Preprocessing of the Axiomatic System for More Efficient Automated Proving and Shorter Proofs*

Sana Stojanović

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

Abstract. One of the main differences between pen-and-paper proofs and computer proofs is in the number of simple facts derived. In automated proof generation, the number of simple facts can be large. We are addressing this problem by preprocessing of the axiomatic system that should enable reduction in the number of simple and redundant facts to some extent. We implemented two types of preprocessing techniques, one concerning symmetric predicates, and another restricting introduction of witnesses during proof search. Both techniques were used within a coherent logic prover ArgoCLP. Evaluations performed on geometrical domain show that use of these techniques makes automated process more efficient and generated proofs often significantly shorter.

Keywords: Predicate symmetry, Axiom reformulation, Coherent logic, Automated and formal theorem proving.

1 Introduction

One of common challenges in automated and interactive theorem proving is coping with proving "simple facts". In traditional pen-and-paper theorem proving, "simple facts" – trivial facts that don't need deep proofs and theory-specific arguments – are typically assumed or neglected. However, in automated theorem proving "simple facts" can significantly increase the search space, while in interactive theorem proving they can make a significant burden in following the main line of the proof. Because of this, there is a number of methods and techniques for dealing with simple facts both in automated and interactive theorem proving. In this paper, we will address this issues in the context of automated generation of formal (machine verifiable) but also readable proofs.

We address two sorts of simple facts – those that rely on symmetry properties of certain predicates and those that rely on specific reformulations of certain axioms or lemmas. Symmetries and properties of symmetrical predicates have

* This work has been partly supported by the grant 174021 of the Ministry of Science of Serbia and by the SNF SCOPES grant IZ73Z0_127979/1.

been widely studied and used in automated reasoning [5,6,7], but in this paper we will consider one variant sufficient for our purposes. In cases when the prover introduces witnesses during the proof search, a large number of (potentially unnecessary) witnesses can lead to larger search space and redundant steps.

As an answer to those problems we propose two simple preprocessing techniques. We built these two techniques into ArgoCLP [12] – a prover based on coherent logic and forward reasoning that generates machine verifiable proofs in Isar [11] and proofs in natural language. In the evaluation, we are focusing on Euclidean geometry. The evaluation shows that application of these two techniques improves the efficiency of the prover and generate proofs which are often significantly shorter. Moreover, the readable proofs that ArgoCLP generates become more similar to the proofs that can be found in mathematics textbooks.

This paper is organized as follows: in Section 2 we give background information on coherent logic and on prover ArgoCLP; in Section 3 we describe the importance of symmetric predicates, their detection and use; in Section 4 we describe a technique dealing with axioms that introduce several witnesses; in Section 5 we give an overview of our experiments; in Section 6 we give an overview of related work; and in Section 7 we give our conclusions and plans for the future work.

2 Coherent Logic and ArgoCLP Prover

Coherent Logic. Coherent logic (CL) was initially defined by Skolem and in recent years was popularized by Bezem [3]. It is a fragment of first-order logic, consisting of implicitly universally quantified formulae of the following form:

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y_1} B_1(\boldsymbol{x}, \boldsymbol{y_1}) \vee \ldots \vee \exists \boldsymbol{y_m} B_m(\boldsymbol{x}, \boldsymbol{y_m}) \tag{2.1}$$

where: $n \geq 0$, $m \geq 0$, $\boldsymbol{x}$ denotes a sequence of variables, 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, 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. A *term* is a constant or a variable. An *atomic formula* is either $\perp$ 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.* CL deals with the sets of facts — ground atomic expressions.

The reasoning in coherent logic is constructive and proof objects can easily be obtained, therefore CL is suitable for producing both readable and formal proofs. A large number of theories and theorems can be formulated directly in CL. There exists a linear translation from FOL to CL that preserves logical equivalence.

