ToMma, pc[[i, 2, 2]] /. ToMma}]],
      {i, Length[pc]}]; ShowGraph[ver]]

GraphicSimplifier[•prop[l_, ___, •flist[•lf["", constr_]]],
    ___, opts___?OptionQ] := Module[{cc = constr, ver},
    {ver} = {Verify} /. {opts} /. Options[GraphicSimplifier];
    cc = ReplacePart[cc, List, Position[cc, ™And, Heads → True]];
    cc = Flatten[ReplacePart[cc, List, Position[cc, ™Implies, Heads → True]]];
    InitGeom; Last[cc //. TransElements[True]]; ShowGraph[ver]]
```

It uses the function *ShowGraph* and through it some other functions gathered in the *GeomGraphics* package. These functions transform the given constructions in polynomial form, generate random coordinates for the free points if no coordinates are given, solve the resulted equation system and plot the graph of the geometric configuration.

As each point is defined by one or two equations and these equations contain only coordinates of already defined points the solving of the equation system can be speeded

up using an iterative procedure,solving successively one equation or systems of two equations with two unknowns.

```
ShowGraph[ver___] := Module[{i, j, ddy, ddx, l}, Grcoord[greq[[1]],
      {Random[Integer, {−200, 200}], Random[Integer, {−200, 200}]}];
    Do[l = Table[grcoordinates[[i, 2 j, 2]], {j, Length[grcoordinates[[i]]] / 2}];
      ddy = Max[l] − Min[l];
      l = Table[grcoordinates[[i, 2 j − 1, 2]], {j, Length[grcoordinates[[i]]] / 2}];
      ddx = Max[l] − Min[l]; grdx = Floor[ddx / 50]; grdy = Floor[ddx / 35];
      Show[Graphics[ReleaseHold[grobjectsh] /. grcoordinates[[i]],
          AspectRatio → Automatic, PlotRange → All,
          TextStyle → {FontFamily → "Courier", FontSize → 12}]];
      If[ver, Do[GrverifyCond[geomobj[[j]], i], {j, Length[geomobj]}]],
      {i, Length[grcoordinates]}]]

Grcoord[a_, coord_] :=
  Module[{i, j, grcoord, grhyp, hypvar, grusedpoints}, grcoord = grcoordinates;
    grusedpoints = {}; grhypset = grhypset /. False → True;
    grps1 = {}; If[MemberQ[greq, a], j = First[First[Position[greq, a]]], j = 1];
    Do[grhyp = grhypset[[i, 2]]; hypvar = grhypset[[i, 1]];
      If[Not[(ReleaseHold[grhyp /. Equal → TildeEqual] /. grcoordinates[[1]] /.
              b_ ≃ c_ → Round[b] == Round[c]) === True],
        If[a === hypvar || Not[GrcalcHyp[grhypset[[i]]]],
          Message[Grcoord::errhyp, a]; grcoordinates = grcoord;
          Return[]]], {i, Length[grhypset]}];
    For[i = j, i ≤ Length[greq], i++, grps1 = Append[grps1, grps[[i]]];
      grusedpoints = Union[grusedpoints, FormulaVariables[grps1]];
      grworkpoint = greq[[i]]; If[i == Length[greq] ||
          UnsameQ[greq[[i]], greq[[i + 1]]], changed = True; Grcalc[grps1]]]]

Grcalc[l_] := Module[{i, j, x1, y1, x2, y2, sol, sol1, s1, s2},
    grpointset = Union[grpointset]; grobjectsh = grobject; grpssh = l;
    For[i = 1, i ≤ Length[grpointset], i++, GrpointType[grpointset[[i]]]];
    If[changed,
      Switch[grworkpointtype,
        2, ElimCoord[grworkpoint];
        sol = Table[Grsolve[grcoordinates[[i]], {gx_grworkpoint, gy_grworkpoint}],
            {i, Length[grcoordinates]}];
        If[grtype_grworkpoint == 1 && glink_grworkpoint == 1 && Length[sol[[1]]] > 1,
          sol = Table[s1 = sol[[i, 1]]; s2 = sol[[i, 2]];
              If[(gxx_grworkpoint − s1[[1, 2]])^2 + (gyy_grworkpoint − s1[[2, 2]])^2 <
                  (gxx_grworkpoint − s2[[1, 2]])^2 + (gyy_grworkpoint − s2[[2, 2]])^2,
                {s1}, {s2}], {i, Length[grcoordinates]}]],
        1, x1 = {}; y2 = {};
        Do[AppendTo[x1, gx_grworkpoint → N[gx_grworkpoint /. grcoordinates[[i]]]];
          AppendTo[y2, gy_grworkpoint → N[gy_grworkpoint /. grcoordinates[[i]]]],
```

```
    {i, Length[grcoordinates]}];
  ElimCoord[grworkpoint];
  sol = Table[Grsolve[Append[grcoordinates[[i]], x1[[i]]], {gy_grworkpoint}],
      {i, Length[grcoordinates]}];
  sol1 = Table[
      Grsolve[Append[grcoordinates[[i]], y2[[i]]], {gx_grworkpoint}],
      {i, Length[grcoordinates]}];
  sol = Table[If[Flatten[sol[[i, j]]] == {}, y1 = ∞, y1 = N[sol[[i, j, 1, 2]]]];
      If[Flatten[sol1[[i, j]]] == {}, x2 = ∞, x2 = N[sol1[[i, j, 1, 2]]]];
      If[((gy_grworkpoint /. y2[[i]]) - y1)^2 < (x2 - (gx_grworkpoint /. x1[[i]]))^2,
        {N[x1[[i]]], gy_grworkpoint → N[y1]},
        {N[y2[[i]]], gx_grworkpoint → N[x2]}],
      {i, Length[grcoordinates]}, {j, Length[sol[[i]]]}];
  If[grworkpointtype > 0, grcoordinates = Table[Join[grcoordinates[[i]],
      N[sol[[i, j]]]], {i, Length[grcoordinates]}, {j, Length[sol[[i]]]}];
  If[grcoordinates ≠ {{}}, grcoordinates = Flatten[grcoordinates, 1]];
  ElimDegeneratedGraph; changed = False]]
```

When the constructions include conditions that can be expressed algebraically by inequalities, the *Mathematica* function *Experimental'CylindricalAlgebraicDecomposition* is used to obtain coordinates for the points that satisfy these inequalities.

```
GrsolveIneq[a_, grhyp_] :=
  Module[{cond, list}, cond = Experimental'CylindricalAlgebraicDecomposition[
      ReleaseHold[grhyp] /. grcoordinates[[1]], {gx_a, gy_a}];
    PrintDebug[5, cond]; If[cond === False, Message[GrsolveIneq::err, a]];
    list = GrvalIneq[cond, {}]]

GrvalIneq[cond_, l_] := Module[{l1},
  Switch[Head[cond],
    Or, GrvalIneq[cond[[Round[Length[cond] / 2]]], l],
    And, l1 = GrvalIneq[cond[[1]], l]; GrvalIneq[cond[[2]], l1],
    Inequality, Append[l, cond[[3]] ->
      Random[Integer, {Ceiling[cond[[1]] /. l], Floor[cond[[5]] /. l]}]],
    Greater | GreaterEqual, Append[l, cond[[1]] -> Random[Integer,
      {Ceiling[cond[[2]] /. l], Ceiling[cond[[2]] /. l] /. l + 100}]],
    Less | LessEqual, Append[l, cond[[1, 1]] ->
      Random[Integer, {Floor[cond[[2]] /. l] - 100, Floor[cond[[2]] /. l]}]]]]
```

## 5.4 Algebraic Methods

### 5.4.1 Formulation of the Problem

Consider a certain kind of geometry in the sense of Hilbert. Starting from the defining axiomatic system we may introduce a coordinate system over some number field, which will turn any geometric entities and relations into algebraic ones. In the case the geometry admits some axiom of infinity and some Pascalian axiom, then the number system is a commutative field of characteristic 0. The algebraic relations occurring in a geometric statement will then be polynomial equations, polynomial inequations and polynomial inequalities, with coefficients in the associated field. Thus the geometric statement can be translated into the form:

$$\text{HYP} \qquad H_1(x_1, \ldots, x_n) \times 0, \ldots, H_s(x_1, \ldots, x_n) \times 0$$

$$\text{CON} \qquad C(x_1, \ldots, x_n) \times 0$$

where × stands for any of $=$, $\neq$, $<$ and $\leq$, and

$x_1, \ldots, x_n$ range over some geometric entities, such as coordinates of points, areas of triangles, etc.

The set $\{x_1, \ldots, x_n\}$ consists usually of two distinct sets:

parameters: $\{u_1, \ldots, u_p\}$

geometric dependents: $\{y_1, \ldots, y_q\}$

where $p + q = n$.

The proof of a theorem amounts to finding some subsidiary conditions

$$D_1(x_1, \ldots, x_n) \times 0, \ldots, D_t(x_1, \ldots, x_n) \times 0,$$

which correspond to the non-degenerate conditions of the theorem and determining whether

$$\begin{aligned}(\forall y_1) \cdots (\forall y_q)\,(&H_1(u_1, \ldots, u_p, y_1, \ldots, y_q) \times 0 \wedge \cdots \wedge H_s(u_1, \ldots, u_p, y_1, \ldots, y_q) \times 0 \\ &\wedge D_1(u_1, \ldots, u_p, y_1, \ldots, y_q) \times 0 \wedge \cdots \wedge D_t(u_1, \ldots, u_p, y_1, \ldots, y_q) \times 0 \\ &\Rightarrow C(u_1, \ldots, u_p, y_1, \ldots, y_q) \times 0)\end{aligned}$$

is true over the field associated or its extension.

As presented in section 3.2, in the case of Wu's method and Gröbner basis method we shall consider only that class of geometric statements that do not involve order relation. Thus, in the algebraic form only polynomial equations appear. In this case the algebraic form of a geometric statement becomes:

$$\text{HYP} \qquad H_1(x_1, \ldots, x_n) = 0, \ldots, H_s(x_1, \ldots, x_n) = 0$$

$$\text{CON} \qquad C(x_1, \ldots, x_n) = 0.$$

This corresponds to the algebraic decision problem:

Given a finite system of polynomials $F_1, \ldots, F_s$ and any other non-zero polynomial $G$ with coefficients in some fixed field $\mathbb{K}$, decide whether $G = 0$ is a formal consequence of $F_1 = 0, \ldots, F_s = 0$ i.e., whether any common zero of $F_1, \ldots, F_s$ is a zero of $G$ (over some extension field of $\mathbb{K}$).

The question of subsidiary conditions (non-degenerate conditions) under which the stated propositions are true is important and heavily studied in geometric theorem proving. In this thesis we state all propositions in generic form. Wu's method, in principle, generates non−degenerate conditions automatically, but we do not interpret them geometrically in this thesis. The Gröbner basis method, to a certain extent, can also generate non−degenerate conditions, see [43]. However, since we use the built-in Gröbner basis function of *Mathematica*, we do not show these conditions in this implementation.

### *5.4.2 Algebraic Form - Algebraic Simplifier*

To be able to use the algebraic methods, the problem has to be transformed from its internal form obtained in *geomobj* into polynomial form. We have to separate the coordinates into free and bound variables (denoted by $u_i$ and $x_i$, respectively) and to find an appropriate coordinate system.

The transformation into algebraic form is done by some functions of the *Algform* package. This package works with several global variables that are modified as side-effects of the functions. The main function is *TransConstr* that associates to each construction one or two polynomials, creating *pshyp* and *psc*. In the same time it creates the data structures needed for determining which coordinates are free and which are bound and for finding a convenient coordinate system.

Example:

```
TransConstr[{a_, "online", {b_, c_}}] := (NoteqPoints[b, c];
    equations = Append[equations, {a, "l", {b, c}}];
    pshyp = Append[pshyp, Hold[(y_b − y_a) (x_c − x_a) − (y_c − y_a) (x_b − x_a)]])

TransConstr[{a_, "divpoint", {b_, c_}, r_}] :=
  (If[t_a < 2, t_a ++, Message[BoundPoint::err, a]; Throw[False]]; NoteqPoints[b,
      c]; equations = Join[equations, {{a, "l", {b, c}}, {a, "e", {b, c}}}]; pshyp =
      Join[pshyp, {Hold[(1 − r) x_a − x_b + r x_c], Hold[(1 − r) y_a − y_b + r y_c]}])

TransConstr[{a_, "oncircle", {b_, c_}}] :=
  Module[{cc, i}, NoteqPoints[b, c]; i = Addcircle[b, c, a];
    Eqcircle[a, b, circles[[i, 2, 1]]]]

TransConstr[{___, "collinear", {a_, b_, c_}}] :=
  (equations = Append[equations, {"", "C", {a, b, c}}];
    psc = Append[psc, Hold[(y_b − y_a) (x_c − x_a) − (y_c − y_a) (x_b − x_a)]])
```

Function *PreProcess* analyzes the given configuration of points, determines whether all the points are essentially used and eliminates superfluous points (which were constructed but not referenced) and equations that are not needed. It determines which coordinates are free and which are bound and finds an appropriate coordinate system and ordering of the variables. According to the value of the boolean parameter *find* the coordinate system will be chosen by the program trying an optimization (see section 6.1) or simply the first constructed point is considered to be the origin and the next is considered to lie on the *y*-axis.

```
PreProcess[find_, usedatabase_] :=
  (If[find && Union[xaxis, yaxis] == {}, FindAxisT];
    If[Not[usedbase], UselessPoints]; If[Union[xaxis, yaxis] == {},
      FindAxis[find]]; ElimInvalidEq; ps = Delete[ps, Position[ps, 0]];
    var = UniquenessCond; var = VarList[find, var])
```

Using the *Theorema* function *Simplify* with option *by→AlgebraicSimplifier* an algebraic representation of the problem with explanations can be obtained. This simplifier admits as option *MyAxis* with possible values *True* and *False* (giving the value of parameter *find* explained above) and returns the polynomials representing the geometric problem and the ordered list of variables.

Example:

```
Simplify[Proposition["Pappus"],
    by → AlgebraicSimplifier, SimplifierOptions → {MyAxis → False}];
```

========== Output Generated by the Program ==========

Algebraic Form:

To transform the geometric problem into algebraic form we have to chose first an orthogonal coordinate system.

Let's have the origin in point $A$, and points {} and $\{B, C\}$ on the two axes.

Using this coordinate system we have the following points:

$\{\{A, 0, 0\}, \{B, 0, u_1\}, \{A1, u_2, u_3\}, \{B1, u_4, u_5\},$
$\{C, 0, u_6\}, \{C1, u_7, x_1\}, \{P, x_2, x_3\}, \{Q, x_4, x_5\}, \{S, x_6, x_7\}\}$

The algebraic form of the given construction is:

(1)

$$\forall_{x_1, x_2, x_3, x_4, x_5, x_6, x_7} \left(u_3\, u_4 + -u_2\, u_5 + -u_3\, u_7 + u_5\, u_7 + u_2\, x_1 + -u_4\, x_1 == 0 \wedge u_5\, x_2 + -u_4\, x_3 == 0 \wedge -u_1\, u_2 + u_1\, x_2 + -u_3\, x_2 + u_2\, x_3 == 0 \wedge x_1\, x_4 + -u_7\, x_5 == 0 \wedge -u_2\, u_6 + -u_3\, x_4 + u_6\, x_4 + u_2\, x_5 == 0 \wedge u_1\, u_7 + -u_1\, x_6 + x_1\, x_6 + -u_7\, x_7 == 0 \wedge -u_4\, u_6 + -u_5\, x_6 + u_6\, x_6 + u_4\, x_7 == 0 \Rightarrow x_3\, x_4 + -x_2\, x_5 + -x_3\, x_6 + x_5\, x_6 + x_2\, x_7 + -x_4\, x_7 == 0\right)$$

========== End of Generated Output ==========

### *5.4.3 Wu's Method*

Wu's method, its theoretical background and some improvements are presented in detail in [44], [10], [40], and the literature cited there. We shall confine ourselves here to presenting our implementation.

Since in *Mathematica* there is no built−in implementation of the characteristic sets, as a first step this implementation had to be done. The main function for computing the characteristic set of a given polynomial set is *CharSetFact* of package *Algebraic*. This function has as input a set of polynomials *ps* and a boolean parameter that specifies if the incremental computation method should be used (the incremental method will be presented in detail in section 6.2) and outputs a characteristic set of *ps*. For convenience, package Algebraic contains some global variables that are used by many functions and sometimes modified as side-effect of some functions (such variables are: *var*, *fact*, *maxt*, etc.).

```
CharSetFact[ps_, False] :=
  Module[{qs = ps, rs = ps, cs, i, rs1}, fact = {}; While[rs ≠ {}, cs = BasicSet[qs];
      If[Contradictory[cs], rs = {},
        rs = Table[If[FreeQ[cs, qs[[i]]], PremPFact[qs[[i]], cs]], {i, Length[qs]}];
        rs = Delete[rs, Position[rs, Null]]; rs = Delete[rs, Position[rs, 0]];
        rs1 = Table[SimplifyConst[rs[[i]], var], {i, Length[rs]}];
        qs = Union[qs, rs1]]]; cs]
```

During the computation "constant" factors (containing only free variables) of the polynomials are eliminated and gathered in *fact* for a possible analysis.

Using this function, the proof is done by the *Theorema* function *Prove* using the *GeometryProver* with option *Method→"Wu"*. The options admitted are:

- *MyAxis*, with default value *True*, to ask whether a coordinate system should be chosen by the system or not,

- *Iterate*, with default value *True*, to ask whether the iterated version of characteristic set computation should be used or not.

The basic prover for Wu's method is:

```
Wu[•lf[¢WuCalc, ps_, CS_, var1_, •finfo["Remainder"]], dummy___] :=
  Block[{var = var1, fact = {}, maxt, val}, Module[{r, t},
      maxt = 0; t = Timing[r = PremPFact[Last[ps], CS]][[1]];
      Print[t, "  ", maxt]; If[r === 0, val = ¢Proved, val = ¢Failed];
      ProofStep[Prinfo[¢WuCalc,
          {} → •lf[UniqueLabel[], r, •finfo["Remainder"]], Last[ps], maxt, t],
        Psit[•lf[¢WuFinal, ps, CS, var, •finfo[val]], dummy] ]]]

Wu[•lf[¢WuCalc, ps_, var1_, n_, •finfo["Wu"]], dummy___] :=
  Block[{var = var1, fact, maxt, ini, index, psitList = {}, ps1, concl},
    Module[{CS, t}, maxt = 0;
      t = Timing[CS = CharSetFact[Drop[ps, −n], iterate]][[1]];
```

```
      Print[t, "  ", maxt]; ini = Table[Ini[CS[[i]]], {i, Length[CS]}];
      index = Table[IndexSet[CS[[i]]], {i, Length[CS]}];
      If[Contradictory[CS], FinalProofStep[Prinfo[¢WuFinal,
            {} → •lf[UniqueLabel[], CS, •finfo["CS"]]], ¢Failed ],
        If[n == 1,
          ProofStep[Prinfo[¢WuCalc, {} → •lf[UniqueLabel[], CS, •finfo["CS"]],
              fact, ini, index, maxt, t, iterate],
            Psit[•lf[¢WuCalc, ps, CS, var, •finfo["Remainder"]], dummy] ],
          ps1 = Drop[ps, −n]; Do[concl = ps[[(Length[ps] − i) /. ToMma]];
            AppendTo[psitList,
              ProofStep[Prinfo[¢WuCalc,
                  {} → •lf[UniqueLabel[], concl, •finfo["CS"]], (n − i) /. ToMma],
                Psit[•lf[¢WuCalc, Append[ps1, concl], CS, var,
                    •finfo["Remainder"]], dummy]]], {i, n − 1, 0, −1}];
          ProofStep @@ PrependTo[psitList, Prinfo[¢WuCalc,
              {} → •lf[UniqueLabel[], CS, •finfo["CS"]],
              fact, ini, index, maxt, t, iterate]]]]]]

Wu[•lf[l_, constr_, •finfo["Wu"]], kb_, dummy___] :=
  Module[{ps, xaxis, yaxis, points, var, i, j, ps1, n},
    {ps, xaxis, yaxis, points, var} = TranslateToAlgebraicForm[
        constr, find, False]; i = Length[geomobj]; n = 0;
    While[geomobj[[i, 1]] === "", n++; i−−];
    ps2 = Table[™Equal [ps[[j]],  0], {j, Length[ps] − n + 1, Length[ps]}];
    ps1 = Table[™Equal [ps[[i]],  0], {i, Length[ps] − n}];
    ProofStep[Prinfo[¢AlgebraicForm,
        {} → •lf[UniqueLabel[], ™ForAll[•range @@ Map[•simpleRange, var],
            True, ™Implies[™And @@ ps1, ™And @@ ps2]],
          •finfo[]], xaxis, yaxis, points],
      Psit[•lf[¢WuCalc, ps, var, n, •finfo["Wu"]], •asml[], dummy]]]

Wu[•lf[¢WuFinal, ps_, CS_, var1_, •finfo[¢Failed]], dummy___] :=
  FinalProofStep[Prinfo[¢WuFinal, {} → {}], ¢Failed ]

Wu[•lf[¢WuFinal, ps_, CS_, var1_, •finfo[¢Proved]], dummy___] :=
  FinalProofStep[Prinfo[¢WuFinal, {} → {}], ¢Proved ]
```

The prover produces a notebook with all the details of the proof.

Example:

```
Proposition["Pappus",  any[A, B, A1, B1, C, C1, P, Q, S],
  point[A, B, A1, B1] ∧
    pon[C, line[A, B]] ∧
    pon[C1, line[A1, B1]] ∧
    inter[P, line[A, B1], line[A1, B]] ∧
    inter[Q, line[A, C1], line[A1, C]] ∧
    inter[S, line[B, C1], line[B1, C]]
   ⇒ collinear[P, Q, S]]

