Automated Generation of Formal and Readable Proofs in Geometry Using Coherent Logic

Sana Stojanović, Vesna Pavlović, and Predrag Janičić

Faculty of Mathematics, University of Belgrade
Studentski trg 16, 11000 Belgrade, Serbia
{sana,vesnap,janicic}@matf.bg.ac.rs

Abstract. We present a theorem prover ArgoCLP based on coherent logic that can be used for generating both readable and formal (machine verifiable) proofs in various theories, primarily geometry. We applied the prover to various axiomatic systems and proved dozens of theorems from standard university textbooks on geometry. The generated proofs can be used in different educational purposes and can contribute to the growing body of formalized mathematics. The system can be used, for instance, in showing that modifications of some axioms does not change the power of an axiom system. The system can also be used as an assistant for proving appropriately chosen subgoals of complex conjectures.

1 Introduction

Geometry has initiated a number of revolutions in mathematics. Also, it has always had a very important role in mathematical education because of paradigmatic reasoning that it requires. For a similar reason, for decades it has been a challenging domain for computer theorem proving, with most attention payed to Euclidean geometry. As early as from 1950's, there were interesting approaches to automatically proving geometry theorems, but real successes came in last decades of twentieth century. For example, theorem provers for Euclidean geometry based on Wu's method automatically proved hundreds of complex theorems [8] and this method is often considered the most efficient method for automated theorem proving overall. Today, there are two main directions in computer theorem proving in geometry:

- Interactive theorem proving using proof assistants such as Isabelle [29] or Coq [34]. The proofs in this context are made mainly manually, but are automatically verified by a computer. Interactive proving is often very demanding and time consuming, it requires an experienced user, and it is typically non-trivial to reuse pieces of existing proofs. Even more, since there is no automation (or it is of very limited power), if one wants to formulate and prove the same theorem in just a slightly modified theory, that would often require doing the same amount of work all over again.
- Automated theorem proving using algebraic methods (such as Wu's method [40] or Gröbner bases method [6, 22]) or coordinates-free methods (such as

the area method [10] or the full-angle method [11]). In this context, proofs are often generated very efficiently, but they are far from traditional, human-readable proofs.

The above two directions have somewhat different motivations: the former aims at building a corpus of verified mathematical knowledge, while the latter aims at applications in education (e.g., within dynamic geometry software) or in industry (when it is more important to know that a certain conjecture was proved than to have its proof). Nevertheless, there are also goals in the intersection of the above two directions. It would be beneficial (both for the growing body of formalized mathematics and for educational purposes) to have formal, machine verifiable geometry proofs automatically generated, if possible — efficiently and in the traditional geometry manner. In this paper we address these combined goals and describe our theorem prover ArgoCLP (*Automated Reasoning GrOup Coherent Logic Prover*) that automatically generates traditional, human readable, but in the same time formal proofs of geometry theorems (for various axiom systems). The generated step-by-step proofs are very similar to the proofs given in standard geometry textbooks. The theorem prover uses coherent logic as its underlying logic. A suitable domain of the prover are foundational properties typically expressed in terms of applications of individual axioms. Hence, we do not aim at conjectures involving, for instance, metrical quantities, typically successfully proved by algebraic provers. Instead, we primarily aim at automatically proving theorems that are a current subject of manual formal proving by mathematicians. For instance, our system can be used, in showing that modifications of some axioms does not change the power of an axiom system. In addition, we believe that our theorem prover can serve as a machine assistant that can help mathematicians to prove complex theorems suitably broken apart into several smaller ones.

Organization of the paper. In Section 2 we give brief background information on some geometry axiomatizations, on formal mathematics, and on coherent logic. In Section 3 we present our algorithms for proving theorems in coherent logic, in Section 4 we briefly discuss the implementation of our theorem prover ArgoCLP, and in Section 5 we present applications of our prover to four axiom systems for Euclidean space geometry. In Section 6 we discuss the related work and in Section 7 we draw final conclusions and present some of the ideas for further work.

2 Background

Axiomatizations of Geometry. Euclid, with his book "Elements", is considered to be the first who proposed and used an axiomatic method in mathematics [18]. He succeeded to derive, using purely logical rules, many geometry properties that were known long time before him. This system, partly naive from today's point of view, was used for centuries.

In 1899 in his seminal book "Der Grundlagen der Geometrie", Hilbert proposed a new axiom system to elementary geometry that fixed many flaws and

weaknesses of Euclid's system [19]. This Hilbert's work is one of the landmarks for XX century mathematics, but it is still not up to contemporary standards. The axiom system uses three sorts of primitive objects: points, lines and planes, while the set of axioms is divided into five groups (incidence axioms, axioms of order, axioms of congruence, axioms of parallels, and continuity axioms). Each group of axioms is accompanied with some fundamental theorems that can be proved using preceding axioms. One of more modern variants of Hilbert's system was given by Borsuk and Szmielev [5].

In mid-twenty century, Tarski presented a new axiomatisation (actually — several variants) for elementary geometry (without all continuity features ensured), along with a decision procedure for that theory [33, 31]. The main characteristics of Tarski's axiom system is that it is very simple: it is based only on one sort of primitive objects — points, it has only two predicates and eleven axioms.

Formal Mathematics. Over the last years, in all areas of mathematics and computer science, with a history of huge number of flawed published proofs, formal, machine verifiable proofs (given in object-level form — in terms of axioms and inference rules) have been gaining more and more importance. Formal proofs have important role in management of mathematical knowledge (e.g., in digitization of mathematical heritage), in education and e-learning, but also in industrial applications where correctness of some algorithms or calculations is critical. There are growing efforts in developing formal proofs, with many extremely complex theorems proved, with repositories of proved theorems, and also with many software tools for producing and checking formal proofs. Among the most popular theorem proving assistants (systems that implement formal logic and that verify proofs) nowadays are Isabelle [29], Coq[1] [34], Mizar[2] [35], HOL-light[3] [17]. The level of automation in proof assistants is typically very limited.