ArgoCLP Prover. ArgoCLP [12] is a generic theorem prover based on coherent logic that automatically produces formal proofs in Isar and readable proofs in

English that resemble proofs that can be found in mathematics textbooks. It can be used with any set of coherent axioms. The proof procedure is simple forward chaining with iterative deepening. Since negations are not supported in coherent logic, for every predicate symbol R, typically an additional symbol $\overline{R}$ is introduced (that stands for $\neg R$) and the following two axioms are added to the set of axioms (for every predicate R): $R(x) \wedge \overline{R}(x) \Rightarrow \bot$, and $R(x) \vee \overline{R}(x)$ (section A.1 of the appendix).

The following example shows a proof of one geometry theorem in natural language generated by the ArgoCLP prover.

Example 1. Proof generated by the ArgoCLP prover.

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

Proof

Let us prove that $p = r$ by reductio ad absurdum.

1. Assume that $p \neq r$.

 2. 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).

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

   4. From the facts that $p \neq q$, and the point A is incident 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).

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

   Contradiction.

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

   7. 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_21).

   8. 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 line r is incident to the plane α, and the lines q and r do not intersect, it holds that $p = r$ (by axiom ax_E2).

   9. From the facts that $p = r$, and $p \neq r$ we get a contradiction.

   Contradiction.

Therefore, it holds that $p = r$.

This proves the conjecture.

Theorem proved in 9 steps and in 0.02 s.

3 Dealing with Symmetric Predicates

Dealing with symmetric predicates is standardly supported in many automated theorem provers, in different forms. In this paper we are attempting to automatically add support for symmetric predicates as preprocessing technique.

Definition 1. *(Symmetric predicate) An n-ary predicate R is symmetric (in all arguments) if the following (universally quantified) statement holds in the considered theory for every permutation σ:*

$$R(x_1, \ldots, x_n) \Leftrightarrow R(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$$

Theorem 1. *An n-ary predicate R is symmetric if and only if the following two (universally quantified) statements hold:*

$$R(x_1, x_2, x_3, \ldots, x_n) \Leftrightarrow R(x_2, x_1, x_3 \ldots, x_n) \tag{3.1}$$

$$R(x_1, x_2, x_3 \ldots, x_n) \Leftrightarrow R(x_2, x_3, \ldots, x_n, x_1) \tag{3.2}$$

In case that these two formulae are lemmas, formulae that express symmetry of all other permutations are lemmas as well. Those lemmas and their proofs are important for automated theorem proving and for generating formal proofs. The set of all lemmas that express symmetry of a predicate will be referred to as *the symmetry lemmas.*

Preprocessing Phase. The statements (3.1) and (3.2) can be automatically generated from the set of predicates of the axiomatic system, and the prover can then automatically check whether these formulae are lemmas of the considered theory (since not all of them will be lemmas, a time restriction must be set). For symmetric predicates we can automatically generate a set of all symmetry lemmas which can then be used in the automatic generation of formal proofs in a manner described bellow.

Automated Proving Phase. In order to efficiently manage symmetric predicates, all permutations of arguments of ground atomic formula will be represented by a single permutation — a sorted one (for instance, using lexicographic ordering). For example, for the predicate of collinearity and constants A, B, C, ground atomic formulae $col(B, A, C)$ and $col(A, C, B)$ will both be represented with the ground atom $col(A, B, C)$, and are considered the same during proof search.

Object Level Proof Construction Phase. During proof generation these steps are complemented with the following two lemmas:

$col(B, A, C) \Rightarrow col(A, B, C)$
$col(A, B, C) \Rightarrow col(C, A, B)$

Assuming that these lemmas were already proven by the prover, a complete (formal) proof can be generated.

