Annals of Mathematics and Artificial Intelligence
https://doi.org/10.1007/s10472-018-9597-7

From informal to formal proofs in Euclidean geometry

Sana Stojanović-Đurđević[1]

Published online: 29 August 2018

Abstract
In this paper, we propose a new approach for automated verification of informal proofs in Euclidean geometry using a fragment of first-order logic called coherent logic and a corresponding proof representation. We use a TPTP inspired language to write a semi-formal proof of a theorem, that fairly accurately depicts a proof that can be found in mathematical textbooks. The semi-formal proof is verified by generating more detailed proof objects expressed in the coherent logic vernacular. Those proof objects can be easily transformed to Isabelle and Coq proof objects, and also in natural language proofs written in English and Serbian. This approach is tested on two sets of theorem proofs using classical axiomatic system for Euclidean geometry created by David Hilbert, and a modern axiomatic system *E* created by Jeremy Avigad, Edward Dean, and John Mumma.

Keywords Informal proofs · Coherent logic · Euclidean geometry · Interactive theorem proving · Automated theorem proving

Mathematics Subject Classification (2010) 03B35 · 68T15

1 Introduction

In recent years, we have become aware that writing proofs using pen and paper alone is not always reliable, and writing proofs with the help of a computer has gained in importance. Formalization of important mathematical textbooks is an ongoing work, and there is a constantly growing repository of important theorems and parts of many theories that are proven using interactive theorem provers [10, 11, 18, 19]. Nevertheless, theorem proving with the help of a computer is still not widely used by mathematicians, nor by high school and university students. In that community, the vast majority of proofs are pen-and-paper proofs and high school and university textbooks rely on them exclusively.

There are noticeable differences between proofs of theorems written with interactive theorem provers and pen-and-paper proofs. Formal theorem proving increases our confidence

The author is partly supported by the grant ON174021 of the Ministry of Science of Serbia.

✉ Sana Stojanović-Đurđević
sana@matf.bg.ac.rs

[1] Faculty of Mathematics, University of Belgrade, Studentski trg 16, 11000 Belgrade, Serbia

in the correctness of a proof, while proofs in mathematical textbooks intuitively explain to the reader why a certain theorem holds. Or, like Marc Bezem and Dimitri Hendriks noted: *a proof explains why the theorem is true, and a formal proof does so in great detail* [5].

Learning the language of an interactive theorem prover is not trivial and, despite recent efforts [23], proof scripts still cannot be shared between different interactive theorem provers. Furthermore, formalization of a mathematical textbook is a challenging process since verifying the proofs using interactive theorem provers is often far from trivial. The main reason for this is that textbook proofs are often imprecise and can have a lot of missing steps. On the other hand, those proofs have been relied on for several hundred years, they are essential for everyday mathematical practice, and in most cases should not be changed. Filling in the missing steps, which formal theorem proving requires, would make those proofs too cumbersome for an average reader.

In this paper we offer a different way to formulate and verify informal proofs for some geometry theorems in coherent logic (a fragment of first order logic called geometric logic). Instead of learning the language of a specific interactive theorem prover and filling in the missing parts of a proof,[1] an informal proof will be translated into a semi-formal proof written in a *TPTP-like* [52] language. The language of semi-formal proofs is simple and fairly accurately depicts the language of informal mathematics used in geometry textbooks. The set of automated theorem provers will be used to generate the XML proof scripts of the coherent logic vernacular [47]. Those proof scripts can be translated, using proper XSLT style-sheets, into formal proofs verifiable in interactive theorem provers Isabelle and Coq, and also readable, natural language proofs in English and Serbian.

We will test this approach using two axiomatic systems for Euclidean geometry. First we will use the newly created axiomatic system *E* and several proofs of Euclid's postulates, and then we will use a classical Hilbert's axiomatic system with the set of theorem proofs found in Serbian high school geometry textbooks.

The experiments we conducted showed that our approach can be useful for automated verification of the semi-formal proofs that resemble geometry textbook proofs. The whole package that includes the program, both axiomatic systems, the sets of theorem proofs, and the XSLT files used for translation of XML proof scripts to the target languages can be found on-line.[2] The approach is described and tested on geometry theorems, but is not limited to geometry and can be used with any coherent logic theory. Automated verification of a semi-formal proof of a theorem is carried out using a previously defined set of axioms and definitions expressed in coherent logic. The semi-formal proof and the axiomatic system that is used for verification are specified by the user and are included through external files (they are not hard-coded into the system). This paper extends our previous short paper that describes the use of the basic version of the system [46, 50] (sadly in Serbian only).

This paper is organized as follows: the *Background* section will give an overview of tools used in this paper. The *Formal rendering of an informal proof* section will describe the language used to write semi-formal proofs and the necessary translations that were made. The *Approach for automated verification of semi-formal proofs* section will give detailed explanation of the proposed approach. The approach is applied on two different sets of theorems, and observations made during those verifications are presented in the following two sections. In the *Related work* section we discuss similar approaches, and in the *Conclusions*

[1] Various interactive theorem provers have tools and tactics with different levels of automation that can be used for this purpose [9, 14].

[2] http://argo.matf.bg.ac.rs/?content=argochecker

and future work section we draw final conclusions and present some ideas for the future work.

2 Background

In this section, we give a brief overview of computer-assisted theorem proving and the coherent logic vernacular that is used for generating formal proof scripts.

2.1 Computer-assisted theorem proving

Over the last few decades, many powerful programs for interactive and automated theorem proving have been developed, and today, computer-assisted theorem proving is used more than ever.

Interactive theorem provers are programs created for computer aided verification of proofs of theorems with respect to the given underlying logic. The proofs are mostly written by the user with certain amount of automation, but even so, the process of writing a proof generally requires considerable time. Also, reuse of the existing proofs, in case of small changes in a theory, can be difficult. The most popular interactive theorem provers nowadays are HOL-Light [20], Isabelle [37], Coq [53], and Mizar [56].

Programs for automated theorem proving are extremely efficient and capable of working with a large number of formulas. They are mostly used to prove or disprove extremely difficult conjectures. The most prominent step in automated theorem proving was the proof of the Robbins conjecture by William McCune [29] (in the 1996). Significant results were achieved in the field of equational algebra with the assistance of automated theorem provers. They were used to solve some open problems in quasigroup and loop theory [28, 39]. Nevertheless, automated theorem provers still have some shortcomings. The proof objects are usually somewhat neglected, they are not generated in a standardized form, and reusing the proof is generally not possible. Automated theorem provers can have bugs, and therefore do not provide a high degree of certainty in the correctness of the results. Still, they can be used for filtering large sets of axioms and theorems of the theory [9, 31, 32]. Over the last few years, automated theorem provers have been used more frequently in combination with interactive theorem provers. In certain cases this collaboration discovered new mathematical principles. Experiments with MML[3] (Mizar Mathematical Library) using an AI/ATP system [24] produced shorter proofs for some theorems. This was due to symmetry between concepts used in the previously proven theorems that were detected by the AI/ATP system, which was not fully utilized by the human author.

Another direction in the development of theorem provers was motivated by the need to generate human-friendly, textbook-like, readable proofs. In some areas of computer-assisted theorem proving, obtaining readable proofs is often not a priority. That said, readable proofs play an essential role in everyday mathematical practice as they help humans gain more insight in the underlying mathematical theory. The importance of readable proofs is recognized through the development of programs for both automated and interactive theorem proving. New automated theorem provers with human style output [17, 36, 48] have been introduced, and there are increasing efforts to make interactive theorem provers even more human-friendly [55].

[3] http://www.mizar.org/

2.2 Coherent logic vernacular

Coherent logic (CL) is a fragment of the first order logic suitable for automated theorem proving and generation of standardized readable proofs [4]. Initially, CL was used by Skolem and recently is used by many different authors for expressing and formalizing significant parts of mathematics, especially geometry [2, 5, 16, 51]. There exists a translation of linear complexity from first order logic to CL that preserves logical equivalence (and does not involve Skolemization) [4, 40]. CL consists of the implicitly universally quantified formulas of the following form [5]:

$$A_1 \wedge \ldots \wedge A_n \Rightarrow \exists \mathbf{x_1} B_1 \vee \ldots \vee \exists \mathbf{x_m} B_m \tag{1}$$

where $n \geq 0$, $m \geq 0$, and each vector $\mathbf{x_j}$ denotes a sequence of variables $x_1, x_2, \ldots, x_{k_j}$ ($k_j \geq 0$), A_i (for $1 \leq i \leq n$) are first-order atoms, and B_j (for $1 \leq j \leq m$) are conjunctions of first-order atoms $C_1 \wedge \ldots \wedge C_{l_j}$ ($l_j \geq 0$). If $m > 1$, a coherent logic formula is called a disjunctive formula. Existential quantification is allowed in the conclusion of the formula, so CL can be viewed as an extension of the resolution logic. The conjecture is being proved directly and is left unchanged, and there is no need for Skolemization. The proofs in coherent logic are intuitive and the reasoning is constructive.

For the sake of simplicity, we will use only function symbols of arity zero (i.e. constants). A witness is a new constant, not appearing in the axioms used nor in the conjecture that is 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. A closed atomic formula over constants is called a fact. CL deals with the sets of facts — ground atomic expressions.

CL does not involve negation. Negation of a single atom $\neg A$ can be represented in the form $A \Rightarrow \perp$. In order to handle negation in general, additional predicate symbols will be introduced to abbreviate subformulae. Moreover, for every predicate symbol R (that appears in the form $\neg R$) a new symbol $\overline{R}$ is introduced. Symbol $\overline{R}$ will stand for $\neg R$, and the following axioms are postulated [40]:

$$\forall \mathbf{x}(R(\mathbf{x}) \wedge \overline{R}(\mathbf{x}) \Rightarrow \perp), \forall \mathbf{x}(R(\mathbf{x}) \vee \overline{R}(\mathbf{x})) \tag{2}$$

Usually coherent logic is used as a one-sorted theory. In this paper we will focus on geometry, and since geometry is a many-sorted theory, we will use the usual reduction of a many-sorted logic to a one-sorted logic that introduces new predicate symbols for every sort (*point*, *line*, etc.).