Readable formal proofs and Isar. Most of the theorem proving assistants use proof scripts that explicitly list all axioms and inference rules used in every single proof step. Despite many results and successes in formalizing fragments of mathematics and computer science, they are still not used by a wide scientific community. The Intelligible semiautomated reasoning (Isar) approach [39] to readable formal proof documents aims to bridge the gap between internal notions of proof given by state-of-the-art interactive theorem proving systems and an appropriate level of abstraction for user-level work. Isar is an alternative to traditional proof tactic scripts, it provides a proof language interface layer which is much more readable for the users. The Isabelle/Isar system provides an interpreter for the Isar formal proof document language, and readable Isar proof documents are converted and executed as series of low-level Isabelle inference steps. Therefore, Isar allows the user to express proofs in a somewhat human-

[1] `http://coq.inria.fr/`
[2] `http://www.mizar.org`
[3] `http://www.cl.cam.ac.uk/~jrh13/hol-light/`

friendly way but they are still automatically verifiable by the underlying proof system.

Coherent Logic. Coherent logic (CL) was initially defined by Skolem and in recent years it gained new attention [2, 15, 3]. CL allows certain existential quantifications so it can be considered as an extension of resolution logic. In contrast to resolution method, the conjecture being proved is kept unchanged and directly proved (refutation, Skolemization and transformation to clausal form are not used). Proofs in CL are natural and intuitive and reasoning is constructive, so proof objects can be easily obtained [2]. Therefore, CL is a suitable framework for producing both readable and formal proofs. A number of theories and theorems can be formulated directly and simply in CL. More details on features of CL can be found, for instance, in [2, 15].

Formally, CL is a fragment of first-order logic (FOL) consisting of formulae of the following form:

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}_1\ B_1(\boldsymbol{x}, \boldsymbol{y}_1) \vee \ldots \vee \exists \boldsymbol{y}_m\ B_m(\boldsymbol{x}, \boldsymbol{y}_m)$$

which are implicitly universally quantified and where: $0 \leq n$, $0 \leq m$, $\boldsymbol{x}$ denotes a sequence of variables $x_1, x_2, \ldots, x_k$, A_i (for $1 \leq i \leq n$) denotes an atomic formula (involving some of the variables from $\boldsymbol{x}$), $\boldsymbol{y}_j$ denotes a sequence of variables $y_1^j, y_2^j, \ldots, y_{k_j}^j$, and B_j (for $1 \leq j \leq m$) denotes a conjunction of atomic formulae (involving some of the variables from $\boldsymbol{x}$ and $\boldsymbol{y}_j$). There are no function symbols with arity greater than 0. Function symbols of arity 0 are called *constants*. A *witness* is a new constant, not appearing in axioms used nor in the conjecture being proved. The name *constant* covers both constants that are parts of the signature and witnesses. A *term* is a constant or a variable. An *atomic formula* is either $\bot$ or $p(t_1, \ldots, t_n)$, where p is a predicate symbol of arity n and t_i $(1 \leq i \leq n)$ are terms. An atomic formula over constants is called a *fact*.

The only inference rules (in the style of natural deduction, a variant of the rules given in [3]) used in CL are as follows:

$$\frac{A_1(\boldsymbol{a}) \wedge \ldots \wedge A_n(\boldsymbol{a})}{A_i(\boldsymbol{a})}\ \wedge E \qquad \frac{A_1 \vee \ldots \vee A_n \quad \begin{matrix}[A_1]\\ \vdots\\ B\end{matrix} \quad \ldots \quad \begin{matrix}[A_n]\\ \vdots\\ B\end{matrix}}{B}\ \vee E \qquad \frac{\bot}{A}\ efq$$

$$\frac{A_1(\boldsymbol{a}) \quad \ldots \quad A_n(\boldsymbol{a}) \quad A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}_1\ B_1(\boldsymbol{x}, \boldsymbol{y}_1) \vee \ldots \vee \exists \boldsymbol{y}_m\ B_m(\boldsymbol{x}, \boldsymbol{y}_m)}{B_1(\boldsymbol{a}, \boldsymbol{w}_1) \vee \ldots \vee B_m(\boldsymbol{a}, \boldsymbol{w}_m)}\ ax$$

where $\boldsymbol{a}$ is a vector of constants and $\boldsymbol{w}_j$ are vectors of witnesses (i.e., fresh constants). When applied, the rule $\wedge E$ infers $A_i(\boldsymbol{a})$ for each i such that $1 \leq i \leq n$. The rule (ax) is applied only if there are no vectors $\boldsymbol{w}_j$ of constants such that $B_1(\boldsymbol{a}, \boldsymbol{w}_1) \vee \ldots \vee B_m(\boldsymbol{a}, \boldsymbol{w}_m)$ holds.

A formula

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}_1\ B_1(\boldsymbol{x}, \boldsymbol{y}_1) \vee \ldots \vee \exists \boldsymbol{y}_m\ B_m(\boldsymbol{x}, \boldsymbol{y}_m)$$

is a *CL-theorem*, if from premises $A_1(\boldsymbol{a}), \ldots, A_n(\boldsymbol{a})$ (where $\boldsymbol{a}$ denotes a sequence of fresh constants) all conjuncts of a formula $B_j(\boldsymbol{x}, \boldsymbol{w})$ can be derived for some j $(1 \leq j \leq m)$ and for some vector of constants $\boldsymbol{w}$.

There is a breadth-first proof procedure for coherent logic that is sound and complete: a coherent formula F can be proved if and only if F is true in all Tarskian models (with non-empty domains) of the set of the axioms and the facts $A_1(\boldsymbol{a}), \ldots, A_n(\boldsymbol{a})$ [2].

3 ArgoCLP Proof Procedures

In this section we describe proof procedures that are used or can be used in our theorem prover ArgoCLP for CL (a description of the implemented procedures and techniques is given in Section 4). It is a generic theorem prover for coherent logic, so it can use any set of coherent axioms (not just geometrical). Sorts can be used (but, alternatively, corresponding unary predicates may be used). Negations (not allowed by the basic definition of CL) can be used in a limited way: atomic formulae occurring in axioms and in conjectures may be negated. Accordingly, additional predicate symbol $nonR$ is introduced for each predicate symbol R and the following additional axioms for each predicate symbol R are used:

$R(\boldsymbol{x}) \vee nonR(\boldsymbol{x})$

$R(\boldsymbol{x}) \wedge nonR(\boldsymbol{x}) \Rightarrow \bot$

The first axiom schema brings the classical logic (in contrast to the basic deduction system for coherent logic given in Section 2). It is important for geometry reasoning, as the intuitionistic setting has a very limited power for Hilbert style geometry [12].