4 Dealing with Axiom Reformulations

Consider a (coherent) axiom of the following form:

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y} B(\boldsymbol{x}, \boldsymbol{y}) \tag{4.1}$$

where $n \geq 0$, $\boldsymbol{y} = \{y_1, y_2, \ldots, y_k\}$ and $k \geq 2$. During the proof search, in the context of forward chaining, this axiom *introduces k witnesses*:

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists y_1 \exists y_2 \ldots \exists y_k B(\boldsymbol{x}, y_1, \ldots, y_k)$$

In case when $k = 1$ the coherent logic prover does not apply axiom of the form $A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists y_1 B(\boldsymbol{x}, y_1)$ if there is a such that $B(\boldsymbol{x}, a)$ holds, but in the case of k existential quantifiers ($k > 1$) it checks for all k witnesses. I.e., this axiom will not be applied if all of the witnesses instantiating $y_1, \ldots, y_k$ already exist, but if at least one of them is missing, the axiom will introduce k new witnesses. Our goal is to introduce less than k witnesses in case when some of the witnesses that satisfy the axiom already exists.

Let the formula B be a conjunction of atoms, such that it can be represented as $B = B_1 \wedge B_2$, where B_1 is a conjunction of all atoms that have *only* variables from $\boldsymbol{x}$ and y_1 (if such atoms do not exist, B_1 is $\top$), and B_2 is a non-empty conjunction of all other atoms from B:

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists y_1 \ldots \exists y_k (B_1(\boldsymbol{x}, y_1) \wedge B_2(\boldsymbol{x}, y_1, \ldots, y_k))$$

Consider the transformed version of the previous statement:

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \wedge B_1(\boldsymbol{x}, y_1) \Rightarrow \exists y_2 \ldots \exists y_k B_2(\boldsymbol{x}, y_1, \ldots, y_k) \tag{4.2}$$

This statement introduces less witnesses that the original one. In a general case, such statement is not a consequent of axioms and is not provable within a given theory. In case that it is provable, it will be used as a lemma but with higher priority than the original axiom.

Preprocessing Phase. Axioms of the form 4.1 can be automatically recognized (from the set of axioms) and a transformed statement can be generated for each of them. The prover can check if a transformed statement is provable (since not all generated statements will be provable, time restriction must be set). If the transformed statement is provable, the transformation can be applied for the new formula. This process is iterated while there can be found non empty formula B_2.

Automated Proving Phase. The generated lemmas are used as axioms, but during a proof search the prover gives higher priority to those lemmas over the original axiom.

Example 2. Considering the proofs of theorems based on Hilbert's axioms, we notice that certain axioms are rarely applied in their original form. For example, the axiom *I3: On every line there lie two different points*, is more often applied in the following manner *(I3a): If there is a point A on the line p, then there is a point B which is different from A and lies on p.* This statement does not correspond to a verbatim application of that axiom. We should actually introduce two new points B and C, such that B and C are different and that both of them lie on p. Nevertheless, this manner of application of axiom $I3$ is standardly used in mathematical proofs. The statement $I3a$ can actually be proven as a theorem, and that would justify using it.

A problem that occurs with this approach is that not all generated statements will be theorems. Let us consider the following example:

Example 3. Axiom *I8: There exist three non-collinear points*, will generate the following two statements:

1. *Given a point A, there exist points B and C such that A, B and C are non-collinear.*
2. *Given points A and B, there exists a point C such that A, B and C are non-collinear.*

The first statement is a theorem, but the second statement is not a theorem. It lacks an additional condition in its premises, i.e., points A and B must be different. In such case, user may be prompted to try to assist and to add missing premises.

5 Applications in Euclidean Geometry

In this section we discuss both the efficiency of the presented preprocessing techniques, and the effects that they have on the power of the prover. Both techniques were implemented in the prover ArgoCLP. The tool that implements preprocessing techniques is separated from the prover itself. That way, the generation of auxiliary theorems is performed only once for one axiomatic system and need not be performed every time when proving a theorem. All experiments were performed on AMD Opteron 2GHz with 96GB RAM[1]. The system was applied on a Hilbert style axiomatic system (only axioms of the first group of Hilbert's axioms were used). Most of Hilbert's axioms and theorems are directly expressible in coherent logic[2].