Coherent logic vernacular (CLV) is an XML-based format used for proof representation in coherent logic [47]. The vernacular is simple and expressive and can be translated to different formal and natural languages. This feature enables us to share the same mathematical knowledge between different proof assistants. Several XSLT files are used to transform the proofs to desired formats. At the moment there are four XSL transformations that are used to generate formal proofs in Isabelle and Coq languages (*VernacularISAR.xsl*, *VernacularCoqTactics.xsl*), and readable proofs in English and Serbian formatted in LaTeX (*VernacularTex.xsl, VernacularSrpskiTex.xsl*). The constructed XSLT style-sheets are relatively easy to build and maintain. Each of the files is between 500 and 1000 lines long. A DTD (*Document Type Definition*) file *Vernacular.dtd* is used to describe the structure of the proofs. Parts of the file are listed below, describing the notion of a theory, a theorem, and a proof.

```
...
<!--******** Theory **************-->
<!ELEMENT theory (theory_name, signature, axiom*) >
<!ELEMENT theory_name (#PCDATA)>
<!ELEMENT signature (type*, relation_symbol*, constant*) >
<!ELEMENT relation_symbol (type*)>
<!ATTLIST relation_symbol name CDATA #REQUIRED>
<!ELEMENT type (#PCDATA)>
<!ELEMENT axiom (cl_formula)>
<!ATTLIST axiom name CDATA #REQUIRED>
...
...
...
<!--******** Theorem **************-->
<!ELEMENT theorem (theorem_name, cl_formula, proof+)>
<!ELEMENT theorem_name (#PCDATA)>
<!ELEMENT conjecture (name, cl_formula)>
<!--******** Proof **************-->
<!ELEMENT proof (proof_step*, proof_closing, proof_name?)>
<!ELEMENT proof_name EMPTY>
<!ATTLIST proof_name name CDATA #REQUIRED>
<!--******** Proof steps **************-->
<!ELEMENT proof_step (indentation,modus_ponens)>
<!ELEMENT proof_closing (indentation, (case_split|efq|from),
(goal_reached_contradiction|goal_reached_thesis))>
...
```

The process of transforming generated XML proofs to formal and natural language proofs must be done with care, but is fairly straightforward. The proof steps use only the following rules (a modification of rules from [5]): *modus ponens*, *case splits*, *assumption*, and *ex falso quodlibet*. Each of the proof steps is easily translated into the target languages. For readability reasons, we prefer to use native negation as opposed to the CL defined negation predicate symbols. We use the specific *layout.xml* file to define a natural language scheme that is used in a natural language proof. At the moment, the XSLT style-sheets translate all the steps of the generated XML proof scripts. Sometimes, this can result in proofs that are too detailed. We plan to simplify such proofs in the future by potentially ignoring the irrelevant proof steps.

Coherent logic prover ArgoCLP [48] is a generic theorem prover that can be used with an arbitrary coherent logic theory. It reads problems given in the TPTP form, and exports the proof objects in the XML format of the coherent logic vernacular. All axioms and the conjecture being proved must be in the coherent logic form. For each conjecture, the ArgoCLP prover generates three individual XML files: *theory*, *proof*, and *theorem*. The *theory* file contains the signature and the axioms that are used. If the ArgoCLP prover is successful, the *proof* file contains the theorem and the generated proof. If it is unsuccessful it contains just the conjecture. An additional XML file named *frontpage.xml* is used to store some basic information (name of the theory, author's names, prover that is used). The *theorem* file is used as an outline for other files. The details of the *theorem* file are displayed below. The *xsl-stylesheet* processing instruction is used to specify which of the XSLT style-sheets is used for transformation and it is recommended to use separate directories for translation to different target languages. The *chapter* tag is used for grouping of files if needed. Such file structure enables easy integration of several theorem proofs (that use the same *frontpage* and *theory* files) into a single larger file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE main SYSTEM "Vernacular.dtd">
<?xml-stylesheet href="VernacularISAR.xsl" type="text/xsl"?>

<main>
<xi:include href="frontpage.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>
<xi:include href="theory_th_11_01.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>

<chapter name="th_11_01">
<xi:include href="proof_th_11_01.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>
</chapter>
</main>
```

Generated files first must be validated, using the following command:

```
xmllint --dtdvalid Vernacular.dtd --noout --xinclude
th_11_complete_proof.xml
```

and then translated to the target language (in this case, Isabelle):[4]

```
xsltproc --xinclude th_11_complete_proof.xml >
th_11_complete_proof.thy
```

3 Formal rendering of an informal proof

In this section we describe the language of the semi-formal proof and the translation from the textbook proof to the semi-formal proof that can be verified within our approach. The length of the semi-formal proofs should not be much larger than the length of the proofs found in mathematical textbooks. We want to preserve the explanatory value of the proof. The semi-formal proof can have some details missing, with enough information to get automatically generated complete formal proofs from it (with all the missing details filled in).

Theorems and axiomatic systems Automated verification of the semi-formal proofs is possible only for theorems that belong to coherent logic, i.e. formulae of the following form:

$$A_1 \wedge \ldots \wedge A_n \Rightarrow \exists \mathbf{x_1} B_1 \vee \ldots \vee \exists \mathbf{x_m} B_m$$

The axiomatic system that is used for verification must be in the coherent logic form as well. A significant number of geometry theorems and axioms belong to coherent logic and those that do not belong can, in most cases, be easily translated into coherent logic. There is a general translation procedure of formulas of first order logic to coherent logic [40], but here we use the translations specific to Euclidean geometry [2, 51]. Translations to coherent logic will be performed manually, while formulating the semi-formal proof.

[4] Translated LaTeX file uses a custom *argoclp.sty* file to display the proper indentation of proof steps.

Translation from textbook sentences to formulae Textbook proofs often use the language of everyday vernacular, and not just the mathematical vernacular. In general, it is not easy to interpret a textbook proof. Also, the level of detail in different textbook proofs can significantly vary. In our setting, we choose to extract only the relevant information from the proof. Every time a context is changed in the proof, or a new relevant step is introduced, we will also introduce a new proof step (for example, the existence of the intersection point of two lines, or the conclusion that certain points are collinear). All additional information about the axioms and definitions that are used will be ignored.

At the moment, our language allows only direct linear proofs. Case distinctions are ignored, only their conclusions are included. Also, we ignore reasoning by contradiction and we replace it by just stating the positive conclusion. In both cases, the sub-proof will not be a part of the semi-formal proof. However, the conclusion is kept and automated theorem provers are used to discover the missing case distinctions. This means that automated theorem provers will have to fill in some parts of the proof. To accomplish this task, they use the complete set of definitions and axioms which includes the disjunctive formulas (axioms with the disjunctive conclusions) and the rule of excluded middle (axioms of the form (2) are postulated for all the predicate symbols).[5]

The language of the semi-formal proof The language of the semi-formal proof is based on the inference rules of coherent logic. In general, proof steps of the semi-formal proof can be described as a subset of the first order formula with equality. Each proof step can be either a left hand side or a right hand side of a coherent logic formula. The semi-formal proof is a finite sequence of formal statements. At the moment, only linear semi-formal proofs are allowed. In the future, we plan to enable reasoning by cases.

The first step of the semi-formal proof is assuming the premiss of the conjecture to be proved, i.e. a conjunction of first-order atoms (left hand side of a CL formula, without the quantification[6]).

Subsequent step[s] is a possibly existentially quantified conjunction of first-order atoms (right hand side of a CL formula for $m = 1$).

The last step of the semi-formal proof is the goal of the informal proof, i.e. a disjunction of a possibly existentially quantified conjunctions (right hand side of a CL formula for $m \geq 1$).

The conclusion of the conjecture that is proved will not always be the last step of the semi-formal proof. If the conjecture is existentially quantified, the proof will show the existence of some proof objects with certain properties. In such cases it can happen that the objects are constructed earlier in the proof, while the rest of the proof will show that the objects satisfy the required properties.

Sometimes the proof of a theorem is not available. In that case, the semi-formal proof will only have two formulas, the premiss of the conjecture and the conclusion of the conjecture to be proved. If a proof of a theorem is available and the system can not verify the complete proof, the user can try to split subsequent steps (a conjunctions of atomic formulae) into smaller steps and run the system again.

[5] See for example the proof of *Auxiliary 1* in the Section 5.

[6] Universal quantification in the coherent logic formulae is implicit, but has the scope of the whole formula, and not just the left hand side of the formula.

Table 1 Examples of the semi-formal proof steps

Assume A and B are two distinct points. `assume [A,B] : (point(A) & point(B) & A!=B)`
Show that points A, B, C are colinear. `have (col(A,B,C))`
Construct a plane R such that the point A and the line P lie on the plane R. `let [R] : (plane(R) & inc_po_pl(A,R) & inc_l_pl(P,R))`
Show that points A, B, and C lie on the plane R. `have (inc_po_pl(A,R) & inc_po_pl(B,R) & inc_po_pl(C,R))`
Construct two different points A and B such that A and B lie on the line L. `let [A,B] : (point(A) & point(B) & A!=B & inc_po_l(A,L) & inc_po_l(B,L))`

The syntax The syntax used for representation of the semi-formal proof steps is simple and self explanatory, and selected so that it enables easy integration into the TPTP conforming formulas. The user works in a language similar to the language of automated theorem provers.

There are three different types of proof steps. The first step always starts with the word *"assume"* and it gives the context of the theorem that is proved. It can be read as: *Under the assumption of.* All subsequent steps can start with the word *"let"*, which introduces new objects into the proof and can be read as: *Construct object[s] with the following property*;[7] or with the word *"have"* which introduces new relations over the existing objects and can be read as: *Show that the following relations hold.*[8]

While formulating the semi-formal proof, the user can choose his own signature, but there are a few limitations due to the use of the coherent logic theorem prover ArgoCLP. Predicate symbols must be unique (whether or not they are used with different sorts of objects) and predicate symbols of the positive form can not start with a letter n. For example, relation of incidence will be represented with several predicate symbols for various sorts: inc_po_l for points and lines, inc_po_pl for points and planes, and inc_l_pl for lines and planes. Since coherent logic does not use negation, negative form of a predicate symbol is introduced as described in the Section 2, and labeled with the letter n preceding a predicate symbol for the positive form. For example, if the predicate symbol used for collinearity is col, then the predicate symbol used for non-collinearity will be $ncol$.

Some semi-formal proof steps expressed in English, using those specific predicates, are given in Table 1. Looking at the mathematical textbooks (especially geometry textbooks), one can notice a lot of statements that can be represented by our semi-formal proof steps.

In the text below we will point out the most important translations made during the formulation of the semi-formal proofs (translations were made on some conjectures, as well as on certain proof steps).

[7] The sort of the constructed objects should be stated at the beginning of the formula.

[8] Different semantically equivalent sentences can be used as well.

Unique objects A theorem that states that *there exists one and only one point with certain property* is split into two theorems: the first one that states that *such point exists*, and the second one that states that *if there are two points with that property, those two points are the same*. The two theorems are equivalent to the original one.

While formulating the second semi-formal proof, a large part of the first semi-formal proof is used. This is the same principle used in textbooks, so we can say that there is one-to-one correlation with proofs from the book.

Nondegeneracy assumptions In Euclidean geometry a nondegenerate configuration of objects in the conjecture being proved is often assumed. In the semi-formal proof of a theorem, that is used with our approach, the nondegeneracy assumptions will have to be made explicit.

Also, some proof steps require *the construction of new objects* or, more precisely, giving name to an existing object. In high-school geometry, and also in the Elements, lines, planes, and circles (in the Elements) are named by listing two points that lie on the line, by three points that lie on the plane, and by three points that lie on the circle. Every step with such naming will produce few new steps in our semi-formal proof.

Those are the sentences of the following form: *there exists a point D that does not belong to the line AB*. Before formulating the corresponding semi-formal proof step we need to construct a new object of the *line* sort determined by the two distinct points *A* and *B* that will be used instead of *the line AB*, with the assumption that the points *A* and *B* are different. In the textbook proofs, those assumptions are sometimes left implicit. But, in the semi-formal proof this type of a proof step is necessary. The sentence will be translated into the following text: *"For different points A and B, let L be the line determined by the points A and B, let D be a point that does not belong to the line L"*, and represented by the following semi-formal proof steps:

```
have (A!=B)
let [L] : (line(L) & inc_po_l(A,L) & inc_po_l(B,L))
let [D] : (point(D) & ninc_po_l(D,L))
```

Reasoning by cases Looking at the proofs from geometry textbooks, one can notice that reasoning by cases is often omitted, without loss of generality, due to the similarity between different cases. In formal theorem proving, this is not an option. Still, in geometry reasoning by cases is rarely used extensively, and at this moment, our approach does not support proof steps that use reasoning by cases. But, if reasoning by cases is explicitly used in a proof, such proofs can be linearized. The user can skip those proof steps and try to verify the remaining parts of the proof. If the proof can not be verified without the steps that explicitly use reasoning by cases, then the proof will have clear sub-proofs for each case and it will be reasonable to introduce additional weaker theorems based on those cases (and formulate corresponding semi-formal proofs).

All this shows that translation from the textbook proof of a theorem to the semi-formal proof must be done with care. Even though semi-formal proofs that we can verify are not identical to the proofs found in mathematical textbooks, the necessary translations are minor, and they often present the inevitable first step in a transformation from the textbook proof to the formal proof.

4 Approach for automated verification of semi-formal proofs

In this section we describe the approach for automated verification of the semi-formal proofs. The semi-formal proof and the set of axioms and definitions used for verification are provided through external input files. The definitions will have the same status as the axioms. The file with axioms and definitions must be given in the TPTP format in the coherent logic form. The semi-formal proof must be given in the format described in the previous section.

The system is using publicly available automated theorem provers. It also uses additional files of the coherent logic vernacular to validate the generated XML files and to translate them to target languages. The whole process is completely automated. The system can be downloaded from the web-page[9] and accessed from the command line.

The approach uses the framework for automated theorem proving and the features of the coherent logic vernacular. The framework for automated theorem proving is using several automated theorem provers. The resolution theorem provers Vampire [41] and E [43] are used for their efficiency, as preprocessors for filtering a large set of axioms and definitions. The coherent logic theorem prover ArgoCLP [48] is used to generate a readable proof script in the coherent logic vernacular. The generated proof scripts can be translated, using simple XSLT style-sheets, into languages of different interactive theorem provers (Isabelle and Coq) and different natural languages (English and Serbian). Several Python and shell scripts are used for automated manipulation with input and output (generated) files.

The steps of the semi-formal proof will be verified individually. Some steps will be trivial, like the application of a single axiom, but in some cases one step can hide an auxiliary lemma whose proof contains a lot of missing facts that need to be derived so that the whole proof can be verified.

The verification of a semi-formal proof, given the selected set of axioms and definitions, can be described as follows:

1. Coherent logic auxiliary lemmas are formulated for each step of the semi-formal proof:
   - The premiss of the current lemma is the conjunction of the previous steps of the semi-formal proof, and the conclusion of the lemma is the current step.
   - A theory that will be proved is formed. The theory consists of the set of all axioms and definitions that are used, and the set of auxiliary lemmas.
   - For each auxiliary lemma, a TPTP file will be created that is used with automated theorem provers. The TPTP file will consist of the given set of axioms and definitions, and the current lemma as the goal.
2. Several theorem provers will try to prove all auxiliary lemmas and generate proof objects in the XML form of the coherent logic vernacular:[10]
   - Resolution theorem provers Vampire and E will be used to prove the lemma using a generated TPTP file both in the original and the reverse order of lines (the order of premisses often impacts the proving process). The list of used axioms, returned by the resolution provers, will be used again until the smallest list is identified.
   - Coherent logic prover ArgoCLP will be invoked with the acquired list of axioms and the proof in the XML form of the coherent logic vernacular will be generated.

[9] http://argo.matf.bg.ac.rs/?content=argochecker

[10] Proving an auxiliary lemma is performed using the framework for theorem proving used in a larger formalization project [51].

3. Generated proofs of all lemmas will be combined into one XML document:
   - Coherent logic prover ArgoCLP generates three XML files for each auxiliary lemma: *theorem*, *proof*, and *theory* file (as described in Section 2).
   - All generated proof scripts must be validated against the rules of the CLV. For each step of the semi-formal proof, i.e. for each auxiliary lemma, we check if the generated *theorem* file meets the given syntactical restrictions (using the DTD file *Vernacular.dtd*).
   - Formal and natural language files for individual lemmas are generated using the XSLT style-sheets.[11]
   - The generated *proof* files for all lemmas are merged into a new XML file using the *chapter* tag (like the one presented below).
4. The generated document will be validated against the *Vernacular.dtd* file, and translated into formal languages of Isabelle and Coq, and English and Serbian natural languages (by using the appropriate XSLT style-sheets).

The following code outlines the proof steps of the semi-formal proof presented at the end of this section (named *th_11_complete_proof.xml* and it is available on-line).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE main SYSTEM "Vernacular.dtd">
<?xml-stylesheet href="VernacularISAR.xsl" type="text/xsl"?>

<main>
<xi:include href="frontpage.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>
<xi:include href="theory_trivial.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>

<chapter name="Hilbert">
<xi:include href="proof_th_11_01.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>
<xi:include href="proof_th_11_02.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>
</chapter>
</main>
```

The proofs made by resolution theorem provers (in step 2) are hard to read, and the interpretation and verification of such proofs is especially difficult. It is possible to directly use proof by contradiction from the resolution theorem provers, but this is a considerably challenging task [7]. Instead, we are using the ArgoCLP prover and the CLV translations for generating readable and formal proofs, and those translations are straightforward and easy.

The architecture of the system is displayed in Fig. 1. The whole process of automated proving, validation of XML files, and proof translation to the target languages is fully automated, without any user interaction. The described system is implemented in C++ and Python. Formulation of auxiliary lemmas, validation of XML files, and final integration of XML documents into one file are implemented in Python (and several shell scripts), while the framework for automated verification of each lemma is implemented in C++. Since automated theorem provers cannot guarantee that they will prove a conjecture, a time limit

[11] This step is not necessary, but it can be useful for examining the proof of individual steps. For example, if the original proof is too long and we are interested in inspecting only certain proof steps.

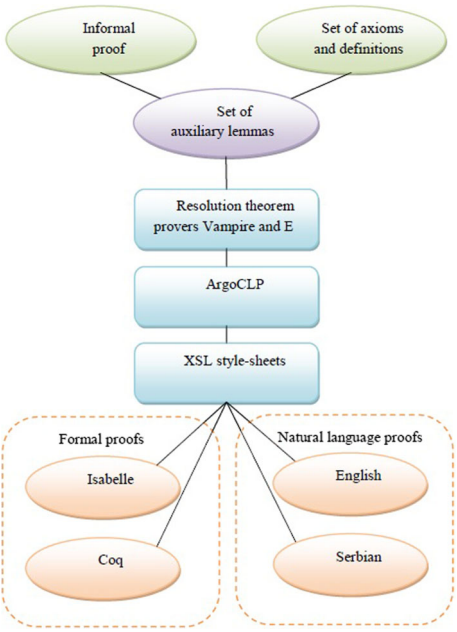

Fig. 1 The architecture of the system

must be set. Even if the system cannot prove some steps of the semi-formal proof, a formal document will still be generated but proofs for those steps will be missing (labeled with *sorry* in Isabelle, and with *Admitted* in Coq). In that case, the user can fill out the missing part of the formal proof, or he can formulate a new semi-formal proof and then try again.

Natural language proofs One of the advantages of using the coherent logic vernacular is the option of specifying the layout for certain predicate symbols. This is done by altering the specific layout file `layout.xml` [47] directly and it is commonly used for translation to natural language proofs. For example, a defined layout for `cong(A,B,C,D)` can be $(A, B) \cong (C, D)$, and defined layout for `inc_po_1(A,L)` can be $A \in l$. If the user does not want to change the layout, predicate names will be taken from the input files.