3.1 Basic Proof Procedure

An alternative to the breadth-first proof procedure is a simple proof procedure with forward chaining and with iterative deepening. Axioms are applied according to the inference rule (ax) given in Section 2. Definitions available are used as they were axioms. The axioms are applied in the waterfall manner: when one axiom has been successfully applied, then search for applicable axioms starts again from the first axiom. All constants are enumerated and there is a dedicated counter s that controls applications of axioms — an axiom can be applied only if all its (universally quantified) variables are matched with constants with order less then s. Initially, s equals the number of constants appearing in the premises of the conjecture. The value s is increased once no axiom can be applied and the proof procedure continues. If no axiom can be applied anymore and the conjecture has not been proved, this means that the conjecture is not CL-theorem

(however, for many non-CL-theorems the proof procedure does not terminate). It can be proved (in a similar manner as it was proved for the breadth-first procedure in [2]) that this proof procedure is sound and complete: a coherent formula F can be proved if and only if F is true in all Tarskian models (with non-empty domains) of the set of the axioms and the facts $A_1(\boldsymbol{a}), \ldots, A_n(\boldsymbol{a})$ [2]. In addition, it can be proved that this proof procedure is sound and complete with respect to the inference system given in Section 2, i.e., a formula F can be proved if and only if F is CL-theorem.

Despite the completeness property, proving some conjectures in some theories is practically impossible with this basic proof procedure (i.e., impossible with reasonable memory and time resources).

3.2 Improved Proof Procedure

Efficiency of the basic proof procedure given above can be improved in a number of ways, while still preserving completeness. Here some possible improvements are listed, all of which aim at keeping control on the search space (i.e., on the number of introduced witnesses) and decreasing to some extent a combinatorial explosion (caused by derived facts that are irrelevant).

Ordering of axioms. The axioms are grouped into the following groups (it is assumed that $n > 0$, $m > 0$, and that one group of axioms excludes previous groups as its special cases):

non-productive non-branching axioms: axioms of the form:
$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x})$
$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow B(\boldsymbol{x})$

non-productive branching axioms: axioms of the form:
$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow B_1(\boldsymbol{x}) \vee \ldots \vee B_m(\boldsymbol{x})$

productive non-branching axioms: axioms of the form:
$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}\ B(\boldsymbol{x}, \boldsymbol{y})$

productive branching axioms: axioms of the form:
$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}_1\ B_1(\boldsymbol{x}, \boldsymbol{y}_1) \vee \ldots \vee \exists \boldsymbol{y}_m\ B_m(\boldsymbol{x}, \boldsymbol{y}_m)$

strongly productive non-branching axioms: axioms of the form:
$\exists \boldsymbol{y}\ B(\boldsymbol{x}, \boldsymbol{y})$

strongly productive branching axioms: axioms of the form:
$\exists \boldsymbol{y}_1\ B_1(\boldsymbol{x}, \boldsymbol{y}_1) \vee \ldots \vee \exists \boldsymbol{y}_m\ B_m(\boldsymbol{x}, \boldsymbol{y}_m)$

Axioms can be automatically assigned their types and are used in the proving process with priorities given to the groups as in the above ordering. There is no imposed ordering of axioms within a group (although their ordering within groups can also impact efficiency).

Early pruning. When testing an axiom for applicability, it is not necessary to instantiate all its variables and only then check if all relevant facts were already derived. Instead, a check for relevant facts can be performed as soon as possible, in order to enable early rejection of some axiom instances and pruning of the search space. For instance, when applying the following axiom:
$\forall x : \mathit{line}\ \ \forall y : \mathit{line}\ \ \forall X : \mathit{point}\ \ \forall Y : \mathit{point}$

$(incident(X, x) \wedge incident(Y, x) \wedge incident(X, y) \wedge incident(Y, y) \wedge X \neq Y \Rightarrow x = y)$

instead of matching x, y, X, and Y with all admissible constants, x and X will be first unified with admissible constants, and the matching will backtrack immediately if the fact instantiated from $incident(X, x)$ has not been already derived. Generally, relevant facts are checked as soon as all involved arguments have been instantiated.

Breaking axioms that introduce several witnesses. As said in Section 2, an axiom like:

$\forall x : line \ \exists X : point \ \exists Y : point \ (incident(X, x) \wedge incident(Y, x) \wedge X \neq Y)$

will not be applied for a specific line a (instantiating x) if there are already constants $A : point$ and $B : point$ such that $incident(A, a)$, $incident(B, a)$, and $A \neq B$. However, for efficiency reasons, it is beneficial not to apply the above axiom even if there is a constant $A : point$ such that $incident(A, a)$ holds, and there is no constant $B : point$ such that $incident(B, a)$ and $A \neq B$. Instead, the following variant of the above axiom should be used:

$\forall x : line \ \forall X : point \ (incident(X, x) \Rightarrow \exists Y : point \ (incident(Y, x) \wedge X \neq Y))$

Therefore, instead of one axiom, two axioms will be used, with the general one having lower priority. The same mechanism can be applied for all axioms that involve more than one existential quantifier. Breaking such axioms into several versions is not always straightforward as in the above example. For example, the axiom:

$\exists X : point \ \exists Y : point \ \exists Z : point \ \exists U : point \ noncoplanar(X, Y, Z, U)$

should be broken into four axioms, with one of them:

$\forall X : point \ \forall Y : point \ \forall Z : point \ \exists U : point \ noncoplanar(X, Y, Z, U)$

However, this conjecture is invalid (in Euclidean geometry) and additional premise ($noncolinear(X, Y, Z)$) is required. Because of this, if an axiom can be broken into several variants, each of them should be proved (again by the CL prover) before used. If some variant cannot be proved (i.e., if it cannot be proved within some time limit), the user may be asked to modify it. Notice that additional axioms introduced this way actually change the original axiom system, but since the new axiom system is equivalent to the original one (each of its axioms can be proved as a theorem by the other one and vice versa), this modification is legitimate (the new axioms can be considered only as lemmas).

Dealing with equality. For theories involving equality, the axioms of equality are not used explicitly. Instead, equivalence classes of equality of constants are maintained. Thanks to this, it suffices to work only with a canonical representative of a class instead of all objects that belong to that class. In the beginning of the proving process, every object represents its own class and the classes are maintained using Tarjan's *union-find* structures [32].

