Automated Proofs using Bracket Algebra
with
Cinderella and *OpenMath*

Dan Roozemond BSc
Eindhoven University of Technology
Department of Mathematics and Computer Science

Abstract

This paper describes the results of a project intended to make it possible to put forward geometrical theorems by pointing and clicking, and then obtain a proof for that theorem automatically. This goal was achieved by adding various options to *Cinderella* [1], a computer program with which one can create geometrical configurations. Its internal 'Randomized prover' is able to discover theorems automatically.

In the project the functionality was added to find proofs for these theorems with the aid of the computer algebra package *GAP* [9]. Communication between these two programs and the various steps in generating the proof is done by means of *OpenMath* [5, 7]. The proofs are represented by bracket calculations as proposed in [8].

1 Introduction

Proof is the idol before whom the pure mathematician tortures himself.
Sir Arthur Eddington (1882 - 1944)

Cinderella is a computer program, with which one can create geometrical configurations. Cinderella has a built in 'Randomized prover', that is able to discover geometrical theorems and return a probabilistic proof [2]. However, these proofs are not verifiable.

The main goal of the project covered here was to make it possible to put forward a theorem by pointing and clicking (in Cinderella), and then obtain a mathematically sound proof of that theorem. Moreover, this proof should be verifiable. This goal was achieved by using the following three packages:

Cinderella "Software for doing geometry on the computer, designed to be both mathematically robust and easy to use" [1].

OpenMath "A new, extensible standard for representing the semantics of mathematical objects" [5] - The communication between Cinderella and GAP was implemented with OpenMath. This was done using the Riaca OpenMath library for Java [7].

GAP "GAP – Groups, Algorithms, and Programming" [9].

In this paper we first introduce the OpenMath standard in Section 2. Sections 3 and 4 explain how bracket calculations are used for the representation of geometrical configurations and theorems. Section 5 addresses the structure of the prover. Section 6 gives some notes on the translation from a geometric configuration in Cinderella into bracket equations as well as the implementation using GAP. A few examples of the theorem prover in action can be found in Section 7.

2 The OpenMath Standard

The OpenMath standard is made for the representation of mathematics in such a way that mathematical objects can easily be exchanged between computer programs.

> OpenMath is an emerging standard for representing mathematical objects with their semantics, allowing them to be exchanged between computer programs, stored in databases, or published on the worldwide web. While the original designers were mainly developers of computer algebra systems, it is now attracting interest from other areas of scientific computation and from many publishers of electronic documents with a significant mathematical content. [5, Overview]

A rough overview of the standard can be found in Figure 1. The 3 layers are explained as follows:

Language The OpenMath language defines the 'grammar'. It defines notions like Variables, Constants, Errors, and Functions.

Content Dictionary A Content Dictionary (CD) is (or can be) defined for each area of Mathematics. For example the 'arith1' CD describes the notions of 'minus', 'plus', 'power', etc.

Phrasebooks A Phrasebook provides communication between OpenMath and another program. Phrasebooks exist for, for example, Mathematica, GAP and Singular. A specific Phrasebook consists of three parts:

- An *encoder* to encode OpenMath objects into commands that the program understands,
- A *decoder* to translate program output into OpenMath objects,
- The physical communication between the program and the Java (or C, or C++) program containing the OpenMath objects.

The interested reader is encouraged to have a look at `http://www.openmath.org` for an extensive overview of the OpenMath standard. In this project the (experimental) `plangeo` codec [6] is used, since it contains elements for representing planar geometry.

3 Brackets and Projective Geometry

When we restrict ourselves to geometric theorems that are invariant under projective transformations, we can prove geometric theorems much faster than when we would have used Gröbner bases. The method we use is based on a paper by Jürgen Richter-Gebert in 1995 [8]. For now, we only consider configurations and theses of the form:

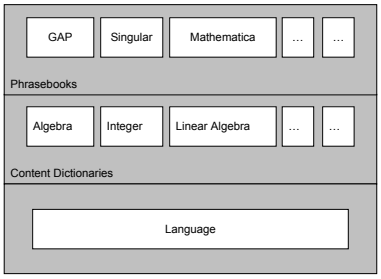

Figure 1: The OpenMath framework

- The three points A, B, and C lie on one line (the points A, B, and C are collinear), denoted by '$h(A, B, C)$',
- The three lines through A and B, C and D, and E and F, respectively, go through one point, denoted by '$m((A, B), (C, D), (E, F))$',
- The six points A, B, C, D, E, and F lie on one conic, denoted by '$c(A, B, C, D, E, F)$'.

We again observe the homogeneous coordinates in the plane, elements of $\mathbb{P}^3$. This means the coordinates are in $(\mathbb{R}^3\backslash\{0\})/\mathbb{R}\backslash\{0\}$, in words: all scalar multiples of a vector denote the same point. We will denote the determinant

$$\begin{vmatrix} x_A & x_B & x_C \\ y_A & y_B & y_C \\ z_A & z_B & z_C \end{vmatrix}$$

corresponding to the points A, B, and C by $[ABC]$. This notation is referred to as the *bracket notation*, a determinant $[ABC]$ as *a bracket*. In the mathematical foundation in this section parts from the masters thesis by one of Jürgen Richter-Gebert's students, Andreas Umbach, were consulted [10].