Axiomatic System. Our axiomatic system is based on Hilbert's axiomatic system [8]. Since ArgoCLP works with coherent logic, some Hilbert's axioms had to be transformed in coherent logic form. The main transformation is elimination of negation. As discussed in Section 2, for each predicate[3] R new predicate $\overline{R}$

[1] All materials can be found in `http://www.matf.bg.ac.rs/~sana/system.zip`

[2] Avigad, Dean, and Mumma [1] also noticed that a strong syntactic restriction on formulae, similar to the coherent logic, is adequate to representing the axioms and theorems of Euclid's plane geometry.

[3] Incidence of point with a line, incidence of point with a plane, incidence of line with a plane; intersection of two lines, intersection of two planes; collinearity of three points, coplanarity of four points, relation between for three points, congruence between pairs of points.

is added and the following axiom (definition) is added to the axiomatic system: $R \wedge \overline{R} \Rightarrow \bot$. For some predicates R it is easy to define these new predicates $\overline{R}$ explicitly[4]. Alternatively, the following axiom can be used (which is less preferred, because of cases split it introduces): $R \vee \overline{R}$. For predicates that are defined, this formula can be proven as a theorem[5]. Those definitions are mainly trivial and they are presented in appendix A.1.

Automated Detection of Symmetric Predicates. Symmetric property of predicates R and $\overline{R}$ are proven separately because symmetry lemmas for both of those predicates are used in completion of proofs.

Only predicates whose arguments are all of the same type are processed (positive and negative form of intersection of lines, intersection of planes, between, collinearity, coplanarity, and congruence). A total of 20 statements of the form (3.1) and (3.2) was generated (for predicates of arity two, these statements are identical). Only those predicates for which both statements were proved can be used as symmetric in proving geometry theorems.

In the set of the listed predicates, the following eight are symmetric: positive and negative form of intersection of lines, intersection of planes, collinearity and coplanarity. ArgoCLP succeeded in proving that seven out of these eight predicates are symmetric (all but $\overline{\textit{coplanarity}}$) with the average execution time 3.6 seconds and the average number of steps 170. For predicates that are proven to be symmetric, all lemmas that express symmetry of the predicate by permutations not covered by (3.1) and (3.2) are generated and again ArgoCLP was used to generate their proofs. There are 40 such lemmas in total (with proving time under 2 seconds).

Automated Reformulation of Axioms. The set of axioms that are automatically recognized as axioms which introduce more than one witness is:

I3a There exist at least two different points on a line.
I3b There exist at least three points that are non-collinear.
I8 There exist at least four points which are non-coplanar.[6]

Statements that are automatically generated from this set, in the manner described in section 4, are:

I3a1 $(\forall p : \mathit{Line})(\forall A : \mathit{Point}) A \in p \Rightarrow (\exists B : \mathit{Point})(A \neq B \wedge B \in p)$
I3b1 $(\forall A : \mathit{Point})(\exists B : \mathit{Point})(\exists C : \mathit{Point})\neg \mathit{col}(A, B, C)$
I3b2 $(\forall A : \mathit{Point})(\forall B : \mathit{Point})(\exists C : \mathit{Point})\neg \mathit{col}(A, B, C)$

[4] $\overline{\mathit{col}}(A, B, C) \leftrightarrow (\exists p)(A \in p \wedge B \in p \wedge C \notin p)$.

[5] For example, if $A \in p \vee \overline{A \in p}$ holds, then it is trivial to prove $\mathit{col}(A, P, Q) \vee \overline{\mathit{col}}(A, P, Q)$ (with appropriately defined predicate $\overline{\mathit{col}}$).

[6] These are not original formulations of the axioms. Axioms had to be slightly modified in order to express them in coherent logic. Original formulation of these axioms is: [I3a] There exist at least two points on a line, [I3b] There exist at least three points that do not lie on a line, [I8] There exist at least four points which do not lie in a plane.

