mathematics

MDPI

Article

A Machine Proof System of Point Geometry Based on Coq

Siran Lei [1], Hao Guan [1], Jianguo Jiang [2], Yu Zou [1] and Yongsheng Rao [1,*]

[1] Institute of Computing Science and Technology, Guangzhou University, Guangzhou 510006, China; siranl@e.gzhu.edu.cn (S.L.); guanhao@gzhu.edu.cn (H.G.); zouyu@gzhu.edu.cn (Y.Z.)
[2] School of Mathematics, Liaoning Normal University, Dalian 116029, China; jjgbox@sina.com
[*] Correspondence: rysheng@gzhu.edu.cn

Abstract: An important development in geometric algebra in recent years is the new system known as point geometry, which treats points as direct objects of operations and considerably simplifies the process of geometric reasoning. In this paper, we provide a complete formal description of the point geometry theory architecture and give a rigorous and reliable formal verification of the point geometry theory based on the theorem prover Coq. Simultaneously, a series of tactics are also designed to assist in the proof of geometric propositions. Based on the theoretical architecture and proof tactics, a universal and scalable interactive point geometry machine proof system, PointGeo, is built. In this system, any arbitrary point-geometry-solvable geometric statement may be proven, along with readable information about the solution's procedure. Additionally, users may augment the rule base by adding trustworthy rules as needed for certain issues. The implementation of the system expands the library of Coq resources on geometric algebra, which will become a significant research foundation for the fields of geometric algebra, computer science, mathematics education, and other related fields.

Keywords: point geometry; formal method; interact theorem proving; Coq; machine proof

MSC: 68T99

check for updates

Citation: Lei, S.; Guan, H.; Jiang, J.; Zou, Y.; Rao, Y. A Machine Proof System of Point Geometry Based on Coq. *Mathematics* **2023**, *11*, 2757. https://doi.org/10.3390/math11122757

Academic Editor: Christos G. Massouros

Received: 27 May 2023
Revised: 14 June 2023
Accepted: 16 June 2023
Published: 18 June 2023

1. Introduction

The proof of geometric theorems is one of mathematics' fundamental and most challenging tasks. In recent years, reasoning and proof have gradually moved away from the traditional paper-and-pencil model. The higher thinking activities of the human brain can be mechanized, and machines can perform tedious and complex verification. Machine proofs of geometric theorems have also played an important role in mathematical research and intelligence education, with a wide range of variations and applications. Early work on machine proofs of geometric theorems was represented by Wu's method [1] as an algebraic proof method. Algebraic methods other than Wu's method include the Gröbner basis method [2] and the numerical parallelism method [3], etc. In order to support automation, these methods often have different degrees of limitations, and the polynomial transformations that transform the reasoning of geometric problems into algebra [4] also lose the geometric meaning of the proof process, which greatly reduces the readability. Subsequently, database-based search methods have also made great progress and search methods can produce traditional mathematical proofs and discover new theorems, and these are considered to be the first choice for intelligent educational software development [5]. However, the search method is not a complete method and still suffers from difficulties in composition and a lack of auxiliary quantities in reasoning, which represent major limitations.

For the proof of geometric theorems, the number of terms that need to be expressed using coordinate-based algebraic methods is often large, so there are various problems, such as high computational complexity and difficulty in understanding complex geometric propositions. The geometric algebraic method is a method that can directly calculate geometric objects (such as points, lines, etc.) as terms, which has the characteristics of a simple expression and strong geometric meaning, so the method of geometric theorem proof based on geometric algebraic principles can overcome the above difficulties to a certain extent. In particular, the advanced invariant methods of geometric algebraic methods significantly reduce the number of terms that need to be used using coordinate-based algebraic methods, but strong expertise is also required to read the proofs [6]. In addition, the combination of physics and mathematics, prime geometry, has obvious geometric significance and advances the development of geometric algebra. In 2018, Zhang proposed an intuitive new geometric algebraic system called "point geometry", which has a simple theoretical system, easy-to-understand laws and properties, and geometric algebraic methods such as prime geometry, the coordinate method, and the vector method [7,8]. For example, the proof of the triangle median theorem can be completed with only one line of proof, as in Example 1.

Example 1 (The triangle median theorem)**.** *As shown in Figure 1, the three midlines AM, BN, CP of triangle ABC intersect to a point. To prove this, we assume point A is the origin O, and we let point G be the intersection of midline BN and midline CP. Firstly, given that $B = 2P$ and $C = 2N$, it follows that $2M = B + C = 2P + C = 2N + B = 3G$. Then, we can conclude that point G is on the line AM and that $3AG = 2AM$.*

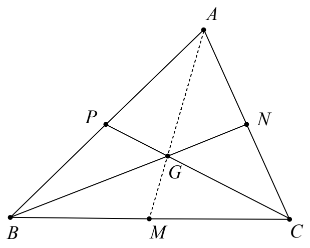

Figure 1. The geometric diagram of the triangle median theorem.

Theorem proving is a powerful technique in computer science and mathematics that expresses and verifies systems in a machine-readable form, providing significant advantages in terms of correctness and precision, particularly for complex problems. Theorem proving can reduce errors and tedious reasoning processes and provides mathematicians and computer scientists with a more intuitive and reliable means of verification, playing a crucial role in computer science and mathematics [9–12]. However, for sufficiently complex problems, complete automation is often challenging, and for complicated proof procedures, relying solely on pen and paper is no longer reliable. Therefore, interactive theorem proving has become a more reliable and efficient means of proof. There are many advanced interactive theorem provers available, such as Coq [13,14], Isabelle [15], HOL [16], etc. Among them, Coq, based on the calculus of inductive constructions, is a powerful theorem prover with strong expressive power, covering multiple domains, including program verification and mathematical theorem proving. It provides an efficient, reliable, and automated proof environment. In the field of computer science, Coq has been widely used for program verification, language design, formal semantics, security protocol analysis, program repair, and many other applications, as documented in various research studies [17–21]. Coq is also widely used in mathematics to formalize [22,23] and verify theorems and lemmas. A comparative study by Narboux et al. [24] in 2022 explored traditional pencil-and-paper proofs versus formal proof methods in high school geometry, revealing potential challenges in formalization due to different formulas and proofs. Fortunately, however, recent research has shown that Coq's advantages in terms of verifiability and interactivity have made it widely adopted for the verification of mathematical theorems. Notably, Boldo et al. [25] fully formalized the Beppo Levi theorem and Fatou's lemma using the Coq proof assistant in 2021. Additionally, Fu et al. [26] proposed a Coq-based formalization of infinitesimal calculus, utilizing it to formally verify important results such as the Newton–Leibniz formula and Taylor's theorem. Coq has also been employed to formalize various other mathematical theorems, including the topology space relation model [27], Tonelli's theorem [28], and so on. In the field of automated theorem proving for geometry, the geometry library of Coq provides a rich set of geometric concepts and theorems, covering areas such as Euclidean geometry, non-Euclidean geometry, and vector spaces, offering strong support for the automated proving of geometric theorems, as well as new possibilities for teaching and research in geometry. For instance, Julien Narboux et al. used Coq to implement the process of the area method and proposed GeoCoq, a formalized geometry theory based on the Tarski axiom system [29–32]. Furthermore, Pedro Quaresma et al. proposed a new readability criterion for formal proofs produced by automated theorem provers for geometry, inspired by their modernization of Lemoine's Geometrography, to enhance the readability of automated proofs in geometry [33]. These works have not only promoted the development of geometry but also demonstrated the importance of interactive proofs in automated theorem proving for geometry.