3.1 Collinearity

First, we focus on the collinearity conditions, described by $h(A, B, C)$. It is commonly known that three points A, B, and C are collinear if and only if $[ABC] = 0$.

To create a proof (as shown in the next section) we need to make a connection between several conditions. This can be done using the following theorem. In order to make reading easier, we write k for the point defined by coordinates $(x_k, y_k, z_k)^T$.

Theorem 3.1. *Let* $1, 2, 3, 4$*, and* 5 *be 5 points in the plane, such that* $1, 4$*, and* 5 *are not collinear. Then the following equivalence holds:*

$$[123] = 0 \Leftrightarrow [124][135] = [125][134]$$

Proof The bracket is a 3-linear alternating form: Observe

$$a = [123][145] - [124][135] + [125][134].$$

Fix 1 in a. Then a is a 4-linear alternating form on $\{2,3,4,5\}$. However, there is no such thing as a non degenerate 4-linear alternating form in a 3-dimensional space, so $a = 0$. Since this implies

$$[123][145] = [124][135] - [125][134]$$

the theorem is proved. □

Using this theorem, we can translate a set of conditions of the form $h(A,B,C)$ to *bi-quadratic equations*:

$$[ABD][ACE] = [ABE][ACD]$$

for any D and E such that A, D, and E are not collinear.

This method of describing a geometry theorem implicitly introduces a number of non-degeneracy conditions. For example, the fact that 1, 4, and 5 are not collinear. On the one hand, this is an advantage, as we do not have to express this kind of non-degeneracy conditions explicitly. On the other hand, this is a disadvantage, as we might add some non-degeneracy conditions we are not aware of and which might be unnecessary. However, in this specific situation the disadvantage seems less important, since the user will construct a certain theorem in Cinderella, thus (in general) avoiding degenerated cases himself.

3.2 Concurrency and Conics

In this section we show how to translate the assertions $m((A,B),(C,D),(E,F))$ and $c(A,B,C,D,E,F)$ to bracket equations.

Theorem 3.2. *Observe the assertion $m((A,B),(C,D),(E,F))$. This means that the lines through A and B, C and D, and E and F go through one point. This assertion implies that all these 6 points and 3 lines are distinct, and that the point of concurrency is not one of A,B,C,D,E,F, thus implicitly adding non-degeneracy conditions every time we use this assertion.*

This assertion is equivalent to,

$$[ABC][CDE][EFA] = -[ABE][CDA][EFC],$$

and to

$$[ABF][CDE] = [ABE][CDF].$$

Proof Observe the assertion $m((A,B),(C,D),(E,F))$, i.e. the lines through A and B, C and D, and E and F go through one point, say Z. This is equivalent to the combination of the three assertions

$$h(A,B,Z),\ h(C,D,Z) \text{ and } h(E,F,Z).$$

Using Theorem 3.1 we find the following three equations. Notice how we have to use the fact that all 6 points are distinct and none of them is the point of concurrency.

$$\begin{aligned} [ABC][AZE] &= [ABE][AZC], \\ [CDE][CZA] &= [CDA][CZE], \\ [EFA][EZC] &= [EFC][EZA]. \end{aligned} \tag{1}$$

We multiply the left- and right-hand sides and cancel terms that occur on both sides, and obtain

$$[ABC][CDE][EFA] = -[ABE][CDA][EFC], \tag{2}$$

thus proving the first equation.

As $m((A,B),(C,D),(E,F))$ is equivalent to (for example) the assertion $m((A,B),(C,D),(F,E))$ we obtain from Equation 2:

$$-[ABF][CDA][FEC] = [ABC][CDF][FEA]. \tag{3}$$

Again, we multiply the left- and right-hand sides from Equations 2 and 3 and cancel terms that occur on both sides, and we obtain

$$[ABF][CDE] = [ABE][CDF], \tag{4}$$

which proves the second equation of the theorem. □

We just showed two possible encodings of the $m(..)$-assertion. However, it appears that using only the second one suffices in practice. An assertion $m(..)$ gives us only three different instances of Equation 4, all other permutations are equivalent to one of those three.

Remark 3.3. Because of the implicit degeneration conditions introduced, we have to be careful when using $m(..)$ as an assertion representing the thesis. Additionally, in practice it appears a proof using $m(..)$ as configuration assertions is harder to understand than a proof using $h(..)$ as configuration assertions. However, the $m(..)$-assertion still has the huge advantage that it represents three $h(..)$-assertions, thus considerably reducing the amount of configuration assertions and configuration equations.

As this shows that using the $m(..)$-assertion has both advantages we do not want to loose and disadvantages we do not want to have, we chose to leave the choice to the user of Cinderella. When trying to obtain a proof, he can decide whether he wants to use $m(..)$-assertions in the configuration, and whether he wants to use $m(..)$-assertions in the thesis. This enables the user to find the 'golden mean' between the shortness and the clarity of the proof. △

The next theorem describes how to encode six points on a conic into bracket expressions.