For example, if there are constants $A : point$, $p : line$, $q : line$ and $\alpha : plane$ such that $incident(A, p)$, $incident(q, \alpha)$, and $p = q$ hold, the following axiom can be applied:

$\forall X : point \quad \forall x : line \quad \forall \chi : plane \quad (incident(X, x) \wedge incident(x, \chi) \Rightarrow incident(X, \chi))$

and, for $X = A$, $x = p$, $\chi = \alpha$, the fact $incident(A, \alpha)$ can be derived. Namely, for this instantiation of variables, when checking if the fact $incident(p, \alpha)$ hold, the representatives of p and α are first determined — say, q and α — and since $incident(q, \alpha)$ holds, the axiom can be applied.

Although the axioms of equality are not used explicitly during the search process, they are used in building a proof trace from which a complete (machine verifiable) proof object can be constructed.

Dealing with symmetrical predicate symbols. A predicate R is symmetrical (in argument positions i and j) if it holds (universal quantification is assumed):

$R(x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) \Leftrightarrow R(x_1, \ldots, x_j, \ldots, x_i, \ldots, x_n)$

For symmetrical predicates, only *representatives* of facts can be considered. For instance, instead of storing both $colinear(A, B, C)$ and $colinear(C, B, A)$, it suffices to store only $colinear(A, B, C)$. A representation of a class of facts can be determined in the following way: using the ordering of constants, sort arguments in symmetrical positions, and choose the minimum as the representative. This step is performed whenever a fact over a symmetrical predicate should be checked. This mechanism can be used in conjunction with the mechanism of equivalence classes w.r.t. equality to further reduce the number of facts stored. Like the equality axioms, statements ensuring that a predicate is symmetrical are not used during the search process, but they are used in building the proof trace from which a complete (machine verifiable) proof object can be constructed.

Whether a predicate is symmetrical can be checked automatically: all relevant conjectures are generated and then tried to be proved. Instead of proving conjectures for all permutations of symmetrical arguments, it is sufficient to prove conjectures for permutation group generators. For instance, when trying to prove that the predicate *coplanar* is symmetrical on all four arguments, it is sufficient to prove conjectures only for permutation group generators (universal quantification is assumed):

$coplanar(x_1, x_2, x_3, x_4) \Leftrightarrow coplanar(x_2, x_3, x_4, x_1)$

$coplanar(x_1, x_2, x_3, x_4) \Leftrightarrow coplanar(x_2, x_1, x_3, x_4)$

For example, if there are constants $A : point$, $B : point$, $C : point$ and $D : point$ with the ordering $A < B < C < D$, and the facts $noncolinear(C, B, D)$ and $colinear(A, D, C)$ derived, if the fact $A = B$ is derived, the equivalence classes of these two objects will be merged and a contradiction can be detected. Namely, if A is the representative of a class containing A and B, by using the equivalence classes, the representative of $noncolinear(C, B, D)$ is $noncolinear(C, A, D)$, and, by symmetry properties, its representative is $noncolinear(A, C, D)$. By symmetry, the representative of $colinear(A, D, C)$ is $colinear(A, C, D)$, so, from $noncolinear(A, C, D)$ and $colinear(A, C, D)$, a contradiction can be derived.

Reuse of proved theorems. Proved conjectures that a predicate is symmetrical (along with their proofs) are used within wider proofs. However, this can be done also for other theorems of the theory proved by the system.

Even with all these techniques, many complex theorems cannot be proved in a reasonable time. Also, generated proofs contain many irrelevant derivations.

3.3 Techniques That Do Not Preserve Completeness

In order to improve efficiency of the prover, at least for some conjectures, some techniques that do not preserve completeness may be used:

Restriction on branching axioms. Branching axioms of the form $R(\boldsymbol{x}) \vee nonR(\boldsymbol{x})$ are generated and used only for primitive (and not for defined) predicates.

Restriction on axioms. In the proof procedure, only axioms that involve just predicates occurring in the conjecture are used. Another, relaxed variant of this restriction is: in the proof procedure, only axioms that involve at least one predicate occurring in the conjecture are used.

4 ArgoCLP Implementation

The prover ArgoCLP is implemented in C++. It consists of around 5 thousand lines of code, organized within 23 classes. Both the signature and the set of axioms are imported into the program through files, so the prover can be used for different CL theories. A conjecture is specified by:

theory's signature: names of sorts are stated after the keyword *types*, for example:
`types point line plane`
followed by the list of predicate symbols given along with the list of types of each argument. For example, the *incidence* predicate over points and lines would be given as:
`datatype inc_po_l point line`
It is assumed that *eq_type* denotes equality over two objects of a type *type*.

set of axioms: axioms are given in the following form:
```
point(1) point(2) ~eq_point(1,2)
=> line(3) inc_po_l(1,3) inc_po_l(2,3)
```
(variables are represented by natural numbers, universal quantification is assumed for variables appearing on the left hand side of the implication, existential quantification is assumed for variables appearing only on the right hand side of the implication).

set of definitions: definitions are used for convenience and should have the same form as axioms. For instance:
```
point(1) point(2) point(3) line(4)
inc_po_l(1,4) inc_po_l(2,4) inc_po_l(3,4)
=> colinear(1,2,3)
```

conjecture: it is given in the same form as axioms. For example:

```
point(1) point(2) point(3) line(4)
inc_po_l(1,4) inc_po_l(2,4) bet(1,2,3)
=> inc_po_l(3,4)
```

Most of the techniques listed in Section 3.2 are already implemented within ArgoCLP: grouping and prioritizing axioms, early pruning, support for equality reasoning, support for symmetrical predicates. Lemmas obtained by breaking axioms that introduce several witnesses can be verified within the prover, but their automated generation (with possible assistance of the user) is still under development. Also, symmetrical predicates are used as explained, but automatic detection of symmetrical predicates and automatic generation of required properties is not fully implemented yet.

The user can state (through a configuration file) which of the techniques from Section 3.2 and Section 3.3 should be used in the proof search:

equality flag indicates whether the built-in equality reasoning will be used.

excluded middle flags indicate whether axioms of excluded middle are to be used and, more specifically, if only axioms of excluded middle for primitive predicate symbols (and not for defined ones) will be used.

flags for non completeness-preserving techniques indicate whether only axioms that involve only predicates occurring in the conjecture should be used (this does not apply to equalities if the equality flag is set); whether only axioms that involve at least one predicate occurring in the conjecture should be used (this does not apply to equalities if the equality flag is set); whether the constant counter is being incremented before trying to apply any of strongly productive axioms.