Examples The semi-formal proofs, the axiomatic system, and the automatically generated formal documents in Isabelle and Coq, as well as the natural language proofs in English and Serbian are available on-line,[12] both for Hilbert's axiomatic system, and for the axiomatic system E. The system was tested on a server with 48 AMD Opteron(tm) Processor 6168. The time limit for verification of individual proof steps is set to 60 seconds for all automated theorem provers.

Here we will list one theorem, its textbook proof (this is an actual text taken from the textbook [13]), a CL semi-formal proof, an automatically generated Isabelle proof, and a natural language proof in English. The total time for verification and generation of final proofs is 72 seconds.

[12] http://argo.matf.bg.ac.rs/?content=argochecker

Theorem 11: *For a line p and a point A that does not belong to the line p, prove that all lines that contain the point A and intersect the line p lie on a single plane.*

Textbook proof:
Instructions: Prove that all these lines belong to the plane determined by the point A and the line p.[13]

Even though this "proof" is more a hint than an actual proof, it can be used to formulate the CL semi-formal proof that can be verified with our approach. It is obvious that the introduction of a new object of the line sort is needed (the line q in the following statement), and that we can say that we are proving the following theorem:

Theorem 11': *For a line p and a point A that does not belong to the line p, and a line q that contains the point A and intersects the line p, prove that the line q lies on the plane determined by the point A and the line p.*

Now, even with this meager hint we can easily write the following semi-formal proof in our syntax. Numbers are not part of the proof, they are used to improve the readability. We have to stress that the construction of the plane in the step 1 is the necessary intermediate step in the proof.

```
0. assume [P,A,Q]:(line(P) & point(A) & ninc_po_l(A,P) & line(Q)
                  & inc_po_l(A,Q) & int_l_l(P,Q))
1. let [R]:(plane(R) & inc_po_pl(A,R) & inc_l_pl(P,R))
2. have (inc_l_pl(Q,R))
```

The automatically generated natural language proof (in LaTeX) is listed below. We decided to improve the readability of the proofs by using the dominant naming scheme. Points are denoted by capital Latin letters, lines are denoted by small Latin letters, and planes are denoted by capital Greek letters.[14] Using the *layout.xml* file we denoted the layout for *inc_po_l(A,P)*, *inc_po_pl(A,R)*, *inc_l_pl(P,R)* to be $A \in p$, $A \in \alpha$, $p \in \alpha$ (in that order); and layout for *int_l_l(P,Q)* to be *"lines p and q intersect"*.

Theorem 1 (**th_11_01**) *Assuming that $A \notin p$ and $A \in q$ and lines p and q intersect there exist plane α, such that $A \in \alpha$ and $p \in \alpha$.*

Proof

1. There exist a point B and a point C where $B \neq C$ and $B \in p$ and $C \in p$ (using ax_I3a).
2. From the facts $B \neq C$ and $B \in p$ and $C \in p$ and $A \notin p$ it holds that $\neg col(B, C, A)$ (using ax_D1a).
3. From the fact $\neg col(B, C, A)$ it holds that $\neg col(C, A, B)$ (using ax_sym_ncol1).
4. From the fact $\neg col(C, A, B)$ it holds that $\neg col(A, B, C)$ (using ax_sym_ncol1).
5. From the fact $\neg col(A, B, C)$ there exist a plane α, where $A \in \alpha$ and $B \in \alpha$ and $C \in \alpha$ (using ax_I4a).

[13] In Serbian: *Uputstvo: Dokazati da sve ove prave pripadaju ravni određenoj tačkom A i pravom p.*

[14] We wanted the user to be able to recognize the sort of an object immediately, at the first glance, just like when reading mathematical textbooks.

6. From the facts $B \neq C$ and $B \in p$ and $C \in p$ and $B \in \alpha$ and $C \in \alpha$ it holds that $p \in \alpha$ (using ax_I6).
7. The conclusion follows from the facts $A \in \alpha$ and $p \in \alpha$.

□

Theorem 2 (**th_11_02**) *Assuming that $A \notin p$ and $A \in q$ and lines p and q intersect and $A \in \alpha$ and $p \in \alpha$ it holds that $q \in \alpha$.*

Proof

1. From the fact lines p and q intersect there exist a point B where $p \neq q$ and $B \in p$ and $B \in q$ (using ax_D6).
2. From the facts $p \in \alpha$ and $B \in p$ it holds that $B \in \alpha$ (using ax_D11).
3. It holds that $A = B$ or $A \neq B$.
   4. Assume that: $A = B$.
      5. From the facts $B \in p$ and $A = B$ it holds that $A \in p$.
      6. From the facts $A \notin p$ and $A \in p$ we get contradiction.
   7. Assume that: $A \neq B$.
      8. From the facts $A \neq B$ and $A \in q$ and $B \in q$ and $A \in \alpha$ and $B \in \alpha$ it holds that $q \in \alpha$ (using ax_I6).
      9. The conclusion follows from the fact $q \in \alpha$.
10. The conjecture follows in all cases.

□

The automatically generated formal proof verifiable in the interactive theorem prover Isabelle is listed below:

```
lemma th_11_01:
assumes "¬ inc_po_l A p" and "inc_po_l A q" and "int_l_l p q"
shows "∃(alpha::plane).(inc_po_pl A alpha ∧ inc_l_pl p alpha)"
proof -

obtain B::point and C::point where "B ~= C" and "inc_po_l B p" and "inc_po_l C p"
using ax_I3a [of "p"] by auto
from `B ~= C` and `inc_po_l B p` and `inc_po_l C p` and `¬ inc_po_l A p` have "¬
col B C A" by (rule ax_D1a)
from `¬ col B C A` have "¬ col C A B" by (rule ax_sym_ncol1)
from `¬ col C A B` have "¬ col A B C" by (rule ax_sym_ncol1)
from `¬ col A B C` obtain alpha::plane where "inc_po_pl A alpha" and "inc_po_pl
B alpha" and "inc_po_pl C alpha" using ax_I4a [of "A" "B" "C"] by auto
from `B ~= C` and `inc_po_l B p` and `inc_po_l C p` and `inc_po_pl B alpha` and
`inc_po_pl C alpha` have "inc_l_pl p alpha" by (rule ax_I6)
from `inc_po_pl A alpha` and `inc_l_pl p alpha` have ?thesis by auto
from this show ?thesis .
qed

lemma th_11_02:
assumes "¬ inc_po_l A p" and "inc_po_l A q" and "int_l_l p q" and "inc_po_pl
A alpha" and "inc_l_pl p alpha"
shows "(inc_l_pl q alpha)"
proof -

from `int_l_l p q` obtain B::point where "p ~= q" and "inc_po_l B p" and "inc_po_l
B q" using ax_D6 [of "p" "q"] by auto
from `inc_l_pl p alpha` and `inc_po_l B p` have "inc_po_pl B alpha" by (rule
ax_D11)
```

```
have "A = B ∨ A ~= B" by (subst disj_commute, rule excluded_middle)
    show ?thesis
    proof(cases "A = B")
      case True
       from `inc_po_l B p` and `A = B` have "inc_po_l A p" by simp
       from `¬ inc_po_l A p` and `inc_po_l A p` have "False" by (rule notE)
       from this show ?thesis by (rule FalseE)
    next
      case False
       from `A ~= B` and `inc_po_l A q` and `inc_po_l B q` and `inc_po_pl A alpha`
        and `inc_po_pl B alpha` have "inc_l_pl q alpha" by (rule ax_I6)
       from `inc_l_pl q alpha` show ?thesis by assumption
    qed
qed
end
```

4.1 Comparison with the Sledgehammer tool

The approach presented in this paper has some similarities with the Sledgehammer approach [8, 9], which is implemented within the proof assistant Isabelle and works with resolution theorem provers E, SPASS [54], Vampire, and Z3 [34]. The list of axioms found by those provers is passed to the internal prover Metis that constructs a formal proof. The alternative is to use an algorithm that translates proofs generated by resolution theorem provers [6, 7]. The translated proofs are referred to as natural deduction proofs, extended with case analyses and nested subproofs, and they look like regular Isabelle proofs.

Our approach has a different agenda from Sledgehammer. Sledgehammer is a very powerful tool with several automated theorem provers and additional tools at its disposal. The Sledgehammer tool is more powerful than our system, but our approach is more suited for the beginners.

We wanted to make a simple system. The goal is to verify proofs which resemble those found in mathematical textbooks, without using the interactive theorem provers directly. We believe that our semi-formal proofs are more suitable for users without any experience with interactive theorem provers. While Sledgehammer can be used only by Isabelle users, our automatically generated proofs can be used by both Isabelle and Coq users, directly or as part of a larger formalization. Furthermore, unlike Sledgehammer, our approach offers natural language proofs.

5 From Euclid's proofs to formal proofs

Euclid's *Elements* is one of the iconic textbooks for elementary geometry. Euclid was the first that faithfully used axiomatic methods and derived many geometric properties relying only on logic rules [15]. Euclidean geometry was used undisputed for more than two thousand years, until the nineteenth century when the use of diagrams (i.e. diagrammatic reasoning) came to be viewed as flawed. This resulted in new axiomatisations made by Hilbert (1899) and later Tarski (1959). Those axiomatizations were trying to eliminate the use of diagrams and fill the gaps that were thought to be overlooked by Euclid. But in 2009 Jeremy Avigad, Edward Dean, and John Mumma developed a new axiomatic system *E* for Euclid's *Elements* [2]. They showed that diagrammatic reasoning in Euclid's proofs is actually controlled and guided by a distinct logic. They asserted a reliable method for diagrammatic reasoning using a simple set of relations, construction rules, and inference rules,

Table 2 Some basic relations in the formal language E

$on(a, L)$	point a is on line L
$sameside(a, b, L)$	point a and b are on the same side of line L
$between(a, b, c)$	points a, b, and c are distinct and collinear, and b is between a and c
$onc(a, \alpha)$	point a is on circle α
$inside(a, \alpha)$	point a is inside circle α
$center(a, \alpha)$	point a is the center of circle α
$intersects(L, M)$	lines L and M intersect
$intersects(L, \alpha)$	line L intersects circle α
$intersects(\alpha, \beta)$	circles α and β intersect
$cong(a, b, c, d)$	$\overline{ab} = \overline{cd}$

and showed that the proofs in their system faithfully represent the proofs from the *Elements*. All references to the *Elements* refer to the Heath translation [15].