Theorem 3.4. *Observe the assertion $c(A,B,C,D,E,F)$, meaning that the six points A,B,C,D, E, and F are on one conic. This assertion implies that all these six points are distinct and no three of the points are collinear, thus implicitly adding non-degeneracy conditions every time we use this assertion.*

This assertion is equivalent to the following bracket equation:

$$[ACE][BDE][ABF][CDF] = [ABE][CDE][ACF][BDF].$$

Proof First, observe four distinct points, A, B, C, and D, and the two degenerate conics c_1 and c_2. The conic c_1 is given by the line through A and B and the line through C and D, the conic c_2 is given by the line through A and C and the line through B and D. For an arbitrary point x we have

$$x \in c_1 \quad \text{if and only if } x \text{ on } AB \text{ or } x \text{ on } CD, \text{ so} \quad [ABx][CDx] = 0$$

$$\begin{array}{rcl} [ABC][ADE][BDF][CEF] & = & [ABD][ACE][BCF][DEF], \\ [ABE][ACD][BDF][CEF] & = & [ABD][ACE][BEF][CDF], \\ [ABE][BCD][ADF][CEF] & = & [ABD][BCE][AEF][CDF], \\ [ABD][AEF][BCF][CDE] & = & [ABF][ADE][BCD][CEF], \\ [ABE][ACF][BDF][CDE] & = & [ABF][ACE][BDE][CDF]. \end{array} \quad (9)$$

Table 1: A basis for $c(..)$-assertions in the configuration

$$x \in c_2 \quad \text{if and only if } x \text{ on } AC \text{ or } x \text{ on } BD, \text{ so} \quad [ACx][BDx] = 0. \quad (5)$$

Now, for all $\lambda, \mu \in \mathbb{R}$, the equation

$$\lambda[ABx][CDx] + \mu[ACx][BDx]$$

describes a conic through A, B, C, and D. Now let

$$\begin{array}{rcl} \lambda & = & [ACE][BDE] \\ \mu & = & -[ABE][CDE], \end{array} \quad (6)$$

and observe the expression

$$[ACE][BDE][ABx][CDx] - [ABE][CDE][ACx][BDx]. \quad (7)$$

Since this is a bi-quadratic expression of degree two, which evaluates to zero for $x \in \{A, B, C, D, E\}$, this defines a conic on the points A,B,C,D, and E. This means F is on that conic if and only if

$$[ACE][BDE][ABF][CDF] = [ABE][CDE][ACF][BDF], \quad (8)$$

which concludes the proof. □

In general, a single $c(..)$-assertion gives us $6! = 720$ possible equations. However, in the configuration a basis of the subspace spanned by these 720 equations will suffice. Such a basis is a set of equations such that the other equations can be obtained by multiplying sides and removing pairs that occur on both sides. With basic linear algebra we can obtain a basis for this [10, p. 36]. The basis can be found in Table 1. So, every $c(..)$-assertion only adds five equations to the set of configuration equations. However, if the thesis is a $c(..)$-assertion, we have to include *all* 720 possible equations, as each of those equations is equivalent to the thesis.

4 Non-Projective Geometry

In this section we present some theory that enables us to represent assertions in non-projective geometry in brackets. Someone might think that calculating with expressions that are invariant with respect to linear transformations, as described in the previous chapter, automatically makes it impossible to prove any theorems containing for example circles. This is not such a strange thought, since circles might become conics (and lose their circularity) under linear transformations. However, by adding two special points to the configuration, we can prove such theorems.

4.1 Complex Numbers

With the following procedure we can use complex numbers to express conditions involving distances, angles, etc. Given a point $P = (x, y) \in \mathbb{R}^2$ in the plane, we define $z_p \in \mathbb{C} := x + iy$. Moreover, $z_p = r \cdot e^{i\varphi}$ for certain $r, \varphi \in \mathbb{R}$. We know $z \in \mathbb{R} \Leftrightarrow z = \overline{z}$, and $z \in i\mathbb{R} \Leftrightarrow z = -\overline{z}$.

We go back to homogeneous coordinates and introduce two new 'points': $I = (i, -1, 0)$ and $J = (-i, -1, 0)$. Now observe the bracket $[ABI]$, where $A = (x_a, y_a, 1)$ and $B = (x_b, y_b, 1)$:

$$[ABI] = \begin{vmatrix} x_a & x_b & i \\ y_a & y_b & -1 \\ 1 & 1 & 0 \end{vmatrix} = x_a + iy_a - x_b - iy_b = z_a - z_b. \tag{10}$$

Likewise, the bracket $[ABJ]$ evaluates to

$$[ABJ] = \begin{vmatrix} x_a & x_b & -i \\ y_a & y_b & -1 \\ 1 & 1 & 0 \end{vmatrix} = x_a - iy_a - x_b + iy_b = \overline{z_a - z_b}. \tag{11}$$

Example (Collinearity) Now suppose A, B, and C are collinear. Observe the complex numbers $z_1 = r_1 e^{i\varphi_1}$ of the vector $(B - A)$, and $z_2 = r_2 e^{i\varphi_2}$ of $(C - A)$. The points A, B, and C are collinear if and only if the two angles φ_1 and φ_2 are either the same or opposed to each other. This means $\varphi_1 = \varphi_2$ or $\varphi_1 = \pi + \varphi_2$, which means $z_1/z_2 \in \mathbb{R}$, or equivalently

$$\frac{B - A}{C - A} = \overline{\left(\frac{B - A}{C - A}\right)} \tag{12}$$

Using Equations 10 and 11 this is equal to

$$\frac{[BAI]}{[CAI]} = \frac{[BAJ]}{[CAJ]}, \tag{13}$$

which evaluates to

$$[ABI][ACJ] = [ACI][ABJ], \tag{14}$$

which indeed fulfills the claim at Theorem 3.1. △

4.2 Circles

We will now show how to encode the fact that four points are on one circle in brackets.