I8a $(\forall A : Point)(\exists B : Point)(\exists C : Point)(\exists D : Point)\neg cop(A, B, C, D)$
I8b $(\forall A : Point)(\forall B : Point)(\exists C : Point)(\exists D : Point)\neg cop(A, B, C, D)$
I8c $(\forall A : Point)(\forall B : Point)(\forall C : Point)(\exists D : Point)\neg cop(A, B, C, D)$

Two of these statements were proven (theorems $I3a1$ and $I3b1$), and formal proofs in Isar were generated. Statements $I3b2$ and $I8b$ lack the condition $A \neq B$, and statement $I8c$ lacks the condition $\neg col(A, B, C)$ in order to be theorems. Total time of this preprocessing phase is 20 minutes (when time limit is set to 5 minutes).

Improvement in the Performance of ArgoCLP. The described techniques were tested on the prover ArgoCLP with small benchmark set of 24 theorems[7] that were all proved with ArgoCLP. The median and average proving time were 2 minutes and 23 minutes (due to four hard lemmas), while the average number of steps was 1374.

1. *Effect of Symmetric Predicates.* For this evaluation we used only the automatically proven symmetry theorems (that express symmetry for 7 out of 8 symmetric predicates). With this technique average proving time was reduced by 56% and the average number of steps by 83% (average proving time was 10 minutes and average number of steps 230).
2. *Exploitation of Axiom Reformulation.* For this evaluation we used only the reformulated versions of axioms that were proven automatically (two lemmas). We compare the performance of the prover using both techniques to its performance when using only the technique for dealing with symmetries. The average proving time was reduced by 15% and the average number of steps by 37% (average proving time was 8.5 minutes and average number of steps 143.).
3. *The total improvement* when using both techniques compared to the original prover is 63% in the average proving time and 89% in the average number of steps.

6 Related Work

Techniques for handling symmetric predicates are widely used in automated reasoning and constraint programming. The symmetry of problem instances (propositional formulae, CSPs, etc.), are intensively exploited in theorem proving. For instance, Arai and Masukawa [2] designed a ground theorem prover Godzila that quickly finds symmetries in combinatorial problems. Cadoli and Mancini [4,9] considered specifications of constraint problems as logical formulae, and used automated theorem prover to determine existence of symmetries and functional dependencies.

[7] For newly introduced predicates that are defined with new axioms, excluded middle rule is a theorem and is a part of this set.

Dealing with symmetries in geometry dates back to 1959 and Gelernter [7]. He was dealing with the problem of efficient machine manipulation of formal systems in which the predicates display a high degree of symmetry, and successfully eliminated symmetry-redundant goals.

Chou, Gao, and Zhang developed a deductive database method [6] that can be used to prove or discover nontrivial geometry theorems. They noticed that most geometric predicates (such as collinearity) satisfy properties such as transitivity and symmetry, which leads to a large database and repetitive representation of information. They used equivalence classes and canonical forms (in a way that is similar to ours) to represent facts in the database and reduced size of the database by a factor of 100.

Caferra, Peltier, and Puitg [5] used a similar approach in geometry theorem proving in order to reduce the number of facts conveying the same information. They incorporated human techniques in the prover which increased its power and made user interaction more natural. They encoded equational theories such as commutativity and circularity in the unification algorithm. Only the minimal representative of equivalence class (determined by commutativity and circularity) is stored into the database which results in a significant speed up.

Stojanović, Pavlović, and Janičić [12] developed a prover ArgoCLP but analyzed symmetries in a Hilbert style axiomatic system by hand. The information on all symmetric predicates (all eight predicates, positive and negative form of four symmetric predicates) was given to the prover as a new set of lemmas. Also, reformulation of axioms was done manually and all reformulated formulae (six) were added to the set of lemmas (after human intervention and inclusion of the missing relations). In this paper we showed that to some extent this process can be done automatically, without human intervention.