Our work presents a formal description and verification of a new algebraic system for point geometry using the Coq theorem prover. The result is a reliable, versatile, and extensible machine proof system for point geometry called PointGeo. The paper is divided into three main parts. The first part provides the formalization and verification of the knowledge base of point geometry, allowing geometric theorems, properties, and problems to be expressed in the Coq language and verified by the Coq kernel to ensure the validity of every step (Sections 2 and 3). The second part describes the design of the automatic proof tactics in the system. The third part discusses the overall structure and proof methods of PointGeo (Section 4).

2. The Formal Description of Definitions

The theory of point geometry provides a more straightforward notation and advanced logic that permits direct operations on geometric points as objects. This makes it easier to solve geometric issues and provides richer geometric meaning. However, due to the complexity of geometry, it is challenging to develop a complete algorithm to solve all geometry problems automatically. For this reason, it is crucial to develop an interactive and scalable system to solve geometric problems. In this section, we will first introduce the formal description of definitions in point geometry that the Coq kernel can understand.

2.1. Type and Proof

In Coq, every item has a type, and each type is also an item. There are many different types—for example, atomic types such as the natural number type nat, the integer type Z, and the real number type R, and arrow types such as $A \rightarrow B$. An arrow type can also be expressed as the type of a function. If the type of a function f is $A \rightarrow B$, i.e., f: $A \rightarrow B$, then f maps any variable of type A to a value of type B. In addition, the type operator can also construct a binary group (a, b) of type $A \times B$, where a is of type A and b is of type B.

The description of a geometric problem includes the hypotheses and the conclusion, and its initial proof state is formed as $E_p, \Gamma \vdash^? g$. E_p denotes the point geometry theoretical environment of PointGeo. Γ is the context, a sequence of statements such as $[x_1 : T_1; x_2 : T_2; \cdots ; x_n : T_n]$, which generally includes all hypotheses of the geometric proposition. g denotes the conclusion. The proof of the geometric problem needs to be advanced by applying the tactics interactively, and the process of proof may generate multiple subgoals g_i. The proof is completed if all the goals are proven, i.e., $g_i = []$.

In fact, CIC is a higher-order typed lambda calculus with natural support for inductive data types based on the Curry–Howard isomorphism, which ties lambda-term-type inference to natural deductive proofs in complete intuitionistic predicate logic. In other words, judgment is type reasoning. Specifically, intuitionism holds that a logical judgment is valid if and only if the constructing subject can verify it. This corresponds to what is represented in Coq: if t exists such that $E, \Gamma \vdash t : g$, then t is said to be a proof of g, where t is a lambda term of type g . In other words, the proof of g is to build a lambda item of type g. This is consistent with geometric algebra concepts such as Wu's method and the identity-based method for point geometry. The main idea is that if the conclusion polynomial can be constructed from the conditional polynomial, then the conclusion is proven. However, the identity-based method is incomplete for more sophisticated constraints, such as those of higher order or with quantifiers. Fortunately, PointGeo's platform, Coq, supports dependent types, which means that there is sufficient descriptive power here to describe geometric problems or rules and implement proofs of these problems.

2.2. Basic Definition of Point Geometry and Its Formal Description

To precisely describe points in point geometry theory, they are uniquely specified here by pairs of real number information regarding the point's position,

$$Point \in Set \triangleq R \times R \quad R\_R\_to\_Point : R \rightarrow R \rightarrow Point \quad (r_1, r_2) : Point,$$

where $Point$ is the type of point; Set is the type of $Point$. $R\_R\_to\_point$ is a function that maps pairs of real numbers to points. "(_ , _) " is the Coq notion of $R\_R\_to\_point$, i.e., if r_1, r_2 belong to R, then (r_1, r_2) is a point. Specifically, the origin O is described as $R\_R\_to\_point(0, 0)$, i.e., $O = (0, 0)$. In addition, functions Px and Py are designed to extract the first and second components of the point, respectively, and they indicate that if $P = (r_1, r_2)$, then $Px(P) = r_1$, $Py(P) = r_2$; in other words, for any point P, we have $P = (Px(P), Py(P))$.

The basic operations of point geometry depend on the choice of the origin. The relationship between the coordinates of the points relative to the origin represents the relationship between the points. The definition of addition and scalar multiplication in point geometry is as follows.

Definition 1 (Addition of points)**.** *If $A = (x_A, y_A)$, $B = (x_B, y_B)$, and $C = (x_A + x_B, x_A + y_B)$, then the addition of A and B is C, which is denoted by $A + B = C$.*

Definition 2 (Scalar multiplication)**.** *If $A = (x, y)$, $B = (\lambda x, \lambda y)$, then the scalar multiplication of A is denoted by $B = \lambda A$.*

Corresponding to the addition and scalar multiplication of points in PointGeo are the binary functions Pplus and Pmults:

$$Pplus((p_1 : Point), (p_2 : Point)) \triangleq ((Px(p_1) + Px(p_2)), (Py(p_1) + Py(p_2)))$$
$$Pmults(\lambda : R), (p : Point) \triangleq ((\lambda * Px(p)), (\lambda * Py(p)))$$

$Pplus$ and $Pmults$ are defined by applying the addition and multiplication operations, respectively, to the position coordinates of the point, Px and Py, through the R_R_to_Point function. In the scope of points, the function symbols for the addition and scalar multiplication of points are '$+$' and '$*$', respectively, written '$p_1 + p_2$' and '$\lambda * p$', where p, p_1, and p_2 are of type $Point$ and λ is of type R. The addition and scalar multiplication of points inherit the relevant laws of the real numbers, such as the law of union and distribution, and these laws have been verified in Coq. The source code of basic rules is given in the file Basics.v (the components of this work will be discussed in Section 5). In addition, the source code for this work can be obtained from https://github.com/RanranL/Pointgeo (accessed on 6 May 2023).

Additionally, the monomial function Pneg, defined as 'If $P = (x, y)$, then $Pneg(P) = (-x, -y)$', represents the point's negative form. Based on the Pneg, the subtraction of points is defined as follows:

$$Pminus(P_1, P_2) \triangleq Pplus((P_1), Pneg(P_2))$$

The function symbol for Pminus is '$-$', written '$p_1 - p_2$'. For instance, for points p_1 and p_2, the expression '$p_1 - p_2$' can be considered equivalent to '$p_1 + (-p_2)$'. In vector geometry, $\overrightarrow{AB}$ is equivalent to $\overrightarrow{OB} - \overrightarrow{OA}$, where O is the origin, $\overrightarrow{OB}$ is equivalent to B in point geometry, and $\overrightarrow{OA}$ is equal to point A in point geometry, so vector $\overrightarrow{AB}$ is equivalent to '$B - A$' in point geometry. It follows that vector geometry and point geometry are interconvertible.