Theorem 4.1. *Suppose the four points A, B, C, and D are on one circle, then the following bracket equation holds:*

$$[ACI][BDI][ADJ][BCJ] = [BCI][ADI][ACJ][BDJ].$$

This assertion will be denoted by $ci(A, B, C, D)$.

Note that this matches the bracket equation for a conic through the points A, B, C, D, I, and J (See Theorem 3.4).

Proof It is a well known theorem that four points A, B, C, and D are on one circle if and only if the angle between AC and BC is equal to the angle between AD and BD. We now switch to complex numbers as described in the start of this section, and find that this is equivalent to

$$\frac{z_A - z_C}{z_B - z_C} \Big/ \frac{z_A - z_D}{z_B - z_D} \in \mathbb{R},$$

with arguing as in the example on collinearity above. This equation can be rewritten to

$$\frac{z_A - z_C}{z_B - z_C} \Big/ \frac{z_A - z_D}{z_B - z_D} = \overline{\frac{z_A - z_C}{z_B - z_C} \Big/ \frac{z_A - z_D}{z_B - z_D}}.$$

Using Equations 10 and 11 this transforms to

$$[ACI][BDI][BCJ][ADJ] = [BCI][ADI][ACJ][BDJ],$$

which concludes the proof. □

5 The Prover

Suppose we are given a certain theorem in planar geometry, containing points and some collinearity conditions. This means we know that certain brackets (i.e. expressions of the form $[ABC]$) are equal to zero. We define B to be the set of all brackets, i.e. all combinations of three points from the geometry theorem, so $|B| = \binom{p}{3}$, where p denotes the number of points in the configuration.

Example Suppose we have a configuration with the points A, B, C, and D. Then

$$B := \{[ABC], [ABD], [ACD], [BCD]\},$$

and $|B| = 4 = \binom{4}{3}$. △

Suppose we have a geometry statement, and by Theorems 3.1, 3.2 and 3.4 we obtained a set of n equations following from the configuration:

$$\begin{aligned} c_{1l} &\equiv c_{1r}, \\ c_{2l} &\equiv c_{2r}, \\ &\vdots \\ c_{nl} &\equiv c_{nr}, \end{aligned} \tag{15}$$

where 'l' denotes the left hand side of the equation, and 'r' the right hand side. Each of the factors of the c_{il} and c_{ir} denotes a determinant of three points in the geometry statement, so each $c_{i,l/r}$ is a product of elements of B. Note that we use the equivalence sign '$\equiv$' rather than the normal equation sign '$=$' to make it clear that we are calculating with brackets, elements of B, rather than with the elements in $\mathbb{R}$ or $\mathbb{Q}$ they evaluate to. From Theorem 3.1 it follows directly that each of the factors of the c_{il} and c_{ir} is not equal to zero.

Moreover, we have an (at least one) equation that implies the thesis we want to test:

$$t_l \equiv t_r. \tag{16}$$

Note that all factors in $t_{l,r}$ should be a factor of at least one $c_{i,l/r}$. This means that the brackets in the thesis equation should occur somewhere in the configuration equations.

Remark 5.1. Note that it is almost always possible to express the thesis in various different equations. For the remainder of the section we will just pick one, for ease of reading. In practice we will test all of them, checking which gives us the shortest proof, if any. △

Now suppose we have a certain oracle that gives us a vector $g \in \mathbb{Q}^n$, $g \neq 0$ such that

$$\frac{1}{t_l}\prod_{i=1}^{n}(c_{il})^{g_i} \equiv \frac{1}{t_r}\prod_{i=1}^{n}(c_{ir})^{g_i}. \tag{17}$$

By multiplying both sides by the greatest common divisor q of the denominators in $g_1, \ldots, g_n$, thus clearing the denominators, we obtain the following equation:

$$\left(\frac{1}{t_l}\right)^q \prod_{i=1}^{n}(c_{il})^{v_i} \equiv \left(\frac{1}{t_r}\right)^q \prod_{i=1}^{n}(c_{ir})^{v_i}, \text{ where } v_i = q \cdot g_i, \text{ so } v_i \in \mathbb{Z}. \tag{18}$$

Remark 5.2. In words: For each of the n equations we multiply a certain power of the left sides with each other, and the same power of the right sides. Then all terms cancel, except for $(t_l)^q$ on the left side, and $(t_r)^q$ on the right side. △

By the definition of the $c_{i,l/r}$ we know

$$(c_{il})^a \equiv (c_{ir})^a \quad \forall 1 \leq i \leq n, \forall a \in \mathbb{Z}, \tag{19}$$

and since $q \neq 0$ we obtain from Equation 18:

$$\left(\frac{1}{t_l}\right)^q \equiv \left(\frac{1}{t_r}\right)^q, \quad \text{so } t_l \equiv t_r. \tag{20}$$

This means such a vector $g \in \mathbb{Q}^n$ gives us a verifiable *proof* that the thesis logically follows from the configuration. In the next section it will be shown how we can obtain such a g.

5.1 Obtaining the Proof

It will be shown that a vector g as in 17 can be found by solving linear equations. We recall that B is the set of all brackets, and define $b = |B|$ and x_i such that $\{x_1, \ldots, x_b\} = B$.