Meikle and Fleuriot [10] developed a semi-automated approach[8] to geometry theorem proving. They automated in Isabelle/HOL some of the geometric reasoning (symmetry of collinearity) that is often required in verification tasks by extending Isabelle's simplifier and classical reasoner. They noticed that some of those theorems alone could generate infinite loops and that special attention is needed. The potential problem with symmetry theorems and infinite loops are not present with our approach either. The theorems that can generate an infinite loop will not be used by our prover during the proof search, but only to complete the proofs in places where the knowledge about symmetry is needed. In contrast to their approach where proofs of theorems that express symmetry of collinearity were proved by hand, in our approach, all symmetry theorems that are used were proven automatically.

We are not aware of prior work related to axiom reformulation.

[8] Semi-automated theorem proving is using automated techniques that are implemented within interactive theorem proving assistant. During interactive process, the user has the option of using the help of tool for automation if it manages to derive some useful goals.

7 Conclusions and Future Work

We proposed two techniques for modification of an axiomatic system that improve efficiency of forward chaining automated theorem provers. These techniques were implemented and tested within ArgoCLP prover and the evaluation was performed on the theorems of Euclidean geometry using Hilbert like axiomatic system.

Automatic detection of symmetries managed to detect 7 out of 8 symmetric predicates. The automated reformulation of axioms managed to detect 2 out of 6 useful lemmas. We showed that the exploitation of this knowledge during the work of the prover can give better performance compared to the original prover. The execution time was shortened by 63%. In some cases the proposed techniques do not provide any speedup, but they never result in decreased efficiency.

With these simple techniques the generated proofs are shorter, simpler, and closer to proofs from mathematics textbooks. These techniques can be useful, for instance, for checking equivalence of different axiomatic systems (Hilbert's axiomatic system has several different interpretation).

Apart from predicates that are symmetric on all arguments, there are predicates which are symmetric only in some arguments (*between, congruence*). They are not discussed in this paper but can be treated in the same manner.

Most of the statements generated by reformulation of axioms in our experiments are not provable because they are missing additional condition in premises (4 out of 6 generated statements are not provable). One way of dealing with this problem is user assistance and modification of those statements manually. In the future, we plan to devise a heuristic for automatic discovery of additional conditions.

References

1. Avigad, J., Dean, E., Mumma, J.: A formal system for Euclid's Elements. The Review of Symbolic Logic (2009)
2. Arai, N.H., Masukawa, R.: How to Find Symmetries Hidden in Combinatorial Problems. In: Proceedings of the Eighth Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (2000)
3. Bezem, M., Coquand, T.: Automating coherent logic. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 246–260. Springer, Heidelberg (2005)
4. Cadoli, M., Mancini, T.: Using a Theorem Prover for Reasoning on Constraint Problems. In: Bandini, S., Manzoni, S. (eds.) AI*IA 2005. LNCS (LNAI), vol. 3673, pp. 38–49. Springer, Heidelberg (2005)
5. Caferra, R., Peltier, N., Puitg, F.: Emphasizing Human Techniques in Automated Geometry Theorem Proving A Practical Realization. In: Richter-Gebert, J., Wang, D. (eds.) ADG 2000. LNCS (LNAI), vol. 2061, pp. 268–305. Springer, Heidelberg (2001)
6. Chou, S.-C., Gao, X.-S., Zhang, J.-Z.: A Deductive Database Approach to Automated Geometry Theorem Proving and Discovering. Journal of Automated Reasoning (2000)