Language of *E* There are six sorts of objects in the language of E: points, lines, circles, segments, angles, and areas. Small Latin letters are used for points, capital letters for lines, and Greek letters for circles. In addition to the equality symbol, there are some basic relations between elements of these sorts.[15] Language of E allows negation. Basic assertion or a literal is an atomic formula or a negated atomic formula. Assertions over the first three sorts are called "diagrammatic assertions", and assertions over the last three sorts are called "metric assertions". Relations over the first three sorts of objects are given in Table 2. Sorts of segments, angles, and areas are not used in this paper, but segments are represented by a pair of points and several additional predicates are introduced.

Theorems and proofs in *E* Theorems in E have the following logical form:

$$\forall \mathbf{a}, \mathbf{L}, \boldsymbol{\alpha}\ (\varphi(\mathbf{a}, \mathbf{L}, \boldsymbol{\alpha}) \Rightarrow \exists \mathbf{b}, \mathbf{M}, \boldsymbol{\beta}\ \psi(\mathbf{a}, \mathbf{b}, \mathbf{L}, \mathbf{M}, \boldsymbol{\alpha}, \boldsymbol{\beta})), \tag{3}$$

where φ is a conjunction of literals, ψ is either a conjunction of literals or the symbol $\perp$, and vectors $\mathbf{a}$, $\mathbf{L}$, and $\boldsymbol{\alpha}$ (and $\mathbf{b}$, $\mathbf{M}$, and $\boldsymbol{\beta}$) represent a sequence of points, lines, and circles. This form is a subset of the coherent logic form.

There are several propositions and their auxiliaries that are proved in the axiomatic system E. Words "Have" and "Hence", that appear in the proofs, are not used as part of the logical system, they are only used to improve the readability. "Have" is used to introduce new metric assertions that are inferred from the diagram, and "Hence" is used to introduce assertions that follow from previous metric assertions. The word "Let" is used to introduce new objects on the diagram. Also, comments in the brackets are optional and are not used as part of a proof. Abbreviations "Q.E.F." and "Q.E.D." are taken from the Elements and they mark the distinction between a proof that ends with a construction, and a proof that demonstrates certain properties.[16]

Original Euclid's formulation of Proposition 1 of Book I is: *On a given straight line, construct an equilateral triangle*. In this paper we will not use the proof from the Elements, but the following proof in the axiomatic system E.

[15] http://www.andrew.cmu.edu/user/avigad/Papers/euclid/e.smt

[16] Original phrases are "that which it was required to do" (abbreviated by "Q.E.F." for "quod erat faciendum" by Heath), and "that which it was required to prove" (abbreviated by "Q.E.D." for "quod erat demonstratum").

Proposition I.1 *Assume a and b are two distinct points. Construct point c such that $\overline{ab} = \overline{bc}$ and $\overline{bc} = \overline{ca}$.*

Proof

Let α be the circle with center a passing through b.
Let β be the circle with center b passing through a.
Let c be a point on the intersection of α and β.
Have $\overline{ab} = \overline{ac}$ [since they are radii of α].
Have $\overline{ba} = \overline{bc}$ [since they are radii of β].
Hence $\overline{ab} = \overline{bc}$ and $\overline{bc} = \overline{ca}$.

□

From this proof, we can formulate the following semi-formal proof (with only minor modifications due to the syntax of our system). All steps of the proof are automatically verified.

```
0. assume [A,B] : (point(A) & point(B) & A != B)
1. let [K1] : (circle(K1) & center(A,K1) & onc(B,K1))
2. let [K2] : (circle(K2) & center(B,K2) & onc(A,K2))
3. let [C] : (point(C) & onc(C,K1) & onc(C,K2)
           & intersectscc(K1,K2))
4. have (cong(A,B,A,C))
5. have (cong(B,A,B,C))
6. have (cong(A,B,B,C) & cong(B,C,C,A))
```

Avigad et al. noticed that the previous proposition is often used by Euclid with an assumption that the point c is distinct from a and b, and that c is not on the line determined by a and b. Those two statements are auxiliary lemmas that are easily stated and proved in the language of E. Here we list the proof of the first lemma in the axiomatic system E.

Auxiliary 1 *Assume a and b are two distinct points, $\overline{ab} = \overline{bc}$, and $\overline{bc} = \overline{ca}$. Then $c \neq a$ and $c \neq b$.*

Proof

Suppose $c = a$.
  Hence $a = b$.
  Contradiction.
Hence $c \neq a$.
Suppose $c = b$.
  Hence $a = b$.
  Contradiction.
Hence $c \neq b$.

□

The proofs of both auxiliary lemmas are using reasoning by cases, which is not supported by our language at the moment. Therefore, we will consider the first lemma as if there is no proof available (and let the system try to prove this conjecture on its own[17]).

[17] As we mentioned in the Section 3, automatically generated proofs can use case distinctions if disjunctive formulas are provided by the user (as part of the axiomatic system).

In the semi-formal proof of the second lemma we will add, as intermediate steps, the conclusions from the first lemma. Semi-formal proofs are listed below. Both lemmas are successfully automatically verified.

```
0. assume [A,B,C]:(point(A) & point(B) & A != B & point(C)
                  & cong(A,B,B,C) & cong(B,C,C,A))
1. have (C != A & C != B)
```

Auxiliary 2 *Assume a and b are two distinct points, a is on L, b is on L, and* $\overline{ab} = \overline{bc}$*, and* $\overline{bc} = \overline{ca}$*. Then c is not on L.*

```
0. assume [A,B,C,L] : (point(A) & point(B) & line(L) & A != B
                     & on(A,L) & on(B,L) & point(C)
                     & cong(A,B,B,C) & cong(B,C,C,A))
1. have (C != A)
2. have (C != B)
3. have (non(C,L))
```

These three proofs were automatically verified within our system. In the paper by Avigad et al. there is only one more theorem which covers the first three sorts with a proof formulated in the language of E[18] — Proposition I.2. The authors recognize that the proof of this proposition is surprisingly complicated considering that it is the second proof in the book. They identify a few missing steps in the proof, and they note: *"where Euclid carries out a complex construction without further justification, our system requires an explicit (but brief) argument, amidst the construction, to ensure that a certain point lies inside a certain circle"*. This divergence is due to the Euclid's implicit use of diagrammatic information in the proof. Our system did not manage to verify four steps of the semi-formal proof. Two auxiliary lemmas were not proven by the resolution theorem provers, and two lemmas were proven by the resolution provers but not by the coherent logic prover.[19]

There are several examples of proofs in the formal system E that the authors provided. The approach described in our paper managed to verify three of them completely and one of them partially. The language of E is closely related to the coherent logic and the language described in this paper. Proofs in the language of E look a lot like semi-formal proofs described in this paper. The language of E does not use functions and neither do we. Proofs in Euclid's Elements are constructive, mostly linear and, like authors noticed, Euclid only occasionally uses proofs by cases or proofs by contradiction. Theorems of E do not allow disjunctive conclusions, but disjunctive reasoning (case splits on atomic formula) in the proof is allowed.

The number of steps in each semi-formal proof, the number of verified steps, and the total verification time are given in Table 3. Although the approach did not verify all steps in the semi-formal proofs, it was successful in handling the basic diagrammatic inferences in Euclid's proofs.

[18] There are in total eight proofs formulated in the axiomatic system E, but the last four proofs are proofs over the angle sort.

[19] In the first step of the proof (in the axiomatic system E) previously proven Proposition I.1 is used, but it is too difficult for automated theorem provers, and we do not use previously proven theorems as part of the system at this point.

Table 3 Number of steps in the semi-formal proof, the number of verified steps, and the verification time

Theorem name	Semi-formal proof steps	Number of proven steps	Time (in seconds)
Proposition 1	6	6	8
Auxiliary 1	1	1	7
Auxiliary 2	3	3	105
Proposition 2	17	13	130

6 From high school geometry to formal proofs

Just like the proofs in Euclid's Elements, high school proofs sometimes rely on diagrams and intuition as authors often present only the crucial part of the proof. While the proofs in the formal axiomatic system E can be directly used in our system, the proofs from high school textbooks first must be translated into the semi-formal proofs. The axiomatic system that is used is Hilbert-like axiom system.

Hilbert's axiomatic system Hilbert published one of the most influential books on geometry, *Foundations of Geometry (der Grundlagen der Geometrie)* in 1899. His goal was to create a formal system: *"a complete and simple set of axioms and to deduce from them the most important geometric theorems"* [21]. There are eleven editions of Hilbert's *Grundlagen*, containing a lot of significant changes and upgrades over the original.[20]

Hilbert considers three different sets of elements: points, lines, and planes, and establishes an axiomatic system based on those elements and their mutual relations. There are five groups of axioms: *Axioms of Incidence*, *Axioms of Order*, *Axioms of Congruence*, *Axiom of Parallels*, and *Axioms of Continuity*. Parts of Hilbert's axiomatization use second order logic, but in this paper we use only the first two groups of Hilbert's axioms. They belong to first order logic and are easily expressed in coherent logic.

The first group of Hilbert's axioms describes the incidence relations between points, lines, and planes. The second group of axioms describes the ordering of points using the *between* concept. In most cases theorems that can be proved using those two groups of axioms belong to coherent logic as well, or can easily be translated. We test the approach on proofs of theorems found in geometry textbooks used in Serbian high schools [13, 33, 49] that are proved with the first two groups of Hilbert's axioms.[21] Those proofs are, for the most part, easily transformed into the semi-formal proofs that can be verified by our system.

In later editions of Hilbert's Grundlagen it is noted that "two points" are always distinct points so the addition of non-degeneracy assumptions, that we mentioned earlier, is valid. One of the necessary translations to coherent logic is transforming the sentence "there exists one and only one x such that $P(x)$ holds" into two sentences. In general (for complicated formula P) this sentence is not coherent, but in most cases in Euclidean geometry P is actually a conjunction of first order atoms, and the following equivalence can be used:

$$\exists! x\, P(x) \equiv \exists x\, P(x) \wedge (\forall x \forall y\, P(x) \wedge P(y) \rightarrow x = y)$$

[20] This produced some inconsistencies in cases when the axioms were changed, but the proofs of the theorems were not updated accordingly. Detailed discussion on Hilbert's axiomatic system is beyond the scope of this paper and will be presented in the separate paper.