The outer product of the points is also defined by subtraction.

Definition 3 (Outer product)**.** *The outer product of the two points of the convention is* $AB = B - A$.

The functions Vec and OutP represent vectors and two-point outer products, respectively, of type '$Point \to Point \to Point$':

$$Vec(P_1, P_2) \triangleq Pminus(P_2, P_1) \quad OutP(P_1, P_2) \triangleq Pminus(P_2, P_1)$$

The outer product of points also satisfies the law of union and distribution, and its function symbol is '$\times$', written '$p_1 \times p_2$'. Furthermore, it is clear from the definition that the outer product of two points and the subtraction of points, vectors, and points are equivalent:

$$Pminus(P_1, P_2) \Leftrightarrow Vec(P_2, P_1) \Leftrightarrow Out(P_2, P_1)$$

Definition 4 (Outer product of three points)**.** *If* $A = (x_A, y_A), B = (x_B, y_B), C = (x_C, y_C)$, *then* $ABC = x_A y_B + x_B y_C + x_C y_A - x_A y_C - x_B y_A - x_C y_B$.

The outer product of three points is also the area of the triangle formed by the three points with the sign, and its intuitive geometric meaning is as follows: 'In the right-handed coordinate system, if the triangle vertices A, B, and C are in anti-clockwise order, then $ABC > 0$; otherwise, $ABC < 0$. If A, B, and C are collinear , then $ABC = 0$'. The outer product of three points is expressed using the ternary function 's':

$$s(p_1 : Point)(p_2 : Point)(p_3 : Point) \in \mathbb{R} \triangleq Px(p_1) * Py(p_2) + Px(p_2) * Py(p_3) +$$

$$Px(p_3) * Py(p_1) - Px(p_1) * Py(p_3) - Px(p_2) * Py(p_1) - Px(p_3) * Py(p_2)$$

For addition in point geometry, not only the scalar multiplication and the outer product but also the inner product is an operation that satisfies the distributive law, which is defined as follows.

Definition 5 (Scalar product)**.** *If* $A = (x, y), B = (u, v)$ *then* $A \cdot B = ux + vy$.

The binary function Scalarprod represents the scalar product of two points; the function symbol is '$\cdot$'.

$$Scalarprod((p_1 : Point), (p_2 : Point)) \in \mathbb{R} \triangleq Px(p_1) * Px(p_2) + Py(p_1) * Py(p_2)$$

For a given point P, the square of P is defined using the scalar product of the point with itself, denoted as $Scalarprod(P, P)$. We represent the square of point P as $Pnorm\_sqr(P)$, which is later expressed as $\sqrt{P^2}$ for simplicity of description; $Pnorm(P)$ means, i.e., $|P|$; $distance(P_1, P_2)$ means the length of the line segment P_1P_2. The point square, the modulus, and the length of the line segment at two points are defined formally as

$$Pnorm\_sqr(P) \triangleq Scalarprod(P, P)$$

$$Pnorm(P) \triangleq Sqrt(Pnorm\_sqr(P))$$

$$distance(P_1, P_2) \triangleq Sqrt(Pnorm\_sqr(Pminus(P_2, P_1)))$$

One innovative approach to solving difficulties with angles is to multiply points with complex numbers; the definition of the multiplication of complex and point is as follows.

Definition 6 (Multiplication of complex and point). *If $A = (x, y)$ in the Cartesian coordinate system and the complex $\alpha = u + vi$, then define $\alpha A = uA + i(uA)$.*

For a complex number, $z = u + iv$, where '$+i$' is a binary function of type $R \to R \to C$. Similar to Px and Py, there is $Cre(z) = u$ and $Cim(z) = v$. CPmult denotes the complex number multiplied by a point:

$$CPmult(z, p) \triangleq ((Cre(z) * Px(p) - Cim(z) * Px(y)), (Cre(z) * Py(p) + Cim(z) * Px(y)))$$

3. Formal Description and Proof of Geometric Propositions

This section will give a brief introduction to the proof model. We start by describing the geometric construction, a set of geometric predicates used to describe a geometric proposition.We then continue with a group of rules for proofs and finally show how to complete a proof for a geometric proposition.

3.1. Geometric Predicates

Geometric objects come first, followed by the relationships between them. Geometric predicates are intuitive predicates used to express static relations. Complex geometries are often constructed from simple geometries, and in order to describe geometric propositions, this subsection describes the construction of the main geometric predicates in point geometry.

Starting from the collinearity of three points, if there exists a real number λ such that $\overrightarrow{P_1P_2} = \lambda\overrightarrow{P_1P_3}$, then, in vector geometry, it can be concluded that the three points P_1, P_2, and P_3 are collinear, so the formal representation of vector collinearity is $Vcollin$.

Predicate 1 (Vcollin). $\exists\lambda : R, \text{Vec}(P_1, P_2) = \lambda * \text{Vec}(P_1, P_3)$.

Since $Vec(P_1, P_2)$ is equivalent to $Pminus(P_2, P_1)$, the collinear formalism of three points in point geometry is described as $collin$.

Predicate 2 (Collin). $\exists\lambda : R, P_2 - P_1 = \lambda * (P_3 - P_1)$.

Note that if P_1 is the origin O, then clearly O, P_2, and P_3 are collinear if and only if there exists a λ with $P_2 = \lambda * P_3$.

If two lines AB and CD intersect, as long as there is the equation $u + v = r + s$ such that $uP_1 + vP_2 = rP_3 + sP_4$, there is P_F, P_G satisfying $(u + v)P_F = uP_1 + vP_2 = rP_3 + sP_4 = (r + s)P_G$. In fact, it is easy to see from the above linear combination of two points that as long as $u + v = r + s \neq 0$, the points P_F and P_G are on the lines P_1P_2 and P_3P_4, which also implies $P_F = P_G$. Therefore, $P_F(P_G)$ is the intersection point of P_1P_2 and P_3P_4. In conclusion, the information of the intersection point P_F of two lines can be derived from the linear combination of four points, such as '$P_F = (u/(u + v))P_1 + (v/(u + v))P_2$'. $Xcollin$ is a geometric predicate describing the intersection point of two lines.

Predicate 3 (Xcollin). $u + v = r + s \to u * P_1 + v * P_2 = r * P_3 + s * P_4 \to (r + s) * P_X = r * P_3 + s * P_4$.

Consider the geometric meaning of the point geometry equation $P_1 + P_2 = \lambda P_3$ when $\lambda = 1$, 2, 3 as $P_1OP_2P_3$ as a parallelogram, P_3 as the midpoint of P_1P_2, and P_3 as the center of gravity of triangle OP_1P_2, respectively; then, the parallelogram ($Parall$), the midpoint ($Midpoint$), and the center of gravity of triangle (Ptr_median) can be constructed.

Predicate 4 (Parall). $P_1 + P_3 = P_2 + P_4$.

Predicate 5 (Midpoint). $P_1 + P_2 = 2 * P_3$.

