A Readability Criterion for Humans and Machines

Pedro Quaresma[1][0000−0001−7728−4935]
Pierluigi Graziani[2][0000−0002−8828−8920]⋆q

[1]University of Coimbra, pedro@mat.uc.pt
[2]University of Urbino, pierluigi.graziani@uniurb.it

Abstract. Previous research has investigated the criteria for evaluating the readability of geometric proofs generated by theorem provers, with a particular focus on human readability. In particular, Graziani and Quaresma [14] focused on the readability of geometric proofs produced by theorem provers implementing the area method. Building on their work, this study analyzes the machine's performance during the proof process, specifically regarding CPU time. The aim is to assess whether this analysis uncovers significant differences in readability criteria between humans and machines and to identify potential improvements to better align human and machine readability.

1 Introduction

Automated Theorem Proving (ATP) is a well-established area of research in mathematics, boasting numerous methods and results alongside many open problems that highlight its vitality. Among the various domains within ATP, geometry stands out due to its history and the challenging problems it proposes.

Two prominent challenging problems in ATP are particularly noteworthy: the *simplicity of a proof* and the *readability of a proof*.

- The first problem seeks a criterion to quantify the simplicity of a proof.
- The second problem seeks a criterion to quantify the readability of a proof.

Regarding the simplicity problem, historical contributions from scholars such as David Hilbert, who proposed a framework for identifying the simplest possible proofs [2,3,17], and Émile Lemoine [9], who introduced the notion of *Geometrography* to measure the simplicity of geometric constructions underscore the enduring importance of this concept. Recent advancements in automated theorem proving have refined these studies on simplicity. For instance, Pierluigi Graziani and Pedro Quaresma's modernisation of Lemoine's Geometrography [14,15,16] exemplifies contemporary efforts to integrate historical methods with cutting-edge technology. Their modernisation leverages tools and automated proving

⋆ The first author was partially financed through national funds by FCT - Fundação para a Ciência e a Tecnologia, I.P., in the framework of the Project UIDB/00326/2020 and UIDP/00326/2020.

methods, such as the *Geometry Constructions LaTeX Converter* (*GCLC*) and the *area method* [4,6] to create a more robust framework for evaluating the simplicity of geometric constructions and of formal proofs produced by automated theorem provers.

Concerning the readability problem, it is a cornerstone of mathematical communication. Readability refers to the ease with which a reader can comprehend a written text, distinct from legibility, which pertains to recognising individual characters. To our knowledge, there are two precise proposals to measure the readability of a proof: the first by Chou et al. [1], and the second by Freek Wiedijk [18], known as the de Bruijn factor. Also, in this case, recent advancements in automated theorem proving have refined these studies on readability. Graziani and Quaresma [14], dissatisfied with the previous criteria—one being too restrictive and the other too general—applied methodologies from the study of simplicity to explore the readability of geometric proofs, proposing not only their criterion but also a general methodology for analysing this characteristic in the context of proofs produced via the area method. By enhancing the readability of these proofs, they aimed to make complex mathematical concepts more accessible to a broader audience, thereby bridging the gap between automated systems and human mathematicians. Improving readability not only aids human understanding but also enhances the ability of automated systems to communicate their results effectively.

The study of the simplicity and the readability issues led Graziani and Quaresma to analyse interesting elements in the geometric proofs. Given a mathematical proof as a sequence of steps (the proof script), in addition to the coefficient of simplicity, $\mathrm{CS}_{\mathrm{proof}}$, giving the simplicity coefficient for the overall proof, it is possible to consider other coefficients, e.g.,

- $\mathrm{CS}_{\mathrm{gcl}}$, simplicity coefficient for the geometric construction (the conjecture);
- $\mathrm{CT}_{\mathrm{proof}}$, the total number of steps in the proof;
- $\mathrm{CS}_{\mathrm{proofmax}}$, the highest simplicity coefficient of the lemmas/definitions applications, it gives the simplicity coefficient for the most difficult step of the proof;
- $\mathrm{CD}_{\mathrm{typeproof}}$, the number of different types of lemmas used in the proof;
- $\mathrm{CD}_{\mathrm{highproof}}$, the number of different steps of high difficulty in the proof;

and also

- The proof script (all the steps in the formal proof);
- The corresponding line chart or proof trace, given by the simplicity coefficient of each step.

Graziani and Quaresma called these coefficients the, *Geometrographic Coefficients*.

Let's consider, for example, the Ceva's theorem.