Suppose we have a configuration given by $c_{1l}, c_{1r}, \ldots, c_{nl}, c_{nr}$ and a thesis given by t_l and t_r. Recall that $c_{i,l/r}$ and $t_{l/r}$ are products of elements of B. Introduce the $b \times n$ matrix X with coefficients in $\mathbb{Z}$, defined as follows:

$$\text{for all } 1 \leq k \leq b, 1 \leq i \leq n : X_{ki} := \begin{cases} 1 & \text{if } x_k \text{ is a factor of } c_{il}, \\ -1 & \text{if } x_k \text{ is a factor of } c_{ir}, \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

The vector $Y \in \mathbb{Z}^b$ is defined in the same way from our thesis.

$$\text{for all } 1 \leq k \leq b : Y_k := \begin{cases} 1 & \text{if } x_k \text{ is a factor of } t_l, \\ -1 & \text{if } x_k \text{ is a factor of } t_r, \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

Now observe the following system of linear equations

```
1.  A := FreePoint;        10. F := Meet(a,c);
2.  B := FreePoint;        11. e := Join(E,F);
3.  a := Join(A,B);        12. G := Meet(b,d);
4.  C := FreePoint;        13. f := Join(A,G);
5.  b := Join(B,C);        14. g := Join(A,D);
6.  D := FreePoint;        15. h := Join(B,E);
7.  c := Join(C,D);        16. H := Meet(e,f);
8.  E := FreePoint;        17. k := Join(H,C);
9.  d := Join(D,E);
```

Figure 2: Pappos' Theorem as a Cinderella Algorithm
(Capital characters denote points, small characters denote lines)

$$X \cdot g = Y, \tag{23}$$

with a solution vector g. Since X and Y have integer values, we know that $g \in \mathbb{Q}^n$. It is straightforward to see that g satisfies Equation 17. Thus, we have a procedure that enables us to obtain a proof by solving linear equations, which is *much* faster than having to use Gröbner bases.

Remark 5.3. In general, the problem whether a geometric theorem is true or false is still equal to the decision if $t_l - t_r$ is in the ideal I generated by $c_{il} - c_{ir}$ $(i = 1, \ldots, n)$. This ideal membership problem is normally decided by means of Gröbner bases, but experimenting with Gröbner bases in this context thought us that they are too slow to be practical in our situation. However, a g as in Equation 23 shows that

$$t_l - t_r \equiv \sum_{i=1}^{n} g_i(c_{il} - c_{ir}). \tag{24}$$

which proves the ideal membership. If such a vector g does not exist however, we have no information on the ideal membership. This is why we *lose the possibility to prove a theorem to be false*, as a theorem is called 'false' only when $t_l - t_r \notin \sqrt{I}$. △

6 On the Implementation

In this section some notes are given on the implementation of the prover described in Sections 3 and 4. It is meant to give an *overview* of how the transition from a geometric theorem in Cinderella to a proof of that theorem can be realized.

6.1 Translating the Assertion

The translation from a geometric theorem in Cinderella into a set of assertions of the forms described in Sections 3 and 4, takes place in two steps.

Firstly, an algorithm in Cinderella (see for example Figure 2) is translated into an OpenMath `plangeo.assertion`-object. This can be done rather straightforward, as every step of the algorithm corresponds to a single OpenMath Application. For example,

```
a := Join(A,B)
```

can directly be translated into the OpenMath object below.

```
<OMA>
  <OMS name="line" cd="plangeo1"/>
  <OMV name="a"/>
  <OMA>
    <OMS name="incident" cd="plangeo1"/>
    <OMV name="a"/>
    <OMV name="A"/>
  </OMA>
  <OMA>
    <OMS name="incident" cd="plangeo1"/>
    <OMV name="a"/>
    <OMV name="B"/>
  </OMA>
</OMA>
```

Thus, walking through Cinderella's algorithm step by step, we obtain an OpenMath `plangeo1.assertion` representing the configuration. The thesis added to this object is the last non-trivial incidence the randomized prover concluded.

Cinderella is able to convert the following Cinderella algorithms into OpenMath elements: `Join`, `Meet`, `Mid`, `PointOnLine`[1], `Through`[2], `Orthogonal`, `Parallel`, `CircleMP`[3], `ConicBy5`, `IntersectionConicLine`, `IntersectionConicConic`, `CircleBy3`, `PointOnCircle`, `OtherIntersectionCC`[4], and `OtherIntersectionCL`[5]. Note that not all of these algorithms can be encoded to bracket equations. However, it is useful to translate as much objects as possible to OpenMath, as this OpenMath object can be used by other applications.

In this step we will restrict ourselves to elements we can translate to the assertions given in Sections 3 and 4. This means that if the OpenMath object from the previous step has elements such as `Mid`, an error will be raised and the translation will be broken off at this point. However, the OpenMath object is still valid, and might be used by an application that can handle more statements. Moreover, this two-phased design makes it possible for the prover to handle any theorem in projective geometry that can be expressed in OpenMath, not just the ones that can be constructed in Cinderella!