Prove[Proposition["Pappus"],
  by → GeometryProver, ProverOptions → {Method → "Wu" }]
```

========== Output Generated by the Program ==========

Prove:

(Proposition (Pappus)
$$\forall_{A,B,A1,B1,C,C1,P,Q,S} (\text{point}[A, B, A1, B1] \wedge \text{pon}[C, \text{line}[A, B]] \wedge \text{pon}[C1, \text{line}[A1, B1]] \wedge$$
$$\text{inter}[P, \text{line}[A, B1], \text{line}[A1, B]] \wedge \text{inter}[Q, \text{line}[A, C1], \text{line}[A1, C]] \wedge$$
$$\text{inter}[S, \text{line}[B, C1], \text{line}[B1, C]] \Rightarrow \text{collinear}[P, Q, S])$$

with no assumptions.

To prove the above statement we shall use Wu's Method. First we have to transform the problem into algebraic form.

Algebraic Form:

To transform the geometric problem into algebraic form we have to chose first an orthogonal coordinate system.

Let's have the origin in point $A$ and points {} and $\{B, C\}$ on the two axes.

Using this coordinate system we have the following points:

$$\{\{A, 0, 0\}, \{B, 0, u_1\}, \{A1, u_2, u_3\}, \{B1, u_4, u_5\},$$
$$\{C, 0, u_6\}, \{C1, u_7, x_1\}, \{P, x_2, x_3\}, \{Q, x_4, x_5\}, \{S, x_6, x_7\}\}$$

The algebraic form of the given construction is:

(1)
$$\forall_{x_1,x_2,x_3,x_4,x_5,x_6,x_7} (u_3 u_4 + -u_2 u_5 + -u_3 u_7 + u_5 u_7 + u_2 x_1 + -u_4 x_1 == 0 \wedge$$
$$u_5 x_2 + -u_4 x_3 == 0 \wedge -u_1 u_2 + u_1 x_2 + -u_3 x_2 + u_2 x_3 == 0 \wedge$$
$$x_1 x_4 + -u_7 x_5 == 0 \wedge -u_2 u_6 + -u_3 x_4 + u_6 x_4 + u_2 x_5 == 0 \wedge$$
$$u_1 u_7 + -u_1 x_6 + x_1 x_6 + -u_7 x_7 == 0 \wedge$$
$$-u_4 u_6 + -u_5 x_6 + u_6 x_6 + u_4 x_7 == 0 \Rightarrow$$
$$x_3 x_4 + -x_2 x_5 + -x_3 x_6 + x_5 x_6 + x_2 x_7 + -x_4 x_7 == 0)$$

First we compute the characteristic set of the hypothesis polynomials. We use successive steps, introducing polynomials for each point in a separate step.

Characteristics of the characteristic set:

Time needed to compute the characteristic set: 2.17 Seconds.

During the computation the following factors were removed:
$$\{-1, u_2 + -u_4, u_4, u_5 + -u_6, u_7, u_1 u_2 + -u_1 u_4 + u_3 u_4 + -u_2 u_5 + -u_3 u_7 + u_5 u_7\}$$

The maximum number of monomials in a polynomial during the computation was: 56.

The set of initials of the polynomials in the characteristic set is:
$$\{u_2 + -u_4, u_1 u_4 + -u_3 u_4 + u_2 u_5, u_1 u_4 + -u_3 u_4 + u_2 u_5,$$
$$-u_2 u_3 u_4 + u_2^2 u_5 + u_3 u_4 u_7 + -u_2 u_5 u_7 + u_2 u_6 u_7 + -u_4 u_6 u_7,$$
$$-u_2 u_3 u_4 + u_2^2 u_5 + u_3 u_4 u_7 + -u_2 u_5 u_7 + u_2 u_6 u_7 + -u_4 u_6 u_7,$$
$$u_1 u_2 u_4 + -u_1 u_4^2 + u_3 u_4^2 + -u_2 u_4 u_5 + -u_3 u_4 u_7 +$$
$$u_2 u_5 u_7 + -u_2 u_6 u_7 + u_4 u_6 u_7, u_1 u_2 u_4 + -u_1 u_4^2 + u_3 u_4^2 +$$
$$-u_2 u_4 u_5 + -u_3 u_4 u_7 + u_2 u_5 u_7 + -u_2 u_6 u_7 + u_4 u_6 u_7\}$$

The index set of the polynomials in the characteristic set is:
{{6, 1, 1}, {4, 2, 1}, {4, 3, 1}, {8, 4, 1}, {10, 5, 1}, {12, 6, 1}, {18, 7, 1}}

The characteristic set is:

(2)

$$\{u_3\,u_4 + -u_2\,u_5 + -u_3\,u_7 + u_5\,u_7 + u_2\,x_1 + -u_4\,x_1,$$
$$-u_1\,u_2\,u_4 + u_1\,u_4\,x_2 + -u_3\,u_4\,x_2 + u_2\,u_5\,x_2,$$
$$-u_1\,u_2\,u_5 + u_1\,u_4\,x_3 + -u_3\,u_4\,x_3 + u_2\,u_5\,x_3,$$
$$-{u_2}^2\,u_6\,u_7 + u_2\,u_4\,u_6\,u_7 + -u_2\,u_3\,u_4\,x_4 + {u_2}^2\,u_5\,x_4 +$$
$$u_3\,u_4\,u_7\,x_4 + -u_2\,u_5\,u_7\,x_4 + u_2\,u_6\,u_7\,x_4 + -u_4\,u_6\,u_7\,x_4,$$
$$u_2\,u_3\,u_4\,u_6 + -{u_2}^2\,u_5\,u_6 + -u_2\,u_3\,u_6\,u_7 + u_2\,u_5\,u_6\,u_7 + -u_2\,u_3\,u_4\,x_5 +$$
$${u_2}^2\,u_5\,x_5 + u_3\,u_4\,u_7\,x_5 + -u_2\,u_5\,u_7\,x_5 + u_2\,u_6\,u_7\,x_5 + -u_4\,u_6\,u_7\,x_5,$$
$$-u_1\,u_2\,u_4\,u_7 + u_1\,{u_4}^2\,u_7 + u_2\,u_4\,u_6\,u_7 + -{u_4}^2\,u_6\,u_7 + u_1\,u_2\,u_4\,x_6 +$$
$$-u_1\,{u_4}^2\,x_6 + u_3\,{u_4}^2\,x_6 + -u_2\,u_4\,u_5\,x_6 + -u_3\,u_4\,u_7\,x_6 + u_2\,u_5\,u_7\,x_6 +$$
$$-u_2\,u_6\,u_7\,x_6 + u_4\,u_6\,u_7\,x_6,\ -u_1\,u_2\,u_4\,u_6 + u_1\,{u_4}^2\,u_6 + -u_3\,{u_4}^2\,u_6 +$$
$$u_2\,u_4\,u_5\,u_6 + -u_1\,u_2\,u_5\,u_7 + u_1\,u_4\,u_5\,u_7 + u_1\,u_2\,u_6\,u_7 + -u_1\,u_4\,u_6\,u_7 +$$
$$u_3\,u_4\,u_6\,u_7 + -u_4\,u_5\,u_6\,u_7 + u_1\,u_2\,u_4\,x_7 + -u_1\,{u_4}^2\,x_7 + u_3\,{u_4}^2\,x_7 +$$
$$-u_2\,u_4\,u_5\,x_7 + -u_3\,u_4\,u_7\,x_7 + u_2\,u_5\,u_7\,x_7 + -u_2\,u_6\,u_7\,x_7 + u_4\,u_6\,u_7\,x_7\}$$

Next we compute the pseudoremainder of the conclusion polynomial $x_3\,x_4 + -x_2\,x_5 + -x_3\,x_6 + x_5\,x_6 + x_2\,x_7 + -x_4\,x_7$ with respect to the obtained characteristic set.

The remainder is: 0

Time needed to compute the remainder: 1.01 Seconds.

The maximum number of monomials in a polynomial during the computation was: 166.

As the obtained remainder is 0 the statement is generically true.

□

========== End of Generated Output ==========

### *5.4.4 Gröbner Bases Method*

An other algebraic method used in geometry theorem proving is the Gröbner bases method, as presented in [28] and [25] (for a theoretical presentation of Gröbner bases see [4]). We shall confine ourselves here to presenting our implementation.

As *Mathematica* contains a very good implementation of Gröbner bases, we use this function in our implementation.

Using this function the proof is done by the *Theorema* function *Prove* using the *GeometryProver* with option *Method*→"*GroebnerProver*". The admitted options are:

- *MyAxis*, with default value *True*, to ask whether a coordinate system should be chosen by the system or not,

- *Refutation*, with default value *False*, to ask whether the refutation method should be used or not,

- *ReverseVars*, with default value *True*, to ask whether reversed variable ordering should be applied or not (that is, the lowest rank variable should be the *y* coordinate of the last constructed point).

The basic prover for the Gröbner bases method is:

```
Groebner[•lf[¢GroebnerCalc, ps_, var1_, •finfo[False]], dummy___] :=
  Module[{GB, t, t1, val, p, var},
    If[reverse, var = Reverse[var1], var = var1];
    t = Timing[GB = GroebnerBasis[Drop[ps, −1],
              var, MonomialOrder → DegreeReverseLexicographic,
              CoefficientDomain → RationalFunctions]][[1]];
    Print[t, "   ", Length[GB]];
    t1 = Timing[p = System`Simplify[PolynomialReduce[Last[ps], GB,
                  var, MonomialOrder → DegreeReverseLexicographic,
                  CoefficientDomain → RationalFunctions][[2]]];
         If[p === 0, val = ¢Proved, val = ¢Failed]][[1]]; Print[t1];
    FinalProofStep[Prinfo[¢GroebnerFinal,
      {} → •lf[¢GroebnerFinal, {GB, p}, finfo], t, t1], val ]]

Groebner[•lf[¢GroebnerCalc, ps_, var_, •finfo[True]], dummy___] :=
  Module[{ps1}, ps1 = Table[™Equal [ps[[i]], 0], {i, Length[ps] − 1}];
    AppendTo[ps1, ™Not[™Equal[Last[ps], 0]]];
    ProofStep[Prinfo[¢GroebnerCalc, {} →
        •lf[UniqueLabel[], ™Not[™Exists[•range @@ Map[•simpleRange, var],
            True, ™And @@ ps1]], •finfo[™Not]]],
      Psit[•lf[¢GroebnerCalc, ps, var, •finfo[True, "Rabin"]], dummy]]]

Groebner[•lf[¢GroebnerCalc, ps_, var_, •finfo[True, "Rabin"]], dummy___] :=
  Module[{ps1}, ps1 = ReplacePart[ps, 1 − •var[v0] Last[ps], −1];
    ps1 = Table[™Equal [ps1[[i]], 0], {i, Length[ps1]}];
    ProofStep[Prinfo[¢GroebnerCalc, {} → •lf[UniqueLabel[], ™Not[
          ™Exists[•range @@ Map[•simpleRange, Append[var, •var[v0]]],
            True, ™And @@ ps1]], •finfo["Rabin"]]],
      Psit[•lf[¢GroebnerCalc, ps, var, •finfo[True, ™Not]], dummy]]]

Groebner[•lf[¢GroebnerCalc, ps_, var1_, •finfo[True, ™Not]], dummy___] :=
  Module[{GB, t, val, var},
    If[reverse, var = Prepend[Reverse[var1], •var[v0]],
      var = Append[var1, •var[v0]]];
    t = Timing[GB = GroebnerBasis[ReplacePart[ps, 1 − •var[v0] Last[ps], −1],
              var, MonomialOrder → DegreeReverseLexicographic,
              CoefficientDomain → RationalFunctions]][[1]];
    Print[t, "   ", Length[GB]]; If[MemberQ[GB, 1], val = ¢Proved,
      val = ¢Failed];
    FinalProofStep[Prinfo[¢GroebnerFinal,
      {} → •lf[UniqueLabel[], GB, finfo], t], val ]]

Groebner[•lf[l_, constr_, •finfo["GroebnerProver"]], dummy___] :=
  Module[{ps, xaxis, yaxis, points, var, i, ps1}, {ps, xaxis, yaxis, points, var} =
      TranslateToAlgebraicForm[constr, find, False];
    ps1 = Table[™Equal [ps[[i]], 0], {i, Length[ps] − 1}];
```

```
ProofStep[Prinfo[¢AlgebraicForm,
    {} → •lf[UniqueLabel[], ™ForAll[•range @@ Map[•simpleRange, var],
          True, ™Implies[™And @@ ps1, ™Equal[Last[ps], 0]]],
        •finfo[]], xaxis, yaxis, points],
  Psit[•lf[¢GroebnerCalc, ps, var, •finfo[ref]], dummy]]]
```

The prover produces a notebook with all the details of the proof.

Example 1:

```
Proposition["Pappus",  any[A, B, A1, B1, C, C1, P, Q, S],
  point[A, B, A1, B1] ∧
    pon[C, line[A, B]] ∧
    pon[C1, line[A1, B1]] ∧
    inter[P, line[A, B1], line[A1, B]] ∧
    inter[Q, line[A, C1], line[A1, C]] ∧
    inter[S, line[B, C1], line[B1, C]]
   ⇒ collinear[P, Q, S]]

Prove[Proposition["Pappus"], by → GeometryProver,
  ProverOptions → {Method → "GroebnerProver"}]
```

========== Output Generated by the Program ==========

Prove:

(Proposition (Pappus))

$$\forall_{A,B,A1,B1,C,C1,P,Q,S} (\text{point}[A, B, A1, B1] \wedge \text{pon}[C, \text{line}[A, B]] \wedge \text{pon}[C1, \text{line}[A1, B1]] \wedge \text{inter}[P, \text{line}[A, B1], \text{line}[A1, B]] \wedge \text{inter}[Q, \text{line}[A, C1], \text{line}[A1, C]] \wedge \text{inter}[S, \text{line}[B, C1], \text{line}[B1, C]] \Rightarrow \text{collinear}[P, Q, S])$$

with no assumptions.

To prove the above statement we shall use the Gröbner basis method. First we have to transform the problem into algebraic form.

Algebraic Form:

To transform the geometric problem in algebraic form, we have to chose first an orthogonal coordinate system.

Let's have the origin in point $A$, and points {} and {$B$, $C$} on the two axes.

Using this coordinate system we have the following points:

{{$A$, 0, 0}, {$B$, 0, $u_1$}, {$A1$, $u_2$, $u_3$}, {$B1$, $u_4$, $u_5$},
{$C$, 0, $u_6$}, {$C1$, $u_7$, $x_1$}, {$P$, $x_2$, $x_3$}, {$Q$, $x_4$, $x_5$}, {$S$, $x_6$, $x_7$}}

The algebraic form of the given construction is:

=== closed cell ===

We have to compute the Gröbner basis of the hypothesis polynomials

The polynomials of the Gröbner basis are

$$\{u_3\,u_4 + -u_2\,u_5 + -u_3\,u_7 + u_5\,u_7 + (u_2 + -u_4)\,x_1,$$
$$-u_1\,u_2\,u_4 + (u_1\,u_4 + -u_3\,u_4 + u_2\,u_5)\,x_2,$$
$$-u_1\,u_2\,u_5 + (u_1\,u_4 + -u_3\,u_4 + u_2\,u_5)\,x_3,\ -{u_2}^2\,u_6\,u_7 + u_2\,u_4\,u_6\,u_7 +$$
$$(-u_2\,u_3\,u_4 + {u_2}^2\,u_5 + u_3\,u_4\,u_7 + -u_2\,u_5\,u_7 + u_2\,u_6\,u_7 + -u_4\,u_6\,u_7)\,x_4,$$
$$u_2\,u_3\,u_4\,u_6 + -{u_2}^2\,u_5\,u_6 + -u_2\,u_3\,u_6\,u_7 + u_2\,u_5\,u_6\,u_7 +$$
$$(-u_2\,u_3\,u_4 + {u_2}^2\,u_5 + u_3\,u_4\,u_7 + -u_2\,u_5\,u_7 + u_2\,u_6\,u_7 + -u_4\,u_6\,u_7)\,x_5,$$
$$u_1\,u_2\,u_4\,u_7 + -u_1\,{u_4}^2\,u_7 + -u_2\,u_4\,u_6\,u_7 + {u_4}^2\,u_6\,u_7 + (-u_1\,u_2\,u_4 + u_1\,{u_4}^2 +$$
$$-u_3\,{u_4}^2 + u_2\,u_4\,u_5 + u_3\,u_4\,u_7 + -u_2\,u_5\,u_7 + u_2\,u_6\,u_7 + -u_4\,u_6\,u_7)\,x_6,$$
$$u_1\,u_2\,u_4\,u_6 + -u_1\,{u_4}^2\,u_6 + u_3\,{u_4}^2\,u_6 + -u_2\,u_4\,u_5\,u_6 + u_1\,u_2\,u_5\,u_7 +$$
$$-u_1\,u_4\,u_5\,u_7 + -u_1\,u_2\,u_6\,u_7 + u_1\,u_4\,u_6\,u_7 + -u_3\,u_4\,u_6\,u_7 +$$
$$u_4\,u_5\,u_6\,u_7 + (-u_1\,u_2\,u_4 + u_1\,{u_4}^2 + -u_3\,{u_4}^2 + u_2\,u_4\,u_5 +$$
$$u_3\,u_4\,u_7 + -u_2\,u_5\,u_7 + u_2\,u_6\,u_7 + -u_4\,u_6\,u_7)\,x_7\}$$

We reduce the conclusion polynomial with respect to the obtained GB.

The reduced polynomial is: 0.

As the obtained polynomial is 0, the statement is generically true.

Statistics:

The length of the Gröbner basis is 7.

Time needed to compute the Gröbner basis: 0.11 Seconds.

Time needed to reduce the conclusion polynomial with respect to the GB: 0.5 Seconds.

□

========== End of Generated Output ==========

Example 2 (the same problem with refutation):

```
Prove[Proposition["Pappus"], by → GeometryProver,
  ProverOptions → {Method → "GroebnerProver", Refutation → True}]
```

========== Output Generated by the Program ==========

Prove:

(Proposition (Pappus))

$$\forall_{A,B,A1,B1,C,C1,P,Q,S}\ (\text{point}[A, B, A1, B1] \wedge \text{pon}[C, \text{line}[A, B]] \wedge \text{pon}[C1, \text{line}[A1, B1]] \wedge$$
$$\text{inter}[P, \text{line}[A, B1], \text{line}[A1, B]] \wedge \text{inter}[Q, \text{line}[A, C1], \text{line}[A1, C]] \wedge$$
$$\text{inter}[S, \text{line}[B, C1], \text{line}[B1, C]] \Rightarrow \text{collinear}[P, Q, S])$$

with no assumptions.

To prove the above statement we shall use the Gröbner basis method. First we have to transform the problem into algebraic form.

Algebraic Form:

To transform the geometric problem into algebraic form, we have to chose first an orthogonal coordinate system.

Let's have the origin in point $A$, and points {} and {$B$, $C$} on the two axes.

Using this coordinate system we have the following points:

{{$A$, 0, 0}, {$B$, 0, $u_1$}, {$A1$, $u_2$, $u_3$}, {$B1$, $u_4$, $u_5$},
{$C$, 0, $u_6$}, {$C1$, $u_7$, $x_1$}, {$P$, $x_2$, $x_3$}, {$Q$, $x_4$, $x_5$}, {$S$, $x_6$, $x_7$}}

The algebraic form of the given construction is:

(1)

$$\forall_{x_1,x_2,x_3,x_4,x_5,x_6,x_7} (u_3\,u_4 + -u_2\,u_5 + -u_3\,u_7 + u_5\,u_7 + u_2\,x_1 + -u_4\,x_1 == 0 \wedge u_5\,x_2 + -u_4\,x_3 == 0 \wedge -u_1\,u_2 + u_1\,x_2 + -u_3\,x_2 + u_2\,x_3 == 0 \wedge x_1\,x_4 + -u_7\,x_5 == 0 \wedge -u_2\,u_6 + -u_3\,x_4 + u_6\,x_4 + u_2\,x_5 == 0 \wedge u_1\,u_7 + -u_1\,x_6 + x_1\,x_6 + -u_7\,x_7 == 0 \wedge -u_4\,u_6 + -u_5\,x_6 + u_6\,x_6 + u_4\,x_7 == 0 \Rightarrow x_3\,x_4 + -x_2\,x_5 + -x_3\,x_6 + x_5\,x_6 + x_2\,x_7 + -x_4\,x_7 == 0)$$

This problem is equivalent to

(2)

$$\neg \exists_{x_1,x_2,x_3,x_4,x_5,x_6,x_7} (u_3\,u_4 + -u_2\,u_5 + -u_3\,u_7 + u_5\,u_7 + u_2\,x_1 + -u_4\,x_1 == 0 \wedge u_5\,x_2 + -u_4\,x_3 == 0 \wedge -u_1\,u_2 + u_1\,x_2 + -u_3\,x_2 + u_2\,x_3 == 0 \wedge x_1\,x_4 + -u_7\,x_5 == 0 \wedge -u_2\,u_6 + -u_3\,x_4 + u_6\,x_4 + u_2\,x_5 == 0 \wedge u_1\,u_7 + -u_1\,x_6 + x_1\,x_6 + -u_7\,x_7 == 0 \wedge -u_4\,u_6 + -u_5\,x_6 + u_6\,x_6 + u_4\,x_7 == 0 \wedge x_3\,x_4 + -x_2\,x_5 + -x_3\,x_6 + x_5\,x_6 + x_2\,x_7 + -x_4\,x_7 \neq 0)$$

to remove the last inequality, we use the wellknown Rabinowitsch trick. Let $v_0$ a new variable. Then the problem becomes:

(3)

$$\neg \exists_{x_1,x_2,x_3,x_4,x_5,x_6,x_7,v_0} (u_3\,u_4 + -u_2\,u_5 + -u_3\,u_7 + u_5\,u_7 + u_2\,x_1 + -u_4\,x_1 == 0 \wedge u_5\,x_2 + -u_4\,x_3 == 0 \wedge -u_1\,u_2 + u_1\,x_2 + -u_3\,x_2 + u_2\,x_3 == 0 \wedge x_1\,x_4 + -u_7\,x_5 == 0 \wedge -u_2\,u_6 + -u_3\,x_4 + u_6\,x_4 + u_2\,x_5 == 0 \wedge u_1\,u_7 + -u_1\,x_6 + x_1\,x_6 + -u_7\,x_7 == 0 \wedge -u_4\,u_6 + -u_5\,x_6 + u_6\,x_6 + u_4\,x_7 == 0 \wedge 1 + -v_0\,(x_3\,x_4 + -x_2\,x_5 + -x_3\,x_6 + x_5\,x_6 + x_2\,x_7 + -x_4\,x_7) == 0)$$

To prove this statement, we have to compute the Gröbner bases of the above polynomials.

The polynomials of the Gröbner bases are {1}.

As the obtained Gröbner bases is {1}, the statement is generically true.

Statistics:

The length of the Gröbner bases is 1.

Time needed to compute the Gröbner bases: 0.34 Seconds.

□

========== End of Generated Output ==========

## *5.5 Coordinate-free Techniques - the Area Method*

### *5.5.1 Basic Notions*

The area method is a coordinate-free technique for proving geometry theorems based on point elimination. The basic geometry invariants used are the signed area and Pythagoras differences of oriented triangles and ratios of oriented segments.

The method can deal with geometric statements of constructive type, where each new point is introduced by one construction using only previously defined points. We presented the set of accepted constructions in section 3.2. The conclusion can be any geometric property that can be expressed by the help of the defined geometric quantities involving only the constructed points.

We use capital letters (or combination of capital letters and numbers) to denote points in the Euclidean plane.

We denote by $\bullet L_{\{A,B\}}$ the length of the oriented segment from $A$ to $B$ and by $\bullet S_{\{A,B,C\}}$ the signed area of the oriented triangle $ABC$. Thus $\bullet L_{\{A,B\}} = -\bullet L_{\{B,A\}}$ and

$$\bullet S_{\{A,B,C\}} = \bullet S_{\{B,C,A\}} = \bullet S_{\{C,A,B\}} = -\bullet S_{\{C,B,A\}} = -\bullet S_{\{B,A,C\}} = -\bullet S_{\{A,C,B\}}.$$

For an oriented quadrilateral $ABCD$, we define its area as $\bullet S_{\{A,B,C,D\}} = \bullet S_{\{A,B,C\}} + \bullet S_{\{A,C,D\}}$.

**Definition 4**

*In an oriented triangle ABC the Pythagorean difference $\bullet P_{\{A,B,C\}}$ is defined as*

$$\bullet P_{\{A,B,C\}} = \bullet L_{\{A,B\}}^2 + \bullet L_{\{C,B\}}^2 - \bullet L_{\{A,C\}}^2$$

It is easy to check that $\bullet P_{\{A,A,B\}} = 0$; $\bullet P_{\{A,B,C\}} = \bullet P_{\{C,B,A\}}$; $\bullet P_{\{A,B,C\}} = 2 \bullet L_{\{A,B\}}^2$. For an oriented quadrilateral ABCD we define

$$\bullet P_{\{A,B,C,D\}} = \bullet P_{\{A,B,D\}} - \bullet P_{\{C,B,D\}} = \bullet L_{\{A,B\}}^2 + \bullet L_{\{C,D\}}^2 - \bullet L_{\{B,C\}}^2 - \bullet L_{\{D,A\}}^2$$

**Definition 5**

*We shall understand by geometric quantities the ratio of the length of two oriented segments on one line or on two parallel lines (denoted by $\bullet R_{\{A,B,C,D\}}$), the signed area of an oriented triangle or a quadrilateral and the Pythagorean difference of an oriented triangle or a quadrilateral.*

In the proofs we use the elimination lemmas presented in [11].

### *5.5.2 Area Form - Area Simplifier*

To be able to use the algebraic methods, the problem has to be transformed from its internal form obtained in *geomobj* into o form that contains only the basic constructions accepted by the AreaProver. The conclusion has to be expressed in the form of a polynomial in the geometric quantities mentioned above.

The transformation into area form is done by some functions of the *Area* package. This package works with several global variables that are modified as side-effects of the functions. The main function is *Translate* which, by the help of *Translate*, *TransConstruction* and *TransConclusion* performs the necessary transformations and returns the list of constructions, the conclusion polynomial form and the list of the constructed points.

Examples of the functions used:

```
TransArea[hyp_] := Module[{geomobj}, InitGeom; InitArea;
    If[MemberQ[{™And, ™Implies, And, Implies}, Head[hyp]],
      geomobj = ReplacePart[hyp, List, 0] /. TransElements;
      While[UnsameQ[Head[Last[geomobj][[1]]], List],
        geomobj = Drop[geomobj, −1]]; geomobj = Last[geomobj],
      geomobj = hyp /. TransElements];
    Print["geomobj   ", geomobj];
    FindElements[geomobj];
    Translate[geomobj];
    constructions = System‘Simplify[constructions];
    expr = TransConclusion[Last[geomobj]] //. OrdPoints;
    {constructions, expr, variants}]

Translate[geomobj1_] :=
  Module[{i, go, go1, geomobj = geomobj1}, i = 1; If[Length[circles] > 0, i = 3;
      go = {"circle", Intersection[{points[[4]], points[[5]]}, circles[[1, 3]]], {}, {}};
      While[MemberQ[circles[[1, 4]], i],
        go[[2]] = Append[go[[2]], geomobj[[i, 1]]]; i++]; If[i < 4, i = 1,
        constructions = {go}; If[UnsameQ[points[[1]], circles[[1, 1]]],
          {points[[1]], points[[2]]} = {points[[2]], points[[1]]}]]];
    AppendTo[geomobj, {{}, {}}];
    While[i < Length[geomobj], go = geomobj[[i]];
      PrintDebug[5, go[[1]], go[[2]]];
      If[SameQ[go[[1]], geomobj[[i + 1, 1]]], i++; go1 = geomobj[[i]];
        TransConstruction[go, go1, i], TransConstruction[go]]; i++;]]

TransConstruction[{p_, "online", {a_, b_}}] :=
  (PrintDebug[5, {"online  ", p, "  ", a, b}]; AddPratio[p, a, a, b, r_p])

TransConstruction[{p_, "online", {a_, b_}}, {p_, "online", {c_, d_}}, i_] :=
  (PrintDebug[5, {"inter  ", p}]; AddInter[p, a, b, c, d])

TransConstruction[{p_, "online", {a_, b_}}, {p_, "onpline", {l___}}, i_] :=
  (PrintDebug[5, {"interp  ", p}]; AddInterP[p, a, b, l])
```

```
TransConstruction[{p_, "online", {a_, b_}}, {p_, "ontline", {c_, d_, e_}}, i_] :=
  (PrintDebug[5, {"intert  ", p}]; If[SameLine[{a, b}, {d, e}],
      AddFoot[p, c, d, e], InsertPoint[β_p]; lines = Append[lines, {p, c, β_p}];
      AddTline[β_p, c, d, e, r_i]; AddInter[p, a, b, c, β_p]])

TransConstruction[{p_, "online", {a_, b_}},
    {p_, "onaline", {c_, d_, e_, f_, g_}}, i_] :=
  (PrintDebug[5, {"intera  ", p}]; InsertPoint[β_p]; lines = Append[lines, {p, c, β_p}];
    AddAline[β_p, c, d, e, f, g]; AddInter[p, a, b, c, β_p])

TransConstruction[{p_, "online", {a_, b_}}, {p_, "oncircle", {o_, c_}}, i_] :=
  Module[{j, d}, PrintDebug[5, {"interc  ", p}];
    j = AreaFindCircle[o, c]; d = Intersection[circles[[j, 3]], {a, b}];
    If[Length[d] == 2, Message[TransConstruction::cir, a, b, o, c]];
    If[Length[d] == 1, AddInterC[p, a, b, o, d[[1]]], AddInterC2[p, a, b, o, c]]]

TransConstruction[{p_, "online", {a_, b_}}, {p_, "eqdist", {c_, d_, e_}}, i_] :=
  (PrintDebug[5, {"intere  ", p}];
    If[SameLine[{p, c}, {d, e}],
      AddPratio[p, c, d, e, 1]; AddVariants[
        {{Length[constructions], 2, 5}, 1}, {{Length[constructions], 2, 5}, -1}],
      Message[TransConstruction::eq]])

TransConclusion[concl_] :=
  Module[{co, i, l1, l2, expr}, co = ReleaseHold[concl];
    Switch[co[[2]],
      "collinear", expr = •S_co[[3]],
      "parallel", expr = •S_{co[[3,1]],co[[3,3]],co[[3,4]]} - •S_{co[[3,2]],co[[3,3]],co[[3,4]]},
      "disteq", (*l1={co[[3,1]],co[[3,2]]};l2={co[[3,3]],co[[3,4]]};
        i=ParallelNr[l1,l2];If[i≤Length[plines],
          expr=•R_{co[[3,1]],co[[3,2]],lines[[plines[[i,1]],1]],lines[[plines[[i,1]],2]]} -
              •R_{co[[3,3]],co[[3,4]],lines[[plines[[i,1]],1]],lines[[plines[[i,1]],2]]},
          If[SameLine[l1,l2],expr=Abs[•R_{co[[3,1]],co[[3,2]],co[[3,4]],co[[3,3]]}]-1,*)
        expr = •P_{co[[3,1]],co[[3,2]],co[[3,1]]} - •P_{co[[3,3]],co[[3,4]],co[[3,3]]}(*]]*),
      "perpendicular", expr = •P_{co[[3,1]],co[[3,3]],co[[3,4]]} - •P_{co[[3,2]],co[[3,3]],co[[3,4]]},
      "expr" | "compute", expr = co[[3]],
      "distprodeq", expr = •P_{co[[3,1]],co[[3,2]],co[[3,1]]} •P_{co[[3,3]],co[[3,4]],co[[3,3]]} -
          •P_{co[[3,5]],co[[3,6]],co[[3,5]]} •P_{co[[3,7]],co[[3,8]],co[[3,7]]}, "ineq", expr = 1,
      "", Null];
    expr /. ToMma]
```

Using the *Theorema* function *Simplify* with option *by→AreaSimplifier* the area representation of the problem can be obtained. The notebook also contains explanations.

Example:

```
Simplify[Proposition["Pappus"], by → AreaSimplifier];
```

========== Output Generated by the Program ==========

Using only constructions admitted by the area method we have to prove that for any points {*A*, *A1*, *B*, *B1*, *C*, *C1*, *P*, *Q*, *S*} constructions

{*A*, *B*, *A1*, *B1*} free points
*C A* ∥ *A B* and $\frac{\overline{A\,C}}{\overline{A\,B}}=r_C$ with ndg. {*A* ≠ *B*}
*C1 A1* ∥ *A1 B1* and $\frac{\overline{A1\,C1}}{\overline{A1\,B1}}=r_{C1}$ with ndg. {*A1* ≠ *B1*}
*P* = *A B1* ⋂ *A1 B* with ndg. {*A* ≠ *B1*, *A1* ≠ *B*, *A B1* ⋈ *A1 B*}
*Q* = *A C1* ⋂ *A1 C* with ndg. {*A* ≠ *C1*, *A1* ≠ *C*, *A C1* ⋈ *A1 C*}
*S* = *B C1* ⋂ *B1 C* with ndg. {*B* ≠ *C1*, *B1* ≠ *C*, *B C1* ⋈ *B1 C*}

imply that

points {*P*, *Q*, *S*} are collinear.

========== End of Generated Output ==========

### *5.5.3 Proving by the Area Method*

The area method is presented in detail in [11] and [13]. We shall confine ourselves here to presenting our implementation.

In the implementation we use the *Theorema* function *Prove* using the *GeometryProver* with option *Method*→"*Area*". First it transforms the problem into a form accepted by the area method as presented in section 5.2.3. In the last step of the proof area coordinates of the free points relative to an arbitrary coordinate system are introduced. According to option *OrtCoordSyst* we can take either an arbitrary orthogonal coordinate system, or consider the first three constructed points as defining a "skew" coordinate system. The admitted options are:

- *OrtCoordSyst*, with default value *True*, to ask whether an orthogonal coordinate system should be used in the final step or not,

- *ElimFactors*, with default value *True*, to ask whether nonzero factors should be eliminated during the proof steps or not,,

- *VerifyNdg*, with default value *True*, to ask whether the non−degenerate conditions should be verified or not.

- *DetailedPrint*, with default value *True*, to ask whether the intermediate expressions obtained by the elimination and ordering steps should be included in the proofobject and thus in the proof notebook or not.

The basic prover for the area method is:

```
Area[•lf[¢AreaElim, constructions_ ⇒ 0, •finfo["Proof"]], dummy___] :=
    FinalProofStep[
      Prinfo[¢AreaProved, {} → •lf[¢AreaProved, {}, •finfo["Proof"]]], ¢Proved ];
```

```
Area[•lf[¢AreaElim, constructions_ ⇒ 0, •finfo["Ndg"]], dummy___] :=
    FinalProofStep[
      Prinfo[¢AreaFailed, {} → •lf[¢AreaProved, {}, •finfo["Ndg"]]], ¢Failed ];

Area[•lf[¢AreaElim, constructions_ ⇒ expr_, finfo_], dummy___] /;
      constructions === {"point"} := Module[{ex1, ex2, ex = expr},
      ex = ex /. •L2{a__} :→ ElimCoordL2[•L2{a}];
      ex = System‘Simplify[ex];
      If[! ort,
        ex = ex /. {points[[4]] → αO, points[[5]] → αU, points[[6]] → αV}];
      ex = ex /. •L{a__}^k_ :→ ElimCoordL[•L{a}, k];
      ex = ex /. •S{a__}^k_ :→ ElimCoordS[•S{a}, k];
      If[ort, ex1 = ex, ex1 = ex //. OrdPoints];
      ex2 = System‘Simplify[ex1]; Print["Vege   ", ex2];
      ProofStep[Prinfo[¢AreaElim, {} → •lf[¢AreaElim, {}, finfo], ex, ex1, ex2],
        Psit[•lf[¢AreaFinish, expr ⇒ ex2, finfo], dummy]]];

Area[•lf[¢AreaElim, constructions_ ⇒ expr_, finfo_], dummy___] /;
      constructions === {"circle"} := Module[{ex1, ex2, ex = expr},
      ex = ex /. •L2{a__} :→ ElimCoordL2[•L2{a}];
      ex = System‘Simplify[ex];
      ex = ex /. •L{a__} :→ ElimCirL[•L{a}];
      If[! ort, ex = ex /. {points[[5]] → αO, points[[6]] → αU, points[[7]] → αV}];
      ex = ex /. •S{a__}^k_ :→ ElimCoordS[•S{a}, k];
      If[ort, ex1 = ex, ex1 = ex //. OrdPoints];
      ex2 = System‘Simplify[ex1];
      ProofStep[Prinfo[¢AreaElim, {} → •lf[¢AreaElim, {}, finfo], ex, ex1, ex2],
        Psit[•lf[¢AreaFinish, expr ⇒ ex2, finfo], dummy]]];

Area[•lf[¢AreaElim, constructions_ ⇒ expr_, finfo_], dummy___] :=
    Module[{cc, ll = Last[constructions], ex = expr, ex1, ex2},
      ComputeUYUV[ll];
      ex = ex /. ElimPoint[ll, ort];
      If[usel2, ex = ex /. •P{a__} :→ ElimP[•P{a}]];
      If [ll[[1]] === "point" && ! ort,
        ex = ex /. {αO → points[[4]], αU → points[[5]], αV → points[[6]]}];
      Print[ll[[2, 1]]];
      ex1 = ex //. OrdPoints;
      ex2 = System‘Simplify[ex1];
      If[optelim, ex2 = ElimNonZeroFact[ex2]];
      cc = Drop[constructions, −1];
      While[
        Length[cc] > 1 && FreeQ[ex2, Last[cc][[2, 1]], 20], cc = Drop[cc, −1]];
      Switch[ll[[1]], "point", cc = {"point"}, "circle", cc = {"circle"}];
      If[ex2 === 0, cc = {}];
      ProofStep[Prinfo[¢AreaElim, {} → •lf[¢AreaElim, {}, finfo], ll, ex, ex1, ex2],
        Psit[•lf[¢AreaElim, cc ⇒ ex2, finfo], dummy]]];
```

```
Area[•lf[¢AreaFinish, expr1_ ⇒ expr_, •finfo["Compute", concl_]],
      dummy___] :=
    Module[{ex}, If[Length[expr1] <= Length[expr], ex = expr1, ex = expr];
      FinalProofStep[Prinfo[¢AreaComputed,
          {} → •lf[¢AreaComputed, ex, •finfo["Compute", concl]]], ¢Proved ]];
Area[•lf[¢AreaFinish, _ ⇒ 0, •finfo["Proof"]], dummy___] :=
    Module[{}, FinalProofStep[Prinfo[¢AreaProved,
          {} → •lf[¢AreaProved, {}, •finfo["Proof"]]], ¢Proved ]];
Area[•lf[¢AreaFinish, _ ⇒ 0, •finfo["Ndg"]], dummy___] :=
    Module[{}, FinalProofStep[
        Prinfo[¢AreaFailed, {} → •lf[¢AreaProved, {}, •finfo["Ndg"]]], ¢Failed ]];
Area[•lf[¢AreaFinish, _ ⇒ _, •finfo["Proof"]], dummy___] :=
    Module[{}, FinalProofStep[Prinfo[¢AreaFailed,
          {} → •lf[¢AreaFailed, {}, •finfo["Proof"]]], ¢Proved ]];
Area[•lf[¢AreaFinish, _ ⇒ _, •finfo["Ndg"]], dummy___] :=
    Module[{}, FinalProofStep[
        Prinfo[¢AreaProved, {} → •lf[¢AreaFailed, {}, •finfo["Ndg"]]], ¢Proved ]];

Area[•lf[¢AreaProblem , constructions_ ⇒ expr_, finfo_], dummy___] :=
    Module[{psitList = {}, i, j, cc, ex, ff}, If[compexpr, optelim = False;
        ff = •finfo["Compute", expr], ff = •finfo["Proof"]];
      PrependTo[psitList,
        ProofStep[Prinfo[¢AreaConstructions,
            {}⟶•lf[¢AreaConstructions, constructions ⇒ expr, ff],
            optndg, constructions, expr],
          Psit[•lf[¢AreaElim, constructions ⇒ expr, ff], dummy]]];
      If[optndg, ff = •finfo["Ndg"]; cc := constructions;
        Do[ex = constructions[[i, 3, j]] //. OrdPoints;
          While[Length[cc] ≥ i, cc = Drop[cc, −1]];
          PrependTo[psitList,
            ProofStep[Prinfo[¢AreaConstructions,
                {}⟶•lf[¢AreaConstructions, cc ⇒ ex, ff], constructions[[i]]],
              Psit[•lf[¢AreaElim, cc ⇒ ex, ff], dummy]]],
          {i, Length[constructions], 2, −1}, {j, Length[constructions[[i, 3]]]}]];
      ProofStep @@ PrependTo[psitList, Prinfo[¢AreaStep,
          {}⟶•lf[¢AreaStep, constructions ⇒ expr, finfo]]]];

Area[•lf[l_, ™ForAll[___, ™Implies[ hyp_, concl_]], •finfo["Area"]],
      dummy___] := Module[{constructions, variants, expr,
        ff, i, j, k, cc, n, psitList = {}}, finfo = •finfo["Area"];
      {constructions, expr, variants} = TransArea[(hyp ∧ concl) /. ToMma];
      expr = expr //. OrdPoints;
      If[compexpr, ff = •finfo["Compute", expr], ff = •finfo["Proof"]];
      If[Length[variants] ≤ 1, ProofStep[
          Prinfo[¢AreaProblem, {}⟶•lf[l, constructions ⇒ expr, ff], optndg],
          Psit[•lf[¢AreaProblem, constructions ⇒ expr, finfo], dummy]],
        Do[n = Length[variants[[i]]];
          cc = ReplacePart[constructions, Table[variants[[i, j, 2]], {j, n}],
              Table[variants[[i, j, 1]], {j, n}], Table[{j}, {j, n}]];
          PrependTo[psitList, ProofStep[Prinfo[¢AreaProblem,
                {}⟶•lf[l, cc ⇒ expr, ff], optndg],
              Psit[•lf[¢AreaProblem, cc ⇒ expr, finfo], dummy]]],
```

```
        {i, Length[variants], 1, −1}]; k = Length[variants];
      ProofStep @@ PrependTo[psitList, Prinfo[¢AreaConstructions,
          {} ⟶ •lf[¢AreaProblem, cc ⇒ expr, finfo], k]]];

Area[•lf[l_, hyp_ ⇒ concl_, •finfo["Area"]], dummy___] :=
  Area[•lf[l, ™ForAll[™Implies[ hyp, concl]]], •finfo["Area"]], dummy]
```

Example:

```
Proposition["Pappus", any[A, B, A1, B1, C, C1, P, Q, S],
  point[A, B, A1, B1] ∧
    pon[C, line[A, B]] ∧
    pon[C1, line[A1, B1]] ∧
    inter[P, line[A, B1], line[A1, B]] ∧
    inter[Q, line[A, C1], line[A1, C]] ∧
    inter[S, line[B, C1], line[B1, C]]
   ⇒ collinear[P, Q, S]]

Prove[Proposition["Pappus"],
  by → GeometryProver, ProverOptions → {Method → "Area"}]