Theorem 1 (Ceva's Theorem, [6]). *Let ΔABC be a triangle and P be any point in the plane. Let $D = AP \cap CB$, $E = BP \cap AC$, and $F = CP \cap AB$. Show*

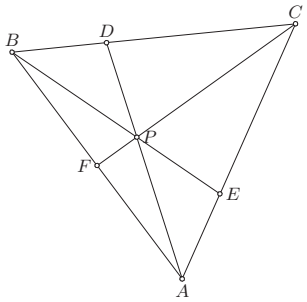


Fig. 1. Geometric Construction, Ceva's Theorem

that: $\frac{\overline{AF}}{\overline{FB}} \times \frac{\overline{BD}}{\overline{DC}} \times \frac{\overline{CE}}{\overline{EA}} = 1$. *P should not be in the lines parallels to AC, AB and BC and passing through B, C and A respectively.*

The proof script of Ceva's theorem, produced by the *GCLC* prover based on the area method, has all the details explained, and it fills two pages, almost three pages if the notes about the non-degeneracy conditions and the proof itself are considered (see [14]).

The Geometrographic coefficients of Ceva's Theorem Proof are the following:

$$\textbf{Ceva's Theorem} \begin{cases} \mathrm{CS}_{\mathrm{proof}} = 220 \\ \mathrm{CS}_{\mathrm{gcl}} = 22 \\ \mathrm{CT}_{\mathrm{proof}} = 32 \\ \mathrm{CS}_{\mathrm{proofmax}} = 84 \\ \mathrm{CD}_{\mathrm{typeproof}} = 3 \\ \mathrm{CD}_{\mathrm{highproof}} = 0 \end{cases}$$

The line chart (or proof trace) is shown in figure 2. In it, the sequences of algebraic simplifications are condensed in only one step (for a more condensed view of the graph).

What does this line chart describe? This line chart is based on assumptions about certain parameters' weight (efforts) on human readability (high, medium, and low) of a geometric proof. It graphically represents the progress of the proof with weights associated with the various steps, expressing their difficulty via their simplicity (complexity) coefficient.

It can be asked whether these assumptions and weights reflect values solely related to the human being proving the theorem or if they also pertain to the effort exerted by the machine's processors in constructing the proof. In other words, if the line chart resembles an electroencephalogram, as Graziani and Quaresma wrote [14], can it be considered also a kind of electroencephalogram of the machine during the theorem proving? Or are the encephalograms related to the human-proving activity and the machine-proving activity different from each other?

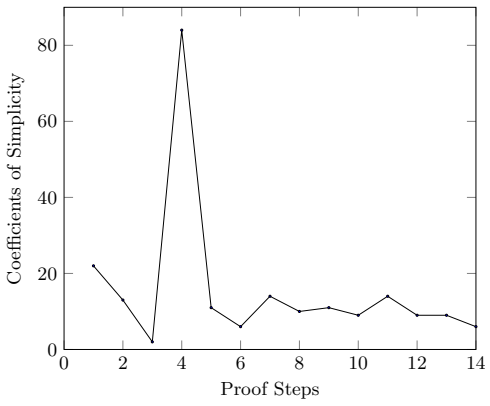


Fig. 2. Ceva's Theorem, Geometrography Proof Trace

Building on these foundational concepts and issues, this paper explores a novel objective with respect to [14]: comparing the complexity of line charts produced in geometric constructions with the computational effort required by the processors executing the theorem proving. By examining the correlation between human readability and the underlying machine computational processes, the authors aim to understand whether the weights/efforts assumed for humans reflect the machine's efforts in its proof process. This exploration is particularly pertinent in modern computational resources, where the processor's effort in executing theorem-proving tasks can vary significantly based on task complexity. It is possible to identify patterns and correlations indicating the computational load involved by systematically analysing the charts representing these geometric proofs. Such insights can inform the development of more efficient algorithms and systems, ultimately contributing to the advancement of automated theorem-proving technologies.

Overview of the paper. The paper is organised as follows: Section 2 provides an introduction to *GCLC*; Section 3 summarises Graziani and Quaresma's research on simplicity and readability; Section4 describes the methodology for analysing processor activity; Section 5 compares the graphical representation of the proof of the theorem and processor activity; Section 6 presents the conclusions and future work.

2 *GCLC* (from Geometry Constructions → LaTeX Converter)

GCLC (Geometry Constructions → LaTeX Converter),[1] developed by Predrag Janičić [4], is a specialised software tool designed to facilitate the creation of geo-

[1] `https://github.com/janicicpredrag/gclc`

metric constructions and their seamless conversion into LaTeX format. This tool bridges the gap between geometric visualisation and high-quality typesetting, making it invaluable for educational and research purposes in mathematics.