The `plangeo1.assertion` from the previous step is processed in the following way:

1. Find all elements (point, lines, conics) in the configuration, say $\{E_1, \ldots, E_p\}$,
2. Find all incidences, and link them to the elements, thus finding a set of incidences $F_i \subset \{E_1, \ldots, E_p\}$, where $1 \leq i \leq p$. This means that element E_i is incident to all elements in the set F_i,
3. We set $G := \{1, \ldots, p\}$,

[1] A new point on an existing line
[2] A new line through an existing point
[3] MP stands for MidPoint
[4] The two intersections between two conics
[5] The two intersections between a conic and a line

4. While $G \neq \emptyset$:
   (a) Get an index $k \in G$, where first indices corresponding to conics are processed, then circles, then points, and finally lines,
   (b) Encode the element identified by k. For example: If E_k is a conic, and F_k contains 6 points, an element of the form $c(..)$ is added to the configuration,
   (c) Set $G := G \backslash \{k\}$
   (d) Set $F_i := F_i \backslash \{E_k\}$, for all $i \in G$,
   (e) Set $G := G \backslash \{i\}$ for all $i \in G$ for which $F_i = \emptyset$.
5. Find the incidence the thesis describes. Depending on if this is an incidence between a point and a line, a point and a conic or a point and a circle, the type of the thesis will differ.

Notice that the element 'points' in Item 4a may be ignored if the user stated that he does not want $m(..)$ assertions in the configuration (See Remark 3.3). Using the above procedure, a configuration from Cinderella is translated automatically to a configuration consisting of assertions, ready to be converted into brackets.

6.2 The Prover

The procedure explained in the previous sections was implemented in Cinderella [1], with the aid of OpenMath [5, 6, 7] and GAP [9]. A geometric theorem in Cinderella is translated into a proof in bracket algebra according to the following steps:

1. **Cinderella**: A configuration described by points and lines, some objects may have coordinates,
2. **OpenMath**: A configuration described by points and lines, as in `plangeo`, some objects may have coordinates,
3. **OpenMath**: A matrix X and a vector Y as in Equations 21 and 22, respectively,
4. **GAP**: A matrix X and a vector Y as in Equations 21 and 22, respectively,
5. **GAP**: A vector g as in Equation 17,
6. **OpenMath**: A vector g as in Equation 17,
7. **Cinderella**: A vector g as in Equation 17,
8. **Cinderella**: A string representing the proof, where the coordinates of the vector g have been translated back into their equivalents in bracket expressions.

These translations were implemented using Java, the Riaca OpenMath Library [7] and GAP [9]. The result was integrated within Cinderella and will be a part of Cinderella 2, which will be ready someday in the future with more exciting new options!

7 Examples

In this section we give some examples of geometric theorems proved using Cinderella and GAP. These theorems were created in Cinderella, 'discovered' by the internal Randomized Prover, and then, via OpenMath, given to the prover. Thus, there has been no optimization whatsoever by the user.

7.1 Pappos

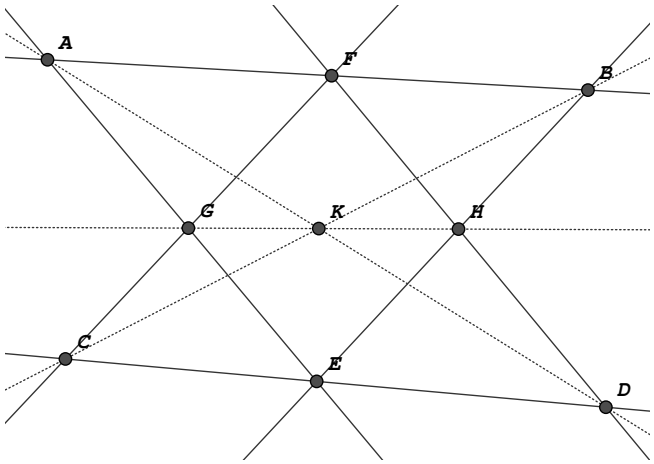

Figure 3: Pappos' Theorem

We consider Pappos' Theorem, see Figure 3. The thesis is that the lines through B and C, through A and D, and through G and H go through one point (K). The full output of the prover is as follows:

```
Conditions:
{h(C, F, G)} On line h
{h(D, F, H)} On line g
{h(B, E, H)} On line f
{h(A, E, G)} On line e
{h(C, D, E)} On line c
{h(A, B, F)} On line a

Assertion:
{m((B, C), (A, D), (G, H))} Through point K

Number of configuration equations: 200
Number of possible theses: 3

Found a proof for thesis 1. Length: 11.
Found a proof for thesis 2. Length: 11.
Found a proof for thesis 3. Length: 10.

(1) [A.C.F][B.C.G] ==  [B.C.F][A.C.G] <== {h(C, F, G)}
(1) [B.D.F][A.D.H] ==  [A.D.F][B.D.H] <== {h(D, F, H)}
(1) [B.C.E][A.B.H] ==  [A.B.E][B.C.H] <== {h(B, E, H)}
(1) [A.B.E][B.D.H] ==  [B.D.E][A.B.H] <== {h(B, E, H)}
(1) [A.B.E][A.C.G] ==  [A.C.E][A.B.G] <== {h(A, E, G)}
(1) [A.D.E][A.B.G] ==  [A.B.E][A.D.G] <== {h(A, E, G)}
(1) [B.C.D][A.C.E] ==  [A.C.D][B.C.E] <== {h(C, D, E)}
```

```
(1) [A.C.D][B.D.E] ==  [B.C.D][A.D.E] <== {h(C, D, E)}
(1) [A.B.C][A.D.F] ==  [A.B.D][A.C.F] <== {h(A, B, F)}
(1) [A.B.D][B.C.F] ==  [A.B.C][B.D.F] <== {h(A, B, F)}
------------------------------------------------------------------------
(1) [B.C.G][A.D.H] ==  [B.C.H][A.D.G] <== {m((B, C), (A, D), (G, H))}

Checking proof... done.
Result: [B.C.G][A.D.H] == [A.D.G][B.C.H]

Found first proof in 2.08 seconds, final proof in 3.27 seconds.
```

In the remainder of the section only the proofs are given, the rest of the prover output is omitted.