7. Gelernter, H.: A Note on Syntatic Symmetry and the Manipulation of Formal Systems by Machine. Information and Control (1959)
8. Hilbert, D.: Grundlagen der Geometrie, Leipzig (1899)
9. Mancini, T., Cadoli, M.: Detecting and Breaking Symmetries by Reasoning on Problem Specifications. In: Zucker, J.-D., Saitta, L. (eds.) SARA 2005. LNCS (LNAI), vol. 3607, pp. 165–181. Springer, Heidelberg (2005)
10. Meikle, L., Fleuriot, J.: Automation for Geometry in Isabelle/HOL. In: Proceedings of PAAR, FLOC 2010 (2010)
11. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL. LNCS, vol. 2283. Springer, Heidelberg (2002)
12. Stojanović, S., Pavlović, V., Janičić, P.: A Coherent Logic Based Geometry Theorem Prover Capable of Producing Formal and Readable Proofs. In: Schreck, P., Narboux, J., Richter-Gebert, J. (eds.) ADG 2010. LNCS (LNAI), vol. 6877, pp. 201–220. Springer, Heidelberg (2011)

A Hilbert's Axiomatic System I Group of Axioms

For readability, instead of writing $\overline{A \in p}$ we will write $A \notin p$, and instead of writing $\overline{col}(A, B, C)$ we will write $\neg col(A, B, C)$.

Universal quantifiers and types are omitted in the formule for readability. Capital letters are used for points, small letters are used for lines, and Greek letters are used for planes.

I1 $A \neq B \Rightarrow (\exists p : line)(A \in p \wedge B \in p)$
I2 $A \neq B \wedge A \in p \wedge B \in p \wedge A \in q \wedge B \in q \Rightarrow p = q$
I3a $(\exists A : point)(\exists B : point)(A \neq B \wedge A \in p \wedge B \in p)$
I3b $(\exists A : point)(\exists B : point)(\exists C : point)\neg col(A, B, C)$
I4a $\neg col(A, B, C) \Rightarrow (\exists \alpha : plane)(A \in \alpha \wedge B \in \alpha \wedge C \in \alpha)$
I4b $(\exists A : point)A \in \alpha$
I5 $\neg col(A, B, C) \wedge A \in \alpha \wedge B \in \alpha \wedge C \in \alpha \wedge A \in \beta \wedge B \in \beta \wedge C \in \beta \Rightarrow \alpha = \beta$
I6 $A \neq B \wedge A \in p \wedge A \in \alpha \wedge B \in p \wedge B \in \alpha \Rightarrow p \in \alpha$
I7 $\alpha \neq \beta \wedge A \in \alpha \wedge A \in \beta \Rightarrow (\exists B : point)(A \neq B \wedge B \in \alpha \wedge B \in \beta)$
I8 $(\exists A : point)(\exists B : point)(\exists C : point)(\exists D : point)\neg cop(A, B, C, D)$

A.1 Additional Axioms

1. $A = B \vee A \neq B$
2. $p = q \vee p \neq q$
3. $\alpha = \beta \vee \alpha \neq \beta$
4. $A \in p \vee A \notin p$
5. $A \in \alpha \vee A \notin \alpha$
6. $A \in p \wedge B \in p \wedge C \in p \Rightarrow col(A, B, C)$
7. $col(A, B, C) \Rightarrow (\exists p : line)(A \in p \wedge B \in p \wedge C \in p)$
8. $A \neq B \wedge A \in p \wedge B \in p \wedge C \notin p \Rightarrow \neg col(A, B, C)$
9. $A \in \alpha \wedge B \in \alpha \wedge C \in \alpha \wedge D \in \alpha \Rightarrow cop(A, B, C, D)$
10. $cop(A, B, C, D) \Rightarrow (\exists \alpha : plane)(A \in \alpha \wedge B \in \alpha \wedge C \in \alpha \wedge D \in \alpha)$
11. $\neg col(A, B, C) \wedge A \in \alpha \wedge B \in \alpha \wedge C \in \alpha \wedge D \notin \alpha \Rightarrow \neg cop(A, B, C, D)$