Along the proving process, ArgoCLP generates a proof trace with all relevant information. This proof trace can be exported to different output formats. Currently, ArgoCLP can generate (formally verifiable) proof objects in Isabelle/Isar form (that are accompanied by the axioms also exported from ArgoCLP), and to even more readable, natural language form (in English, in LaTeX format). In addition, there is a mechanism for eliminating from a proof trace all inference steps (including branching steps[4]) that were not relevant, yielding a ,,clean" (often significantly shorter) proof trace. Such clean proof traces, can be again exported to Isabelle/Isar or natural language form.

Example 1. Let us consider the following conjecture (of Hilbert-style Euclidean geometry): for three lines p, q, and r and a plane α which contains them all holds that if $p \neq q$ and $q \neq r$ and p and q do not intersect and q and r do not intersect and if there exists a point A which belongs to the plane α and to the lines p and r, then $p = r$.

The conjecture is specified in the following form:

[4] A branching step is relevant only if both branches use the assumed case, otherwise, the branching can be eliminated and a branch that does not use the assumed case can be kept.

```
premises
# TH_8
% for three lines and a plane which contains them all holds that
% if first and second are distinct and second and third are distinct
% and first and second do not intersect and second and third do not
% intersect and if there exists a point which belongs to the plane
% and to the first and third line, then first and third line are equal

line(1)
line(2)
line(3)
plane(4)
~eq_line(1,2)
~eq_line(2,3)
~int_l_l(1,2)
~int_l_l(2,3)
inc_l_pl(1,4)
inc_l_pl(2,4)
inc_l_pl(3,4)
point(5)
inc_po_pl(5,4)
inc_po_l(5,1)
inc_po_l(5,3)

conclusions

eq_line(1,3)
```

A key fragment of the generated („clean“) Isabelle/Isar proof generated by the prover is given below.

```
...
lemma  TH_8:
assumes "LI1 ~= LI2"
and "LI2 ~= LI3"
and "\<not>int_l_l LI1 LI2"
and "\<not>int_l_l LI2 LI3"
and "inc_l_pl LI1 PL1"
and "inc_l_pl LI2 PL1"
and "inc_l_pl LI3 PL1"
and "inc_po_pl PO1 PL1"
and "inc_po_l PO1 LI1"
and "inc_po_l PO1 LI3"
shows "LI1 = LI3"
proof -

(*1*)
have  "LI1 = LI3 \<or> LI1 ~= LI3"
using ax_g_ex_mid_3 [of "LI1" "LI3"]
by auto
```

```
(*2*)  moreover
{ assume "LI1 = LI3"
(*3*)
from this
have ?thesis
by auto
} note note1 = this
(*4*)  moreover
{ assume "LI1 ~= LI3"
(*5*)  moreover
have  "inc_po_l PO1 LI2 \<or> \<not>inc_po_l PO1 LI2"
using ax_g_ex_mid_7 [of "PO1" "LI2"]
by auto
(*6*)  moreover
{ assume "inc_po_l PO1 LI2"
(*7*)  moreover
from ‘LI1 ~= LI2‘ and ‘inc_po_l PO1 LI1‘ and ‘inc_po_l PO1 LI2‘
have  "int_l_l LI1 LI2"
using ax_D5 [of "LI1" "LI2" "PO1"]
by auto
(*8*)  moreover
from ‘int_l_l LI1 LI2‘ and ‘\<not>int_l_l LI1 LI2‘
have False
by auto
(*9*)
ultimately
have False
by auto
} note note2 = this
(*10*)  moreover
{ assume "\<not>inc_po_l PO1 LI2"
(*11*)  moreover
from ‘\<not>int_l_l LI1 LI2‘
have  "\<not>int_l_l LI2 LI1"
using ax_nint_l_l_21 [of "LI1" "LI2"]
by auto
(*12*)  moreover
from ‘\<not>inc_po_l PO1 LI2‘ and ‘inc_po_pl PO1 PL1‘ and ‘inc_l_pl LI2 PL1‘
and ‘inc_po_l PO1 LI1‘ and ‘inc_l_pl LI1 PL1‘ and ‘\<not>int_l_l LI2 LI1‘
and ‘inc_po_l PO1 LI3‘ and ‘inc_l_pl LI3 PL1‘ and ‘\<not>int_l_l LI2 LI3‘
have  "LI1 = LI3"
using ax_E2 [of "PO1" "LI2" "PL1" "LI1" "LI3"]
by auto
(*13*)  moreover
from ‘LI1 = LI3‘ and ‘LI1 ~= LI3‘
have False
by auto
(*14*)
ultimately
have False
```

```
by auto
} note note3 = this
(*15*)  from note2 and note3 and ‘inc_po_l PO1 LI2 | \<not>inc_po_l PO1 LI2‘
have False
by auto
(*16*)
ultimately
have False
by auto
} note note4 = this
(*17*)  from note1 and note4 and ‘LI1 = LI3 | LI1 ~= LI3‘
have ?thesis
by auto
ultimately
show ?thesis
by auto
qed
```

The („clean“) proof generated in the natural language form (using the natural language description of the theory's signature), along with the natural language formulation generated from the conjecture specification, is given below.

Theorem TH_8:

Assuming that $p \neq q$, and $q \neq r$, and the lines p and q do not intersect, and the lines q and r do not intersect, and the line p is incident to the plane α, and the line q is incident to the plane α, and the line r is incident to the plane α, and the point A is incident to the plane α, and the point A is incident to the li ne p, and the point A is incident to the line r, show that $p = r$.

Proof:

1. It holds that $p = r$ or $p \neq r$ (by axiom of excluded middle).
   2. Assume that $p = r$.

      This proves the conjecture.
   3. Assume that $p \neq r$.

      4. It holds that the point A is incident to the line q or the point A is not incident to the line q (by axiom of excluded middle).

         5. Assume that the point A is incident to the line q.

            6. From the facts that $p \neq q$, and the point A is inci dent to the line p, and the point A is incident to the line q, it holds that the lines p and q intersect (by axiom ax_D5).

            7. From the facts that the lines p and q intersect, and the lines p and q do not intersect we get a contradiction.

               Contradiction.

         8. Assume that the point A is not incident to the line q.

9. From the facts that the lines p and q do not intersect, it holds that the lines q and p do not intersect (by axiom ax_nint_l_l_2 1).