7.2 Pascal

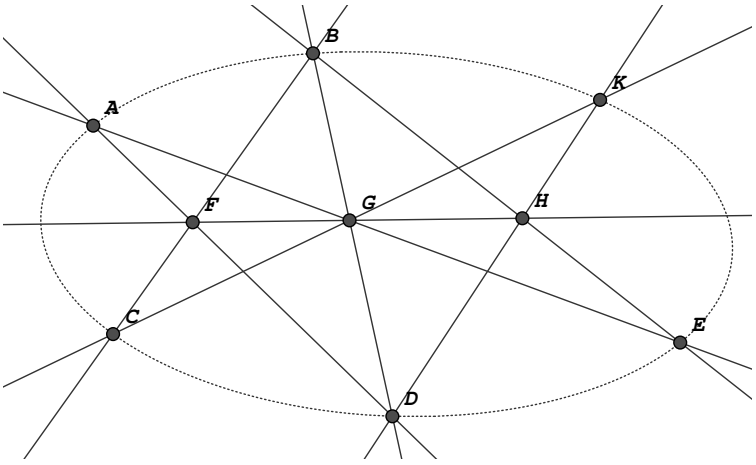

Figure 4: Pascal's Theorem

We consider Pascal's Theorem, as shown in the picture above. The thesis is that the six points A, B, C, D, E and K are on one common conic. The proof is as follows:

```
(1)              [A.C.G][B.C.K] ==  [B.C.G][A.C.K]              <== {h(C, G, K)}
(1)              [B.D.H][A.D.K] ==  [A.D.H][B.D.K]              <== {h(D, H, K)}
(1)              [B.F.G][A.F.H] ==  [A.F.G][B.F.H]              <== {h(F, G, H)}
(1)              [B.C.G][A.B.F] ==  [A.B.C][B.F.G]              <== {h(B, C, F)}
(1)              [A.B.C][B.F.H] ==  [B.C.H][A.B.F]              <== {h(B, C, F)}
(1)              [A.B.D][A.F.G] ==  [A.D.G][A.B.F]              <== {h(A, D, F)}
(1)              [A.D.H][A.B.F] ==  [A.B.D][A.F.H]              <== {h(A, D, F)}
(1)              [A.B.E][B.C.H] ==  [B.C.E][A.B.H]              <== {h(B, E, H)}
(1)              [B.D.E][A.B.H] ==  [A.B.E][B.D.H]              <== {h(B, E, H)}
(1)              [A.C.E][A.B.G] ==  [A.B.E][A.C.G]              <== {h(A, E, G)}
(1)              [A.B.E][A.D.G] ==  [A.D.E][A.B.G]              <== {h(A, E, G)}
-------------------------------------------------------------------------------------
(1) [A.C.E][A.D.K][B.C.K][B.D.E] ==  [B.D.K][B.C.E][A.D.E][A.C.K] <== {co(A, B, C, D, E, K)}
```

7.3 Miguel

Miguel states that if $ABCF$, $BCDE$, $CEFG$, $AFGH$ and $ABDH$ form five circles, then the four points D, E, G and H are on one circle, see Figure 5. This is proved as follows:

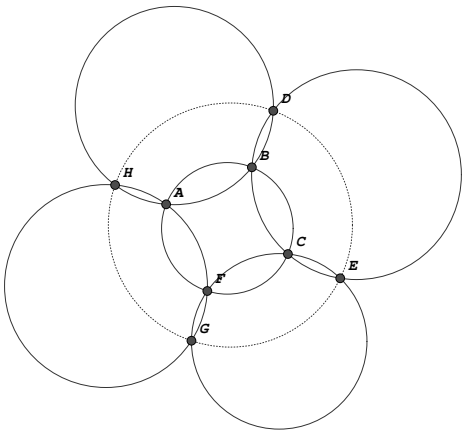

Figure 5: Miguel's six circle theorem

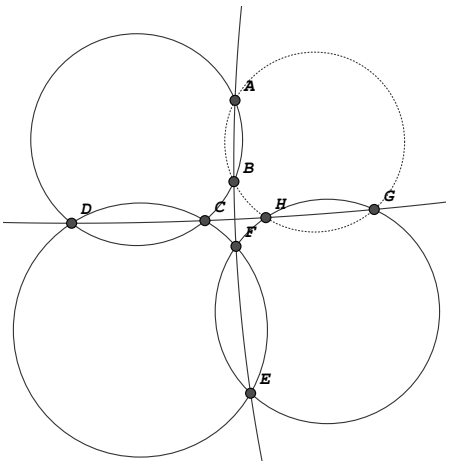

Figure 6: Another six circle theorem

```
(1) [A.F.I][F.G.H][A.H.J][G.I.J] ==  [A.F.H][F.G.I][A.I.J][G.H.J] <== {ci(A, F, G, H)}
(1) [A.F.H][A.I.J][F.G.J][G.H.I] ==  [A.F.J][A.H.I][F.G.H][G.I.J] <== {ci(A, F, G, H)}
(1) [C.E.I][C.F.J][E.G.J][F.G.I] ==  [C.E.J][C.F.I][E.G.I][F.G.J] <== {ci(C, E, F, G)}
(1) [B.C.I][B.D.J][C.E.J][D.E.I] ==  [B.C.J][B.D.I][C.E.I][D.E.J] <== {ci(B, C, D, E)}
(1) [A.B.H][B.D.I][A.I.J][D.H.J] ==  [A.B.I][B.D.H][A.H.J][D.I.J] <== {ci(A, B, D, H)}
(1) [A.B.J][A.H.I][B.D.H][D.I.J] ==  [A.B.H][A.I.J][B.D.J][D.H.I] <== {ci(A, B, D, H)}
(1) [A.B.I][B.C.F][A.F.J][C.I.J] ==  [A.B.F][B.C.I][A.I.J][C.F.J] <== {ci(A, B, C, F)}
(1) [A.B.F][A.I.J][B.C.J][C.F.I] ==  [A.B.J][A.F.I][B.C.F][C.I.J] <== {ci(A, B, C, F)}
------------------------------------------------------------------------------------
(1) [D.E.I][D.H.J][E.G.J][G.H.I] ==  [G.H.J][E.G.I][D.H.I][D.E.J] <== {ci(D, E, G, H)}
```

7.4 Six circles

Observe the geometric configuration in Figure 6. The thesis is that the points A, B, G and H are on one circle. Although this theorem is a lot like Miguel's theorem about six circles, thisone can not be proved to be true by our prover. In [10] Umbach gives a proof in 4 steps, using human reasoning about this configuration. He too, however, is unable to prove this theorem automatically.

8 Conclusion

We made it possible to obtain proofs for theorems put together in Cinderella by any user. However, we must be careful not to forget that we do not have the possibility to prove theorems false, as we did when using Gröbner bases. Sure, the prover can handle *some* theorems in projective and non-projective geometry, but often fails, as for example in Section 7.4. However, in exchange for these disadvantages, we gained the possibility to proof geometric theorems considerably faster than before. Moreover, these proofs are short and, unlike proofs made by Gröbner bases, easy to check by hand.

In the course of the project the power of OpenMath became clear. Because of the existing link between OpenMath and GAP [7] it was extremely easy to use GAP for solving linear equations without any additional programming. Although that is not such a difficult algorithm, there is no need to implement it yourself. The added advantage is that GAP will perform a lot better than our own home-made algorithm. Other advantages of the extensive

use of OpenMath include the possibility to reuse intermediate results, import geometric theorems made by hand, and in the future, use other computer algebra packages than GAP, or import geometric theorems made by other geometry programs. All this can be done rather easily, because of the clarity of the OpenMath standard.

A few questions remain open on proving geometry theorems using bracket algebra. For example, we again consider the fact that, when proving a geometric theorem, we are actually testing ideal membership (see Remark 5.3). In practice however, we are able to find a proof for a fairly large number of geometric theorems, using only linear combinations. The question why we 'get lucky' so often remains open. The power of the prover may be extended in the future, for example using methods proposed recently in the Journal of Symbolic Computation by Li and Wu [3, 4].

Cinderella and its randomized prover have been out there for a few years already, the invariant theory used exists since the twenties, and solving linear equations is not the most recent discovery either. However, the combination of these three items into a single program gives us a rather interesting result. OpenMath made it possible to do this in a structured and extendable manner, helping to achieve the goal of this project. It is now possible to create geometric theorems by pointing and clicking, and then automatically obtain a proof for that theorem. Not only can this proof be obtained very quickly, it is short and easy to check!

References

[1] The Interactive Geometry Software Cinderella. `http://www.cinderella.de`.

[2] Ulrich Kortenkamp. *Foundations of Dynamic Geometry*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1999. `http://www.cinderella.de/papers/diss.pdf`.

[3] Hongbo Li and Yihong Wu. Automated short proof generation for projective geometric theorems with Cayley and bracket algebras: I. Incidence geometry. *Journal of Symbolic Computation*, 36(5):717–762, 2003.

[4] Hongbo Li and Yihong Wu. Automated short proof generation for projective geometric theorems with Cayley and bracket algebras: II. Conic geometry. *Journal of Symbolic Computation*, 36(5):763–809, 2003.

[5] OpenMath. `http://www.openmath.org`.

[6] OpenMath Content Dictionary: plangeo1..5. This CD defines symbols for planar Euclidean geometry. `http://www.win.tue.nl/~amc/oz/om/cds/geometry.html`.

[7] The Riaca OpenMath Library. `http://www.riaca.win.tue.nl`.

[8] Jürgen Richter-Gebert. Mechanical theorem proving in projective geometry. *Annals of Mathematics and Artificial Intelligence*, 13:139–172, 1995.

[9] Martin Schönert et al. *GAP – Groups, Algorithms, and Programming*. Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, fifth edition, 1995.

[10] Andreas Umbach. Automatisches erzeugen geometrischer beweise. Master's thesis, Institut für Theoretische Informatik ETH Zürich, 2000.