[21] Hilbert's theorems from the first two groups belong to the set of theorems analysed in this paper.

Table 4 Relations in Hilbert's axiomatic system

$inc\_po\_l(A, L)$	point A is on line L
$inc\_po\_pl(A, P)$	point A is on plane P
$inc\_l\_pl(L, P)$	line L is on plane P
$int\_l\_l(L1, L2)$	lines $L1$ and $L2$ intersect
$int\_pl\_pl(P1, P2)$	planes $P1$ and $P2$ intersect
$int\_l\_pl(L, P)$	line L and plane P intersect
$col(A, B, C)$	points A, B, and C are collinear
$cop(A, B, C, D)$	points A, B, C, and D are coplanar
$bet(A, B, C)$	point B lies between points A and C

Predicate symbols that we used to represent relations of Hilbert's axiomatic system are given in Table 4.

Here we will illustrate a translation of one theorem proof taken from an early high school geometry textbook [33]. For the sake of readability, each step of the textbook proof is enumerated and presented in a new line (without the axioms and rules of inference that are used).

Theorem I1 *If a point A does not belong to a line p, there exists a unique plane that contains both the point A and the line p.*

Proof

1. *The line p contains at least two points B and C.*
2. *Points A, B, and C are non-collinear.*
   - 2.1. *Assume the opposite: If points A, B, C are collinear, they lie on the same line.*
   - 2.2. *Points B and C lie on p, so point A will then lie on the line p as well.*
   - 2.3. *This is a contradiction with the assumption of the theorem that point A does not lie on p.*
3. *There exists a plane α that contains them.*
4. *All points of the line p lie on that plane.*
5. *If a plane β contains the point A and the line p,*
6. *then it contains points A, B, and C*
7. *and it is identical to the plane α.*

□

The part of the proof that is using proof by contradiction is marked by indentation and will not be used as part of the semi-formal proof, as we explained in the Section 3. This theorem concludes that *there exists a unique plane*, so we formulate the following two theorems and their semi-formal proofs[22] (expressed using predicate symbols from Table 4):

Theorem 1 *If a point A does not belong to a line p then there exists a plane α such that the point A and the line p lie on that plane.*

[22] Both semi-formal proofs were verified (and formal and readable proofs generated) in 11 seconds.

```
0. assume [P,A] : (line(P) & point(A) & ninc_po_l(A,P))
1. let [B,C] : (point(B) & point(C) & B!=C & inc_po_l(B,P)
             & inc_po_l(C,P))
2. have (ncol(A,B,C))
3. let [R] : (plane(R) & inc_po_pl(A,R) & inc_po_pl(B,R)
           & inc_po_pl(C,R))
4. have (inc_l_pl(P,R) & inc_po_pl(A,R))
```

Theorem 2 *If a point A does not belong to a line p, and there exist two planes α and β such that the point A and the line p lie on both of them, then those two planes are equal.*

```
0. assume [P,A,R1,R2] : (line(P) & point(A) & plane(R1)
                        & plane(R2) & ninc_po_l(A,P)
                        & inc_po_pl(A,R1) & inc_l_pl(P,R1) &
                        inc_po_pl(A,R2) & inc_l_pl(P,R2))
1. let [B,C] : (point(B) & point(C) & B!=C & inc_po_l(B,P)
             & inc_po_l(C,P))
2. have (ncol(A,B,C))
3. have (inc_po_pl(A,R1) & inc_po_pl(B,R1) & inc_po_pl(C,R1) &
         inc_po_pl(A,R2)& inc_po_pl(B,R2) & inc_po_pl(C,R2))
4. have (R1 = R2)
```

The set of theorems found in Serbian high school geometry textbooks is particularly well suited for the approach described in this paper, as textbooks contain both easy and complex proofs. The common problem with mathematical textbooks is that they often jump from easy to difficult theorems too quickly. The level of detail varies in proofs for different theorems. Some theorems are even stated without a proof, or with brief instructions from which we can formulate only two or three semi-formal proof steps. That said, the experiments presented in this paper show that our approach can be useful in those cases as well.

In general, for most theorems, textbook proofs were easily translated into the semi-formal proofs that can be verified with our approach. Sometimes, a slight modification of the textbook proof was needed. Theorems that state that "there exists only one" are processed by formulating two additional theorems. We processed two such examples. Adding new proof steps for construction of new objects (naming of new objects) was performed consciously and frequently. Proof steps that are using reasoning by cases are left out from the semi-formal proof. This was done for three theorems.

The number of steps in each semi-formal proof, the number of verified steps, and the verification time are given in Table 5. The set of axioms that was used contained either the axioms of incidence alone, or the axioms of incidence and the axioms of order (depending on the theorem). In both cases basic definitions used in Hilbert's axiomatic systems were added (definitions for collinearity, coplanarity, etc.). In 12 out of 18 semi-formal proofs, the system successfully verified all the steps of the semi-formal proof. In the other 6, many individual steps were verified as well. In total, out of the 65 automatically generated auxiliary lemmas (that correspond to the semi-formal proof steps), our system automatically verified 56 (86%). On average it took 79 seconds to verify a semi-formal proof, validate the automatically generated XML file, and translate it to formal documents in Isabelle and Coq, and readable documents in English and Serbian.

Table 5 Number of steps in semi-formal proof, number of verified steps, and verification time

Theorem number	Semi-formal proof steps	Number of proven steps	Time (in seconds)
1	4	4	5
2	4	4	6
3	3	3	5
4	2	2	79
5	1	1	14
6	2	2	5
7	17	17	126
8	1	0	125
9	1	1	5
10	1	1	5
11	2	2	73
12	8	8	96
13	1	1	5
14	9	6	213
15	3	2	126
16	1	0	159
17	2	1	134
18	3	1	127

In some cases the system does not verify all auxiliary lemmas, mostly when proofs of those lemmas use a previously proven theorem. If automated provers fail, the current lemma is added without a proof and labeled with *sorry* in Isabelle, and with *Admitted* in Coq. The user can manually fill in the missing proof in the generated Isabelle or Coq file, or change the semi-formal proof of the theorem and run the system again.

Sledgehammer We tested the Sledgehammer on the set of auxiliary lemmas automatically generated by our system. We used the default Sledgehammer settings. For theorem proofs in Hilbert's axiomatic system, out of the 65 automatically generated lemmas, Sledgehammer proved 59 (three more than our system). In some cases Isabelle automatically (without the Sledgehammer) discovered that the goal (the current auxiliary lemma) could be solved directly with an axiom. A few times Sledgehammer was confused by this and found the trivial proof only after the automatically generated proof (found by our system) was run. We classified those lemmas as successful. Sledgehammer gave the same results as our system when tested on theorem proofs in Avigad's axiomatic system.

7 Related work

Formalizing Hilbert's geometry has been the subject of several papers. Dehlinger, Dufourd, and Shreck showed, using the interactive theorem prover Coq, that many theorems from the first two groups of Hilbert's book "der Grundlagen der Geometrie" can not be proved using the intuitionistic approach and without the rule of the excluded middle [12]. They also formulated and verified weaker versions of Hilbert's theorems in an intuitionistic setting, and

later used them as part of the proof of the original theorem (the first step of the proof of the original theorem was always some rule of the excluded middle). Later, Meikle and Fleuriot [30], and Scott and Fleuriot [45] showed, using interactive theorem prover Isabelle, that the formalization of the first three groups of Hilbert's axiomatics is not trivial. They pointed out that many of Hilbert's proofs rely on assumptions that were not stated in the proof, and that some proofs even rely on a graphical presentation of the problem. The rules that they used manipulate finite sets of collinear and planar points, instead of the usual primitives of points, lines, and planes. But, they are still using just the elementary theory. Sets are used just as a convenient representation. In this paper, we did not use such representation. They identified a lot of missing steps in Hilbert's proofs which initially resulted in very complex proofs that showed little resemblance to the original ones. The introduction of sets simplified formal proofs and made them more similar to Hilbert's proofs. They even concluded that gaps in Hilbert's proofs were suitable since they did not capture the essence of the proofs.

The framework for automated theorem proving of mathematical knowledge was used with Tarski's axiomatic system. The framework is created by Stojanović-Đurđević, Narboux, and Janičić, and was used in several experiments [51] with theorems from the first part of the book on foundations of geometry: *Metamathematische Methoden in der Geometrie*, by Wolfram Schwabhäuser, Wanda Szmielew, and Alfred Tarski [44]. The authors thoroughly analyzed Tarski's axioms, definitions, and theorems, and expressed them in coherent logic (which resulted in 238 theorems, starting with 179 theorems of plane geometry). Special attention was paid to the negations, function symbols, and to the use of sets (as an illustration, in Tarski's book, lines are represented as sets of points). Careful comparison and integration with the existing Coq formalization of the Tarski's book [10, 35] was performed (requiring to express auxiliary lemmas formulated in the Coq formalization into coherent logic as well). In the first experiment, all the axioms, definitions, and theorems are listed in the order they appear in the book that is being formalized. While proving the current conjecture, the set of all axioms, definitions, and theorems that precede the conjecture is used by the framework. The authors showed that, in such manner, 37% of the theorems can be completely automatically verified with formal proofs generated. The user can influence the proving process by formulating additional auxiliary lemmas and carefully integrating them within the set of theorems being proved. Auxiliary lemmas were taken from the existing Coq formalization. The addition of auxiliary lemmas has the positive influence on the proving process and rises the percentage of proven theorems to 42%. This approach gave promising results for the formalization of the whole theory. In most cases, the theorems from this paper are too difficult for the (ATP/ArgoCLP) framework itself, so semi-formal proofs of the theorems have to be formulated if we want to generate formal and readable proofs, like we did in this paper.