GCLC describes geometric constructions using a simple and intuitive language known as *gcl* [5]. This language allows users to define points, lines, circles, and other geometric objects and specify their relationships and constructions. The core idea is to use a sequence of primitive construction steps, such as drawing a line between two points, finding the intersection of two lines, or drawing a circle given its centre and a point on the circumference. These primitive steps can be combined to form more complex constructions.

To illustrate the capabilities of *GCLC*, consider the following example, which constructs a basic geometric figure, a triangle with a circumcircle. The *gcl* code for this example is as follows:

```
%%% Constructive steps
point A 50 65
point B 45 35
point C 90 35
med a C B
med b A C
intersec O a b

%%% Drawing steps
cmark_lt A
cmark_lb B
cmark_rb C
drawsegment A B
drawsegment B C
drawsegment C A
drawcircle O A
```

Listing 1.1. *gcl*—Specification—Triangle & circuncircle

The *gcl* language has commands to build the geometric construction and commands to draw the corresponding figure. This code (see Listing 1.1) defines three points, A, B, and C, with their respective coordinates, the segments CB and AC perpendicular bisectors (`med`iatrice), and the intersection of those two lines, the point O, the centre of the circuncentre. The construction geometric construction is independent of the coordinates, which are solely used to draw the figure. The `cmark` command marks these points on the figure. The `drawsegment` commands are used to draw the sides (segments) of the triangle ABC and `drawcircle` command constructs the circumcircle passing through the three vertices of the triangle. As it can be seen by this example, the construction and the drawing are two separate parts of a *gcl* specification. A user cannot draw an non-existing element, but what to draw, or not, is a user's decision. From the automated theorem provers, embedded in *GCLC*, point of view, the drawing section is superfluous.

Once the gcl file (e.g., `triangle.gcl`) is created, it can be processed by *GCLC* to generate a LaTeX file.

This process integrates the geometric figure into the LaTeX document, ensuring the final output is mathematically precise and visually appealing.

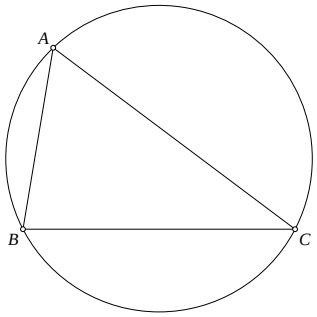


Fig. 3. *gcl*—Drawing—Triangle & circuncircle

GCLC also supports more advanced geometric constructions and transformations see [4,5] and also the manual, distributed with the software.

GCLC also contains a set of embedded automated theorem provers for geometry, e.g. the one based in the *Area Method* [6,7,8].

3 *GCLC* and Geometrography

Geometrography, "alias the art of geometric constructions", aims at providing a tool: (i) to designate every geometric construction by a symbol that manifests its simplicity and exactitude; (ii) to teach the simplest way to execute an assigned construction; (iii) to discuss a known solution to a problem and eventually replacing it with a better solution; (iv) to compare different solutions for a problem, by deciding which is the most exact and the simplest solution from the point of view of Geometrography.

Émile Lemoine [9] in his *Geometrography* defined two coefficients to measure the relative difficulty of performing some geometric constructions. The approach is applied to ruler and compass geometry, i.e., geometric constructions made only with the help of a ruler and a compass. Considering the modifications proposed by John S. Mackay [10], the following **R**uler and **C**ompass constructions and the corresponding coefficients can be analysed.

To place the edge of the ruler in coincidence with one point ..... R_1
To place the edge of the ruler in coincidence with two points ..... $2R_1$
To draw a straight line ..... R_2
To put one point of the compasses on a determinate point ..... C_1

To put the points of the compasses on two determinate points ..... $2C_1$
To describe a circle ..... C_2

Then a given construction is measured against the number of uses of those elementary steps. For a given construction expressed by the equation:

$$l_1R_1 + l_2R_2 + m_1C_1 + m_2C_2$$

where l_i and m_j are coefficients denoting the number of times any particular operation is performed. The number $(l_1 + l_2 + m_1 + m_2)$ is called the *coefficient of simplicity* of the construction, and it denotes the total number of operations performed. The number $(l_1 + m_1)$ is called the *coefficient of exactitude* of the construction and it denotes the number of preparatory operations on which the exactitude of the construction (made with the help of physical, inaccurate, tools) depends [10,11].