12. $p \neq q \wedge A \in p \wedge A \in q \Rightarrow int(p, q)$
13. $int(p, q) \Rightarrow (\exists A : point)(A \in p \wedge A \in q \wedge p \neq q)$
14. $int(p, q) \vee \neg int(p, q)$
15. $\alpha \neq \beta \wedge A \in \alpha \wedge A \in \beta \Rightarrow int(\alpha, \beta)$
16. $int(\alpha, \beta) \Rightarrow (\exists A : point)(A \in \alpha \wedge A \in \beta \wedge \alpha \neq \beta)$
17. $int(\alpha, \beta) \vee \neg int(\alpha, \beta)$
18. $p \notin \alpha \wedge A \in p \wedge A \in \alpha \Rightarrow int(p, \alpha)$
19. $int(p, \alpha) \Rightarrow (\exists A : point)(A \in p \wedge A \in \alpha \wedge p \notin \alpha)$
20. $int(p, \alpha) \vee \neg int(p, \alpha)$
21. $p \in \alpha \wedge A \in p \Rightarrow A \in \alpha$
22. $A \in p \wedge A \notin \alpha \Rightarrow p \notin \alpha$

A.2 Theorems

1. $A \neq B \wedge col(A, B, C) \wedge col(A, B, D) \Rightarrow col(A, C, D)$
2. $col(A, B, C) \vee \neg col(A, B, C)$
3. $A \neq C \wedge A \in p \wedge B \notin p \wedge C \in p \Rightarrow \neg col(A, B, C)$
4. $A \neq C \wedge A \in p \wedge C \in p \wedge \neg col(A, B, C) \Rightarrow B \notin p$
5. $p \neq q \wedge \neg int(p, q) \wedge A \in p \Rightarrow A \notin q$
6. $\alpha \neq \beta \wedge \neg int(\alpha, \beta) \wedge A \in \alpha \Rightarrow A \notin \beta$
7. $p \notin \alpha \wedge A \in p \Rightarrow A \notin \alpha$
8. $\neg col(A, B, C) \Rightarrow (\exists \alpha : plane)(A \in \alpha \wedge B \in \alpha \wedge C \in \alpha)$
9. $cop(A, B, C, D) \wedge \neg col(A, B, C) \wedge A \in \alpha \wedge B \in \alpha \wedge C \in \alpha \Rightarrow D \in \alpha$
10. $p \in \alpha \vee p \notin \alpha$
11. $\neg col(A, B, C) \Rightarrow A \neq B \wedge A \neq C \wedge B \neq C$
12. $(\exists A : point)(\exists B : point)A \neq B$
13. $col(A, A, A)$
14. $(\exists A : point)A \notin p$
15. $p \in \alpha \wedge q \in \alpha \Rightarrow (\exists A : point)(A \in p \wedge A \in q) \vee \neg int(p, q)$
16. $p \neq q \wedge A \in p \wedge A \in q \wedge B \in p \wedge B \in q \Rightarrow A = B$
17. $\neg int(\alpha, \beta) \vee ((\exists p : line)p \in \alpha \wedge p \in \beta)$
18. $\alpha \neq \beta \wedge p \in \alpha \wedge p \in \beta \wedge A \in \alpha \wedge A \in \beta \Rightarrow A \in p$
19. $p \notin \alpha \Rightarrow (\exists A : point)(A \in p \wedge A \in \alpha) \vee \neg int(p, \alpha)$
20. $p \notin \alpha \wedge A \in p \wedge A \in \alpha \wedge B \in p \wedge B \in \alpha \Rightarrow A = B$
21. $A \notin p \Rightarrow (\exists \alpha : plane)(A \in \alpha \wedge p \in \alpha)$
22. $A \notin p \wedge p \in \alpha \wedge A \in \alpha \wedge p \in \beta \wedge A \in \beta \Rightarrow \alpha = \beta$
23. $p \neq q \wedge A \in p \wedge A \in q \Rightarrow (\exists \alpha : plane)(p \in \alpha \wedge q \in \alpha)$
24. $p \neq q \wedge A \in p \wedge A \in q \wedge p \in \alpha \wedge q \in \alpha \wedge p \in \beta \wedge q \in \beta \Rightarrow \alpha = \beta$