Predicate 6 (Tra_median)**.** $3 * P_G = P_1 + P_2 + P_3$.

Now, we can specify the geometric proposition that we wish to verify using geometric predicates. Geometric figures and propositions contain a great deal of information, so it is only necessary to add the relevant information to the context. The relevant hypothesis and conclusion in Example 1 are shown as follows.

- Hypothesis: Point P is midpoint of segment AB.
- Hypothesis: Point N is the midpoint of segment AC.
- Hypothesis: Point M is the midpoint of segment BC.
- Hypothesis: BN and CP intersect at point G.
- Conclusion: Three points, A, B, and C, are collinear.

The conclusion in Example 1 is equivalent to proving that the points A, G, and M are collinear , i.e., they satisfy the 'collin (A, G, M)' relation. The description of Example 1 is as follows.

```
Example First_exam : ∀ A B C M N P G : Point,
A = O → Midpoint P A B → Midpoint N A C → Midpoint M B C → Xcollin N
    B P C G → collin A G M.
```

3.2. Rules and Proof Mode

Thus far, we have seen how to describe geometric propositions in Coq based on the definition of point geometry and geometric predicates. However, this is insufficient to prove a geometric proposition. Additional reliable proof rules and proof tactics are required. We will show how to prove a geometric proposition after giving some typical rules.

3.2.1. Rules

Let us first consider the rule *Peq* for determining the equality of two points.

Rule 1 (Peq)**.** $\forall P_1\ P_2 :\ Point\ , P_1 = P_2 \leftrightarrow Px(P_1) = Px(P_2) \wedge Py(P_1) = Py(P_2)$.

The Peq rule, which expands the equality in point geometry to the equality of the corresponding real coordinates, is used to verify the validity of fundamental rules. Take the verification of the exchange law of the addition of points (Rule 2) as an example.

Rule 2 (Padd_comm)**.** $\forall P_1 P_2 :\ Point\ , P_1 + P_2 = P_2 + P_1$.

Proof of Rule 2. According to the rule Peq, proving that the equation $P_1 + P_2 = P_2 + P_1$ holds is equivalent to proving that $Px(P_1 + P_2) = Px(P_2 + P_1) \wedge Py(P_1 + P_2) = Py(P_2 + P_1)$. □

Furthermore, the origin in point geometry is a rather particular point, and there are several rules about the origin that are easily verified by Peq's rule.

$$Pneg\_0 : -O = O \quad Pmult\_0 : n * O = O \quad Pplus\_0\_l : O + P = P$$

$$Pplus\_0\_r : P + O = P \quad Pminus\_0\_l : O - P = -P \quad Pminus\_0\_r : P - O = P$$

With the basic rules of point geometry as a basis, some more complex geometric propositions can be proven directly based on points, without expanding the coordinates of the points. For example, the co-side theorem, expressed by the product of three points, bridges point geometry and the area method by converting the ratio of triangle areas into the ratio of sides. A variation of the co-side theorem is offered here to simplify the demonstration of the pertinent geometric propositions.

Rule 3 (CommonEdg)**.** *If F is the intersection of lines AB and CD and $F \neq B$, then we have* $\frac{|ACD|}{|BCD|} = \frac{|AF|}{|BF|}$.

Proof of Rule 3. If F is the intersection of lines AB and CD, and $F \neq B$, then for all u, v, r, s, as long as $u + v = r + s$, we have $(u + v)F = uA + vB = rC + sD$. We also have the equation $uACD + vBCD = 0$ when we use CD as the outer product. Furthermore, since $(u + v)F = uA + vB$, it follows that $u(F - A) + v(F - B) = 0$. Therefore, we can conclude that $\frac{|ACD|}{|BCD|} = \frac{|AF|}{|BF|}$ based on the previous deductions. □

Other laws, such as the laws of addition, scalar multiplication, outer products, the complex multiplication of points, and the logical relationships between geometric predicates, are already established in Coq. Moreover, the user may add rules as needed, and when the rules are verified in point geometry theory, they can be used to prove more geometric propositions.

3.2.2. Proof Mode

This part aims to elaborate on a proof mode for a geometric proposition, which can be accomplished by interacting with tactics. The reliability of this proof process is guaranteed by Coq, an interactive proof assistant tool whose kernel ensures the correctness of tactics. To assist with the development of proofs, Coq offers a robust decision process and a library of automated proof tactics. Several proof tactics commonly used in interactive proof development are listed below.

- `intros`: Adds assumptions to the current context.
- `apply H`: Applies the hypothesis or rule `H`.
- `rewrite`: A tactic for replacement.
- `unfold` *def*: Used to expand the definition.
- `destruct`: For discussion by the situation.

Now, consider the proof of Example 1; the state before the start of the proof is shown below.

```
1 goal
-------------------------------------------------------------------------
∀ A B N C P G M : Point, A = O → Midpoint P A B → Midpoint N A C →
    Midpoint M B C → Xcollin N B P C G → collin A G M.
```

The area above the dashed line represents the context Γ, where $\Gamma = []$ at this point, and the area below the dashed line represents the geometric proposition to be proven. Firstly, with the exception of the conclusion '`collin A G M`', which remains to be proven, all other statements will be introduced into the context through the tactic '`intros`'. The left-hand side of the proof state presented below is the result of using the '`intros`', and this state is the initial proof state. Then, with the following tactics, the geometric predicate will be expanded, and the origin will be eliminated according to the following rules.

```
unfold Midpoint, Xcollin, collin in * ;
rewrite H, Pplus_O_l in * ;
do 2 rewrite Pminus_O_r.
```

The proof state enters the geometric-algebraic state, as in the right-hand side of the proof state presented below.

```
                            - A, B, N, C, P, G, M: Point
- A, B, N, C, P, G, M :     - H : A = O
    Point                   - H0: B = 2 * P
- H : A = O                 - H1: C = 2 * N
- H0: Midpoint P A B        - H2: B + C = 2 * M
- H1: Midpoint N A C        - H3: ∀ u v r s: R, (u + v) = (r + s) → u * N
- H2: Midpoint M B C            + v * B = r * P + s * C → (r + s) * G = r
- H3: Xcollin N B P C G         * P + s * C
--------------------------  ----------------------------------------------
collin A G M                ∃ m, G = m * M
```

For example, the equation corresponding to hypothesis H_0 is $2P = A + B$. Using the rewrite tactic, according to hypothesis H, replacing the origin A with O, there is $2P = O + B$. Moreover, using the Pplus_O_l rule, the rewrite tactic can replace $O + B$ with B, yielding H_0.