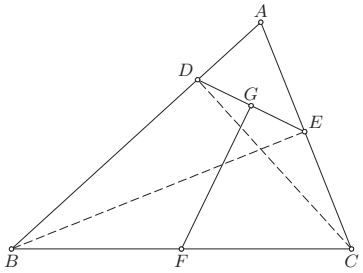
Classical *Geometrography* applies to geometric constructions made with the help of a ruler and a compass. Graziani and Quaresma proposed a modernisation and generalisation of classical *Geometrography* in [14,15,16] using the tools of the dynamic geometry systems (DGS) and using *GCLC*.

Considering the operations: *define a point, anywhere in the plane*, D and *define a given object, using other objects*, C, the following values for the *GCLC* basic constructions are obtained:

`point` – fix a point in the plane ..... D
`line` – uses two points ..... $2C$
`circle` – uses two points ..... $2C$
`intersec` – uses two lines ..... $2C$
`intersec` – uses four points ..... $4C$
`intersec2` – uses a circle and a circle or line ..... $2C$
`midpoint` – uses two points ..... $2C$
`med` – uses two points ..... $2C$
`bis` – uses three points ..... $3C$
`perp` – uses a point and a line ..... $2C$
`foot` – uses a point and a line ..... $2C$
`parallel` – uses a point and a line ..... $2C$
`onsegment` – uses two points ..... $2C$
`online` – uses two points ..... $2C$
`oncircle` – uses two points ..... $2C$

In the modernisation of the *Geometrography*, the *coefficient of exactitude* loses its meaning; the DGS executes the constructions, so they are accurate (exact). However, the *coefficient of simplicity* of the constructions can still be useful. It can be used to classify constructions by levels of simplicity. A new dimension can also be added, the *coefficient of freedom*, given by the degree of freedom a given geometric object has, e.g., "a point in a line" has one degree of freedom, a point in the plane has two degrees of freedom, etc. This new coefficient gives

a value to the dynamism of the geometric construction. Graziani and Quaresma calculated these coefficients for all constructions in the *TGTP* repository [12].[2] For the *GCLC* constructions contained in *TGTP* an average value of simplicity ($\mathrm{CS_{gcl}}$) of 20.8 was obtained. Using the k-means clustering function implemented in the statistics package of *Octave*,[3] three classes of geometric constructions, describing an increasing level of complexity, were defined: simple constructions, $1 \leq \mathrm{CS_{gcl}} \leq 18$; average complexity constructions, $18 < \mathrm{CS_{gcl}} \leq 28$; complex constructions, $\mathrm{CS_{gcl}} > 28$. *TGTP* contain 71 simple constructions; 81 average complexity constructions; 28 complex constructions.



$$\mathrm{CS_{gcl}} = 19 = 3\mathbf{D} + 16\mathbf{C}$$
$$\mathrm{CF_{gcl}} = 6$$

Fig. 4. Geometric Construction, *TGTP* problem `GEO0369`

For example (*TGTP* problem `GEO0369`): "In triangle ΔABC, let F be the midpoint of the side BC, and D and E the feet of the altitudes on AB and AC, respectively. FG is perpendicular to DE at G. Show that G is the midpoint of DE", has a geometric construction with a coefficient of simplicity 19 (see Fig. 4), so an average complexity construction, and coefficient of freedom 6.

Graziani and Quaresma used the same approach to take into consideration of synthetic geometric proofs, i.e., proofs based on a geometric axiomatic theory, using geometric inference rules. Considering the proofs produced by the Geometric Automated Theorem Prover (GATP) *GCLC*, implementing the area method [4,6], they calculated the coefficient of simplicity for all the axioms and lemmas of the theory.

Apart from the geometric constructions in which the proof is based (with the coefficient of simplicity n**Cnst**), other steps must be considered.

(Elementary) Algebraic Simplification.....(**AS**)

[2] `http://hilbert.mat.uc.pt/TGTP/`

[3] GNU Octave, version 6.1.1, package octave-statistics, function means `https://octave.sourceforge.io/statistics/function/kmeans.html`

(Elementary) Geometric Simplification ..... (**GS**)
Application of the Area Method Lemma n ..... ($\mathbf{AML_n}$)

A given proof can thus be measured against the number of those steps.[4] For a given proof expressed by the equation:

$$n_1\mathbf{Cnst} + n_2\mathbf{AS} + n_3\mathbf{GS} + \sum_{j=l_1}^{l_k} \mathbf{AML_j}$$

The coefficient of simplicity for the proof would be:

$$\mathrm{CS}_{\mathrm{proof}} = n_1 + n_2 + n_3 + \sum_{j=l_1}^{l_k} \mathrm{CS}_{\mathrm{proof}}(\mathbf{AML_j})$$

The coefficient of freedom has no meaning in this setting.