- ProofObject -
```

========== Output Generated by the Program ==========

Prove:

(Proposition (Pappus))

$$\forall_{A,B,A1,B1,C,C1,P,Q,S} (\text{point}[A, B, A1, B1] \wedge \text{pon}[C, \text{line}[A, B]] \wedge \text{pon}[C1, \text{line}[A1, B1]] \wedge \text{inter}[P, \text{line}[A, B1], \text{line}[A1, B]] \wedge \text{inter}[Q, \text{line}[A, C1], \text{line}[A1, C]] \wedge \text{inter}[S, \text{line}[B, C1], \text{line}[B1, C]] \Rightarrow \text{collinear}[P, Q, S])$$

with no assumptions.

We shall use the area method.

First we have to transform the problem into the internal form for the area method. We have to prove, that constructions

$\{A, B, A1, B1\}$ free points
$C\,A \parallel A\,B$ and $\frac{\overline{A\,C}}{\overline{A\,B}} = r_C$ with ndg. $\{A \neq B\}$
$C1\,A1 \parallel A1\,B1$ and $\frac{\overline{A1\,C1}}{\overline{A1\,B1}} = r_{C1}$ with ndg. $\{A1 \neq B1\}$
$P = A\,B1 \cap A1\,B$ with ndg. $\{A \neq B1, A1 \neq B, A\,B1 \not\parallel A1\,B\}$
$Q = A\,C1 \cap A1\,C$ with ndg. $\{A \neq C1, A1 \neq C, A\,C1 \not\parallel A1\,C\}$
$S = B\,C1 \cap B1\,C$ with ndg. $\{B \neq C1, B1 \neq C, B\,C1 \not\parallel B1\,C\}$

imply that points $\{P, Q, S\}$ are collinear

Explanations

As the area method supports just a few basic constructions we had to rewrite the initial problem using only these constructions.

The area method consist of eliminating constructed points from the lefthand-side of the conclusion expression in the reverse order of their construction. Elimination of a point

means using certain rewrite rules depending on the geometric quantity the point appears in and the construction by which the point was introduced.

First we have to verify the non-degenerate conditions for each construction, then to prove our statement.

Step 1

We have to verify the non−degenerateconditions for

$P = A\,B1 \cap A1\,B$ with ndg. $\{A \neq B1,\ A1 \neq B,\ A\,B1 \nparallel A1\,B\}$

That is, constructions

$\{A, B, A1, B1\}$ free points
$C\,A \parallel A\,B$ and $\frac{\overline{A\,C}}{\overline{A\,B}} = r_C$ with ndg. $\{A \neq B\}$
$C1\,A1 \parallel A1\,B1$ and $\frac{\overline{A1\,C1}}{\overline{A1\,B1}} = r_{C1}$ with ndg. $\{A1 \neq B1\}$

do not imply

$A\,B1 \parallel A1\,B$

We use the elimination method described in the above explanations.

As the last expression contains no constructed points we have to rewrite the expression in area coordinates of the free points:

Rewriting

$$\frac{1}{\bullet S_{\{A,B,A1\}}}(-((-\bullet S_{\{A,A1,A1\}} + \bullet S_{\{A,A1,B\}})\bullet S_{\{A,B,A\}} + (\bullet S_{\{A,A1,A\}} + -\bullet S_{\{A,A1,B\}})\bullet S_{\{A,B,A1\}} + (-\bullet S_{\{A,A1,A\}} + \bullet S_{\{A,A1,A1\}})\bullet S_{\{A,B,B\}})) + \frac{1}{\bullet S_{\{A,B,A1\}}}((-\bullet S_{\{A,A1,B\}} + \bullet S_{\{A,A1,B1\}})\bullet S_{\{A,B,A1\}} + (\bullet S_{\{A,A1,A1\}} + -\bullet S_{\{A,A1,B1\}})\bullet S_{\{A,B,B\}} + (-\bullet S_{\{A,A1,A1\}} + \bullet S_{\{A,A1,B\}})\bullet S_{\{A,B,B1\}})$$

Ordering

$$-\bullet S_{\{A,B,A1\}} + \frac{1}{\bullet S_{\{A,B,A1\}}}(\bullet S_{\{A,B,A1\}}(\bullet S_{\{A,A1,B1\}} + \bullet S_{\{A,B,A1\}}) + -\bullet S_{\{A,B,A1\}}\bullet S_{\{A,B,B1\}})$$

Simplifying

$$\bullet S_{\{A,A1,B1\}} + -\bullet S_{\{A,B,B1\}}$$

As the expression is still not 0, we try to use the possible relations between areas and lengths of segments:

Rewriting

$$-\bullet S_{\{\alpha O,\alpha U,B1\}} + \bullet S_{\{\alpha O,\alpha V,B1\}}$$

Ordering

$$-\bullet S_{\{\alpha O,\alpha U,B1\}} + \bullet S_{\{\alpha O,\alpha V,B1\}}$$

Simplifying

$$-\bullet S_{\{\alpha O,\alpha U,B1\}} + \bullet S_{\{\alpha O,\alpha V,B1\}}$$

Non-degenerate condition is verified.

Step 2

We have to verify the non−degenerateconditions for

$Q = A\ C1 \cap A1\ C$ with ndg. $\{A \neq C1,\ A1 \neq C,\ A\ C1 \nparallel A1\ C\}$

That is, constructions

$\{A, B, A1, B1\}$ free points
$C\ A \parallel A\ B$ and $\frac{\overline{A\ C}}{\overline{A\ B}} = r_C$ with ndg. $\{A \neq B\}$
$C1\ A1 \parallel A1\ B1$ and $\frac{\overline{A1\ C1}}{\overline{A1\ B1}} = r_{C1}$ with ndg. $\{A1 \neq B1\}$
$P = A\ B1 \cap A1\ B$ with ndg. $\{A \neq B1,\ A1 \neq B,\ A\ B1 \nparallel A1\ B\}$

do not imply

$A\ C1 \parallel A1\ C$

We use the elimination method described in the above explanations.

Eliminating point $C1$:

We eliminate point $C1$ using construction:

$C1\ A1 \parallel A1\ B1$ and $\frac{\overline{A1\ C1}}{\overline{A1\ B1}} = r_{C1}$ with ndg. $\{A1 \neq B1\}$

$$\bullet S_{\{A,A1,C\}} + -(-r_{C1} + \bullet R_{\{A1,B1,A1,B1\}}) \bullet S_{\{A1,C,A1\}} + -(r_{C1} + \bullet R_{\{A1,A1,A1,B1\}}) \bullet S_{\{A1,C,B1\}}$$

After ordering the points we obtain:

$$\bullet S_{\{A,A1,C\}} + r_{C1} \bullet S_{\{A1,B1,C\}}$$

The simplified expression is:

$$\bullet S_{\{A,A1,C\}} + r_{C1} \bullet S_{\{A1,B1,C\}}$$

Eliminating point $C$:

We eliminate point $C$ using construction: $C\ A \parallel A\ B$ and $\frac{\overline{A\ C}}{\overline{A\ B}} = r_C$ with ndg. $\{A \neq B\}$

$$(-r_C + \bullet R_{\{A,B,A,B\}}) \bullet S_{\{A,A1,A\}} + (r_C + \bullet R_{\{A,A,A,B\}}) \bullet S_{\{A,A1,B\}} +$$
$$r_{C1}\ ((-r_C + \bullet R_{\{A,B,A,B\}}) \bullet S_{\{A1,B1,A\}} + (r_C + \bullet R_{\{A,A,A,B\}}) \bullet S_{\{A1,B1,B\}})$$

After ordering the points we obtain:

$$-r_C \bullet S_{\{A,B,A1\}} + r_{C1}\ ((1 + -r_C) \bullet S_{\{A,A1,B1\}} + r_C \bullet S_{\{B,A1,B1\}})$$

The simplified expression is:

$$-r_C \bullet S_{\{A,B,A1\}} + r_{C1}\ (-((-1) + r_C) \bullet S_{\{A,A1,B1\}} + r_C \bullet S_{\{B,A1,B1\}})$$

As the last expression contains no constructed points we have to rewrite the expression in area coordinates of the free points: ====closed cell ====

As the expression is still not 0, we try to use the possible relations between areas and lengths of segments:

Rewriting ====closed cell ====

Ordering ====closed cell ====

Simplifying

$$r_C\ (((-1) + r_{C1}) \bullet S_{\{\alpha O,\alpha U,\alpha V\}} + -r_{C1} \bullet S_{\{\alpha O,\alpha U,B1\}}) + r_{C1} \bullet S_{\{\alpha O,\alpha V,B1\}}$$

Non−degeneratecondition is verified.

Step 3

We have to verify the non−degenerateconditions for

$S = B\,C1 \cap B1\,C$ with ndg. $\{B \neq C1,\ B1 \neq C,\ B\,C1 \nparallel B1\,C\}$

That is, constructions

$\{A, B, A1, B1\}$ free points
$C\,A \parallel A\,B$ and $\frac{\overline{A\,C}}{\overline{A\,B}} = r_C$ with ndg. $\{A \neq B\}$
$C1\,A1 \parallel A1\,B1$ and $\frac{\overline{A1\,C1}}{\overline{A1\,B1}} = r_{C1}$ with ndg. $\{A1 \neq B1\}$
$P = A\,B1 \cap A1\,B$ with ndg. $\{A \neq B1,\ A1 \neq B,\ A\,B1 \nparallel A1\,B\}$
$Q = A\,C1 \cap A1\,C$ with ndg. $\{A \neq C1,\ A1 \neq C,\ A\,C1 \nparallel A1\,C\}$

do not imply

$B\,C1 \parallel B1\,C$

We use the elimination method described in the above explanations.
====closed cell ====

Non−degeneratecondition is verified.

Step 4

As all the non−degenerateconditions are verified now we have to prove

$\{A, B, A1, B1\}$ free points
$C\,A \parallel A\,B$ and $\frac{\overline{A\,C}}{\overline{A\,B}} = r_C$ with ndg. $\{A \neq B\}$
$C1\,A1 \parallel A1\,B1$ and $\frac{\overline{A1\,C1}}{\overline{A1\,B1}} = r_{C1}$ with ndg. $\{A1 \neq B1\}$
$P = A\,B1 \cap A1\,B$ with ndg. $\{A \neq B1,\ A1 \neq B,\ A\,B1 \nparallel A1\,B\}$
$Q = A\,C1 \cap A1\,C$ with ndg. $\{A \neq C1,\ A1 \neq C,\ A\,C1 \nparallel A1\,C\}$
$S = B\,C1 \cap B1\,C$ with ndg. $\{B \neq C1,\ B1 \neq C,\ B\,C1 \nparallel B1\,C\}$
⇒points $\{P, Q, S\}$ are collinear

Eliminating point $S$: ====closed cell ====

Eliminating point $Q$: ====closed cell ====

Eliminating point $P$: ====closed cell ====

Eliminating point $C1$:

We eliminate point $C1$ using construction:

$C1\,A1 \parallel A1\,B1$ and $\frac{\overline{A1\,C1}}{\overline{A1\,B1}} = r_{C1}$ with ndg. $\{A1 \neq B1\}$
====closed cell ====

After ordering the points we obtain: ====closed cell ====

The simplified expression is:

$$\begin{aligned}&(((-1)+r_{C1})\,r_{C1}\,\bullet S_{\{A1,B1,C\}}\,(\bullet S_{\{A,A1,C\}}\,\bullet S_{\{A,B,B1\}}\,\bullet S_{\{B,A1,B1\}}+\\&\qquad \bullet S_{\{A,B,A1\}}\,(\bullet S_{\{A,B,B1\}}\,\bullet S_{\{A1,B1,C\}}+-\bullet S_{\{A,A1,B1\}}\,\bullet S_{\{B,B1,C\}})))/\\&((\bullet S_{\{A,A1,C\}}+r_{C1}\,\bullet S_{\{A1,B1,C\}})\,(\bullet S_{\{A,B,A1\}}+-\bullet S_{\{B,A1,B1\}})\\&\quad(((-1)+r_{C1})\,\bullet S_{\{A1,B1,C\}}+\bullet S_{\{B,B1,C\}}))\end{aligned}$$

Eliminating point $C$:

We eliminate point $C$ using construction: $C\,A \parallel A\,B$ and $\frac{\overline{A\,C}}{\overline{A\,B}} = r_C$ with ndg. $\{A \neq B\}$

$$
\begin{aligned}
&(((-1)+r_{C1})\,r_{C1}\,((-r_C+\bullet R_{\{A,B,A,B\}})\bullet S_{\{A1,B1,A\}}+(r_C+\bullet R_{\{A,A,A,B\}})\bullet S_{\{A1,B1,B\}})\\
&\quad(((-r_C+\bullet R_{\{A,B,A,B\}})\bullet S_{\{A,A1,A\}}+(r_C+\bullet R_{\{A,A,A,B\}})\bullet S_{\{A,A1,B\}})\bullet S_{\{A,B,B1\}}\bullet S_{\{B,A1,B1\}}+\\
&\quad\bullet S_{\{A,B,A1\}}\,(\bullet S_{\{A,B,B1\}}\,((-r_C+\bullet R_{\{A,B,A,B\}})\bullet S_{\{A1,B1,A\}}+\\
&\quad\quad(r_C+\bullet R_{\{A,A,A,B\}})\bullet S_{\{A1,B1,B\}})+-\bullet S_{\{A,A1,B1\}}\\
&\quad\quad((-r_C+\bullet R_{\{A,B,A,B\}})\bullet S_{\{B,B1,A\}}+(r_C+\bullet R_{\{A,A,A,B\}})\bullet S_{\{B,B1,B\}}))))/\\
&((-r_C+\bullet R_{\{A,B,A,B\}})\bullet S_{\{A,A1,A\}}+(r_C+\bullet R_{\{A,A,A,B\}})\bullet S_{\{A,A1,B\}}+\\
&\quad r_{C1}\,((-r_C+\bullet R_{\{A,B,A,B\}})\bullet S_{\{A1,B1,A\}}+(r_C+\bullet R_{\{A,A,A,B\}})\bullet S_{\{A1,B1,B\}}))\\
&\quad(\bullet S_{\{A,B,A1\}}+-\bullet S_{\{B,A1,B1\}})\\
&\quad(((-1)+r_{C1})\,((-r_C+\bullet R_{\{A,B,A,B\}})\bullet S_{\{A1,B1,A\}}+(r_C+\bullet R_{\{A,A,A,B\}})\bullet S_{\{A1,B1,B\}})+\\
&\quad(-r_C+\bullet R_{\{A,B,A,B\}})\bullet S_{\{B,B1,A\}}+(r_C+\bullet R_{\{A,A,A,B\}})\bullet S_{\{B,B1,B\}}))
\end{aligned}
$$

After ordering the points we obtain:

$$
\begin{aligned}
&(((-1)+r_{C1})\,r_{C1}\,((1+-r_C)\bullet S_{\{A,A1,B1\}}+r_C\bullet S_{\{B,A1,B1\}})\\
&\quad(-r_C\bullet S_{\{A,B,A1\}}\bullet S_{\{A,B,B1\}}\bullet S_{\{B,A1,B1\}}+\bullet S_{\{A,B,A1\}}\,(-(1+-r_C)\bullet S_{\{A,A1,B1\}}\bullet S_{\{A,B,B1\}}+\\
&\quad\quad\bullet S_{\{A,B,B1\}}\,((1+-r_C)\bullet S_{\{A,A1,B1\}}+r_C\bullet S_{\{B,A1,B1\}}))))/\\
&((\bullet S_{\{A,B,A1\}}+-\bullet S_{\{B,A1,B1\}})\,((1+-r_C)\bullet S_{\{A,B,B1\}}+\\
&\quad((-1)+r_{C1})\,((1+-r_C)\bullet S_{\{A,A1,B1\}}+r_C\bullet S_{\{B,A1,B1\}}))\\
&\quad(-r_C\bullet S_{\{A,B,A1\}}+r_{C1}\,((1+-r_C)\bullet S_{\{A,A1,B1\}}+r_C\bullet S_{\{B,A1,B1\}})))
\end{aligned}
$$

The simplified expression is:

0

The initial problem was proved, as we obtained 0 == 0.

□

========== End of Generated Output ==========

### *5.5.4 Computing Expressions Containing Geometric Quantities Using the Area Method*

The area method is very convenient for computing expressions involving geometric quantities relative to a specified construction. The construction has to be given using the *Theorema* construct *KnowledgeBase*.

As computing with the area method consists of eliminating constructed points from the given expression and interpreting the obtained result not as a failing proof but as the value of the expression function *AreaCompute* calls the basic prover *Area* with *ProverOptions*→{*ComputeExpression*→*True*}, which will force option *ElimFactors* to *False*. Afterwards it will extract the computed expression from the generated proofobject and return it as the value of the function.

Besides the options admitted by the basic prover *Area* presented in section 5.5.3 *AreaCompute* also admits for *EvaluatorOptions ShowSteps* with default value *True* to ask whether a notebook containing the computations should be generated or not.

Example

```
KnowledgeBase["example", any[A, B, C, D, A1, A2, B1, r],
  point[A, B, C] ∧
    inter[D, pline[A, B, C], pline[C, A, B]] ∧
    divpoint[A1, C, D, r/(r − 1)] ∧
    divpoint[B1, D, A, r/(r − 1)] ∧
    inter[A2, line[A, A1], line[B, B1]]]


Compute[trarea[A, B, A2]/qarea[A, B, C, D],
  by → AreaCompute, using → KnowledgeBase["example"],
  EvaluatorOptions → {DetailedPrint → False}]
```

$$\frac{1 + -r}{4 + -4\,r + 2\,r^2}$$

========== Output Generated by the Program ==========

Compute:

(Expression (example))

$$\frac{\text{trarea}[A,B,A2]}{\text{qarea}[A,B,C,D]}$$

using the following constructions:

(KnowledgeBase (example))

point[$A$, $B$, $C$] ⋀ inter[$D$, pline[$A$, $B$, $C$], pline[$C$, $A$, $B$]] ⋀ divpoint[$A1$, $C$, $D$, $\frac{r}{r-1}$] ⋀
divpoint[$B1$, $D$, $A$, $\frac{r}{r-1}$] ⋀ inter[$A2$, line[$A$, $A1$], line[$B$, $B1$]]

We shall use the area method.

First we have to transform the problem in internal form for the area method. Using constructions

{$A$, $B$, $C$} free points
$\gamma_D\ C \parallel A\ B$ and $\frac{\overline{C\ \gamma_D}}{\overline{A\ B}} = r_5$ with ndg. {$A \neq B$}
$\beta_D\ A \parallel B\ C$ and $\frac{\overline{A\ \beta_D}}{\overline{B\ C}} = r_4$ with ndg. {$B \neq C$}
$D = A\ \beta_D \cap C\ \gamma_D$ with ndg. {$A \neq \beta_D$, $C \neq \gamma_D$, $\beta_D\ A \not\parallel \gamma_D\ C$}
$A1\ C \parallel C\ D$ and $\frac{\overline{C\ A1}}{\overline{C\ D}} = r$ with ndg. {$C \neq D$}
$B1\ D \parallel D\ A$ and $\frac{\overline{D\ B1}}{\overline{D\ A}} = r$ with ndg. {$D \neq A$}
$A2 = A\ A1 \cap B\ B1$ with ndg. {$A \neq A1$, $B \neq B1$, $A\ A1 \not\parallel B\ B1$}

we have to compute $\frac{\bullet S_{\{A,B,A2\}}}{\bullet S_{\{A,B,C,D\}}}$

Explanations ==== *closed cell* ====

First we have to verify the non−degenerateconditions for each construction, then to compute our expression.

Step 1

We have to verify the non−degenerateconditions for

$D = A\ \beta_D \bigcap C\ \gamma_D$ with ndg. $\{A \neq \beta_D, C \neq \gamma_D, \beta_D\ A \not\parallel \gamma_D\ C\}$

That is, constructions

$\{A, B, C\}$ free points
$\gamma_D\ C \parallel A\ B$ and $\frac{\overline{C\ \gamma_D}}{\overline{A\ B}} = r_5$ with ndg. $\{A \neq B\}$
$\beta_D\ A \parallel B\ C$ and $\frac{\overline{A\ \beta_D}}{\overline{B\ C}} = r_4$ with ndg. $\{B \neq C\}$

do not imply

$\beta_D\ A \parallel \gamma_D\ C$

==== verifying ndg. ====

Non-degenerate condition is verified.

Step 2

We have to verify the non−degenerateconditions for

$A2 = A\ A1 \bigcap B\ B1$ with ndg. $\{A \neq A1, B \neq B1, A\ A1 \not\parallel B\ B1\}$

That is, constructions

$\{A, B, C\}$ free points
$\gamma_D\ C \parallel A\ B$ and $\frac{\overline{C\ \gamma_D}}{\overline{A\ B}} = r_5$ with ndg. $\{A \neq B\}$
$\beta_D\ A \parallel B\ C$ and $\frac{\overline{A\ \beta_D}}{\overline{B\ C}} = r_4$ with ndg. $\{B \neq C\}$
$D = A\ \beta_D \bigcap C\ \gamma_D$ with ndg. $\{A \neq \beta_D, C \neq \gamma_D, \beta_D\ A \not\parallel \gamma_D\ C\}$
$A1\ C \parallel C\ D$ and $\frac{\overline{C\ A1}}{\overline{C\ D}} = r$ with ndg. $\{C \neq D\}$
$B1\ D \parallel D\ A$ and $\frac{\overline{D\ B1}}{\overline{D\ A}} = r$ with ndg. $\{D \neq A\}$

do not imply

$A\ A1 \parallel B\ B1$

==== verifying ndg. ====

Non-degenerate condition is verified.

Step 3

As all the non−degenerateconditions are verified, we have to compute $\frac{\bullet S_{\{A,B,A2\}}}{\bullet S_{\{A,B,C,D\}}}$, using constructions

$\{A, B, C\}$ free points
$\gamma_D\ C \parallel A\ B$ and $\frac{\overline{C\ \gamma_D}}{\overline{A\ B}} = r_5$ with ndg. $\{A \neq B\}$
$\beta_D\ A \parallel B\ C$ and $\frac{\overline{A\ \beta_D}}{\overline{B\ C}} = r_4$ with ndg. $\{B \neq C\}$
$D = A\ \beta_D \bigcap C\ \gamma_D$ with ndg. $\{A \neq \beta_D, C \neq \gamma_D, \beta_D\ A \not\parallel \gamma_D\ C\}$
$A1\ C \parallel C\ D$ and $\frac{\overline{C\ A1}}{\overline{C\ D}} = r$ with ndg. $\{C \neq D\}$
$B1\ D \parallel D\ A$ and $\frac{\overline{D\ B1}}{\overline{D\ A}} = r$ with ndg. $\{D \neq A\}$
$A2 = A\ A1 \bigcap B\ B1$ with ndg. $\{A \neq A1, B \neq B1, A\ A1 \not\parallel B\ B1\}$

Eliminating point $A2$:

We eliminate point $A2$ using construction:
$A2 = A\ A1 \bigcap B\ B1$ with ndg. $\{A \neq A1, B \neq B1, A\ A1 \nparallel B\ B1\}$

We obtain:

$$\frac{\bullet S_{\{A,B,A1\}}\ \bullet S_{\{A,B,B1\}}}{(\bullet S_{\{A,A1,B1\}} + \bullet S_{\{A,B,A1\}})\ (\bullet S_{\{A,B,C\}} + \bullet S_{\{A,C,D\}})}$$

Eliminating point $B1$: ==== *closed cell* ====

Eliminating point $A1$: ==== *closed cell* ====

Eliminating point $D$: ==== *closed cell* ====

Eliminating point $\beta_D$: ==== *closed cell* ====

We eliminate point $\beta_D$ using construction: $\beta_D\ A \parallel B\ C$ and $\frac{\overline{A\ \beta_D}}{\overline{B\ C}} = r_4$ with ndg. $\{B \neq C\}$

We obtain:

$$(-\bullet S_{\{A,C,\gamma_D\}}\ \bullet S_{\{B,C,\gamma_D\}}\ ((-1) + r)) /$$
$$((\bullet S_{\{A,C,\gamma_D\}} + \bullet S_{\{B,C,\gamma_D\}})\ (\bullet S_{\{B,C,\gamma_D\}} + \bullet S_{\{A,C,\gamma_D\}}\ ((-1) + r)^2))$$

Eliminating point $\gamma_D$:

We eliminate point $\gamma_D$ using construction: $\gamma_D\ C \parallel A\ B$ and $\frac{\overline{C\ \gamma_D}}{\overline{A\ B}} = r_5$ with ndg. $\{A \neq B\}$

We obtain:

$$\frac{1 + -r}{4 + -4\,r + 2\,r^2}$$

Thus

$$\frac{\bullet S_{\{A,B,A2\}}}{\bullet S_{\{A,B,C\}} + \bullet S_{\{A,C,D\}}} = \frac{1 + -r}{4 + -4\,r + 2\,r^2}$$

□

========== End of Generated Output ==========

# 6 Subalgorithms and Heuristics for Improving Existing Methods

## *6.1 Choice of an Appropriate Coordinate System*

For any algebraic method, the first step is the choice of the coordinate system, translation of the geometrical properties into polynomial equations, separating the parameters and the free variables and ordering the free variables. The performance of each algorithm depends heavily on these factors.

One possibility is to let the choice of the coordinate system be a responsibility of the user. In this case the first defined point would be the origin of the coordinate system and the next a point on the *x* axis. The user should find a convenient order of constructing the points, which is not always easy. Each construction determines whether the point introduced by that construction is a free point, has a degree of freedom, or is fully determined. For example *point*[*A*] would introduce a free point, *pon*[*A*, *line*[*C, D*]] a point on a line, that means it can be anywhere on that line, so it has one degree of freedom, it is semi-bound, while point *X* introduced by *inter*[*X*, *line*[*A, B*], *line*[*C, D*]] is uniquely determined, i.e. it is bound. A free point has two parameters as coordinates, a semi-bound point has a parameter and a variable, while a bound point has two variables. The variables are ordered according to the construction order of the points.

It is more convenient to let the program choose the coordinate system according to some heuristics. It is obvious, that the computations will be easier and faster, if more points are on the axes, and this way more points will have 0 as a coordinate. Again, it is preferable to have as origin a point, which is used in constructing more points, or whose coordinates are involved in second order polynomials. So, if we have perpendicular lines, it is useful to take their intersection point as origin. This will introduce some points with a coordinate 0 on both axes, and at the same time we get rid of a second order polynomial (perpendicularity). If the statement contains no perpendicularity, then we can take as x-axis the line that has the greatest number of incidental points, the first constructed of these being the origin. If we have circles, then a good choice for the x-axis would be the line represented by the two defining points of a circle. This would simplify the second order polynomials introduced by the construction of the points on that circle.

If we choose the origin of the coordinate system from the semi-bound or bound points this point has to become a free point, so virtually the construction sequence of the points has to be changed. Chaining backwards a semi-bound or free point that is used when constructing the origin has to be found and bound.

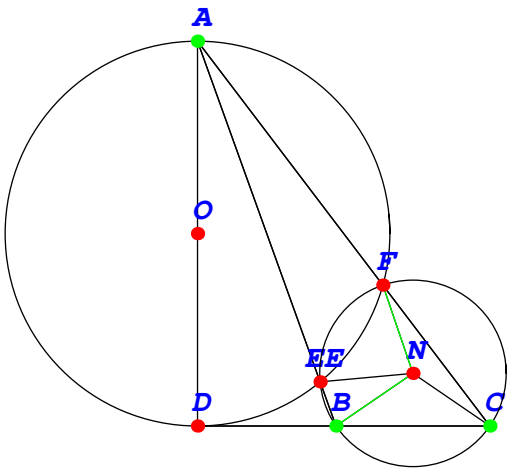
Our program preprocesses the algebraic form of the geometrical statement trying to find a convenient coordinate system and variable ordering. The user may decide if he wants to give the implicit coordinate system by the construction order of the points or wants to use the automatic mechanism of the prover using the option *MyAxis*.

Example:

Let the following geometrical statement be given:

```
Proposition["ex", any[A, B, C, D, O, EE, F, N],
  triangle[A, B, C] ∧
    height[A, D, A, B, C] ∧
    midpoint[O, A, D] ∧
    inter[EE, circle[O, A], line[A, B]] ∧
    inter[F, circle[O, A], line[A, C]] ∧
    eqdist[N, B, N, EE] ∧
    circle[N, B] ∧
    eqdist[N, F, N, C]
 ⇒ distequal[N, F, B, N]]

KnowledgeBase["ex_g", any[A, B, C],
  {{A, {10, 260}}, {B, {100, 10}}, {C, {200, 10}}}]

Simplify[Proposition["ex"],
  by –> GraphicSimplifier, using → KnowledgeBase["ex_g"]]
```

========== Output Generated by the Program ==========




$|NF| \cong |BN|$ for this configuration of the points

========== End of Generated Output ==========

In the implicit case, without using our heuristics we obtain for the algebraic form:

```
Simplify[Proposition["ex"], by → AlgebraicSimplifier,
    SimplifierOptions → {MyAxis → False}];
```

========== Output Generated by the Program ==========

Algebraic Form:

To transform the geometric problem into algebraic form, we have to chose first an orthogonal coordinate system.

Let's have the origin in point $A$, and points {} and {$B$, $EE$} on the two axes.

Using this coordinate system we have the following points:

{{$A$, 0, 0}, {$B$, 0, $u_1$}, {$C$, $u_2$, $u_3$}, {$D$, $x_1$, $x_2$},
{$O$, $x_3$, $x_4$}, {$EE$, 0, $x_5$}, {$F$, $x_6$, $x_7$}, {$N$, $x_8$, $x_9$}}

The algebraic form of the given construction is:

(1)

$$\forall_{v_1, v_2, v_3, v_4, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9} ((-1) + u_2\, v_4 + -v_4\, x_6 == 0 \bigwedge (-1) + u_1\, v_3 + -v_3\, x_5 == 0 \bigwedge (-1) + -v_2\, x_6 == 0 \bigwedge (-1) + -v_1\, x_5 == 0 \bigwedge u_1\, u_2 + -u_1\, x_1 + u_3\, x_1 + -u_2\, x_2 == 0 \bigwedge -u_2\, x_1 + u_1\, x_2 + -u_3\, x_2 == 0 \bigwedge -x_1 + 2\, x_3 == 0 \bigwedge -x_2 + 2\, x_4 == 0 \bigwedge -2\, x_4\, x_5 + x_5^2 == 0 \bigwedge -2\, x_3\, x_6 + x_6^2 + -2\, x_4\, x_7 + x_7^2 == 0 \bigwedge u_3\, x_6 + -u_2\, x_7 == 0 \bigwedge u_1^2 + -x_5^2 + -2\, u_1\, x_9 + 2\, x_5\, x_9 == 0 \bigwedge -u_2^2 + -u_3^2 + x_6^2 + x_7^2 + 2\, u_2\, x_8 + -2\, x_6\, x_8 + 2\, u_3\, x_9 + -2\, x_7\, x_9 == 0 \Rightarrow -u_1^2 + x_6^2 + x_7^2 + -2\, x_6\, x_8 + 2\, u_1\, x_9 + -2\, x_7\, x_9 == 0)$$

========== End of Generated Output ==========

With heuristics:

```
Simplify[Proposition["ex"], by → AlgebraicSimplifier,
    SimplifierOptions → {MyAxis → True}];
```

========== Output Generated by the Program ==========

Algebraic Form:

To transform the geometric problem into algebraic form, we have to chose first an orthogonal coordinate system.

Let's have the origin in point $D$, and points {$B$, $C$} and {$A$, $O$} on the two axes.

Using this coordinate system we have the following points:

{{$D$, 0, 0}, {$A$, 0, $u_1$}, {$B$, $u_2$, 0}, {$C$, $u_3$, 0},
{$O$, 0, $x_1$}, {$EE$, $x_2$, $x_3$}, {$F$, $x_4$, $x_5$}, {$N$, $x_6$, $x_7$}}

The algebraic form of the given construction is:

(1)

$$\forall_{v_1,v_2,v_3,v_4,x_1,x_2,x_3,x_4,x_5,x_6,x_7} ((-1) + u_3\, v_4 + -v_4\, x_4 == 0 \bigwedge (-1) + u_2\, v_3 + -v_3\, x_2 == 0 \bigwedge$$
$$(-1) + -v_2\, x_4 == 0 \bigwedge (-1) + -v_1\, x_2 == 0 \bigwedge -u_1 + 2\, x_1 == 0 \bigwedge$$
$$-u_1^{\,2} + 2\, u_1\, x_1 + x_2^{\,2} + -2\, x_1\, x_3 + x_3^{\,2} == 0 \bigwedge u_1\, u_2 + -u_1\, x_2 + -u_2\, x_3 == 0 \bigwedge$$
$$-u_1^{\,2} + 2\, u_1\, x_1 + x_4^{\,2} + -2\, x_1\, x_5 + x_5^{\,2} == 0 \bigwedge u_1\, u_3 + -u_1\, x_4 + -u_3\, x_5 == 0 \bigwedge$$
$$u_2^{\,2} + -x_2^{\,2} + -x_3^{\,2} + -2\, u_2\, x_6 + 2\, x_2\, x_6 + 2\, x_3\, x_7 == 0 \bigwedge$$
$$-u_3^{\,2} + x_4^{\,2} + x_5^{\,2} + 2\, u_3\, x_6 + -2\, x_4\, x_6 + -2\, x_5\, x_7 == 0 \Rightarrow$$
$$-u_2^{\,2} + x_4^{\,2} + x_5^{\,2} + 2\, u_2\, x_6 + -2\, x_4\, x_6 + -2\, x_5\, x_7 == 0)$$

========== End of Generated Output ==========

We can observe that using our preprocessing, we have just 7 variables, instead of 9. In this example the proof completes almost three times faster then in the implicit case.

## 6.2 Incremental Wu's Method

### 6.2.1 Method

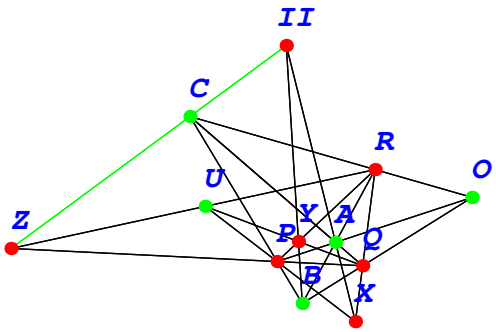
Translating constructive geometry statements into algebraic form we get a special system of polynomials. Each point is introduced by a construction using only previously defined points and each construction is translated into algebraic form using one or two polynomial equalities. Thus the hypothesis polynomials have a "semi" triangular form from the beginning. The computation of the characteristic set is faster if we do not try to work with all the hypothesis polynomials at once, but build the characteristic set step by step. In step 0 we take the empty set as characteristic set. Then in each step one or two polynomials corresponding to a construction are added to the characteristic set obtained in the previous step and the characteristic set of this polynomial set is computed.

This observation, also near at hand, was apparently not used in the literature so far.

### 6.2.2 Example

```
Proposition["Chou26", any[A, B, C, O, U, P, Q, R, X, Y, Z, II],
  triangle[A, B, C] ∧
    point[O, U] ∧
    inter[P, line[A, O], line[B, C]] ∧
    inter[Q, line[B, O], line[C, A]] ∧
    inter[R, line[C, O], line[A, B]] ∧
    inter[X, line[P, U], line[Q, R]] ∧
    inter[Y, line[Q, U], line[R, P]] ∧
    inter[Z, line[R, U], line[P, Q]] ∧
    inter[II, line[A, X], line[B, Y]]
 ⇒ collinear[II, C, Z]]

Simplify[Proposition["Chou26"], by -> GraphicSimplifier]
```

========== Output Generated by the Program ==========




Points ***II C*** and ***Z*** are collinear for this configuration of the points

========== End of Generated Output ==========

Prove[Proposition["Chou26"], by → GeometryProver,
ProverOptions → {Method → "Wu", Iterate → False}]

========== Output Generated by the Program ==========

Prove:

(Proposition (Chou26))

$$\forall_{A,B,C,O,U,P,Q,R,X,Y,Z,II} (\text{triangle}[A, B, C] \wedge \text{point}[O, U] \wedge$$
$$\text{inter}[P, \text{line}[A, O], \text{line}[B, C]] \wedge \text{inter}[Q, \text{line}[B, O], \text{line}[C, A]] \wedge$$
$$\text{inter}[R, \text{line}[C, O], \text{line}[A, B]] \wedge \text{inter}[X, \text{line}[P, U], \text{line}[Q, R]] \wedge$$
$$\text{inter}[Y, \text{line}[Q, U], \text{line}[R, P]] \wedge \text{inter}[Z, \text{line}[R, U], \text{line}[P, Q]] \wedge$$
$$\text{inter}[II, \text{line}[A, X], \text{line}[B, Y]] \Rightarrow \text{collinear}[II, C, Z])$$

with no assumptions.

To prove the above statement we shall use Wu's method. First we have to transform the problem into algebraic form.

Algebraic Form:

To transform the geometric problem into algebraic form we have to chose first an orthogonal coordinate system.

Let's have the origin in point ***P***, and points {} and {***B***, ***C***} on the two axes.

Using this coordinate system we have the following points:

{{***P***, 0, 0}, {***A***, $u_1$, $u_2$}, {***B***, 0, $u_3$}, {***C***, 0, $u_4$}, {***U***, $u_5$, $u_6$}, {***O***, $u_7$, $x_1$},
{***Q***, $x_2$, $x_3$}, {***R***, $x_4$, $x_5$}, {***X***, $x_6$, $x_7$}, {***Y***, $x_8$, $x_9$}, {***Z***, $x_{10}$, $x_{11}$}, {***II***, $x_{12}$, $x_{13}$}}

The algebraic form of the given construction is:

(1)

$$\forall_{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_{10},x_{11},x_{12},x_{13}} (u_2\,u_7 + -u_1\,x_1 == 0 \wedge$$
$$u_3\,u_7 + -u_3\,x_2 + x_1\,x_2 + -u_7\,x_3 == 0 \wedge u_1\,u_4 + u_2\,x_2 + -u_4\,x_2 + -u_1\,x_3 == 0 \wedge u_4\,u_7 + -u_4\,x_4 + x_1\,x_4 + -u_7\,x_5 == 0 \wedge$$
$$-u_1\,u_3 + -u_2\,x_4 + u_3\,x_4 + u_1\,x_5 == 0 \wedge u_6\,x_6 + -u_5\,x_7 == 0 \wedge$$
$$x_3\,x_4 + -x_2\,x_5 + -x_3\,x_6 + x_5\,x_6 + x_2\,x_7 + -x_4\,x_7 == 0 \wedge$$
$$-u_6\,x_2 + u_5\,x_3 + u_6\,x_8 + -x_3\,x_8 + -u_5\,x_9 + x_2\,x_9 == 0 \wedge -x_5\,x_8 + x_4\,x_9 == 0 \wedge$$
$$-u_6\,x_4 + u_5\,x_5 + u_6\,x_{10} + -x_5\,x_{10} + -u_5\,x_{11} + x_4\,x_{11} == 0 \wedge$$
$$x_3\,x_{10} + -x_2\,x_{11} == 0 \wedge$$
$$u_2\,x_6 + -u_1\,x_7 + -u_2\,x_{12} + x_7\,x_{12} + u_1\,x_{13} + -x_6\,x_{13} == 0 \wedge$$
$$u_3\,x_8 + -u_3\,x_{12} + x_9\,x_{12} + -x_8\,x_{13} == 0 \Rightarrow$$
$$u_4\,x_{10} + -u_4\,x_{12} + x_{11}\,x_{12} + -x_{10}\,x_{13} == 0)$$

First we compute the characteristic set of the hypothesis polynomials.

Characteristics of the characteristic set:

Time needed to compute the characteristic set: 384.84 Seconds.

During the computation the following factors were removed:
==== *closed cell* ====

The maximum number of monomials in a polynomial during the computation was: 1769.

The set of initials of the polynomials in the characteristic set is:
==== *closed cell* ====

The index set of the polynomials in the characteristic set is:
{{2, 1, 1}, {4, 2, 1}, {6, 3, 1}, {4, 4, 1}, {6, 5, 1}, {7, 6, 1}, {7, 7, 1},
{26, 8, 1}, {29, 9, 1}, {26, 10, 1}, {29, 11, 1}, {46, 12, 1}, {64, 13, 1}}

The characteristic set is: ==== *closed cell* ====

Next we compute the pseudoremainder of the conclusion polynomial
$u_4\,x_{10} + -u_4\,x_{12} + x_{11}\,x_{12} + -x_{10}\,x_{13}$ with respect to the obtained characteristic set.

The remainder is: 0

Time needed to compute the remainder: 2.59 Seconds.

The maximum number of monomials in a polynomial during the computation was: 162.

As the obtained remainder is 0 the statement is generically true.

□

========== End of Generated Output ==========

Using our incremental version the maximum number of monomials in a polynomial during the computation of the characteristic set decreases from 1769 to 503 executing the computations more than 6 times quicker. The proof of the same problem becomes:

```
Prove[Proposition["Chou26"],
  by → GeometryProver, ProverOptions → {Method → "Wu"}]
```

========== Output Generated by the Program ==========

Prove:

(Proposition (Chou26))

$$\forall_{A,B,C,O,U,P,Q,R,X,Y,Z,II} (\text{triangle}[A, B, C] \wedge \text{point}[O, U] \wedge \text{inter}[P, \text{line}[A, O], \text{line}[B, C]] \wedge \text{inter}[Q, \text{line}[B, O], \text{line}[C, A]] \wedge \text{inter}[R, \text{line}[C, O], \text{line}[A, B]] \wedge \text{inter}[X, \text{line}[P, U], \text{line}[Q, R]] \wedge \text{inter}[Y, \text{line}[Q, U], \text{line}[R, P]] \wedge \text{inter}[Z, \text{line}[R, U], \text{line}[P, Q]] \wedge \text{inter}[II, \text{line}[A, X], \text{line}[B, Y]] \Rightarrow \text{collinear}[II, C, Z])$$

with no assumptions.

To prove the above statement we shall use Wu's method. First we have to transform the problem into algebraic form.

Algebraic Form:

To transform the geometric problem into algebraic form we have to chose first an orthogonal coordinate system.

Let's have the origin in point $P$, and points {} and $\{B, C\}$ on the two axes.

Using this coordinate system we have the following points:

$$\{\{P, 0, 0\}, \{A, u_1, u_2\}, \{B, 0, u_3\}, \{C, 0, u_4\}, \{U, u_5, u_6\}, \{O, u_7, x_1\}, \{Q, x_2, x_3\}, \{R, x_4, x_5\}, \{X, x_6, x_7\}, \{Y, x_8, x_9\}, \{Z, x_{10}, x_{11}\}, \{II, x_{12}, x_{13}\}\}$$

The algebraic form of the given construction is:

(1)

$$\forall_{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,x_{10},x_{11},x_{12},x_{13}} (u_2 u_7 + -u_1 x_1 == 0 \wedge u_3 u_7 + -u_3 x_2 + x_1 x_2 + -u_7 x_3 == 0 \wedge u_1 u_4 + u_2 x_2 + -u_4 x_2 + -u_1 x_3 == 0 \wedge u_4 u_7 + -u_4 x_4 + x_1 x_4 + -u_7 x_5 == 0 \wedge -u_1 u_3 + -u_2 x_4 + u_3 x_4 + u_1 x_5 == 0 \wedge u_6 x_6 + -u_5 x_7 == 0 \wedge x_3 x_4 + -x_2 x_5 + -x_3 x_6 + x_5 x_6 + x_2 x_7 + -x_4 x_7 == 0 \wedge -u_6 x_2 + u_5 x_3 + u_6 x_8 + -x_3 x_8 + -u_5 x_9 + x_2 x_9 == 0 \wedge -x_5 x_8 + x_4 x_9 == 0 \wedge -u_6 x_4 + u_5 x_5 + u_6 x_{10} + -x_5 x_{10} + -u_5 x_{11} + x_4 x_{11} == 0 \wedge x_3 x_{10} + -x_2 x_{11} == 0 \wedge u_2 x_6 + -u_1 x_7 + -u_2 x_{12} + x_7 x_{12} + u_1 x_{13} + -x_6 x_{13} == 0 \wedge u_3 x_8 + -u_3 x_{12} + x_9 x_{12} + -x_8 x_{13} == 0 \Rightarrow u_4 x_{10} + -u_4 x_{12} + x_{11} x_{12} + -x_{10} x_{13} == 0)$$

First we compute the characteristic set of the hypothesis polynomials.We use successive steps, introducing polynomials for each point in a separate step.

Characteristics of the characteristic set:

Time needed to compute the characteristic set: 60.52 Seconds.

During the computation the following factors were removed:

==== *closed cell* ====

The maximum number of monomials in a polynomial during the computation was: 503

The set of initials of the polynomials in the characteristic set is:
==== *closed cell* ====

The index set of the polynomials in the characteristic set is:

{{2, 1, 1}, {4, 2, 1}, {6, 3, 1}, {4, 4, 1}, {6, 5, 1}, {7, 6, 1}, {7, 7, 1},
{26, 8, 1}, {29, 9, 1}, {26, 10, 1}, {29, 11, 1}, {46, 12, 1}, {64, 13, 1}}

The characteristic set is: ==== *closed cell* ====

Next we compute the pseudoremainder of the conclusion polynomial
$u_4\, x_{10} + -u_4\, x_{12} + x_{11}\, x_{12} + -x_{10}\, x_{13}$ with respect to the obtained characteristic set.

The remainder is: 0

Time needed to compute the remainder: 2.62 Seconds.

The maximum number of monomials in a polynomial during the computation was: 162.

As the obtained remainder is 0 the statement is generically true.

□

========== End of Generated Output ==========

# 7 Combining Methods

## *7.1 Proving Geometrical Theorems Involving Order Relation*

Collins' Cylindrical Algebraic Decomposition method (we shall refer to it as CAD method) introduced in [18], improving earlier work of [38] , can be used to prove geometry theorems that involve order relations (that is, the algebraic form consists of polynomial equalities and inequalities) as has been observed in [30], [5] and other papers. In our implementation we use the *Experimental* package of *Mathematica*. Unfortunately only very simple geometric statements can be proved this way, as the method is very time consuming.

To overcome the slowness of Collins' CAD method for complicated polynomials we developed a method that combines the area and the CAD method. First we compute the expressions involved in inequalities using the area method. This way we obtain a new problem equivalent to the original one that is expressed only in terms of the free points (introduced by a *point* construction) of the original constructions. Applying the CAD method to this new problem we can obtain the result in reasonable time even for quite complicated problems.

## *7.2 AreaCAD Method*

### *7.2.1 The Method*

According to definition 3 in section 3.3 a constructive geometry statement involving inequalities consists of a set of constructions, a set of additional hypothesis and a conclusion set. In the general input form some constructions may implicitly contain some additional hypothesis (as in the case of *incircle*, *bisectint*, etc.)

Our method consists of two main steps:

1. Simplifying the original proof problem using the area method, which consists of:

- rewriting the original constructions using only constructions accepted by the area method (see section 5.5.2);
- rewriting the additional hypothesis and the conclusions using geometric quantities recognized by the area method;
- simplifying the resulted expressions by computing them using the area method (see section 5.5.3).

2. Solving the obtained proof problem:

- transforming the obtained expressions into algebraic form;

- finding non−degenerateconditions;

- finding an appropriate coordinate system;

- expressing the proof problem in a form recognized by the CAD algorithm (universally quantified logical formula);

- eliminating the universal quantifier from the obtained formula using the CAD algorithm.

### *7.2.2 Problem Input*

As in the case of the other provers we use the *Theorema* construct *Proposition* to specify the theorem that should be proved. The implemented constructions and conclusions were presented in section 5.2.2.

Example:

```
Proposition["Cad_incircle", any[O, A, B, C, P, X, Y, O1],
  point[A, B, O] ∧
    incircle[O, A, B, C, P] ∧
    midpoint[X, A, B] ∧
    midpoint[Y, A, C] ∧
    inter[O1, tline[X, A, B], tline[Y, A, C]] ∧
    circle[O1, A]
   ⇒ inequation[4 ∗ seglengthsq[O, P] ≤ seglengthsq[O1, A]]]
```

In the implementation we use the *Theorema* function *Prove* using the *GeometryProver* with option *Method*→"AreaCAD".

Example:

```
Prove[Proposition["Cad_incircle"], by → GeometryProver,
  ProverOptions → {Method → "AreaCAD"}]
```

### *7.2.3 Step 1: Simplifying Expressions by the Area Method*

As a first step of our method we have to reduce the original proof problem to an equivalent one that makes use only of the points that were introduced as free points. We compute the expressions that appear as additional hypothesis and conclusions eliminating the constructed points using the elimination steps of area method. Thus we obtain some expressions that depend on the free points and some rational constants. These constants are introduced by the semi-bound points and appear when translating the original constructions to constructions accepted by the area prover.

Thus our original proof problem of finding non-degenerate conditions *N* such that

$$\underset{\substack{A_i \\ i=1,\ldots,p}}{\forall}\ \underset{\substack{B_j \\ j=1,\ldots,m}}{\forall}\ (\text{point}[A_1, \ldots, A_p] \bigwedge C_1[A_1, \ldots, A_p, B_1] \bigwedge \ldots \bigwedge$$
$$C_m[A_1, \ldots, A_p, B_1, \ldots, B_m] \bigwedge H_1[A_1, \ldots, A_p, B_1, \ldots, B_m] \bigwedge$$
$$\ldots \bigwedge H_n[A_1, \ldots, A_p, B_1, \ldots, B_m] \bigwedge N[A_1, \ldots, A_p, B_1, \ldots, B_m] \Rightarrow$$
$$G_1[A_1, \ldots, A_p, B_1, \ldots, B_m] \bigwedge \ldots \bigwedge G_k[A_1, \ldots, A_p, B_1, \ldots, B_m])$$

is transformed to the equivalent problem of finding non-degenerate conditions *N'* such that

$$\underset{\substack{A_i \\ i=1,\ldots,p}}{\forall}\ \underset{\substack{r_j \\ j=1,\ldots,q}}{\forall}\ ((H')_1[A_1, \ldots, A_p, r_1, \ldots, r_q] \bigwedge \ldots \bigwedge$$
$$(H')_n[A_1, \ldots, A_p, r_1, \ldots, r_q] \bigwedge N'[A_1, \ldots, A_p, r_1, \ldots, r_q] \Rightarrow$$
$$(G')_1[A_1, \ldots, A_p, r_1, \ldots, r_q] \bigwedge \ldots \bigwedge (G')_k[A_1, \ldots, A_p, r_1, \ldots, r_q])$$

This step is performed by the function *AreaCAD* making use of the functions of the area prover included in the package *Area*.