The current goal is to introduce a term of G as a proof according to the context. From the geometric proposition, it is clear that point G is the intersection of the lines NB and PC. However, there is no expression in the context that directly expresses the relationship between G and M. It is necessary to establish the equidistant relationships among points N, B, P, and C in order to deduce the expression for point G based on H_3. From assumptions H_0, H_1, and H_2, we know that N, B, P, and C satisfy the equation $HX : 2N + B = 2P + C$. The addition of this equation is performed by the tactic assert below.

```
assert HX: 2 * N +1 * B = 2 * P + 1 * C).{rewrite <- H1, H0. Psolver.}
```

The content in "{ }" is proof of HX, and the tactic 'Psolver' is a decision procedure that can solve linear arithmetic problems with points, which will be introduced in detail in Section 4. From HX and H_3, we know that $3G = 2P + C$, and from H_0 and H_2, we know that $B + C = 2M$, so it is easy to determine that $m = 2/3$; the conclusion can be proven, and this part of the proof is realized by the following tactics.

```
apply H3 in HX; auto. exists (2/3); Psolver.
```

The preceding proof is similar to a step-by-step proof and is hence time-consuming. In the next part, we will introduce numerous automatic tactics to make proofs easier.

4. Proof Tactics

The previous sections focused on the formal description and verification of point geometry theory, including definitions, geometric predicates, rules, and proof models for geometric propositions. We can also design automatic solving and programmable tactics to prove specific types of complicated geometric problems. This section details the functions and methods of several essential geometric proof tactics.

4.1. Oelim

4.1.1. Origin

In point geometry, any point can be chosen as the origin, and the geometric relations between points can be constructed with reference to the origin, which is the cleverness of point geometry. For example, $Parall(P_1P_2P_3P_4)$ means parallelogram $P_1P_2P_3P_4$, which means that wherever the origin is, as long as the four points P_1, P_2, P_3, and P_4 satisfy $P_1 + P_2 = P_3 + P_4$, it means that the quadrilateral $P_1P_2P_3P_4$ is a parallelogram. In particular, if P_1 is taken as the origin, then $P_2 = P_3 + P_4$. The flexibility in the choice of origin makes the point geometry solution process more concise.

For a geometric proposition, once the origin is determined, all points set as the origin in this proposition can be eliminated by the six rules about the origin in Section 3.2.1.

Example 2. *Below is an example of origin elimination.*

$$P_1 = O, Parall\ P_1\ P_2\ P_3\ P_4 \vdash P_2 = P_3 + P_4$$

1	$P_1 = O$	*Premise*
2	$Parall\ P_1\ P_2\ P_3\ P_4$	*Premise*
3	$P_1 + P_2 = P_3 + P_4$	$unfold\ Parall, 2$
4	$O + P_2 = P_3 + P_4$	$rewrite\ 1, 3$
5	$P_2 = P_3 + P_4$	$rewritePplus\_0\_l, 4$

4.1.2. Method of Eliminating the Origin

OELIM is a tactic that automates a process similarly to the elimination of the origin in Example 2. If the current proof state is st, OELIM unfolds the geometric predicate's definition, introduces the conditions and assumptions into the context, and then eliminates the origin. The proof state after the elimination of the origin by the OELIM tactic is denoted as st^{eO}. The method is as in Algorithm 1.

Algorithm 1 OELIM

```
Input: Current proof state $st$.
Output: Algebraic proof state $st^{eO}$
 1: Expand the geometric predicate and introduce the hypothesis into the context.
 2: if $\Gamma \vdash (H :?P = O)$ then
 3:     for each $i \in \{1, \dots, m+1\}$ do          ▷ $m$ is the number of hypotheses
 4:         if $P \in V(Hp_i)$ then                      ▷ Conclusion $g$ is $Hp_{m+1}$
 5:             term$(Hp_i)$ := term$(Hp_i)(P/O)$;
 6:             Elim O with rules of O; return ;
 7:         end if
 8:     end for
 9: else
10:     Elim O with rules of O; return ;
11: end if
```

If the current proof state is denoted as st, with Γ as the current context and the geometric proposition $P : \forall(v_1 : T_1)(v_2 : T_2)\cdots(v_n : T_n), (Hp_1 \to Hp_2 \to \cdots \to Hp_{m+1})$ as the current goal, then the current state is denoted as

$$st = E_p, \Gamma \vdash^? \forall(v_1 : T_1)(v_2 : T_2)\cdots(v_n : T_n), (Hp_1 \to Hp_2 \to \cdots \to Hp_{m+1})$$

where Hp_{m+1} is the proof goal g of the geometric proposition. First, let $\widetilde{Hp_i}$ denote the point geometric algebraic state of each Hp_i after unfolding the geometric predicate, where $\widetilde{Hp}_{m+1}$ denotes the point geometric algebraic form $\widetilde{g}$ of conclusion g; then, step 1 unfolds the geometric predicate by applying the '`unfold`' and introduces the hypothesis into the context by using the '`intros`', so that the proof state becomes

$$\widetilde{st} : E_p, \Gamma :: [(v_i : T_i)]_{i=1}^{n} :: [(H_j : \widetilde{Hp_j})]_{j=1}^{m} \vdash^? \widetilde{g}$$

If a point is given as the origin in the hypothesis, i.e., there is an item "$H :?P = 0$" in the context, then we use the rewrite tactic to replace all occurrences of point P in the context with point O and apply origin elimination rules to remove the origin. If not, then we apply the rule to eliminate the origin directly. At this point, if we use ${Hp_i}^{eO}$ to denote the hypothesis and g^{eO} to denote the conclusion, then we have $\Gamma^{eO} = [(v_i : T_i)]_{i=1}^{n} :: [(H_j : \widetilde{Hp_j}^{eO})]_{j=1}^{m}$, and the proof state after applying the OELIM tactic is as follows:

$$st^{eO} = E_p, \Gamma :: \Gamma^{eO} \vdash^? g^{eO}$$

It is easy to see that the origin of all terms in the algebraic state of st^{eO} has been eliminated. OELIM uses tactics such as *intros*, *unfold*, and *rewrite* to introduce the hypothesis into the context, expand the geometric predicate, and apply the origin-related rules to replace the eliminated origin so that the proof of the topic enters the simplified algebraic state, i.e., $st \Longrightarrow st^{eO}$.

4.2. Linear Point Geometry Decision Procedures Based on Real Numbers

Positivstellensatz generalizes Hilbert's zero theorems for a finite set S of polynomials, and it searches for the refutation of $Cone(S)$ to prove a nonlinear arithmetic problem [34]. PSOLVER is a tactic used to solve point geometry problems using a Positivstellensatz-based decision procedure on linear algebra over the field of real and rational numbers, where the transformation of real numbers to points is implemented by *Peq* (Rule 1).

The method of PSOLVER is implemented by Algorithm 2. Firstly, the method unfolds all not ($\neg$) in the current state, and its resultant state is represented by $\mathrm{UF}_{not}(st)$. Then, PFACTOR initiates the recursive expansion of the formula structure. Firstly, if the goal to be proven, g, has the pattern '$(\_ \to \_)$', the 'intros' tactic is used to introduce a new declaration into the context, resulting in a new proof state denoted as '$Intro(st)$' in this context. Let APPL$Peq(st)$ denote the application of the Peq rule to expand all point geometric equalities in the proof state st. Therefore, the new proof state obtained by applying the Peq rule to the current state is 'APPL$Peq(Intro(st))$'. For the context Γ in the current proof state, for all declarations with type pattern '$(\_ \wedge \_)$', '$(\_ \vee \_)$' or '$(\_ \leftrightarrow \_)$' that need to be discussed on a case-by-case basis, the method uses the destruct tactic , such that $id^{\sim}$ denotes the new sequence of declarations generated after applying the destruct tactic. The state obtained by applying the Peq rule is represented as $\mathrm{APPL}_{Peq}(\mathrm{Intro}(st))$. In the same way, the problem is solved by deconstructing and transforming the goal to be proven with patterns '$(\_ \leftrightarrow \_)$', '$(\_ \wedge \_)$' and '$(\_ \vee \_)$', and the proof state obtained is a real algebraic state.