10. From the facts that the point A is not incident to the line q, and the point A is incident to the plane α, and the line q is incident to the plane α, and the point A is incident to the line p , and the line p is incident to the plane α, and the lines q and p do not intersect, and the point A is incident to the line r, and the l ine r is incident to the plane α, and the lines q and r do not in tersect, it holds that $p = r$ (by axiom ax_E2).

11. From the facts that $p = r$, and $p \neq r$ we get a co ntradiction.

Contradiction.

Theorem proved in 11 steps and in 0.02 s.

5 Applications

We applied ArgoCLP prover to four axiom systems for Euclidean (space) geometry in a uniform manner. These are Hilbert's system [19], Tarski's system [33, 31], system given by Borsuk and Szmielev [5], and our system that is very close to Borsuk's one, but more suitable for CL-based proof procedure. We use the same signature for all the systems (so we could try to prove the same theorems within different systems), which is the union of all the sorts and the predicates used in each of these systems. Of course, if one system does not involve some predicates, it cannot be used for proving their properties (e.g., Tarski's system cannot be used for proving properties of incidence relations, since this system deals only with points). We encoded all axioms from these four systems, except axioms of continuity (for their complexity). Still, a large fragment of geometry can be built without them. We reformulated some axioms in order to avoid complex defined notions such as ray, half-plane, internal angle, etc, but we kept the original meaning of all axioms.

Encoding axioms in its own right is not trivial, because original formulations are often inaccurate, with some conditions only implicitly assumed. For instance, when Hilbert, in his axioms, uses the phrase "two points", he assumes that they are distinct (but does not explicitly state that). Meikle and Fleuriot also underlined this problem [25]. There is a number of problems of this sort and sometimes it is not trivial to show whether a modification would change the set of theorems of the system. Here we do not aim at a thorough comparison between these systems, but rather at illustrating the ArgoCLP prover and to make first steps in showing what fragments of one system can be built within some other system. The prover can also be used to show what modifications of certain axioms can be made.

We applied ArgoCLP on these axiom systems and on a number of theorems from standard geometry courses.[5] As expected, the results depended much on

[5] The prover ArgoCLP, along with descriptions of the used theories and conjectures, is available on-line from `http://argo.matf.bg.ac.rs/downloads.html`

the set of the axioms used. As an illustration, we list 14 theorems (including some that were not proved by the prover within the time limit of 30 seconds) and the obtained results for the four systems. All results were obtained with one fixed configuration of the prover (only axioms that involve just predicates occurring in the conjecture are used and only axioms of excluded middle for primitive predicate symbols are used).

Theorem 1: $\forall p : line \ \forall q : line \ (int(p,q) \Rightarrow \exists \alpha : plane \ (inc(p,\alpha) \wedge inc(q,\alpha)))$

Theorem 2: $\forall p : line \ \forall q : line \ \forall A : point \ \forall B : point \ (p \neq q \wedge inc(A,p) \wedge inc(A,q) \wedge inc(B,p) \wedge inc(B,q) \Rightarrow A = B)$

Theorem 3: $\forall p : line \ \forall \alpha : plane \ \forall A : point \ \forall B : point \ (\neg inc(p,\alpha) \wedge inc(A,p) \wedge inc(A,\alpha) \wedge inc(B,p) \wedge inc(B,\alpha) \Rightarrow A = B)$

Theorem 4: $\forall A : point \ \forall B : point \ \forall C : point \ (\neg col(A,B,C) \Rightarrow A \neq B \wedge A \neq C \wedge B \neq C)$

Theorem 5: $\forall A : point \ \forall B : point \ \forall C : point \ (\neg col(A,B,C) \Rightarrow \exists \alpha : plane(inc(A,\alpha) \wedge inc(B,\alpha) \wedge inc(C,\alpha)))$

Theorem 6: $\forall A : point \ \forall p : line \ (\neg inc(A,p) \Rightarrow \exists \alpha : plane \ (inc(A,\alpha) \wedge inc(p,\alpha)))$

Theorem 7: $\forall A : point \ \forall B : point \ \forall C : point \ \forall D : point \ \forall \alpha : plane \ (comp(A,B,C,D) \wedge \neg col(A,B,C) \wedge inc(A,\alpha) \wedge inc(B,\alpha) \wedge inc(C,\alpha) \Rightarrow inc(D,\alpha))$

Theorem 8: $\forall p : line \ \forall q : line \ \forall r : line \ \forall A : point \ \forall \alpha : plane \ (p \neq q \wedge q \neq r \wedge inc(p,\alpha) \wedge inc(q,\alpha) \wedge inc(r,\alpha) \wedge \neg int(p,q) \wedge \neg int(q,r) \wedge inc(A,\alpha) \wedge inc(A,p) \wedge inc(A,r) \Rightarrow p = r)$

Theorem 9: $\forall A : point \ \forall B : point \ \forall C : point \ \forall p : line \ (inc(A,p) \wedge inc(B,p) \wedge bet(A,B,C) \Rightarrow inc(C,p))$

Theorem 10: $\forall A : point \ \forall B : point \ \forall C : point \ (bet(A,B,C) \Rightarrow \neg bet(A,C,B))$

Theorem 11: $\forall A : point \ \forall B : point \ (A \neq B \Rightarrow \exists C : point \ bet(A,C,B))$

Theorem 12: $\forall A : point \ \forall B : point \Rightarrow cong(A,B,A,B)$

Theorem 13: $\forall A : point \ \forall B : point \ \forall C : point \ \forall D : point \ (cong(A,B,C,D) \Rightarrow cong(C,D,A,B))$

Theorem 14: $\forall A : point \ \forall B : point \ \forall C : point \ \forall D : point \ (cong(A,B,C,D) \Rightarrow cong(B,A,D,C))$

6 Related Work

There is a number of axiom systems for Euclidean geometry. Most of them are variants of Euclid's, Hilbert's or Tarski's system and their comparison often require subtle analyses [37, 26, 27]. Developing new axiom systems is still an active research area, often motivated by machine formalizations. For instance, Avigad, Dean, and Mumma recently proposed an axiomatization [1] that rather faithfully captures basic ideas and methods of inference outlined in Euclid's "Elements", but in a rigorous manner.