```
AreaCAD[•lf["CADArea", constructions_ ⇒ expr_, •finfo["VerArea"]],
    dummy___] := Module[{var, l, verfyndg = True, psitList = {},
      constr = DeleteCases[constructions, {"hyp", ___}],
      hyp = Cases[constructions, {"hyp", ___}]},
    hyp = Table[hyp[[i, 2, 1]], {i, Length[hyp]}];
    hyp = Join[hyp, expr];
    var = {};
    Do[If[geomobj[[i, 1]] =!= "" && geomobj[[i, 2]] =!= "point",
        AppendTo[var, geomobj[[i, 1]]]], {i, Length[geomobj]}];
    Do[l = FormulaVariables[DeleteCases[Variables[hyp[[i]]], r__ ]];
      If[Intersection[l, var] === {},
        AppendTo[psitList, FinalProofStep[Prinfo["CadReady", {} ->
            •lf[¢AreaComputed, hyp[[i]], •finfo["Compute"]]], ¢Proved ]],
        AppendTo[psitList,
          ProofStep[Prinfo["Dummy", {}⟶{}],
            Psit[•lf["AreaCAD", constr ⇒ hyp[[i]],
              •finfo["CAD", verifyndg, True]], dummy]]];
        verifyndg = False], {i, Length[hyp]}];
    ProofStep @@ PrependTo[psitList, Prinfo["CADExpr", {}⟶{}]]];

Area[•lf["AreaCAD",  hyp_ ⇒ concl_, •finfo["CAD", verifyndg_, uL2_]],
    dummy___] := Module[{}, optndg = verifyndg; useL2 = uL2;
    cad = True;  optelim = False; compexpr = True; optprint = False;
    ProofStep[Prinfo["CADAreaProblem",
      {}⟶•lf["CAD", hyp ⇒ concl, •finfo["Compute", concl]], optndg],
      Psit[•lf[¢AreaElim, hyp ⇒ concl //. OrdPoints,
        •finfo["Compute", concl]], dummy]]]
```

Example:

In our previous example we have to prove that:

$\{A, B, O\}$ free points
$\alpha_A\ B \perp A\ O$ and $\alpha_A \in A\ O$ with ndg. $\{A \neq O\}$
$\alpha 1_A\ B \parallel B\ \alpha_A$ and $\frac{\overline{B\ \alpha 1_A}}{\overline{B\ \alpha_A}} = 2$ with ndg. $\{B \neq \alpha_A\}$
$\alpha_B\ A \perp B\ O$ and $\alpha_B \in B\ O$ with ndg. $\{B \neq O\}$
$\alpha 1_B\ A \parallel A\ \alpha_B$ and $\frac{\overline{A\ \alpha 1_B}}{\overline{A\ \alpha_B}} = 2$ with ndg. $\{A \neq \alpha_B\}$
$C = A\ \alpha 1_A \bigcap B\ \alpha 1_B$ with ndg. $\{A \neq \alpha 1_A, B \neq \alpha 1_B, A\ \alpha 1_A \nparallel B\ \alpha 1_B\}$
$P\ O \perp A\ B$ and $P \in A\ B$ with ndg. $\{A \neq B\}$
$\alpha_C\ O \perp A\ C$ and $\alpha_C \in A\ C$ with ndg. $\{A \neq C\}$
$X\ A \parallel A\ B$ and $\frac{\overline{A\ X}}{\overline{A\ B}} = \frac{1}{2}$ with ndg. $\{A \neq B\}$
$Y\ A \parallel A\ C$ and $\frac{\overline{A\ Y}}{\overline{A\ C}} = \frac{1}{2}$ with ndg. $\{A \neq C\}$
$Y\ \gamma_{O1} \perp Y\ A$ and $\frac{\overline{Y\ \gamma_{O1}}}{\overline{Y\ A}} = r_{21}$ with ndg. $\{Y \neq A\}$
$X\ \beta_{O1} \perp X\ A$ and $\frac{\overline{X\ \beta_{O1}}}{\overline{X\ A}} = r_{20}$ with ndg. $\{X \neq A\}$
$O1 = X\ \beta_{O1} \bigcap Y\ \gamma_{O1}$ with ndg. $\{X \neq \beta_{O1}, Y \neq \gamma_{O1}, \beta_{O1}\ X \nparallel \gamma_{O1}\ Y\}$

with additional constraints

$\bullet R_{\{A,P,P,B\}} > 0$
$\bullet R_{\{A,\alpha_C,\alpha_C,C\}} > 0$

imply that $-\overline{A\ O1}^2 + 4\,\overline{P\ O}^2 \leq 0$

which is reduced to proving that

========== Output Generated by the Program ==========

$$\forall_{A,B,O}\left(\frac{\bullet L_{\{A,B\}}^2 + \bullet L_{\{A,O\}}^2 + -\bullet L_{\{B,O\}}^2}{\bullet L_{\{A,B\}}^2 + -\bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2} > 0 \bigwedge \right.$$
$$(-\bullet L_{\{A,B\}}^4 + \bullet L_{\{A,O\}}^4 + 2\bullet L_{\{A,B\}}^2 \bullet L_{\{B,O\}}^2 + -\bullet L_{\{B,O\}}^4) \Big/$$
$$\left(\bullet L_{\{A,B\}}^4 + (\bullet L_{\{A,O\}}^2 + -\bullet L_{\{B,O\}}^2)^2 + -2\bullet L_{\{A,B\}}^2 (\bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2)\right) > 0 \Rightarrow$$
$$-\left(\bullet L_{\{A,B\}}^{12} + -4\bullet L_{\{A,B\}}^{10} (\bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2) + \right.$$
$$64 (\bullet L_{\{A,O\}}^4 + -\bullet L_{\{B,O\}}^4)^2 \bullet S_{\{A,B,O\}}^2 + -256\bullet L_{\{A,B\}}^2$$
$$(\bullet L_{\{A,O\}}^6 + \bullet L_{\{A,O\}}^4 \bullet L_{\{B,O\}}^2 + \bullet L_{\{A,O\}}^2 \bullet L_{\{B,O\}}^4 + \bullet L_{\{B,O\}}^6) \bullet S_{\{A,B,O\}}^2 +$$
$$-4\bullet L_{\{A,B\}}^6 (\bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2) (\bullet L_{\{A,O\}}^4 + \bullet L_{\{B,O\}}^4 + 72\bullet S_{\{A,B,O\}}^2) +$$
$$\bullet L_{\{A,B\}}^8 (6\bullet L_{\{A,O\}}^4 + 8\bullet L_{\{A,O\}}^2 \bullet L_{\{B,O\}}^2 + 6\bullet L_{\{B,O\}}^4 + 80\bullet S_{\{A,B,O\}}^2) +$$
$$\bullet L_{\{A,B\}}^4 (\bullet L_{\{A,O\}}^8 + \bullet L_{\{B,O\}}^8 + 544\bullet L_{\{A,O\}}^2 \bullet L_{\{B,O\}}^2 \bullet S_{\{A,B,O\}}^2 +$$
$$\left.\left. 400\bullet L_{\{B,O\}}^4 \bullet S_{\{A,B,O\}}^2 + 2\bullet L_{\{A,O\}}^4 (\bullet L_{\{B,O\}}^4 + 200\bullet S_{\{A,B,O\}}^2))\right) \Big/\right.$$
$$\left.\left(32\bullet L_{\{A,B\}}^2 (-\bullet L_{\{A,B\}}^2 + \bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2)^2 \bullet S_{\{A,B,O\}}^2\right) \leq 0\right)$$

========== End of Generated Output ==========

### *7.2.4 Step 2: Solving theProblem Using the CAD Method*

For the CAD algorithm we have to transform the obtained problem into polynomial form. To obtain as simple expressions as possible we choose for the origin of the coordinate system the point with the highest number of occurrences in the expressions. The next point is taken as being on the x-axis. We may take this point as having coordinates {1,0}. This way the algebraic expressions become even simpler.

This processing is done by the *AreaCAD* function in step *CadCoord:*

```
AreaCAD[•lf["CadCoord", ___, finfo_], dummy___] := Module[{l, points, ff},
    points = FormulaVariables[exprlist];
    Do[If[Head[x_points[[i]]] =!= Subscript || x_points[[i]][[2]] =!= points[[i]],
      x_points[[i]] =.]; If[Head[y_points[[i]]] =!= Subscript ||
       y_points[[i]][[2]] =!= points[[i]], y_points[[i]] =.], {i, Length[points]}];
    subst = IndependentElem;
    If[UnsameQ[subst, {}], exprlist = exprlist /. subst];
    exprlist = exprlist //. ElimElemCad;
    If[xaxpoints === {}, FindCadAx[points]];
    exprlist = System`Simplify[exprlist //. CoordSyst];
    If[subst === {}, ff = •finfo["ax", xaxpoints], ff = •finfo["subst", subst]];
    If[xaxpoints === {} && subst === {}, ProofStep[Prinfo["Dummy",
        {} ⟶ {}], Psit[•lf["CadForm", exprlist, finfo], dummy]],
      ProofStep[Prinfo["CadCoord", {} ⟶ •lf[UniqueLabel[],
          ™ForAll[•range @@ Map[•simpleRange, Variables[exprlist]],
            True, And @@ PrintExpr["h"] ⇒ And @@ PrintExpr["c"]] /.
          ToVar, ff]], Psit[•lf["CadForm", exprlist, finfo], dummy]]]];
```

For our example we chose $\{x_A \to 0, y_A \to 0, x_O \to 1, y_O \to 0\}$ and obtain

$$\forall_{x_B, y_B} \left( \frac{x_B}{-x_B + x_B^2 + y_B^2} > 0 \bigwedge \frac{((-1)+x_B)\,x_B}{y_B^2} > 0 \Rightarrow \right.$$
$$-(-4\,x_B^7 + x_B^8 + -4\,x_B\,y_B^4\,((-31) + y_B^2) + -4\,x_B^5\,(1 + 3\,y_B^2) + -4\,x_B^3\,y_B^2$$
$$(2 + 3\,y_B^2) + x_B^6\,(6 + 4\,y_B^2) + y_B^4\,((-63) + 2\,y_B^2 + y_B^4) +$$
$$x_B^4\,(1 + 14\,y_B^2 + 6\,y_B^4) + 2\,x_B^2\,(y_B^2 + -27\,y_B^4 + 2\,y_B^6))/$$
$$\left. (8\,((-1) + x_B)^2\,y_B^2\,(x_B^2 + y_B^2)) \le 0 \right)$$

If the denominator of the conclusion not being 0 does not result from the hypothesis this should be considered a non−degenerate condition and added to the hypothesis. At the end of the proof the user has to analyze wether the obtained condition is a non−degeneratecondition or it introduces some essentially new hypothesis.

If the simplified expressions contain square roots even powers of subexpressions should be extracted from the square roots adding the necessary conditions.

As a final preparing step all the universally quantified variables of the expression are renamed using a predefined set of variables.

This processing is done by the *AreaCAD* function in step *CadForm:*

```
AreaCAD[•lf["CadForm", ___, finfo_], dummy___] :=
  Block[{cadhyp, var1}, Module[{cadvar, allvar, cadconcl},
    var1 = Variables[exprlist];
    cadhyp = u1 <= u1;
    exprlist = exprlist //. ElimSqrt[CollectExpr["h"]];
    cadhyp = And @@ Union[List @@ cadhyp];
    If[Head[cadhyp] === And,
      Do[AddIneq[cadhyp[[i]], "h"], {i, 1, Length[cadhyp] − 1}]];
    cadvar = Union[Flatten[Table[If[oplist[[i]] === "h",
               Variables[exprlist[[i]]], {}], {i, Length[exprlist]}]]];
    allvar = var1;
    var1 = Complement[var1, cadvar];
    cadhyp = CollectExpr["h"];
    If[cadvar === {}, cadvar = {var1[[1]]}];
    Do[cadhyp = cadhyp && var1[[i]] ≤ var1[[i]], {i, Length[var1]}];
    Do[cadhyp = cadhyp && cadvar[[i]] ≠ 0, {i, Length[cadvar]}];
    cadconcl = CollectExpr["c"] //. CoordCadU[cadvar];
    cadhyp = cadhyp //. CoordCadU[cadvar];
    allvar = allvar /. CoordCadU[cadvar];
    ProofStep[
      Prinfo["CadForm", {}⟶•lf[UniqueLabel[], ™ForAll[•range @@
                 Map[•simpleRange, cadvar /. CoordCadU[cadvar]] /. ToVar,
               True, (cadhyp /. ToVar) ⇒ (cadconcl /. ToVar)],
             •finfo[CoordCadU[cadvar] /. ToVar]]],
      Psit[•lf["CadCalc", cadhyp ⇒ cadconcl, •finfo[allvar]], dummy]]];
```

For our example with $\{x_B \to u_1, y_B \to u_2\}$ we obtain

$$\forall_{u_1,u_2} \Big( \frac{u_1}{-u_1+u_1^2+u_2^2} > 0 \bigwedge \frac{((-1)+u_1)\,u_1}{u_2^2} > 0 \bigwedge u_1 \neq 0 \bigwedge u_2 \neq 0 \Rightarrow$$
$$-(-4\,u_1^7 + u_1^8 + -4\,u_1\,u_2^4\,((-31)+u_2^2) + -4\,u_1^5\,(1+3\,u_2^2) +$$
$$-4\,u_1^3\,u_2^2\,(2+3\,u_2^2) + u_1^6\,(6+4\,u_2^2) + u_2^4\,((-63)+2\,u_2^2+u_2^4) +$$
$$u_1^4\,(1+14\,u_2^2+6\,u_2^4) + 2\,u_1^2\,(u_2^2 + -27\,u_2^4 + 2\,u_2^6))/$$
$$(8\,((-1)+u_1)^2\,u_2^2\,(u_1^2+u_2^2)) \leq 0\Big)$$

An implication is true if its conclusion is true or if the hypotheses are contradictory. In this second case we get no information on the logical value of the conclusion. Thus we check first the consistency of the hypothesis by an existential quantifier elimination using the *Mathematica Experimental‘ExistsRealQ* function. Then the validity of the universally quantified expression is checked using the *Experimental‘Resolve* function.

### *7.2.5 Output of the Proof*

In a similar way to all the other proof methods a new *Mathematica* notebook is generated with the proof described in all its detail. Making use of the grouped cells the proof may be followed easily, one may look only at the main steps first and visualize the details afterwards.

Example:

========== Output Generated by the Program ==========

Prove:

(Proposition (Cad_incircle))

$\underset{O,A,B,C,P,X,Y,O1}{\forall}$ (point[$A$, $B$, $O$] ∧ incircle[$O$, $A$, $B$, $C$, $P$] ∧ midpoint[$X$, $A$, $B$] ∧
midpoint[$Y$, $A$, $C$] ∧ inter[$O1$, tline[$X$, $A$, $B$], tline[$Y$, $A$, $C$]] ∧ circle[$O1$, $A$] ⇒
inequation[4 seglengthsq[$O$, $P$] ≤ seglengthsq[$O1$, $A$]])

with no assumptions.

As the proposition contains inequalities, we have to use the CAD method. We shall use the area method first to obtain a simpler input for CAD.

First we have to transform the problem in internal form for the AreaCAD. We have to prove, that constructions

{$A$, $B$, $O$} free points
$\alpha_A\ B \perp A\ O$ and $\alpha_A \in A\ O$ with ndg. {$A \neq O$}
$\alpha 1_A\ B \parallel B\ \alpha_A$ and $\frac{\overline{B\ \alpha 1_A}}{\overline{B\ \alpha_A}} = 2$ with ndg. {$B \neq \alpha_A$}
$\alpha_B\ A \perp B\ O$ and $\alpha_B \in B\ O$ with ndg. {$B \neq O$}
$\alpha 1_B\ A \parallel A\ \alpha_B$ and $\frac{\overline{A\ \alpha 1_B}}{\overline{A\ \alpha_B}} = 2$ with ndg. {$A \neq \alpha_B$}
$C = A\ \alpha 1_A \cap B\ \alpha 1_B$ with ndg. {$A \neq \alpha 1_A$, $B \neq \alpha 1_B$, $A\ \alpha 1_A \not\parallel B\ \alpha 1_B$}
$P\ O \perp A\ B$ and $P \in A\ B$ with ndg. {$A \neq B$}
$\alpha_C\ O \perp A\ C$ and $\alpha_C \in A\ C$ with ndg. {$A \neq C$}
$X\ A \parallel A\ B$ and $\frac{\overline{A\ X}}{\overline{A\ B}} = \frac{1}{2}$ with ndg. {$A \neq B$}
$Y\ A \parallel A\ C$ and $\frac{\overline{A\ Y}}{\overline{A\ C}} = \frac{1}{2}$ with ndg. {$A \neq C$}
$Y\ \gamma_{O1} \perp Y\ A$ and $\frac{\overline{Y\ \gamma_{O1}}}{\overline{Y\ A}} = r_{21}$ with ndg. {$Y \neq A$}
$X\ \beta_{O1} \perp X\ A$ and $\frac{\overline{X\ \beta_{O1}}}{\overline{X\ A}} = r_{20}$ with ndg. {$X \neq A$}
$O1 = X\ \beta_{O1} \cap Y\ \gamma_{O1}$ with ndg. {$X \neq \beta_{O1}$, $Y \neq \gamma_{O1}$, $\beta_{O1}\ X \not\parallel \gamma_{O1}\ Y$}

with additional constraints

- $R_{\{A,P,P,B\}} > 0$
- $R_{\{A,\alpha_C,\alpha_C,C\}} > 0$

imply that $-\overline{A\ O1}^2 + 4\,\overline{P\ O}^2 \leq 0$

Step 1

Now we try to obtain a simpler form of the constraints and of the conclusion using the area method.

Explanations ==== *closed cell* ====

Expression 1

Using constructions
{$A$, $B$, $O$} free points
$\alpha_A\ B \perp A\ O$ and $\alpha_A \in A\ O$ with ndg. {$A \neq O$}
$\alpha 1_A\ B \parallel B\ \alpha_A$ and $\frac{\overline{B\ \alpha 1_A}}{\overline{B\ \alpha_A}} = 2$ with ndg. {$B \neq \alpha_A$}

$\alpha_B\ A \perp B\ O$ and $\alpha_B \in B\ O$ with ndg. $\{B \neq O\}$
$\alpha 1_B\ A \parallel A\ \alpha_B$ and $\frac{\overline{A\ \alpha 1_B}}{\overline{A\ \alpha_B}} = 2$ with ndg. $\{A \neq \alpha_B\}$
$C = A\ \alpha 1_A \cap B\ \alpha 1_B$ with ndg. $\{A \neq \alpha 1_A, B \neq \alpha 1_B, A\ \alpha 1_A \nparallel B\ \alpha 1_B\}$
$P\ O \perp A\ B$ and $P \in A\ B$ with ndg. $\{A \neq B\}$
$\alpha_C\ O \perp A\ C$ and $\alpha_C \in A\ C$ with ndg. $\{A \neq C\}$
$X\ A \parallel A\ B$ and $\frac{\overline{A\ X}}{\overline{A\ B}} = \frac{1}{2}$ with ndg. $\{A \neq B\}$
$Y\ A \parallel A\ C$ and $\frac{\overline{A\ Y}}{\overline{A\ C}} = \frac{1}{2}$ with ndg. $\{A \neq C\}$
$Y\ \gamma_{O1} \perp Y\ A$ and $\frac{\overline{Y\ \gamma_{O1}}}{\overline{Y\ A}} = r_{21}$ with ndg. $\{Y \neq A\}$
$X\ \beta_{O1} \perp X\ A$ and $\frac{\overline{X\ \beta_{O1}}}{\overline{X\ A}} = r_{20}$ with ndg. $\{X \neq A\}$
$O1 = X\ \beta_{O1} \cap Y\ \gamma_{O1}$ with ndg. $\{X \neq \beta_{O1}, Y \neq \gamma_{O1}, \beta_{O1}\ X \nparallel \gamma_{O1}\ Y\}$

we have to compute $\bullet R_{\{A,P,P,B\}}$

Eliminating point $P$:

We eliminate point $P$ using construction: $P\ O \perp A\ B$ and $P \in A\ B$ with ndg. $\{A \neq B\}$

We obtain:

$$\frac{\bullet P_{\{B,A,O\}}}{\bullet P_{\{A,B,O\}}}$$

All the constructed points have been eliminated, so we have to rewrite the expression in area coordinates of the free points:

We obtain:

$$\frac{\bullet L_{\{A,B\}}^{\ 2} + \bullet L_{\{A,O\}}^{\ 2} + -\bullet L_{\{B,O\}}^{\ 2}}{\bullet L_{\{A,B\}}^{\ 2} + -\bullet L_{\{A,O\}}^{\ 2} + \bullet L_{\{B,O\}}^{\ 2}}$$

We try to simplify further this expression using the possible relations between areas and lengths of segments:

After simplification we obtain

$$\frac{\bullet L_{\{A,B\}}^{\ 2} + \bullet L_{\{A,O\}}^{\ 2} + -\bullet L_{\{B,O\}}^{\ 2}}{\bullet L_{\{A,B\}}^{\ 2} + -\bullet L_{\{A,O\}}^{\ 2} + \bullet L_{\{B,O\}}^{\ 2}}$$

Thus

$$\bullet R_{\{A,P,P,B\}} = \frac{\bullet L_{\{A,B\}}^{\ 2} + \bullet L_{\{A,O\}}^{\ 2} + -\bullet L_{\{B,O\}}^{\ 2}}{\bullet L_{\{A,B\}}^{\ 2} + -\bullet L_{\{A,O\}}^{\ 2} + \bullet L_{\{B,O\}}^{\ 2}}$$

Expression 2

Using constructions

$\{A, B, O\}$ free points
$\alpha_A\ B \perp A\ O$ and $\alpha_A \in A\ O$ with ndg. $\{A \neq O\}$
$\alpha 1_A\ B \parallel B\ \alpha_A$ and $\frac{\overline{B\ \alpha 1_A}}{\overline{B\ \alpha_A}} = 2$ with ndg. $\{B \neq \alpha_A\}$
$\alpha_B\ A \perp B\ O$ and $\alpha_B \in B\ O$ with ndg. $\{B \neq O\}$
$\alpha 1_B\ A \parallel A\ \alpha_B$ and $\frac{\overline{A\ \alpha 1_B}}{\overline{A\ \alpha_B}} = 2$ with ndg. $\{A \neq \alpha_B\}$
$C = A\ \alpha 1_A \cap B\ \alpha 1_B$ with ndg. $\{A \neq \alpha 1_A, B \neq \alpha 1_B, A\ \alpha 1_A \nparallel B\ \alpha 1_B\}$
$P\ O \perp A\ B$ and $P \in A\ B$ with ndg. $\{A \neq B\}$
$\alpha_C\ O \perp A\ C$ and $\alpha_C \in A\ C$ with ndg. $\{A \neq C\}$
$X\ A \parallel A\ B$ and $\frac{\overline{A\ X}}{\overline{A\ B}} = \frac{1}{2}$ with ndg. $\{A \neq B\}$

$Y\,A \parallel A\,C$ and $\frac{\overline{A\,Y}}{\overline{A\,C}}=\frac{1}{2}$ with ndg. $\{A \neq C\}$
$Y\,\gamma_{O1} \perp Y\,A$ and $\frac{\overline{Y\,\gamma_{O1}}}{\overline{Y\,A}}=r_{21}$ with ndg. $\{Y \neq A\}$
$X\,\beta_{O1} \perp X\,A$ and $\frac{\overline{X\,\beta_{O1}}}{\overline{X\,A}}=r_{20}$ with ndg. $\{X \neq A\}$
$O1 = X\,\beta_{O1} \cap Y\,\gamma_{O1}$ with ndg. $\{X \neq \beta_{O1}, Y \neq \gamma_{O1}, \beta_{O1}\,X \nparallel \gamma_{O1}\,Y\}$

we have to compute $\bullet R_{\{A,\alpha_C,\alpha_C,C\}}$

==== Computing expression 2 ====

Thus

$$\bullet R_{\{A,\alpha_C,\alpha_C,C\}} = \left(-\bullet L_{\{A,B\}}^4 + \bullet L_{\{A,O\}}^4 + 2\,\bullet L_{\{A,B\}}^2\,\bullet L_{\{B,O\}}^2 + -\bullet L_{\{B,O\}}^4\right) \Big/ \left(\bullet L_{\{A,B\}}^4 + \left(\bullet L_{\{A,O\}}^2 + -\bullet L_{\{B,O\}}^2\right)^2 + -2\,\bullet L_{\{A,B\}}^2\left(\bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2\right)\right)$$

Expression 3

Using constructions

$\{A, B, O\}$ free points
$\alpha_A\,B \perp A\,O$ and $\alpha_A \in A\,O$ with ndg. $\{A \neq O\}$
$\alpha 1_A\,B \parallel B\,\alpha_A$ and $\frac{\overline{B\,\alpha 1_A}}{\overline{B\,\alpha_A}}=2$ with ndg. $\{B \neq \alpha_A\}$
$\alpha_B\,A \perp B\,O$ and $\alpha_B \in B\,O$ with ndg. $\{B \neq O\}$
$\alpha 1_B\,A \parallel A\,\alpha_B$ and $\frac{\overline{A\,\alpha 1_B}}{\overline{A\,\alpha_B}}=2$ with ndg. $\{A \neq \alpha_B\}$
$C = A\,\alpha 1_A \cap B\,\alpha 1_B$ with ndg. $\{A \neq \alpha 1_A, B \neq \alpha 1_B, A\,\alpha 1_A \nparallel B\,\alpha 1_B\}$
$P\,O \perp A\,B$ and $P \in A\,B$ with ndg. $\{A \neq B\}$
$\alpha_C\,O \perp A\,C$ and $\alpha_C \in A\,C$ with ndg. $\{A \neq C\}$
$X\,A \parallel A\,B$ and $\frac{\overline{A\,X}}{\overline{A\,B}}=\frac{1}{2}$ with ndg. $\{A \neq B\}$
$Y\,A \parallel A\,C$ and $\frac{\overline{A\,Y}}{\overline{A\,C}}=\frac{1}{2}$ with ndg. $\{A \neq C\}$
$Y\,\gamma_{O1} \perp Y\,A$ and $\frac{\overline{Y\,\gamma_{O1}}}{\overline{Y\,A}}=r_{21}$ with ndg. $\{Y \neq A\}$
$X\,\beta_{O1} \perp X\,A$ and $\frac{\overline{X\,\beta_{O1}}}{\overline{X\,A}}=r_{20}$ with ndg. $\{X \neq A\}$
$O1 = X\,\beta_{O1} \cap Y\,\gamma_{O1}$ with ndg. $\{X \neq \beta_{O1}, Y \neq \gamma_{O1}, \beta_{O1}\,X \nparallel \gamma_{O1}\,Y\}$

we have to compute $-\overline{A\,O1}^2 + 4\,\overline{P\,O}^2$

=== Computing expression 3 ===

Thus

$$\begin{aligned}
&-\bullet P_{\{A,O1,A\}} + 4\,\bullet P_{\{O,P,O\}} \\
&= -\Big(\bullet L_{\{A,B\}}^{12} + -4\,\bullet L_{\{A,B\}}^{10}\left(\bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2\right) + \\
&\quad 64\left(\bullet L_{\{A,O\}}^4 + -\bullet L_{\{B,O\}}^4\right)^2 \bullet S_{\{A,B,O\}}^2 + -256\,\bullet L_{\{A,B\}}^2 \\
&\quad \left(\bullet L_{\{A,O\}}^6 + \bullet L_{\{A,O\}}^4\,\bullet L_{\{B,O\}}^2 + \bullet L_{\{A,O\}}^2\,\bullet L_{\{B,O\}}^4 + \bullet L_{\{B,O\}}^6\right)\bullet S_{\{A,B,O\}}^2 + \\
&\quad -4\,\bullet L_{\{A,B\}}^6\left(\bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2\right)\left(\bullet L_{\{A,O\}}^4 + \bullet L_{\{B,O\}}^4 + 72\,\bullet S_{\{A,B,O\}}^2\right) + \\
&\quad \bullet L_{\{A,B\}}^8\left(6\,\bullet L_{\{A,O\}}^4 + 8\,\bullet L_{\{A,O\}}^2\,\bullet L_{\{B,O\}}^2 + 6\,\bullet L_{\{B,O\}}^4 + 80\,\bullet S_{\{A,B,O\}}^2\right) + \\
&\quad \bullet L_{\{A,B\}}^4\left(\bullet L_{\{A,O\}}^8 + \bullet L_{\{B,O\}}^8 + 544\,\bullet L_{\{A,O\}}^2\,\bullet L_{\{B,O\}}^2\,\bullet S_{\{A,B,O\}}^2 + \right. \\
&\qquad \left.\left. 400\,\bullet L_{\{B,O\}}^4\,\bullet S_{\{A,B,O\}}^2 + 2\,\bullet L_{\{A,O\}}^4\left(\bullet L_{\{B,O\}}^4 + 200\,\bullet S_{\{A,B,O\}}^2\right)\right)\right) \Big/ \\
&\left(32\,\bullet L_{\{A,B\}}^2\left(-\bullet L_{\{A,B\}}^2 + \bullet L_{\{A,O\}}^2 + \bullet L_{\{B,O\}}^2\right)^2 \bullet S_{\{A,B,O\}}^2\right)
\end{aligned}$$

Step 2

Our initial problem becomes

(1)

$$\forall_{A,B,O}\left(\frac{\bullet L_{\{A,B\}}^{2}+\bullet L_{\{A,O\}}^{2}+-\bullet L_{\{B,O\}}^{2}}{\bullet L_{\{A,B\}}^{2}+-\bullet L_{\{A,O\}}^{2}+\bullet L_{\{B,O\}}^{2}}>0\bigwedge\right.$$
$$(-\bullet L_{\{A,B\}}^{4}+\bullet L_{\{A,O\}}^{4}+2\bullet L_{\{A,B\}}^{2}\bullet L_{\{B,O\}}^{2}+-\bullet L_{\{B,O\}}^{4})\Big/$$
$$\left(\bullet L_{\{A,B\}}^{4}+(\bullet L_{\{A,O\}}^{2}+-\bullet L_{\{B,O\}}^{2})^{2}+-2\bullet L_{\{A,B\}}^{2}(\bullet L_{\{A,O\}}^{2}+\bullet L_{\{B,O\}}^{2})\right)>0\Rightarrow$$
$$-\left(\bullet L_{\{A,B\}}^{12}+-4\bullet L_{\{A,B\}}^{10}(\bullet L_{\{A,O\}}^{2}+\bullet L_{\{B,O\}}^{2})+\right.$$
$$64(\bullet L_{\{A,O\}}^{4}+-\bullet L_{\{B,O\}}^{4})^{2}\bullet S_{\{A,B,O\}}^{2}+-256\bullet L_{\{A,B\}}^{2}$$
$$(\bullet L_{\{A,O\}}^{6}+\bullet L_{\{A,O\}}^{4}\bullet L_{\{B,O\}}^{2}+\bullet L_{\{A,O\}}^{2}\bullet L_{\{B,O\}}^{4}+\bullet L_{\{B,O\}}^{6})\bullet S_{\{A,B,O\}}^{2}+$$
$$-4\bullet L_{\{A,B\}}^{6}(\bullet L_{\{A,O\}}^{2}+\bullet L_{\{B,O\}}^{2})(\bullet L_{\{A,O\}}^{4}+\bullet L_{\{B,O\}}^{4}+72\bullet S_{\{A,B,O\}}^{2})+$$
$$\bullet L_{\{A,B\}}^{8}(6\bullet L_{\{A,O\}}^{4}+8\bullet L_{\{A,O\}}^{2}\bullet L_{\{B,O\}}^{2}+6\bullet L_{\{B,O\}}^{4}+80\bullet S_{\{A,B,O\}}^{2})+$$
$$\bullet L_{\{A,B\}}^{4}(\bullet L_{\{A,O\}}^{8}+\bullet L_{\{B,O\}}^{8}+544\bullet L_{\{A,O\}}^{2}\bullet L_{\{B,O\}}^{2}\bullet S_{\{A,B,O\}}^{2}+$$
$$\left.400\bullet L_{\{B,O\}}^{4}\bullet S_{\{A,B,O\}}^{2}+2\bullet L_{\{A,O\}}^{4}(\bullet L_{\{B,O\}}^{4}+200\bullet S_{\{A,B,O\}}^{2}))\right)\Big/$$
$$\left.\left(32\bullet L_{\{A,B\}}^{2}(-\bullet L_{\{A,B\}}^{2}+\bullet L_{\{A,O\}}^{2}+\bullet L_{\{B,O\}}^{2})^{2}\bullet S_{\{A,B,O\}}^{2}\right)\leq 0\right)$$

Choosing a cartesian coordinate system with the x–axis$\{A, O\}$
and performing substitution: $\{x_A \to 0, y_A \to 0, x_O \to 1, y_O \to 0\}$
the problem becomes:

(2)

$$\forall_{x_B,y_B}\left(\frac{x_B}{-x_B+x_B^{2}+y_B^{2}}>0\bigwedge\frac{((-1)+x_B)\,x_B}{y_B^{2}}>0\Rightarrow\right.$$
$$-(-4x_B^{7}+x_B^{8}+-4x_B\,y_B^{4}((-31)+y_B^{2})+-4x_B^{5}(1+3y_B^{2})+-4x_B^{3}\,y_B^{2}$$
$$(2+3y_B^{2})+x_B^{6}(6+4y_B^{2})+y_B^{4}((-63)+2y_B^{2}+y_B^{4})+$$
$$x_B^{4}(1+14y_B^{2}+6y_B^{4})+2x_B^{2}(y_B^{2}+-27y_B^{4}+2y_B^{6}))/$$
$$\left.(8((-1)+x_B)^{2}\,y_B^{2}(x_B^{2}+y_B^{2}))\leq 0\right)$$

If there are square roots of squares in this expression we eliminate them adding the necessary conditions. After performing substitution $\{x_B \to u_1, y_B \to u_2\}$
the problem becomes:

(3)

$$\forall_{u_1,u_2}\left(\frac{u_1}{-u_1+u_1^{2}+u_2^{2}}>0\bigwedge\frac{((-1)+u_1)\,u_1}{u_2^{2}}>0\bigwedge u_1\neq 0\bigwedge u_2\neq 0\Rightarrow\right.$$
$$-(-4u_1^{7}+u_1^{8}+-4u_1\,u_2^{4}((-31)+u_2^{2})+-4u_1^{5}(1+3u_2^{2})+$$
$$-4u_1^{3}\,u_2^{2}(2+3u_2^{2})+u_1^{6}(6+4u_2^{2})+u_2^{4}((-63)+2u_2^{2}+u_2^{4})+$$
$$u_1^{4}(1+14u_2^{2}+6u_2^{4})+2u_1^{2}(u_2^{2}+-27u_2^{4}+2u_2^{6}))/$$
$$\left.(8((-1)+u_1)^{2}\,u_2^{2}(u_1^{2}+u_2^{2}))\leq 0\right)$$

Applying the CAD algorithm to this expression, we obtain that the proposition is true.

□

========== End of Generated Output ==========

# 8 Systematic Exploration of Geometric Configurations

## *8.1 Proving ⟹ exploring*

Theory exploration is an important new paradigm in computer-supported formal mathematics which was introduced in [2] (see also [3]). In this thesis we consider theory exploration for the special case of geometric theories. In this case theory exploration has a well-defined meaning and certain degree of completeness of exploration can be achieved. Given a set of geometric constructions, there are many properties of the resulting geometric elements (points, lines, segments, triangles, etc.) that can be proved. There are some properties that are well-known but there might also be some hidden properties that are not known so far. A good way for discovering new theorems is generating all the properties of a given configuration and then analyzing which of these properties might be unknown so far. By *systematic exploration* of a geometric configuration we mean finding, step by step, all the properties that result from the given constructions. For the moment we are concerned only of some *basic properties*: collinear points, parallel and perpendicular lines, segments with proportional length and triangles with proportional area. Geometry theorem discovering by generating all possible properties using deductive databases is presented in [16] and implemented in Geometry Expert [15]. In contrast to [16], where a pre-assembled knowledge base has been used for the systematic exploration we manage to generate the complete knowledge base of explored theorems fully automatically.

In our implementation we use the newly introduced *Theorema* function *Explore* for a systematic exploration of more and more complex geometric configurations. The only implemented exploration method is based on area method.

Using the area method we can systematically explore given geometric configurations [37]. Starting from an assumption list that specifies some constructions many theorems concerning parallel and perpendicular lines, segments with proportional lengths and triangles with proportional areas are obtained. As a next step further constructions can be specified in a new assumption list and the exploration may continue without recomputing any of the results already obtained.

## *8.2 Exploring by AreaExplore*

### *8.2.1 Problem Input*

We use the *Theorema* construct *Assumptions* (presented in section 5.2.1) to specify the geometric configuration to be analyzed. These assumptions may contain any number of constructions, each one of these introducing a new point using only previously defined points. If necessary graphic elements that should appear on the graphic representation (for lines and circles that exist, but have not been used explicitly in the constructions) may also be included.

Example:

```
Assumptions["Trapezoid1", any[A, B, C, D, M, N],
  triangle[A, B, C] ∧
    pon[D, pline[C, A, B]] ∧
    line[B, C] ∧
    line[A, D] ∧
    midpoint[M, A, D] ∧
    midpoint[N, B, C] ∧
    line[M, N]]
```

We use the *Theorema* function *Explore* (presented in section 5.1.4) with option *by→AreaExplore* to initiate the systematic exploration of the given configuration. As *ExplorerOptions* we may use:

- *GraphicPoints* with implicit value {}, to specify a knowledge base with coordinates of the free points used when displaying the graph of the given geometric configuration.

- *RelatedPoints* with implicit value {}, to specify a set of points. If this option is used the signed areas and Pithagoras differences of the newly introduced points will be computed only for triangles formed by the new point and the enumerated points. This is possible only for points that will not be used further in constructing new points. The use of this option can speed up computation, but some properties will be lost.

Function *Explore* returns a *KnowledgeBase* object that contains the explored graphic configuration and all the information obtained during exploration, including the discovered properties and some information necessary for further exploration steps. If option *producing* is present the new knowledge base will have this label, otherwise a label is generated by adding "_kb" to the label of the assumptions.

When exploring an initial assumptions set only option *by* and, perhaps, *producing* is necessary.

Example:

```
kb1 =
    Explore[Assumptions["Trapezoid1"], by → AreaExplore, producing → Tr1];
```

After the given knowledge base having been explored we may specify further constructions completing the geometric configuration using a new *Assumptions* construct. In this case the previously obtained knowledge base has to be specified in the *Explore* function call by option *using*.

Example:

```
Assumptions["Trapezoid2", any[A, B, C, D, M, N, X, P, Q, S],
  inter[X, line[M, N], line[B, D]] ∧
    inter[P, line[A, C], line[B, D]] ∧
    inter[Q, line[B, C], pline[P, A, B]] ∧
    inter[S, line[P, Q], line[A, D]] ∧
    line[M, C] ∧
    line[A, Q]]

kb2 = Explore[Assumptions["Trapezoid2"],
      by → AreaExplore, using → kb1, producing  −> Tr2];
```

### *8.2.2 Computing Geometric Quantities*

For each geometric construction we generate the signed areas and Pythagoras differences of all triangles that may be obtained using the newly introduced point as one vertex, and another two previously introduced points. We use one elimination step of the area method to express all these geometric quantities in geometric quantities involving only free points. It is possible to use just one elimination step as all geometric quantities involved in the expression obtained after this elimination step have already been computed .

If option *RelatedPoints* is used only signed areas and Pythagoras differences of all triangles that may be obtained using the newly introduced point as one vertex and two of the points given by this option are generated. This option is useful if we have already too many points and we are interested only of properties that result from the relation of the newly introduced point to the given points. Not being necessary to compute signed areas and Pythagoras differences for all possible triangles the computation is faster but some interesting, unexpected properties may be lost. This option cannot be used for points that are used later in constructing new points as some necessary areas or Pythagoras differences might not be included in the knowledge base.

### *8.2.3 Finding Basic Properties*

After all the needed geometric quantities having been generated for the current construction we proceed with finding new basic properties. All the newly obtained geometric quantities are analyzed and compared to each other and the already existing ones. If some geometric quantities are equal to zero or to each other or are proportional we can infer certain basic properties.

Example:

If we find among the generated elements $\bullet S_{\{B,C,N\}} \to 0$ we can infer that points *B, C* and *N* are collinear.

From

$$\bullet S_{\{A,M,N\}} \to \frac{1}{4}\left((-1) + r_D\right) \bullet S_{\{A,B,C\}}$$

and

$$\bullet S_{\{B,M,N\}} \to \frac{1}{4}\left((-1) + r_D\right) \bullet S_{\{A,B,C\}}$$

results that lines *AB* and *MN* are parallel.

Many other relations are also used to find the new properties. The list of the recognized basic properties may be extended in the future to angles, cocircle points, etc.

### *8.2.4 Building the Knowledge Base*

When applying the *Explore* function without a *using* option we start with an empty knowledge base, otherwise the knowledge base specified by *using* is the starting point. For each geometric construction of the input a new element is added to the knowledge base that contains the geometric configuration to be explored at that step, the inferred basic properties and the newly generated geometric quantities with their computed values. After the exploration of all the given constructions is finished the knowledge base is returned by the *Explore* function.

Generating a new element in the knowledge base after each constructed point makes possible to follow the properties induced by each construction, not only the final situation after all constructions given as assumptions. Deleting the last constructed point from a knowledge base is also possible.

### *8.2.5 Output of the Results*

After exploring all the given constructions a new notebook is generated with all the basic properties that could be inferred. Using the *showBy* option of explore function the result can be displayed in the working notebook or it can be suppressed. By default a new notebook is generated. We can display the intermediate results obtained after any of the constructions using the *ShowKnowledgeBase* function with the *AfterConstruction* option.

Example:

After executing the functions of the previous sections we obtain the notebook:

========== Output Generated by the Program ==========

Select construction, then press the button to view the graph

PrintGraph[■]

For any points {***A***, ***B***, ***C***, ***D***, ***M***, ***N***, ***X***, ***P***, ***Q***, ***S***} that satisfy the following constructions:

triangle[***A***, ***B***, ***C***] ∧ pon[***D***, pline[***C***, ***A***, ***B***]] ∧ line[***B***, ***C***] ∧ ,
line[***A***, ***D***] ∧ midpoint[***M***, ***A***, ***D***] ∧ midpoint[***N***, ***B***, ***C***] ∧
line[***M***, ***N***] ∧ inter[***X***, line[***M***, ***N***], line[***B***, ***D***]] ∧
inter[***P***, line[***A***, ***C***], line[***B***, ***D***]] ∧ inter[***Q***, line[***B***, ***C***], pline[***P***, ***A***, ***B***]] ∧
inter[***S***, line[***P***, ***Q***], line[***A***, ***D***]] ∧ line[***M***, ***C***] ∧ line[***A***, ***Q***]

we can prove, that:

There are no incidental points.

Collinear points:
{{***A***, ***D***, ***M***, ***S***}, {***M***, ***N***, ***X***}, {***B***, ***C***, ***N***, ***Q***}, {***B***, ***D***, ***X***, ***P***}, {***A***, ***C***, ***P***}, {***P***, ***Q***, ***S***}}

Parallel lines:

***AB*** ∥ ***CD*** ∥ ***MN*** ∥ ***PQ***

***CM*** ∥ ***AQ***

***BM*** ∥ ***DQ***

***AN*** ∥ ***CS***

***DN*** ∥ ***BS***

There are no perpendicular lines.

Segments with proportional length:

|***AD***| ≅ 2|***AM***| ≅ 2|***DM***|

|***BC***| ≅ 2|***BN***| ≅ 2|***CN***|

|***BD***| ≅ 2|***BX***| ≅ 2|***DX***|

|***AB***| ≅ 2|***MX***|

|***CD***| ≅ 2|***NX***|

|***PQ***| ≅ |***PS***| ≅ $\frac{1}{2}$|***QS***|

Triangles with proportional area:

Area[***ABC***] ≅ Area[***ABD***] ≅ 2Area[***ABM***] ≅ 2Area[***BDM***] ≅ 2Area[***ACN***] ≅ 2Area[***ABN***] ≅ 2Area[***ADX***] ≅ 2Area[***ABX***] ≅ 2Area[***BMQ***] ≅ 2Area[***ANS***] ≅ 4Area[***AMX***] ≅ 4Area[***BMX***] ≅ 4Area[***CMX***] ≅ 4Area[***DMX***]

Area[***ACD***] ≅ Area[***BCD***] ≅ 2Area[***ACM***] ≅ 2Area[***CDM***] ≅ 2Area[***BDN***] ≅ 2Area[***CDN***] ≅ 2Area[***BCX***] ≅ 2Area[***CDX***] ≅ 2Area[***CMQ***] ≅ 2Area[***DNS***] ≅ 4Area[***ANX***] ≅ 4Area[***BNX***] ≅ 4Area[***CNX***] ≅ 4Area[***DNX***]

Area[***BCM***] ≅ Area[***ADN***] ≅ 2Area[***AMN***] ≅ 2Area[***BMN***] ≅ 2Area[***CMN***] ≅ 2Area[***DMN***]

Area[***ADP***] ≅ Area[***BCP***] ≅ Area[***AMQ***] ≅ Area[***ACQ***] ≅ Area[***BDQ***] ≅ Area[***DMQ***] ≅ Area[***ACS***] ≅ Area[***BNS***] ≅ Area[***BDS***] ≅ Area[***CNS***] ≅ $\frac{1}{2}$Area[***ADQ***] ≅ $\frac{1}{2}$Area[***BCS***] ≅ 2Area[***AMP***] ≅ 2Area[***BNP***] ≅ 2Area[***CNP***] ≅ 2Area[***DMP***] ≅ 2Area[***BXQ***] ≅ 2Area[***DXQ***] ≅ 2Area[***BXS***] ≅ 2Area[***DXS***]

Area[***ABP***] ≅ Area[***ABQ***] ≅ Area[***ABS***] ≅ 2Area[***ANP***] ≅ 2Area[***BMP***] ≅ 2Area[***AXS***]

Area[***CDP***] ≅ Area[***CDQ***] ≅ Area[***CDS***] ≅ 2Area[***CMP***] ≅ 2Area[***DNP***] ≅ 2Area[***CXQ***]

Area[***ACX***] ≅ 2Area[***MNP***] ≅ 2Area[***MNQ***] ≅ 2Area[***MNS***]

Area[***AXP***] ≅ Area[***ANQ***] ≅ Area[***BMS***] ≅ 2Area[***MXP***] ≅ 2Area[***MXQ***] ≅ 2Area[***MXS***]

Area[***CXP***] ≅ Area[***DNQ***] ≅ Area[***CMS***] ≅ 2Area[***NXP***] ≅ 2Area[***NXQ***] ≅ 2Area[***NXS***]

Area[***APQ***] ≅ Area[***BPQ***] ≅ Area[***APS***] ≅ Area[***BPS***] ≅ $\frac{1}{2}$Area[***AQS***] ≅ $\frac{1}{2}$Area[***BQS***]

Area[***CPQ***] ≅ Area[***DPQ***] ≅ Area[***CPS***] ≅ Area[***DPS***] ≅ $\frac{1}{2}$Area[***CQS***] ≅ $\frac{1}{2}$Area[***DQS***]

Area[***MPQ***] ≅ Area[***NPQ***] ≅ Area[***XPQ***] ≅ Area[***MPS***] ≅ Area[***NPS***] ≅ Area[***XPS***] ≅ $\frac{1}{2}$Area[***MQS***] ≅ $\frac{1}{2}$Area[***NQS***] ≅ $\frac{1}{2}$Area[***XQS***]

========== End of Generated Output ==========

If option *GraphicPoints* is used the generated notebook will also contain the given coordinates of the free, and sometimes semi−free points, so a more fancy graph of the explored geometric configuration may be obtained. By giving the coordinates of a semi−free point its position relative to the other points may be influenced.

Example:

If we use

```
KnowledgeBase["Trapezoid_g", any[A, B, C, D],
  {{A, {0, 0}}, {B, {250, 0}}, {C, {150, 120}}, {D, {70, 120}}}]
```

and

```
kb2 = Explore[Assumptions["Trapezoid2"],
      by → AreaExplore, using → kb1, producing → Tr2,
      ExplorerOptions → {GraphicPoints → KnowledgeBase["Trapezoid_g"]}];
```

the produced notebook with the graph expanded will be:

========== Output Generated by the Program ==========

Select construction, then press the button to view the graph

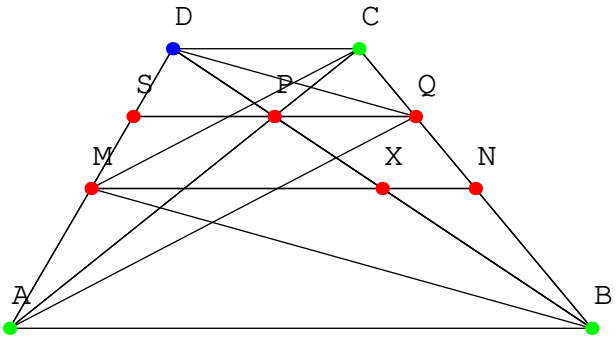
PrintGraph[■]

For any points {***A***, ***B***, ***C***, ***D***, ***M***, ***N***, ***X***, ***P***, ***Q***, ***S***} that satisfy the following constructions:

triangle[***A***, ***B***, ***C***] ∧ pon[***D***, pline[***C***, ***A***, ***B***]] ∧ line[***B***, ***C***] ∧ ,
line[***A***, ***D***] ∧ midpoint[***M***, ***A***, ***D***] ∧ midpoint[***N***, ***B***, ***C***] ∧
line[***M***, ***N***] ∧ inter[***X***, line[***M***, ***N***], line[***B***, ***D***]] ∧
inter[***P***, line[***A***, ***C***], line[***B***, ***D***]] ∧ inter[***Q***, line[***B***, ***C***], pline[***P***, ***A***, ***B***]] ∧
inter[***S***, line[***P***, ***Q***], line[***A***, ***D***]] ∧ line[***M***, ***C***] ∧ line[***A***, ***Q***]

recommended position of points for drawing,

{{***A***, {0, 0}}, {***B***, {250, 0}}, {***C***, {150, 120}}, {***D***, {70, 120}}}




=== The rest of the notebook similar to the previous one ===

========== End of Generated Output ==========

## *8.3 Implementation*

Being closely related to the area method function *AreaExplore* and all its auxiliary functions are implemented in the *Area* package.

Function *ConstrDbase* computes the signed areas and Pythagoras differences of the triangles and prepares the part of the knowledge base that contains these elements.