Algorithm 2 PSOLVER

Input: Current proof state st.
Output: If $st = E_p, \Gamma \vdash^? []$, then the goal is proven; otherwise, do nothing

```
 1: Procedure PFACTOR ($st$)
 2: $st = \mathrm{UF}_{not}(st)$;
 3: if $g$ in pattern of $(\_ \to \_)$ then
 4:     $st = \mathrm{APPL}_{Peq}(\mathrm{Intro}(st))$;
 5:     PFACTOR ($st$);
 6: else
 7:     for each $id$ that $\Gamma \vdash id : Point$ or $id : (\_ \wedge \_)$or $id : (\_ \vee \_)$or $(\_ \leftrightarrow \_)$ do
 8:         $id^{\sim} = \mathrm{Destr}(id)$;
 9:         if $g$ is in pattern of $(\_ \leftrightarrow \_)$ or $(\_ \wedge \_)$ then
10:             $g_i = SP(g)_i$;
11:             $st = APPL_{Peq}(st(g/g_i))$;
12:             PFACTOR ($st$);
13:         else if $g$ is in pattern of $(\_ \vee \_)$ then;
14:             $st = st(g/\,\mathrm{LAR}(g))$;
15:             PFACTOR ($APPL_{Peq}(st)$);
16:         end if
17:     end forreturn ;
18: end if
19: Procedure PSOLVER ($st$)
20: PSOLVER(PFACTOR($st$));
```

4.3. A Proof Method for a Hilbert Intersection Problem

One of the proofs of Example 1 mentioned in Section 3 is tedious. In fact, PointGeo can use only one tactic, HilbertX, to complete the proof and return the solution process. Designing automatic proof tactics such as this can enhance the readability of machine proofs and improve the efficiency in solving them.

4.3.1. Framework of HILBERTX

HILBERTX is an incomplete Hilbert intersection class proof method as in Algorithm 3. For a Hilbert intersection class problem P, the HILBERTX tactic deduces the hidden intersection information and returns the solution process information to efficiently assist in solving the geometric problem. The proof method of the HILBERTX tactic is shown in Figure 2.

HILBERTX solves the problem with intersections and returns the solution information, which is output on lines 1 and 8, respectively. The tactic accepts four optional parameters, a, b, c, and d. The four parameters are entered if and only if there are two intersections in the problem. For a geometric proposition, OELIM is first invoked to eliminate the origin and transform the proof state into a geometric algebraic state. Then, the hidden information term is obtained by HILXCOL_K (Section 4.3.2) and automatically solved by the point geometric linear arithmetic decision process. If the geometric proposition is not solved, the st^{eO} proof state is retained.

Algorithm 3 Framework of HILBERTX

```
Input: Current proof state $st$ and optional parameters $a\ b\ c\ d$.
Output: "Done" if the goal is proven, $st^{eO}$ otherwise
 1: $st^{eO} :=$ Oelim($st$);                                  ▷ return information
 2: if type of $a = ltac\_No\_arg$ then
 3:     $st^P :=$ Psolver $(st^{eO})$;
 4:     if $st^P = E_p, \Gamma \vdash^? []$ then return "Done";
 5:     else return ;
 6:     end if
 7: else
 8:     $st^K :=$ HILXCOL_K $(st^{eo}, a, b, c, d)$;
 9:     $st^{P'} :=$ PSOLVER$(st^K)$;
10:     if $st^{P'} = E_p, \Gamma \vdash^? []$ then return "Done";   ▷ return information
11:     else return ;
12:     end if
13: end if
```

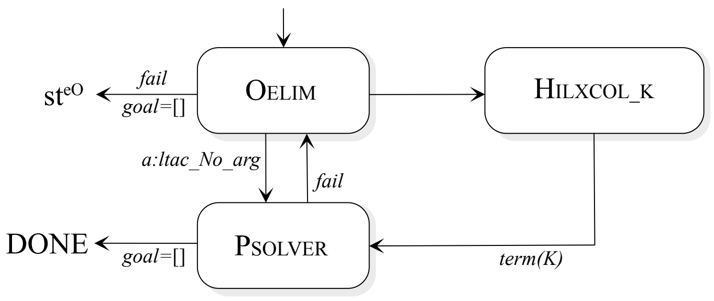

Figure 2. Flowchart of HilbertX proof method.

4.3.2. Generation of Intersection Information Items

If the line AB intersects the line CD, then their intersection K is unique and fixed, and the $Xcollin$ rule can be obtained from the algebraic relations of A, B, C, and D to their algebraic relations with K, i.e., the information term $term_k$ of the intersection K of AB and CD. The algebraic relation of the four points here means finding a set of values of u, v, r, and s such that $u + v = r + s \to u * A + v * B = r * C + s * D$ holds and the information term $term_k$ is the equation $(r + s)K = r * C + s * D$ or $(u + v)K = u * A + v * B$ obtained by $Xcollin$.

4.3.3. Step-by-Step Proof and Proof by Tactic

Example 3. *In parallelogram ABCD, M is the midpoint of AB and P is the intersection of AC and BD. Then,* $\overrightarrow{AC} = 3\overrightarrow{AP}$. *The predicates corresponding to the conditions and conclusions are represented as follows.*

- *parallelogram ABCD: Parallel A B C D.*
- *M is the midpoint of AB: Midpoint M A B.*
- *P is the intersection of AC and BD: Xcollin A C M D P.*
- $\overrightarrow{AC} = 3\overrightarrow{AP}$: *Vec A C* $= 3 *$ *Vec A P*

Since Example 3 is a Hilbert intersection class problem, HilbertX can prove this proposition. The proposition description, proof, and proof process information of Example 3 are as follows.

```
Lemma Para_exam: ∀ A B C D M P : Point,
A=O → Parall A B C D → Midpoint M A B → Xcollin A C M D P → Vec A C =
    3 * Vec A P.
Proof. HilbertX 2 1 2 1. Qed.
------------------------------------
After unfolding predictions and eliminating the O, we obtain the following
    hypotheses:
```

```
H: A = O.
H0: C = B + D.
H1: B = 2 * M.
H2: ∀ u v r s : R,
u + v = r + s → u * O + v * C = r * M + s * D → (r + s ) * P = r * M + s
    * D.
We begin to prove the goal: C = 3 * P.
Firstly, we construct the equation Hx: 2 * O + 1 * C = 2 * M + 1 * D.
By the rule of Xcollin, we obtain the equation of P: Hx: (2 + 1) * P = 2 *
    M + 1 * D.
Finally, the proposition can be proven by the algebraic operations of
    points. Done.
```

The proof of Example 1 mentioned in Section 3.2.2 is tedious and is equivalent to a step-by-step proof. In fact, it is also a Hilbert intersection class problem, which can be proven using only one tactic, HilbertX, which can also return information about the solution process. Table 1 compares the code lengths of the step-by-step proof and the tactic proof for Example 1 and Example 3, and it is clear that the number of proof lines required for the step-by-step proof using the rule is 14, while the tactic requires only one line, which is more convenient and faster.