#	ARGO system	Tarski's system	Borsuk's system	Hilbert's system
1	-	NA	-	-
2	0.01/5/5	NA	0.01/5/5	0.01/5/5
3	0.01/5/5	NA	0.01/5/5	0.01/5/5
4	-	NA	-	-
5	0.01/27/1	NA	0.02/28/1	-
6	-	NA	15.92/524/259	-
7	11.33/125/6	NA	8.31/119/6	-
8	0.02/12/11	NA	0.02/12/11	0.01/12/11
9	-	NA	-	-
10	0.01/2/1	-	0.01/2/1	-
11	-	-	0.08/71/7	-
12	0.01/5/2	0.01/6/2	0.01/6/2	-
13	0.25/13/3	0.16/24/3	0.21/24/3	-
14	1.26/26/7	0.52/30/7	0.58/30/7	-

Table 1. Performance of the prover; entries are given in the form time/n_1/n_2, where n_1 is the number of all axioms applied, and n_2 is the number of axioms applied in a „clean" proof (with eliminated all unnecessary steps) in the natural language form; '-' denotes timeout, NA denotes that the theorem does not belong to the language of the theory; experiments were ran on PC Core 2Quad 2.4GHz with 4GB RAM, running under Linux.

A lot of efforts have been recently invested into formalization of geometry. Dehlinger, Dufourd and Shreck worked on formalization of first two groups of Hilbert's *Grundlagen* in Coq proof assistant following an intuitionistic approach [12]; they came to the conclusion that many theorems could not be proved this way. Meikle and Fleuriot [26] formalized the first three groups of Hilbert's axiomatics in Isabelle/Isar. They showed that some Hilbert's proofs relied on some implicit assumptions (most often based upon a graphical presentation of the problem) and in this way again emphasized the need of having formally verified proofs. Narboux formalized [28] in Coq the first eight chapters of Tarski's book [31] and demonstrated that geometry of Tarski is suitable for mechanization because of its simplicity and production of less degenerated cases. There are also other geometry related formalizations developed in Coq: Kahn's formalization of von Plato's constructive geometry [36, 21], Guilhot's formalization of large portions of high school geometry [16], Duprat's formalisation of an axiom system for compass and ruler geometry [13], formalization of projective geometry by Magaud, Narboux and Schreck [23, 24], etc. All of the mentioned formalizations were done completely manually, with no automation involved.

Automated theorem proving has a history more than fifty years long [9]. The biggest successes were achieved (i.e., the most complex theorems were proved) by algebraic theorem provers based on Wu's [40, 8] method and Gröbner bases method [6, 22, 14]. However, instead of readable, traditional geometry proofs, these methods produce only a yes/no answer with a corresponding algebraic

argument. This is partly changed with coordinate-free methods, such as the area method [10], the full angle method [11, 7], but for many conjectures these methods still deal with extremely complex expressions involving certain geometry quantities. Quaife used a resolution theorem prover to prove theorems in Tarski's geometry [30]. Some challenging conjectures were proved, but no formal or readable proofs were produced.

Coherent logic may serve as a framework that enables automated generation of readable geometry proofs. It is well suited to foundational conjectures, close to the level of axioms. To our knowledge, the first automated theorem prover using CL was developed by Janičić and Kordić [20]. It used a fixed set of geometry axioms close to Borsuk's system [5] and was able to prove tens of foundational theorems from standard geometry textbooks. No formal proofs were generated. The system that we describe in this paper is related to this system but significantly extends it and improves it in several directions.

Over the last several years, CL was explored and popularized by Marc Bezem and his coauthors. Bezem and Coquand [2] developed in Prolog a CL prover that generates proof objects in Coq (some of the problems solved by this CL prover can be found on-line[6]). Berghofer and Bezem developed an internal prover for CL in Isabelle (using shallow embedding), which can be used as a built-in tactic. It has many advantages to "external" provers: it uses existing Isabelle's infrastructure, excludes the need for converting from/to "external" formats and it adds one more level of verification: it assures the correctness of proof procedure itself. Declarative programming languages such as Prolog and Isabelle language used as a functional programming language are well suited to this kind of problems but they can result in a slow executable code, so we believe that C++ implementation can tackle more realistic geometry theorems. As we are aware of, the above provers have not been used for dealing with fragments of geometry addressed in this paper.

7 Conclusions and Further work

We presented a theorem prover ArgoCLP that uses coherent logic as its underlying logic and a forward chaining and iterative deepening in its proof search. The prover can be used for any theory with coherent axioms and for conjectures in the coherent form. It can produce formal, machine verifiable proofs, but also readable proofs given in a natural language form, consisting of steps typical for traditional geometry proofs, so they can be directly used in textbooks. We applied the prover to various axiomatic systems for Euclidean geometry and proved dozens of theorems from standard university textbooks on geometry. Since the generated proofs are both formal and readable, they can be used in different educational purposes, and thanks to automation, the system can serve as a useful tool for building the body of formalized mathematics. The system, in its current version, still does not aim at proving all complex geometry theorems appearing

[6] `http://www.ii.uib.no/~bezem/GL/`

in geometry textbooks, but rather at proving foundational theorems (close to the axiom level) of moderate hardness. For instance, a suitable problem for the prover would be checking if an axiom A could be replaced by another version A' (by proving A with A' and the rest of the system and by proving A' with A and the rest of the system). This is a very important issue for foundations of geometry — there are many axiom systems, sometimes with only slight modifications (following, for instance, different interpretations of the author's intention) and establishing their relationship could be very demanding. Therefore, automation in this process is very much welcome. In addition, the system can be used as an assistant for proving appropriately chosen subgoals of complex conjectures, in a manner that was already applied in proving Hessenberg's theorem by a CL-based theorem prover [4].

We are planning to further develop our prover as there is still much space for improving efficiency. We have implemented a mechanism for cleaning up all irrelevant proof steps from a proof trace, but this cleaning is done only *post festum*, when the conjecture is already proved. We are planning to implement a similar mechanism that would be applied during the proving process since information about relevant/irrelevant facts can be useful in more efficient search guiding in the remaining process (i.e., in future branches). We are also planning to use techniques used in other automated reasoning systems (e.g., SAT solvers) and we expect to obtain significant speed-ups and significant increase in number of theorems that can be proved within reasonable time limits. With a more efficient version of the prover, we are planning to formalize significant portions of different geometries, including the geometry developed by Avigad, Dean, and Mumma [1]. We are also planning to deeply explore different variants of the most significant axioms systems and their relationship by automatically proving axioms of one systems as theorems within another system. That work would answer a number of important questions about formulations of axioms. The domain of our prover is not limited to geometry, so we will apply it to other theories as well. In addition, we are planning to support input from the TPTP form[7] [38] (so we can compare our prover with other automated theorem provers, e.g., resolution-based provers) and we are planning to add support for exporting proof objects to other proof assistants (e.g., Coq).