Axiomatic system *E*, was created for automated verification of postulates from Euclid's *Elements* [2]. There are two computational proof checkers based on the language of *E*. Benjamin Northrop implemented the *E-proof-checker*[23] prover for this formal system [38].[24] The user provides as input the objects that will be used in the proof of a conjecture followed by a series of assertions involving the relations on those objects. The set of axioms of *E* is divided into construction rules and inference rules. The E-proof-checker follows the same

[23] http://www.bennorthrop.com/e/e-proof-checker.php

[24] http://www.phil.cmu.edu/~avigad/formal/paris2_merged.pdf

idea and works in two phases. In the first phase, the prover verifies if all objects can be created with the construction rules. In the second phase, inference rules are used without any guidance. The prover will generate all possible objects and relations and, at the end, verify if the conclusion of the conjecture is demonstrated. The downfall of this prover is that it does not support metric assertions. The second proof checker for the language *E* is a prover *EuclidZ3* created by Kelvin Rojas [42]. This proof checker uses, as a backend, SMT solver *Z3* [34]. SMT solvers are particularly well-suited for metric inferences. While the E-proof-checker verifies proofs in the language *E*, the EuclidZ3 does not parse Euclidean proofs. The EuclidZ3 verifies proofs formulated using collection of Z3 definitions and functions. Unlike the approach described in this paper, both of these proof checkers are created specifically for the axiomatic system *E* and can not work with other axiomatic systems. Also, neither of them generates formal proofs verifiable with interactive theorem provers.

Correcting inaccuracies and filling gaps in Euclid's proofs was done by Beeson, Narboux, and Wiedijk [3]. They developed a new axiomatic system and formulated and verified their proofs in interactive theorem provers HOL Light and Coq. The axiomatic system that they use is very close to Euclid's, and formulated proofs follow the ideas presented in Euclid's proofs. The language of the axiomatic system is similar to the language of Tarski's geometry (with almost all axioms formalized in a *points-only* language). They formalized the first book of the Elements resulting with 213 theorems, starting with 48 of Euclid's propositions. They showed that formalizing the Elements is important but difficult and that both axioms and proofs needed some corrections. They further showed that the order of the propositions had to be changed, and in some cases the proofs deviated notably from the original proofs.

Vyskočil, Kaliszyk, and Urban recently started a new project[25] with the goal to autoformulate an informal mathematical language [25–27]. Their project's scope is much larger than the one presented in this paper. They are using large corpus of corresponding informal/formal formulas and propose the use of various machine learning techniques to train semi-automated translation between informal and formal mathematics. In this paper, we only use simple techniques of forward chaining with iterative deepening implemented in the coherent logic prover ArgoCLP.

8 Conclusions and future work

The approach described in this paper can be used for automatic verification of informal proofs in coherent logic. The system reads a linear semi-formal proof written in the language that resembles mathematical formulas used in geometry textbooks, and generates formal proofs and readable natural language proofs. At the moment, the system generates proofs in languages of two interactive theorem provers (Isabelle and Coq) and two natural languages (English and Serbian).

The primary goal of studying geometry in school is to introduce young students to a conceptual framework for formulating and proving theorems. This framework helps to improve logical thinking and proper reasoning which is applicable beyond the field of geometry itself. We think that our simple approach can significantly aid in fulfilling this educational objective while also being a foray into the field of interactive theorem proving. Even though

[25] https://www.researchgate.net/project/informal2formal

the language of semi-formal proofs is not trivial, we believe that it can easily be mastered by the motivated students.

Both high school geometry and Euclid's proofs were criticized for the same reason, and as Jeremy Avigad et al. noted [1] there is a strong link between Euclid's *Elements* and the proofs used for educational purposes:

> Over the centuries, the style of diagram-based argumentation of Euclid's Elements was held to be the paradigm of rigor, and presentations much like Euclid's are still used today to introduce students to the notion of proof.

The approach described in this paper helped with verifying 18 informal proofs found in Serbian high school geometry textbooks, and 4 informal proofs from Euclid's Elements, which illustrates the potential of this project. Even though we analysed only a small number of proofs in the axiomatic system *E*, we showed that this approach can be used for verification of the basic diagrammatic inferences used in the Elements. Nevertheless, we should be careful in making predictions considering that Euclid's later postulates are noticeably harder and will likely require additional tools.

The approach presented in this paper can be useful for mathematicians who are formalizing parts of mathematical textbooks. Generated formal proofs can be used as part of a larger formalization project and readable proofs can help with understanding an original textbook proof.

Reasoning by cases is currently not supported by the language of semi-formal proofs. In the future, we plan to add proof steps for case splits (disjunctions) and assumptions, as is already done for the coherent logic vernacular [47].

Since we want to keep the language universal, we decided to use mathematical formulas (in a TPTP-like language) to write the semi-formal proof. The alternative would be to use a natural language with a restricted vocabulary. Adding this feature is planed for the future, and it can be done for any natural language (with the corresponding translation to mathematical formulas).

Textbook proofs verified in this paper usually rely on geometric diagrams. Those proofs tend to follow the proof methods used by Euclid. In the future, we plan to add automated generation of corresponding diagrams that help visualize the generated proofs. For this purpose we plan to use the *GCLC* [22] program. GCLC is a tool for producing mathematical illustrations by describing constructions and figures instead of drawing figures (which is more commonly used in other dynamic geometry tools). GCLC has the option of exporting illustrations as LaTeX files, so the generated diagrams can easily be added to our natural language proofs. We also plan to develop additional XSLT style-sheets and add support for languages of other interactive theorem provers (such as Mizar [56]), and other natural languages (such as French and German). Also, we will continue to improve the existing style-sheets with the goal of simplifying the generated proofs (formal and natural language) so that they are even more human-friendly.

Acknowledgements We are grateful to the anonymous reviewers for the extremely helpful feedback on the first version of the paper.

Appendix : Invocation of the system ArgoGeoChecker

– Detailed `README.txt` file can be found on the system's web-page. Here we will give just one example of the invocation of the system. Input file should be named `th_name_proof.txt` and located in the folder `theorems_and_proofs`

- System invocation: `./ArgoGeoChecker axiom_system th_name_proof.txt`
- The shortcut has been added for Hilbert's axiomatic system:
  `./ArgoGeoChecker I th_1_proof.txt`
  The option "I" can be used when we want to use just the first group of Hilbert's axioms for verification of the proof, and the option "II" can be used when we want to use the first and the second group of Hilbert's axioms together (definitions are always used).
- System will automatically generate (in the folder `theorem_and_proofs`) files for all target languages (with extensions *.thy, *.v, *.tex, *.pdf).
- An example:

```
./ArgoGeoChecker I th_1_proof.txt
th_1_01.p| vampire| 0.03| 0.13|             ax_I3a |   0.00||
th_1_02.p| vampire| 0.27| 0.21| ax_sym_ncol ax_D1a |   0.01||
th_1_03.p|       e| 0.28| 0.13|             ax_I4a |   0.02||
th_1_04.p| vampire| 0.16| 0.41|              ax_I6 |   0.00||
Seconds: 6
```

  Here we can see: the name of the auxiliary lemma that is proved — resolution theorem prover that found the smallest axiom set that the lemma can be proved with — Vampire time — E time — set of axioms that was found — ArgoCLP time.
- List of semi-formal proofs in Hilbert's axiomatic system (with the shorcut for the specific set of Hilbert's axioms):

```
./ArgoGeoChecker I th_1_proof.txt
./ArgoGeoChecker I th_2_proof.txt
./ArgoGeoChecker I th_3_proof.txt
./ArgoGeoChecker I th_4_proof.txt
./ArgoGeoChecker I th_5_proof.txt
./ArgoGeoChecker I th_6_proof.txt
./ArgoGeoChecker II th_7_proof.txt
./ArgoGeoChecker II th_8_proof.txt
./ArgoGeoChecker I th_9_proof.txt
./ArgoGeoChecker I th_10_proof.txt
./ArgoGeoChecker I th_11_proof.txt
./ArgoGeoChecker I th_12_proof.txt
./ArgoGeoChecker I th_13_proof.txt
./ArgoGeoChecker I th_14_proof.txt
./ArgoGeoChecker II th_15_proof.txt
./ArgoGeoChecker II th_16_proof.txt
./ArgoGeoChecker II th_17_proof.txt
./ArgoGeoChecker II th_18_proof.txt
```

- List of semi-formal proofs in the axiomatic system *E*:

```
./ArgoGeoChecker axiomatic_system/avigad_axioms.p
 th_prop1_proof.txt
./ArgoGeoChecker axiomatic_system/avigad_axioms.p
 th_prop1aux1_proof.txt
./ArgoGeoChecker axiomatic_system/avigad_axioms.p
```

```
 th_prop1aux2_proof.txt
./ArgoGeoChecker axiomatic_system/avigad_axioms.p
 th_prop2_proof.txt
```