Table 1. The number of proof lines required for a step-by-step proof using the rule is 14, and the proof can be completed in only one line using the tactic, which also returns information about the solution process.

Geometry Propositions	Proof Mode Line Numbers	Tactic Line Numbers
First_exam	7	1
Para_exam	14	1

Consider the comparison of the lengths of the step-by-step proofs and the tactic proofs for the two proof topics, as shown in the following table.

5. Proof System

We propose a point geometric machine proof system, PointGeo, which is capable of integrating multiple geometric-theorem-proving methods and allowing users to add content as needed. Based on a verified point-based geometric theory framework, PointGeo can prove any solvable geometric proposition in point geometry. The design of this system aims to provide an efficient and accurate method of proof, and it provides a new tool and perspective for research in the field of point geometry.

5.1. Structure of the System

PointGeo consists of five main parts, namely basic definitions, fundamental properties, and common rules of point geometry; geometric predicates; proof tactics; and propositional proofs, which are distributed as shown in Table 2.

Table 2. The content of PointGeo.

PointGeo	Content	Line Numbers	Outline
PF.v	Basic definition	95	32
Pbasics.v	Properties	304	67
Plemma.v	Rules	273	61
Pconstruct.v	Predicates	76	26
PTactic.v	Proof tactics	316	41
Proj.v	Machine proof	1783	188

5.2. Proof Method

PointGeo is a scalable proof system that accepts a variety of methods. It is possible to prove geometric propositions based on the basic theory of point geometry, such as Hilbert-intersection-like problems and geometric problems on angular rotations, similar to Morley's theorem.

Example 4 (Morley's theorem)**.** *In ΔABC, let P, Q, and R be the intersection points of the trisection of two angles of the triangle. Then, triangle ΔPQR is an equilateral triangle.*

In order to prove the conclusion of Morley's Theorem, which states that $e^{\frac{i\pi}{3}}(Q-P) = R-P$, we can introduce the notation $e^{\frac{i\pi}{3}} = \omega$. The objective is then to demonstrate that $\omega Q - R = \omega^2 P$. The description of Morley's theorem is presented as follows, and its proof is also completed in PointGeo.

```
Theorem Morley :
∀ (A B C' P Q R' : Point) (a b r : R) (w u v : C),
A = O → triangle0 A B C' (3 * a) (3 * b) → triangle0 A B R' a b →
    triangle0 A Q C' a (PI - a - r) → triangle0 B C' P b r → w = Cexp (0
    +i (PI / 3) → u = Cexp (0 +i (- 2 * a) → v = Cexp (0 +i (- 2 * b) →
    r = (PI / 3 - a - b) → (CPmult w Q) - R' = CPmult (w ^ 2) P.
Proof.... Qed.
```

The definition of the outer product of three points in point geometry theory relates the area to the points. This leads to the co-side theorem, an essential theorem in the area method, in point geometry theory. Therefore, PointGeo supports the use of the area method to prove geometric propositions. Solving geometric propositions using the area method has higher readability and can avoid complex parameters. This feature is well illustrated in the proof of Pappus's theorem, as shown in Example 5.

Example 5 (Pappus' theorem)**.** *Let points A, B, and C be collinear; D, E, and F be collinear; the lines AE and BD intersect at P; AF and CD intersect at Q; and CE and BF intersect at R. Then, P, Q, and R are collinear, as in Figure 3.*

The CommEdg rule derived from the properties of the outer product provides a proof rule for the area-based proof of Pappus's theorem. For example, as shown in Figure 3b, if RC and EQ intersect at point E, i.e., if the condition $Xcollin(R\ C\ E\ Q)$ is satisfied, then applying the CommEdg rule yields the equation $\frac{Rabs(sREQ)}{Rabs(sCEQ)} = \frac{distanceER}{distanceEC}$, i.e.,

$$\frac{|REQ|}{|CEQ|} = \frac{|RE|}{|CE|} : Xcollin\ R\ C\ E\ Q\ E \rightarrow sCEQ \neq 0 \rightarrow \frac{Rabs\ (s\ REQ)}{Rabs\ (sCEQ)} = \frac{distance\ ER}{distance\ EC}.$$

Similarly, the new problem-solving information for (c)~(n) in Figure 3 can be inferred through CommEdg. The formal description and a more detailed formal proof of the Pappus theorem have been completed in PointGeo.

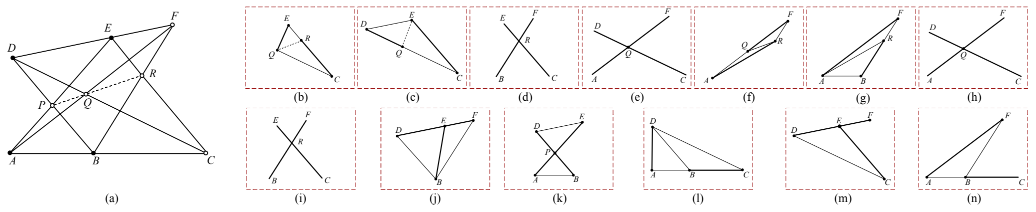

Figure 3. Diagram of the Pappus theorem and the splitting diagram of the ratio of the area to the sides of all the sides used in the proof. (**a**) A diagram illustrating Pappus' theorem. (**b**–**n**) Subgraphs derived from (**a**), representing the graphs associated with the common-edge theorem utilized in each proof step.

Propositional descriptions and proofs based on point geometry theory can also be converted into vectors and proven using the vector method. However, the rules related to vectors are not yet complete. Fortunately, it can be solved quickly by the PSOLVER using the real-related decision process.

6. Conclusions

We formalized and verified the new algebraic geometry system, point geometry, and constructed an interactive geometry-theorem-proving system based on point geometry theory. The system guarantees the reliability of proofs and supports user-generated content with scalability. Moreover, we designed automatic proof tactics to enhance the automation level of interactive proof and enabled the automatic presentation of the proof process by recording the proof status, providing readability. The system also supports multiple geometric-theorem-proving methods. Our next step is to improve various automatic proof methods in PointGeo, such as the point elimination algorithm based on the area method, to make the system more versatile, efficient, and readable.

Author Contributions: Conceptualization, Y.R. and J.J.; methodology, Y.R.; software, S.L. and H.G.; validation, S.L., Y.R. and Y.Z.; formal analysis, S.L.; investigation, Y.Z. and H.G.; resources, J.J.; data curation, S.L. and Y.R.; writing—original draft preparation, S.L.; writing—review and editing, Y.R. and J.J.; visualization, S.L.; supervision, Y.R.; project administration, Y.Z.; funding acquisition, Y.R. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (No. 62172116, 61972109) and the National Key R&D Program of China (No. 2018YFB1005100).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: We are grateful to the anonymous reviewers, whose comments greatly helped to improve the presentation of our research in this article.