References

1. J. Avigad, E. Dean, , and J. Mumma. A formal system for Euclid's Elements. *The Review of Symbolic Logic*, 2009.
2. Marc Bezem and Thierry Coquand. Automating coherent logic. In Geoff Sutcliffe and Andrei Voronkov, editors, *12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning — LPAR 2005*, volume 3835 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.
3. Marc Bezem and Dimitri Hendriks. On the Mechanization of the Proof of Hessenberg's Theorem in Coherent Logic. *Journal of Automated Reasoning*, 40(1), 2008.

[7] `http://www.tptp.org`

4. Marc Bezem and Dimitri Hendriks. On the mechanization of the proof of hessenberg's theorem in coherent logic. *J. Autom. Reasoning*, 40(1), 2008.
5. Karol Borsuk and Wanda Szmielew. *Foundations of Geometry*. Norht-Holland Publishing Company, Amsterdam, 1960.
6. Bruno Buchberger. *An Algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal*. PhD thesis, Math. Inst. University of Innsbruck, Austria, 1965.
7. S.C. Chou, X.S. Gao, and J.Z. Zhang. *Machine Proofs in Geometry*. World Scientific, Singapore, 1994.
8. Shang-Ching Chou. *Mechanical Geometry Theorem Proving*. D.Reidel Publishing Company, Dordrecht, 1988.
9. Shang-Ching Chou and Xiao-Shan Gao. Automated reasoning in geometry. In *Handbook of Automated Reasoning*. Elsevier, 2001.
10. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated production of traditional proofs for constructive geometry theorems. In Moshe Vardi, editor, *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science LICS*, pages 48–56. IEEE Computer Society Press, June 1993.
11. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants, II. theorem proving with full-angles. *Journal of Automated Reasoning*, 17:349–370, 1996.
12. Christophe Dehlinger, Jean-François Dufourd, and Pascal Schreck. Higher-order intuitionistic formalization and proofs in Hilbert's elementary geometry. In *Automated Deduction in Geometry*, volume 2061 of *Lecture Notes in Computer Science*. Springer, 2001.
13. Jean Duprat. Une axiomatique de la géométrie plane en coq. In *Actes des JFLA 2008*, pages 123–136. INRIA, 2008. In french.
14. Bruno Buchberger et.al. Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, 2006.
15. John Fisher and Marc Bezem. Skolem machines and geometric logic. In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *4th International Colloquium on Theoretical Aspects of Computing — ICTAC 2007*, volume 4711 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.
16. Frédérique Guilhot. Formalisation en coq d'un cours de géométrie pour le lycée. In *Journées Francophones des Langages Applicatifs*, Janvier 2004.
17. John Harrison. Hol light: A tutorial introduction. In Mandayam K. Srivas and Albert John Camilleri, editors, *Formal Methods in Computer-Aided Design*, volume 1166 of *Lecture Notes in Computer Science*. Springer, 1996.
18. Thomas L. Heath. *The Thirteen Books of Euclid's Elements*. Dover Publications, New-York, 1956. 2nd ed.
19. David Hilbert. *Grundlagen der Geometrie*. Leipzig, 1899.
20. Predrag Janičić and Stevan Kordić. EUCLID — the geometry theorem prover. *FILOMAT*, 9(3):723–732, 1995.
21. Gilles Kahn. Constructive geometry according to Jan von Plato. Coq contribution, 1995. Coq V5.10.
22. D. Kapur. Using Gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2(4):399–408, 1986.
23. Nicolas Magaud, Julien Narboux, and Pascal Schreck. Formalizing Projective Plane Geometry in Coq. In *Proceedings of ADG'08*, June 2008.
24. Nicolas Magaud, Julien Narboux, and Pascal Schreck. Formalizing desargues' theorem in coq using ranks. In Sung Y. Shin and Sascha Ossowski, editors, *SAC*, pages 1110–1115. ACM, 2009.

25. Laura Meikle and Jacques Fleuriot. Formalizing Hilbert's Grundlagen in Isabelle/Isar. In *Theorem Proving in Higher Order Logics*, pages 319–334, 2003.
26. Laura Meikle and Jacques Fleuriot. Formalizing Hilbert's Grundlagen in Isabelle/Isar. In *Proceedings of TPHOLs*, volume 2758 of *Lecture Notes in Computer Science*. Springer, 2003.
27. Julien Narboux. *Formalisation et automatisation du raisonnement géométrique en Coq*. PhD thesis, Université Paris Sud, September 2006.
28. Julien Narboux. Mechanical Theorem Proving in Tarski's Geometry. In *Proceedings of Automated Deduction in Geometry 2006*, volume 4869 of *Lecture Notes in Computer Science*. Springer, 2007.
29. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer, 2002.
30. Art Quaife. Automated development of tarski's geometry. *Journal of Automated Reasoning*, 5(1):97–118, 1989.
31. Wolfram Schwabhuser, Wanda Szmielew, and Alfred Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, Berlin, 1983.
32. R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, April 1975.
33. Alfred Tarski. What is elementary geometry? In P. Suppes L. Henkin and A. Tarski, editors, *The axiomatic Method, with special reference to Geometry and Physics*, pages 16–29, Amsterdam, 1959. North-Holland.
34. The Coq development team. *The Coq proof assistant reference manual, Version 8.2*. TypiCal Project, 2009.
35. Andrzej Trybulec. Mizar. In Freek Wiedijk, editor, *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.
36. Jan von Plato. The axioms of constructive geometry. *Annals of Pure and Applied Logic*, 76:169–200, 1995.
37. Jan von Plato. Formalization of Hilbert's geometry of incidence and parallelism. *Synthese*, 110:127–141, 1997.
38. Geoff Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337-362, 2009.
39. Markus Wenzel. Isar - a generic interpretative approach to readable formal proof documents. In Yves Bertot, Gilles Dowek, André Hirschowitz, C. Paulin, and Laurent Théry, editors, *Theorem Proving in Higher Order Logics (TPHOLs'99)*, volume 1690 of *Lecture Notes in Computer Science*, pages 167–184. Springer, 1999.
40. Wen-Tsün Wu. On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica*, 21:157–179, 1978.