References

1. Avigad, J.: Understanding proofs. In: Mancosu, P. (ed.) The Philosophy of Mathematical Practice. Oxford University Press, Oxford (2008)
2. Avigad, J., Dean, E., Mumma, J.: A formal system for Euclid's elements. The Review of Symbolic Logic (2009)
3. Beeson, M., Narboux, J., Wiedijk, F.: Proof-checking Euclid. CoRR (2017). arXiv:1710.00787
4. Bezem, M., Coquand, T.: Automating coherent logic. In: Sutcliffe, G., Voronkov, A. (eds.) 12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning — LPAR 2005, Lecture Notes in Computer Science, vol. 3835. Springer (2005)
5. Bezem, M., Hendriks, D.: On the mechanization of the proof of Hessenberg's theorem in coherent logic. J. Autom. Reas., **40**(1) (2008)
6. Blanchette, J.C.: Automatic Proofs and Refutations for Higher-Order Logic. Ph.D. thesis. Department of Informatics, Technische Universität München (2012)
7. Blanchette, J.C.: Redirecting proofs by contradiction. In: Blanchette, J.C., Urban, J. (eds.) Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013, Lake Placid, NY, USA, June 9-10, 2013, EPiC Series, vol. 14, pp. 11–26. EasyChair (2013)
8. Blanchette, J.C., Böhme, S., Paulson, L.C.: Extending sledgehammer with SMT solvers. J. Autom. Reason. **51**(1), 109–128 (2013)
9. Blanchette, J.C., Bulwahn, L., Nipkow, T.: Automatic proof and disproof in Isabelle/HOL. In: Tinelli, C., Sofronie-Stokkermans, V. (eds.) Frontiers of Combining Systems, 8th International Symposium, Proceedings, Lecture Notes in Computer Science, vol. 6989, pp. 12–27. Springer (2011)
10. Braun, G., Narboux, J.: From Tarski to Hilbert. In: Ida, T., Fleuriot, J. (eds.) Automated Deduction in Geometry 2012, Edinburgh (2012). http://hal.inria.fr/hal-00727117
11. Braun, G., Narboux, J.: A synthetic proof of Pappus' theorem in Tarski's geometry. J. Autom. Reason., 1–22 (2016)
12. Dehlinger, C., Dufourd, J.F., Schreck, P.: Higher-order intuitionistic formalization and proofs in Hilbert's elementary geometry. In: Automated Deduction in Geometry, Lecture Notes in Computer Science, vol. 2061. Springer (2001)
13. Despotović, R., Tošić, R., Šešelja, B.: Matematika za I razred srednje škole. Zavod za udžbenike i nastavna sredstva, Beograd (2000)
14. Ekici, B., Katz, G., Keller, C., Mebsout, A., Reynolds, A.J., Tinelli, C.: Extending smtcoq, a certified checker for smt (extended abstract). In: Blanchette, J.C., Kaliszyk, C. (eds.) Proceedings First International Workshop on Hammers for Type Theories, Coimbra, Portugal, July 1, 2016, Electronic Proceedings in Theoretical Computer Science, vol. 210, pp. 21–29. Open Publishing Association (2016)
15. Euclid The Thirteen Books of the Elements, 2nd edn. Dover Publications, New York (1956). Translated with introduction and commentary by Sir Thomas L. Heath, from the text of Heiberg
16. Fisher, J., Bezem, M.: Skolem machines and geometric logic. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) 4th International Colloquium on Theoretical Aspects of Computing — ICTAC 2007, Lecture Notes in Computer Science, vol. 4711. Springer (2007)
17. Ganesalingam, M., Gowers, W.T.: A fully automatic theorem prover with human-style output. J. Autom. Reas., 1–39 (2016). https://doi.org/10.1007/s10817-016-9377-1
18. Gonthier, G., Asperti, A., Avigad, J., Bertot, Y., Cohen, C., Garillot, F., Le Roux, S., Mahboubi, A., O'Connor, R., Ould Biha, S., Pasca, I., Rideau, L., Solovyev, A., Tassi, E., Théry, L.: A machine-checked proof of the odd order theorem. In: Blazy, S., Paulin, C., Pichardie, D. (eds.) ITP 2013, 4th Conference on Interactive Theorem Proving, LNCS, vol. 7998, pp. 163–179. Springer, Rennes (2013). https://doi.org/10.1007/978-3-642-39634-2_14
19. Hales, T.C.: Introduction to the Flyspeck Project, Mathematics, Algorithms, Proofs, Dagstuhl Seminar Proceedings, vol. 05021. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2006)
20. Harrison, J.: HOL light: A tutorial introduction. In: Srivas, M.K., Camilleri, A.J. (eds.) Formal Methods in Computer-Aided Design, Lecture Notes in Computer Science, vol. 1166. Springer (1996)

21. Hilbert, D.: Foundations of Geometry. Open Court Classics, 10th edn (1971)
22. Janičić, P.: GCLC – A tool for constructive Euclidean geometry and more than that. In: Takayama, N., Iglesias, A., Gutierrez, J. (eds.) Proceedings of International Congress of Mathematical Software (ICMS 2006), Lecture Notes in Computer Science, vol. 4151, pp. 58–73. Springer (2006)
23. Kaliszyk, C., Krauss, A.: Scalable LCF-style proof translation. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings, Lecture Notes in Computer Science, vol. 7998, pp. 51–66. Springer (2013)
24. Kaliszyk, C., Urban, J.: Mizar 40 for mizar 40. J. Autom. Reason. **55**(3), 245–256 (2015)
25. Kaliszyk, C., Urban, J., Vyskočil, J.: Learning to Parse on Aligned Corpora (Rough Diamond), pp. 227–233. Springer International Publishing, Cham (2015)
26. Kaliszyk, C., Urban, J., Vyskočil, J.: Automating Formalization by Statistical and Semantic Parsing of Mathematics, pp. 12–27. Springer International Publishing, Cham (2017)
27. Kaliszyk, C., Urban, J., Vyskočil, J., Geuvers, H.: Developing Corpus-Based Translation Methods between Informal and Formal Mathematics: Project Description, pp. 435–439. Springer International Publishing, Cham (2014)
28. Kinyon, M., Veroff, R., Vojtěchovský, P.: Loops with Abelian Inner Mapping Groups: An Application of Automated Deduction, pp. 151–164. Springer, Berlin (2013)
29. McCune, W.: Solution of the robbins problem. J. Autom. Reason. **19**(3), 263–276 (1997)
30. Meikle, L., Fleuriot, J.: Formalizing Hilbert's Grundlagen in Isabelle/Isar. In: Proceedings of TPHOLs, Lecture Notes in Computer Science, vol. 2758. Springer (2003)
31. Meng, J., Paulson, L.C.: Translating higher-order clauses to first-order clauses. J. Autom. Reason. **40**(1), 35–60 (2008)
32. Meng, J., Quigley, C., Paulson, L.C.: Automation for interactive proof: first prototype. Inf. Comput. Special Issue: Comb. Logical Syst. **204**(10), 1575–1596 (2006)
33. Mitrović, M., Ognjanović, S., Veljković, M., Petković, L., Lazarević, N.: Geometrija za prvi razred Matematičke gimnazije. KRUG Beograd (1998)
34. de Moura, L., Bjørner, N.: Z3: An Efficient SMT Solver, pp. 337–340. Springer, Berlin (2008)
35. Narboux, J.: Mechanical theorem proving in Tarski's geometry. In: Proceedings of Automated Deduction in Geometry 2006, Lecture Notes in Computer Science, vol. 4869. Springer (2007)
36. Nikolić, M., Janičić, P.: CDCL-Based Abstract State Transition System for Coherent Logic, pp. 264–279. Springer, Berlin (2012)
37. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL - A Proof Assistant for Higher-Order Logic, Lecture Notes in Computer Science, vol. 2283. Springer (2002)
38. Northrop, B.: Automated Diagrammatic Reasoning: A proof checker for the language of E. Master's thesis, Department of Philosophy Carnegie Mellon University (2011)
39. Phillips, J.D., Stanovský, D.: Automated theorem proving in quasigroup and loop theory. AI Commun. **23**(2-3), 267–283 (2010)
40. Polonsky, A.: Proofs, Types and Lambda Calculus. Ph.D. thesis, University of Bergen (2011)
41. Riazanov, A., Voronkov, A.: The design and implementation of VAMPIRE. AI Commun. **15**(2-3), 91–110 (2002)
42. Rojas, K.: EuclidZ3 – a Computational Proof-Checker for the Language E: Possible Backend for Interactive Geometric Proof Environments. Master's thesis, Department of Philosophy Carnegie Mellon University (2015)
43. Schulz, S.: E - a brainiac theorem prover. AI Commun. **15**(2-3), 111–126 (2002)
44. Schwabhäuser, W., Szmielew, W., Tarski, A.: Metamathematische Methoden in der Geometrie. Springer, Berlin (1983)
45. Scott, P., Fleuriot, J.: An investigation of Hilbert's implicit reasoning through proof discovery in idle-time. In: Schreck, P., Narboux, J., Richter-Gebert, J. (eds.) Automated Deduction in Geometry, pp. 182-200. Springer, Berlin (2011)
46. Stojanović, S.: Formalization and automation of Euclidean geometry (in serbian). Ph.D. thesis, University of Belgrade. Advisor: Predrag Janicić (2016)
47. Stojanović, S., Narboux, J., Bezem, M., Janičić, P.: A vernacular for coherent logic. In: S.W., other (eds.) Intelligent Computer Mathematics - CICM 2014, Lecture Notes in Computer Science, vol. 8543. Springer (2014)
48. Stojanović, S., Pavlović, V., Janicić, P.: A coherent logic based geometry theorem prover capable of producing formal and readable proofs. In: Schreck, P., Narboux, J., Richter-Gebert, J. (eds.) Automated Deduction in Geometry, Lecture Notes in Computer Science, vol. 6877. Springer (2011)
49. Stojanović, V.: Matematiskop 3 - Odabrani zadaci (sa rešenjima) za učenike srednjih škola. Beograd, Naučna knjiga (1988)

50. Stojanović-Đurđević, S.: Automated verification of informal proofs from high school geometry (in Serbian) InfoM (2016)
51. Stojanović-Đurđević, S., Narboux, J., Janičić, P.: Automated generation of machine verifiable and readable proofs: A case study of Tarski's geometry. Annals of Mathematics and Artificial Intelligence (2015)
52. Sutcliffe, G.: The TPTP problem library and associated infrastructure: The FOF and CNF parts, v3.5.0. J. Autom. Reason. **43**(4), 337–362 (2009)
53. The Coq development team: The Coq proof assistant reference manual, Version 8.2. TypiCal Project (2009). http://coq.inria.fr
54. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: Spass version 3.5. In: Automated Deduction - CADE-22 Proceedings, Lecture Notes in Computer Science, vol. 5663, pp. 140–145. Springer (2009)
55. Wenzel, M.: Isar - a generic interpretative approach to readable formal proof documents. In: Bertot, Y., Dowek, G., Hirschowitz, A., Paulin, C., Théry, L. (eds.) Theorem Proving in Higher Order Logics (TPHOLs'99), Lecture Notes in Computer Science, vol. 1690, pp. 167–184. Springer (1999)
56. Wiedijk, F.: A synthesis of the procedural and declarative styles of interactive theorem proving. Logical Methods Comput. Sci., **8**(1) (2012)