Conflicts of Interest: The authors declare no conflict of interest

References

1. Gao, X.S. An introduction to Wu's method of mechanical geometry theorem proving. In Proceedings of the IFIP TC12/WG12. 3 International Workshop on Automated Reasoning, Beijing, China, 13–16 July 1992.
2. Deepak, K.; Sun, Y.; Wang, D.K. An efficient method for computing comprehensive Grobner base. *J. Symb. Comput.* **2013**, *52*, 124–142.
3. Mehne, H.H.; Mirjalili, S. A parallel numerical method for solving optimal control problems based on whale optimization algorithm. *Knowl.-Based Syst.* **2018**, *151*, 59–87. [CrossRef]
4. Plotkin, B. *Universal Algebra, Algebraic Logic, and Databases*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
5. Shi, H. On the resultant formula for mechanical theorem proving. *Math. Mech. Res. Prepr.* **1989**, *4*, 77–86.
6. Chou, S.C.B.; Gao, X.S.; Zhang, J.-Z. Automated generation of readable proofs with geometric invariants: ii. theorem proving with full-angles. *J. Autom. Reason.* **1996**, *17*, 349–370. [CrossRef]
7. Zhang, J. Outlines for point-geometry. *Stud. Coll. Math.* **2018**, *21*, 1–8.
8. Zhang, J.; Peng, X.; Chen, M. Self-evident automated proving based on point geometry from the perspective of Wu's method identity *J. Syst. Sci. Complex.* **2019**, *32*, 78–94. [CrossRef]
9. Fang, B.; Sighireanu, M.; Pu, G.G. Formal modelling of list based dynamic memory allocators. *Sci. China Inf. Sci.* **2018**, *61*, 122103. [CrossRef]
10. Jiang, D.C.; Li, W. The verification of conversion algorithms between finite automata. *Sci. China Inf. Sci.* **2018**, *61*, 028101. [CrossRef]
11. Liu, W.Y.; Bai, Y.J.; Jiao, L. Safety guarantee for time-delay systems with disturbances. *Sci. China Inf. Sci.* **2023**, *66*, 132102. [CrossRef]
12. Rao, Y.; Xie, L.; Guan, H.; Li, J.; Zhou, Q. A Method for Expanding Predicates and Rules in Automated Geometry Reasoning System. *Mathematics* **2022**, *10*, 1177. [CrossRef]
13. Castéran, P.; Bertot, Y. *Interactive Theorem Proving and Program Development. coq'art: The Calculus of Inductive Constructions*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
14. Chlipala, A. *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*; MIT Press: Cambridge, MA, USA, 2022.

15. Nipkow, T.; Wenzel, M.; Paulson, L.C. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*; Springer: Berlin/Heidelberg, Germany, 2002.
16. Harrison, J. Hol light: An overview. In Proceedings of the Theorem Proving in Higher Order Logics: 22nd International Conference, Munich, Germany, 17–20 August 2009.
17. Wang, J.; Fu, M.; Qiao, L.; Feng, X. Formalizing sparcv8 instruction set architecture in coq. *Sci. Comput. Program.* **2020**, *187*, 102371. [CrossRef]
18. Fervari, R.; Trucco, F.; Ziliani, B. Verification of dynamic bisimulation theorems in Coq. *J. Log. Algebr. Methods Program.* **2021**, *120*, 100642. [CrossRef]
19. Tran, D.D.; Ogata, K. Formal verification of tls 1.2 by automatically generating proof scores. *Comput. Secur.* **2022**, *123*, 102900. [CrossRef]
20. Barrière, A.; Blazy, S.; Pichardie, D. Formally Verified Native Code Generation in an Effectful JIT: Turning the CompCert Backend into a Formally Verified JIT Compiler. *Proc. ACM Program. Lang.* **2023**, *7*, 249–277. [CrossRef]
21. Zúñiga, A.; Bel-Enguix, G. On Coevaluation Behavior and Equivalence. *Mathematics* **2022**, *10*, 3800. [CrossRef]
22. Vladimir, V. An experimental library of formalized mathematics based on the univalent foundations. *Math. Struct. Comput. Sci.* **2015**, *25*, 1278–1294.
23. Álvaro, P.; Vladimir, V.; Michael, A.W. A univalent formalization of the p-adic numbers. *Math. Struct. Comput. Sci.* **2015**, *25*, 1147–1171.
24. Narboux, J.; Durand-Guerrier, V. Combining pencil/paper proofs and formal proofs, a challenge for Artificial Intelligence and mathematics education. In *Mathematics Education in the Age of Artificial Intelligence: How Artificial Intelligence Can Serve Mathematical Human Learning*; Richard, P.R., Vélez, M.P., Van, V.S., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp.167–192
25. Boldo, S.; Clément, F.; Martin, V.; Mayero, M.; Mouhcine, H. A Coq Formalization of Lebesgue Induction Principle and Tonelli's Theorem. In Proceedings of the 25th International Symposium on Formal Methods (FM 2023), Lübeck, Germany, 7–9 March 2023.
26. Fu, Y.; Yu, W. Formalizing Calculus without Limit Theory in Coq. *Mathematics* **2021**, *9*, 1377. [CrossRef]
27. Yan, S.; Yu, W. Formal Verification of a Topological Spatial Relations Model for Geographic Information Systems in Coq. *Mathematics* **2023**, *11*, 1079. [CrossRef]
28. Boldo, S.; Clément, F.; Faissole, F.; Martin, V.; Mayero, M. A Coq formalization of Lebesgue integration of nonnegative functions. *J. Autom. Reason.* **2022**, *66*, 175–213. [CrossRef]
29. Narboux, J. A decision procedure for geometry in coq. In Proceedings of the International Conference on Theorem Proving in Higher Order Logics, Park City, UT, USA, 14–17 September 2004; pp. 225–240.
30. Beeson, M.; Narboux, J.; Wiedijk, F. TProof-checking euclid. *Ann. Math. Artif. Intell.* **2019**, *85*, 213–257. [CrossRef]
31. Boutry, P.; Gries, C.; Narboux, J.; Schreck, P. Parallel postulates and continuity axioms: A mechanized study in intuitionistic logic using coq. *J. Autom. Reason.* **2019**, *62*, 1–68. [CrossRef]
32. Boutry, P.; Braun, G.; Narboux, J. Formalization of the arithmetization of euclidean plane geometry and applications. *J. Symb. Comput.* **2019**, *90*, 149–168. [CrossRef]
33. Quaresma, P.; Graziani, P. Measuring the Readability of Geometric Proofs: The Area Method Case. *J. Autom. Reason.* **2023**, *67*, 5. [CrossRef]
34. Fritz, T. A generalization of Strassen's Positivstellensatz. *Commun. Algebra* **2021**, *49*, 482–499. [CrossRef]