```
ConstrDbase[constructions_, realpoints_, lastprocessed_, adbP_, adbS_] :=
  Module[{i, j, l, aa, cc, areadbaseS, areadbaseP, list}, Block[{expr},
      areadbaseS = {}; areadbaseP = {};
      If[lastprocessed == 0, If[constructions[[1, 1]] === "circle",
          l = {•P{realpoints[[1]],realpoints[[2]],realpoints[[1]]}};
          Do[l = Join[l, GenerateAllElements[Drop[realpoints, i]]],
            {i, 3 – Length[realpoints], 0}];
          Do[expr = l[[j]];
            expr = expr //. ElimPoint[constructions[[1]], True] //. OrdPoints;
```

```
    expr = System‘Simplify[expr /. •L{a__} :→ ElimCirL[•L{a}]];
    If[l[[j, 1]] === •P, AppendTo[areadbaseP, l[[j]] → expr],
      AppendTo[areadbaseS, l[[j]] → expr]], {j, Length[l]}],
  l = GenerateFirstElements[realpoints];
  Do[AppendTo[areadbaseP,
      l[[j]] → (l[[j]] /. {•P{a_,b_,a_} → 2 •L{a,b}^2, •P{a_,b_,c_} → •L{a,b}^2 +
                  •L{b,c}^2 − •L{a,c}^2} /. OrdPoints)], {j, Length[l]}]],
If[usepoints === {}, l = GenerateAllElements[realpoints],
  l = GenerateAllElements[usepoints]];
Do[expr = l[[j]]; Do[
    expr = expr /. ElimPoint[constructions[[i]], True];
    expr = expr //. OrdPoints; expr = System‘Simplify[expr];
    expr = System‘Simplify[
        expr //. (adbP //. ToMma) //. (adbS //. ToMma) //. OrdPoints],
    {i, Length[constructions], lastprocessed + 1, −1}];
  expr = expr /. {•P{a_,b_,a_} → 2 •L{a,b}^2,
        •P{a_,b_,c_} → •L{a,b}^2 + •L{b,c}^2 − •L{a,c}^2} //. OrdPoints;
  If[l[[j, 1]] === •P, AppendTo[areadbaseP, l[[j]] → expr],
    AppendTo[areadbaseS, l[[j]] → expr]],
  {j, Length[l]}]]; {areadbaseP, areadbaseS}]]
```

Function *FindMeaning* analyzes all the expressions already obtained for the signed areas and Pythagoras differences and tries to find the existing relations. As an example we present the analyzed cases for Pythagoras differences:

```
FindMeaning[•P{a_,b_,a_} → 0] := leqpoints = {a, b}

FindMeaning[•Pl_ → 0] := Module[{a, b, c, d}, {a, b, d} = l; c = b;
    If[Length[leqpoints] == 2 && Intersection[leqpoints, {a, b, d}] === leqpoints,
      Return[]]; Perpfound[{a, b}, {c, d}]]

FindMeaning[•P{a_,b_,a_} → •P{c_,d_,c_}] := Lfound[{a, b}, {c, d}]

FindMeaning[•P{a_,b_,a_} → x__, •P{c_,d_,c_} → x__] := Lfound[{a, b}, {c, d}]

FindMeaning[•P{a_,b_,a_} → x__, •P{c_,d_,c_} → y__] :=
  Lfound[Sqrt[Abs[System‘Simplify[y / x]]] ⋅ {a, b}, {c, d}] /;
    NumberQ[System‘Simplify[y / x]]

FindMeaning[•P{a_,c_,d_} → •P{b_,x_,y_}] :=
  Perpfound[{a, b}, {c, d}] /; c === x && d === y

FindMeaning[•P{a_,c_,d_} → x__, •P{b_,c_,d_} → x__] :=
  Perpfound[{a, b}, {c, d}] /; x =!= 0 && a =!= d && b =!= d
```

The main function *AreaExplore* reads the old knowledgebase, gathers data from all previous constructions, for each construction calls the processing functions and organizes the obtained data in the new knowledge base.

```
AreaExplore[goal_, kb_, newkb1_, opts___?OptionQ] :=
  Module[{a, b, i, j, jj, substeq, newareadbaseP, newareadbaseS,
      nrS, nrP,  hyp, concl, constr = goal[[4, 1, 2]], info, oldkb,
```

```
    newkb, constructions, expr,  variants, nprocessed, areadbaseP,
    areadbaseS, lastprocessed, leqp, lparl, lperpl, leql, leqa},
  Block[{lcolp, lparlines, leqlength, leqarea, lperplines,
      leqpoints, realpoints, usepoints, coord},
    {usepoints, coord} = {RelatedPoints, GraphicPoints} /. opts /.
        Options[AreaExplore];
    If[coord =!= {}, coord = coord[[4, 1, 2]]];
    usepoints = Map[•var, Flatten[usepoints]];
    If [AtomQ[newkb1], newkb = newkb1, If[newkb1[[1]] === "None",
        newkb = goal[[1]] <> "_kb", newkb = newkb1[[1]]]];
    If[kb[[1]] === "None", oldkb = newkb, oldkb = kb[[1]]];
    {realpoints,  hyp, concl, lastprocessed, areadbaseP, areadbaseS} =
      InitAreaExplore[kb];
    {leqp, lcolp, lparl, lperpl, leql, leqa} = Table[concl[[i, 1]], {i, 6}];
    substeq = Table[leqp[[i, 2, 1]] → leqp[[i, 1, 1]], {i, Length[leqp]}];
    If[FreeQ[{™And, ™Implies, And, Implies}, Head[constr]],
      constr = {constr}];
    constr = constr /. substeq;
    If[oldkb === newkb,
      KnowledgeBase[newkb] = •kb[newkb], KnowledgeBase[newkb] =
        Prepend[Rest[KnowledgeBase[oldkb]], newkb]];
    If[ And @@ Table[MemberQ[realpoints, constr[[i, 1]]], {i, Length[constr]}],
      If[And @@ Table[MemberQ[{line, circle}, Head[constr[[i]]]],
            {i, Length[constr]}], jj = Length[KnowledgeBase[newkb]];
        KnowledgeBase[newkb] = ReplacePart[KnowledgeBase[newkb],
            KnowledgeBase[newkb][[jj, 3, 1]] ∧ constr, {jj, 3, 1}];
        KnowledgeBase[newkb] = KnowledgeBase[newkb] /.
            •coord[_] −> •coord[coord];
        constr = {}, Message[AreaExplore::err]; Return[]]];
    jj = 1;
    While[jj ≤ Length[constr], hyp = hyp ∧ constr[[jj]];
      While[jj < Length[constr] && (constr[[jj, 1]] == constr[[jj + 1, 1]] ||
              MemberQ[{line, circle}, Head[constr[[jj + 1]]]]),
        hyp = hyp ∧ constr[[jj + 1]]; jj++]; jj++;
      {constructions, expr, variants} =
        TransArea[(hyp) /. ToMma, False, False];
      lparlines = {}; leqlength = {}; leqarea = {}; lperplines = {}; leqpoints = {};
      Do[If[FreeQ[realpoints,  geomobj[[i, 1]]],
          realpoints = Append[realpoints,  geomobj[[i, 1]]]],
        {i, Length[geomobj]}]; If[leqp =!= {},
        Do[realpoints = Delete[realpoints, Position[realpoints, leqp[[i, 2, 1]]]],
          {i, Length[leqp]}]];
      If[usepoints =!= {}, AppendTo[usepoints, Last[realpoints]]];
      {newareadbaseP, newareadbaseS} = ConstrDbase[constructions,
          realpoints, lastprocessed, areadbaseP, areadbaseS];
      lastprocessed = Length[constructions];
      nrS = Length[areadbaseS]; nrP = Length[areadbaseP];
      areadbaseS = Join[areadbaseS, newareadbaseS] /. ToMma;
      areadbaseP = Join[areadbaseP, newareadbaseP] /. ToMma;
      Do[ind = True; indl = True; inda = True;
        FindMeaning[areadbaseS[[i]]]; If[areadbaseS[[i, 2]] =!= 0,
```

```
      Do[If[Not[ind && indl], Break[]];
        FindMeaning[areadbaseS[[i]], areadbaseS[[j]]], {j, i − 1}]],
    {i, nrS + 1, Length[areadbaseS]}];
  Do[ind = True; indl = True; inda = True;  FindMeaning[areadbaseP[[i]]];
    If[areadbaseP[[i, 2]] =!= 0,
      Do[If[Not[ind && indl], Break[]];
        FindMeaning[areadbaseP[[i]], areadbaseP[[j]]], {j, i − 1}]],
    {i, nrP + 1, Length[areadbaseP]}];
  If[Length[leqpoints] == 2, Print["Congruent points  ", leqpoints];
    {a, b} = leqpoints; If[Position[realpoints, b][[1, 1]] <
        Position[realpoints, a][[1, 1]], {a, b} = {b, a}];
    leqp = Append[leqp, {{a}, {b}}];
    substeq = Append[substeq, b → a];
    constr = constr /. substeq;
    realpoints = Delete[realpoints, Position[realpoints, a]];
    areadbaseP = Take[areadbaseP, nrP];
    areadbaseS = Take[areadbaseS, nrS];
    newareadbaseP = {}; newareadbaseS = {},
    lcolp = Table[Sort[lcolp[[i]], PointLess[#1, #2] &], {i, Length[lcolp]}];
    lparl = JoinList[lparl, UseFirstPoints[lparlines]];
    lperpl = Union[lperpl, UseFirstPoints[lperplines]];
    leql = JoinList[leql, leqlength];
    leqa = JoinList[leqa, leqarea]];
  concl = •eqpoints[leqp] ∧ •colp[lcolp] ∧ •parl[lparl] ∧
      •perpl[lperpl] ∧ •eql[leql] ∧ •eqa[leqa] ∧ •coord[coord];
  info = Explinfo["", realpoints, hyp → concl,
      {lastprocessed, newareadbaseP, newareadbaseS}];
  AppendTo[KnowledgeBase[newkb], info]]; KnowledgeBase[newkb]]
```

The produced knowledge base contains an •*explinfo* element for each construction. This retains the already constructed points in a •*range* tag, the geometric configuration as •*usedFormulae*, the found properties as •*generatedFormulae*, and the newly computed Pythagoras differences and signed areas as auxiliary information.

Example:

The above obtained *kb1* knowledge base is:

========== Output Generated by the Program ==========

```
•kb[Tr1, •explinfo[, •range[•simpleRange[A],
   •simpleRange[B], •simpleRange[C]],
  •usedFormulae[triangle[A, B, C]], •generatedFormulae[
   •eqpoints[{}] ∧ •colp[{}] ∧ •parl[{}] ∧
    •perpl[{}] ∧ •eql[{}] ∧ •eqa[{}] ∧ •coord[{}]],
  {1, {•P{A,B,C} → •L{A,B}^2 + -•L{A,C}^2 + •L{B,C}^2,
    •P{B,A,C} → •L{A,B}^2 + •L{A,C}^2 + -•L{B,C}^2, •P{A,C,B} →
     -•L{A,B}^2 + •L{A,C}^2 + •L{B,C}^2, •P{A,B,A} → 2 •L{A,B}^2,
    •P{A,C,A} → 2 •L{A,C}^2, •P{B,C,B} → 2 •L{B,C}^2}, {}}],
 •explinfo[, •range[•simpleRange[A], •simpleRange[B],
   •simpleRange[C], •simpleRange[D]],
  •usedFormulae[triangle[A, B, C] ∧
    pon[D, pline[C, A, B]] ∧ line[B, C] ∧ line[A, D]],
  •generatedFormulae[•eqpoints[{}] ∧
    •colp[{{A, B}, {C, D}}] ∧ •parl[{{{A, B}, {C, D}}}] ∧
    •perpl[{}] ∧ •eql[{}] ∧ •eqa[{{{A, B, C}, {A, B, D}},
      {{A, C, D}, {B, C, D}}}] ∧ •coord[{}]],
  {2, {•P{A,D,A} → 2 (rD^2 •L{A,B}^2 + •L{A,C}^2 +
        rD (•L{A,B}^2 + •L{A,C}^2 + -•L{B,C}^2)),
    •P{B,D,B} → 2 (rD^2 •L{A,B}^2 + •L{B,C}^2 +
        -rD (•L{A,B}^2 + -•L{A,C}^2 + •L{B,C}^2)),
    •P{C,D,C} → 2 rD^2 •L{A,B}^2, •P{A,B,D} →
     (1 + -2 rD) •L{A,B}^2 + -•L{A,C}^2 + •L{B,C}^2,
    •P{B,A,D} → (1 + 2 rD) •L{A,B}^2 + •L{A,C}^2 + -•L{B,C}^2,
    •P{A,D,B} → ((-1) + 2 rD^2) •L{A,B}^2 +
      (1 + 2 rD) •L{A,C}^2 + (1 + -2 rD) •L{B,C}^2,
    •P{A,C,D} → -rD (•L{A,B}^2 + •L{A,C}^2 + -•L{B,C}^2),
    •P{C,A,D} → 2 •L{A,C}^2 + rD (•L{A,B}^2 + •L{A,C}^2 + -•L{B,C}^2),
    •P{A,D,C} → rD ((1 + 2 rD) •L{A,B}^2 + •L{A,C}^2 + -•L{B,C}^2),
    •P{B,C,D} → rD (•L{A,B}^2 + -•L{A,C}^2 + •L{B,C}^2),
    •P{C,B,D} → 2 •L{B,C}^2 + -rD (•L{A,B}^2 + -•L{A,C}^2 + •L{B,C}^2),
    •P{B,D,C} →
     rD (((-1) + 2 rD) •L{A,B}^2 + •L{A,C}^2 + -•L{B,C}^2)},
   {•S{A,B,D} → •S{A,B,C}, •S{A,C,D} → -rD •S{A,B,C},
    •S{B,C,D} → -rD •S{A,B,C}}],
 •explinfo[, •range[•simpleRange[A], •simpleRange[B],
   •simpleRange[C], •simpleRange[D],
   •simpleRange[M]], •usedFormulae[
  triangle[A, B, C] ∧ pon[D, pline[C, A, B]] ∧
   line[B, C] ∧ line[A, D] ∧ midpoint[M, A, D]],
  •generatedFormulae[•eqpoints[{}] ∧
```

```
    •colp[{{A, B}, {C, D}, {A, D, M}}] ∧
    •parl[{{{A, B}, {C, D}}}] ∧ •perpl[{}] ∧
    •eql[{{{A, D}, 2 · {A, M}, 2 · {D, M}}}] ∧
    •eqa[{{{A, B, C}, {A, B, D}, 2 · {A, B, M},
       2 · {B, D, M}}, {{A, C, D}, {B, C, D},
       2 · {A, C, M}, 2 · {C, D, M}}}] ∧ •coord[{}]],
 {3, === computed elements ===}],
•explinfo[, •range[•simpleRange[A], •simpleRange[B],
  •simpleRange[C], •simpleRange[D], •simpleRange[M],
  •simpleRange[N]], •usedFormulae[triangle[A, B, C] ∧
   pon[D, pline[C, A, B]] ∧ line[B, C] ∧ line[A, D] ∧
   midpoint[M, A, D] ∧ midpoint[N, B, C] ∧ line[M, N]],
 •generatedFormulae[•eqpoints[{}] ∧ •colp[
    {{A, B}, {C, D}, {A, D, M}, {M, N}, {B, C, N}}] ∧
    •parl[{{{A, B}, {C, D}, {M, N}}}] ∧ •perpl[{}] ∧
    •eql[{{{A, D}, 2 · {A, M}, 2 · {D, M}},
      {{B, C}, 2 · {B, N}, 2 · {C, N}}}] ∧
    •eqa[{{{A, B, C}, {A, B, D}, 2 · {A, B, M},
       2 · {B, D, M}, 2 · {A, C, N}, 2 · {A, B, N}},
      {{A, C, D}, {B, C, D}, 2 · {A, C, M},
       2 · {C, D, M}, 2 · {B, D, N}, 2 · {C, D, N}},
      {{B, C, M}, {A, D, N}, 2 · {A, M, N}, 2 · {B, M, N},
       2 · {C, M, N}, 2 · {D, M, N}}}] ∧ •coord[{}]],
 {4, === computed elements ===}]]
```

========== End of Generated Output ==========

# 9 Conclusions

We managed to integrate many known methods for geometric theorem proving and combination of these methods into one coherent system (*Theorema*). The next natural step we want to investigate is the combination of these algebraically oriented methods with the general predicate logic oriented methods of theorem proving (natural deduction methods, resolution, rewriting, set theory provers, etc), which are also already implemented in *Theorema*. Also, as a natural next step, the integration of solving (constraint solving, model checking, finding of constructions) and proving in geometry will be an important issue for our future work.

# References


[1] Bellemain, F., Laborde, J.: *Cabri Geometry II*. Texas Instruments, Dallas, 1995.

[2] Buchberger, B.: Theory exploration with Theorema. In *Proc. SYNASC 20 (2nd International Workshop on Symbolic and Numeric Algorithms Scientific Computing, Timisoara, Romania, Oct. 4-6, 2000)*, page 1-16, 2000.

[3] Buchberger, B.: Mathematical knowledge management in Theorema. In Buchberger and O. Caprotti, eds., editor, *Electronic Proceedings of the International Workshop on Mathematical Knowledge Management, Sch Hagenberg, Austria, September 24-26, 2001*, 2001.

[4] Buchberger, B.: Gröbner bases: An algorythmic method in polynomial ide theory. In *Multidimensional Systems Theory*, 6, page 184-232. (N.K. Bose ed D. Reidel Publ. Comp., 1985.

[5] Buchberger, B., Collins, G. E., Kutzler, B.: Algebraic methods for geom reasoning. *Ann. Rev. Comp. Sci.*, 3, page 85-119, 1988.

[6] Buchberger, B., Dupre, C., Jebelean, T., Kriftner, F., Nakagawa, K., Vasaru, D. Windsteiger, W.: The Theorema project: A progress report. *Symboli Computation and Automated Reasoning, M. Kerber and M. Kohlhase (eds.)* page 98-113, 2001.

[7] Buchberger, B., T. Jebelean, T.,Kriftner, F., Marin, M., Tomuta E., Vasaru, D An overview on the Theorema project. In W. Kuechlin (ed.), editor, *Pro ISSAC'97 (International Symposium on Symbolic and Algebraic Computatio Maui, Hawaii, July 21-23, 1997)*, page 384-391, ACM Press, 1997.

[8] Cerutti, E., Davis, P. J.: FORMAC meets Pappus: Some observations elementary analytic geometry by computer. *American Mathematical Montl* 76, page 895-905, 1969.

[9] Chou, S.-C.: GEO-prover - A geometry theorem prover developed *Springer's LNCS*, 230, page 679-680, 1986.

[10] Chou, S.-C.: *Mechanical geometry theorem proving*, Reidel, Dordrecht Bosto 1988.

[11] Chou, S.-C., Gao, X.-S., Zhang, J.-Z.: Automated production of traditi proofs for theorems in Euclidean geometry. In *Proc. of Eigth IEEE Symp Logic in Comp. Sci.*, page 48-56, IEEE Computer Society Press, 1993.

[12] Chou, S.-C., Gao, X.-S., Zhang, J.-Z.: Automated geometry theorem proving i vector calculation. In *ACM-ISSAC*, page 284-291, Kiev, 1993.

[13] Chou, S.-C., Gao, X.-S., Zhang, J.-Z.: *Machine proofs in geometry: Automate production of readable proofs for geometry theorems*, volume 6, series Applie Mathematics. World Scientific, Singapore, 1994.

[14] Chou, S.-C., Gao, X.-S., Zhang, J.-Z.: Automated generation of readable proof with geometric invariants I & II. *J. Automat. Reason.*, 17, page 325-370, 1996.

[15] Chou, S.-C., Gao, X.-S., Zhang, J.-Z.: An introduction to Geometry e *Springer's LNAI*, 1104, page 235-239, 1996.

[16] Chou, S.-C., Gao, X.-S., Zhang, J.-Z.: A deductive database approach t automated geometry theorem proving and discovering. *J. Automat. Reason.*, 25 page 219-246, 1996.

[17] Coelho, H., Pereira, L. M.: GEOM: A Prolog geometry theorem p Laboratório Nacional de Engenhaaria Civil Memória no. 525 Ministe Habitacao e Obrass Publicas, Portugal, 1979.

[18] Collins, G. E.: Quantifier elimination for real closed fields by cy algebraic decomposition. *Springer's LNCS*, 33, page 134-165, 1975.

[19] Cyrluk, D. A., Harris, R. M., Kapur, D.: GEOMETER: A theorem prove algebraic geometry. *Springer's LNCS*, 310, page 770-771, 1988.

[20] Elcock, E.W.: Representation of knowledge in a geometry machine. *Mach Intell.*, 8, page 11-29, 1977.

[21] Gelernter, H.: Realization of a geometry theorem-proving machine. In *Proc. Int Conf. Info. Process (Paris, 1959)*, page 273-282.

[22] Gilmore, P. C.: An examination of the geometry theorem machine. *Artif. Intell* 1, page 171-187, 1970.

[23] Hestenes, D.,Sobczyk, G.: *Clifford algebra to geometric calculus*, Dordrecht Boston, 1984.

[24] Hilbert, D.: *The foundations of geometry*, The Open Court Company, Translation by E. J. Townsend.

[25] Kapur, D.: Using Gröbner bases to reason about geometry problems. *J. Sy Comput.*, 2, page 399-408, 1986.

[26] Ko, H.-P.: ALGE-prover II: A new edition of ALGE-prover. Tec 86CRD081 General Electric Co., Schenectady, USA, 1986.

[27] Ko, H.-P., Hussain, M. A.: ALGE-prover: An algebraic geometry theo proving software. Tech. Rep. 85CRD139 General Electric Co., Schenect USA, 1985.

[28] Kutzler, B., Stifter, S.: On the application of Buchberger's algorit automated geometry theorem proving. *J. Symb. Comput.*, 2, page 38 1986.

[29] Kutzler, B., Stifter, S.: A geometry theorem prover based on Buchb algorithm. *Springer's LNCS*, 230, page 693-694, 1986.

[30] Kutzler, B., Stifter, S.: New approaches to computerized proofs of geon theorems. RISC 86-0.5 RISC-Linz, 1986.

[31] Nevins, A.J.: Plane geometry theorem proving using forward chaining. *A Intell.*, 6, page 1-23, 1975.

[32] Poincarè, H.: Review of Hilbert's foundations of geometry (1902). In *Rea numbers, generalizations of the reals, and theories of the continua*, volume 24 of Synthese Library, Kluwer Academic Publisher, 1994.

[33] Richter-Gebert, J.: Mechanical theorem proving in projective geometry. *A Math. Artif. Intell.*, 13, page 139-172, 1995.

[34] Richter-Gebert, J., Kortenkamp, U.H.: *Cinderella – The interactive geom software*, Springer, Heidelberg, 1999.

[35] Ritt, R. F.: *Differential equations from algebraic standpoint*, volume 14, seri AMS Colloquim Publications. New York, 1938.

[36] Ritt, R. F.: *Differential algebra*, series AMS Colloquim Publications York, 1950.

[37] Robu, J.: Systematic exploration of geometric configurations using Theore based on Mathematica. In *Proc. SYNASC '01 (3rd International Workshop Symbolic and Numeric Algorithms for Scientific Computing, Timisoa Romania, Oct. 2-5, 2001)*, page (to appear), 2001.

[38] Tarski, A.: *A decision method for elementary algebra and geometry*, Univ California Press, Berkeley Los Angeles, 2nd edition edition, 1951.

[39] Tarski, A., Givant S.: Tarski's system of geometry. *The Bulletin of Symb. Logic* 5(2), page 175-214, 1999.

[40] Wang, D.: Algebraic factoring and geometry theorem proving. *Spr LNCS*, 814, page 386-400, 1994.

[41] Wang, D.: GEOTHER: A geometry theorem prover. *Springer's LNAI*, 1 page 213-239, 1996.

[42] Wang, D.: Geometry machines: From AI to SMC. *Springer's LNCS*, 1138, pag 213-239, 1996.

[43] Winkler, F.: Gröbner bases in geometry theorem proving and s degeneracy conditions. *Mathematica Pannonica*, 1(1), page 15-32, 1990.

[44] Wu, W.-t.: Basic principles of mechanical theorem proving in eler geometries. *J. Automat. Reason.*, 2, page 221-252, 1986.