Each lemma of the area method, $\mathbf{AML_j}$, has a corresponding simplicity coefficient, the term, $\sum_{j=l_1}^{l_k} \mathrm{CS}_{\mathrm{proof}}(\mathbf{AML_j})$, is the sum of all those values, for all the lemmas used in the proof. To achieve this for each area method lemma, the corresponding simplicity coefficients were calculated [13].

Given that a mathematical proof is a sequence of steps, in addition to the coefficient of simplicity, Graziani and Quaresma defined the other coefficients shown in the first section: e.g., the total number of steps in the proof; the value of the most difficult step in the proof; the number of different steps of high difficulty in the proof; the number of different types of steps (lemmas) in the proof; a proof script; a numerical description of the proof; and a corresponding line chart or proof trace.

It is important to note that to obtain the coefficient $\mathrm{CD}_{\mathrm{highproof}}$ (hp) the area method lemmas implemented in the GATP *GCLC* were analysed, and, using the k-means clustering function implemented in the statistics package of *Octave*, divided into three categories: low difficulty ($hp < 284$), medium difficulty ($284 \leq hp < 1848$) and high difficulty ($hp \geq 1848$).

It is interesting to note how the explored coefficients highlight many salient aspects of the proof, which could be used to analyse the readability of proofs.

Applying the *Geometrography* to the area method proofs contained in the repository *TGTP*, using the GATP *GCLC* with the full level of detail, and using the geometrographic coefficients Graziani and Quaresma argued in favour of the following new readability coefficient [14] for geometric proofs:

Geometrographic Readability Coefficient of Proofs (GRCP)

$$GRCP = ((\mathrm{CS}_{\mathrm{proof}} - \mathrm{CT}_{\mathrm{proof}}) \times (\mathrm{CD}_{\mathrm{highproof}} + \mathrm{CD}_{\mathrm{typeproof}}))$$

[4] By elementary algebraic simplification, it is understood the basic algebraic operations: addition, subtraction, multiplication, division, and their properties of commutativity, associativity, and distributivity. By elementary geometric simplification, the direct application of the definition of the area method quantities is understood. These steps are called *trivial steps*.

This coefficient relates four quantities: the simplicity coefficient of the proof, the total number of steps in the proof, the number of different steps with high-difficulty in the proof, the number of different lemmas used in the proof.

The first factor approximates the simplicity coefficient of the non-trivial steps in the proof. Given that in $\mathrm{CT}_{\mathrm{proof}}$ the construction is equal to 1 step, as well as every application of a lemma, every **AS** step and every **GS** step (i.e., the **AS** and **GS** steps weight is 1), if $\mathrm{CT}_{\mathrm{proof}}$ is subtracted from $\mathrm{CS}_{\mathrm{proof}}$ what will be removed are surely all the trivial steps and also 1 from $\mathrm{CS}_{\mathrm{gcl}}$ and 1 for each application of the lemmas (but the weight of those steps is much grater then 1).

The second factor accounts for the difficult steps. Steps that potentiality make the proof much harder to follow, steps where the normal flow of the proof would be interrupted to jump to the proof of the lemma, resuming after completing the lemma's proof. Adding the number of high-difficulty steps with the number of different lemmas used in the proof gave a multiplying factor for the overall complexity of the proof. A final note about this second factor: a high-difficulty step is, for sure, a lemma application, nevertheless, it is felt that the high-difficulty nature of the lemma is a sufficient reason for this double counting.

Multiplying these factors, the approximation for the overall simplicity coefficient and the number of difficult steps—both elements that are believed to characterise the readability of a proof—the readability coefficient of a proof is obtained.

Therefore, considering 71 theorems and their area method proofs, from the *TGTP* repository and using, again, the k-means clustering function from *Octave*, the proofs can be divided into the following classes of Geometrographic readability:[5]

- readable ($high - readability$), $GRCP \leq 48000$;
- medium-readability $48000 < GRCP \leq 135000$;
- low-readability, $GRCP > 135000$.

For example the *GRCP* for `GEO0001`, Ceva's proofs is: $\mathrm{GRCP}_{\mathrm{GEO0001}} = (220 - 32) \times (0 + 3) = 564 \leq 48000$, so a readable (high-readability) proof.

4 An Analysis of Processors' Computational Effort

The *GCLC* prover is a open source project.[6] For that reason it was possible to access the source code and modify it to include the "watch points". The `chrono` library, for dealing with time, was used (see Listing 1.2). At the beginning of the proof cycle the CPU clock was kept in a variable `start` (see Listing 1.3) and at each step of the proof the CPU clock was kept in a variable `stop`, the `duration` is given by the difference `stop-start` and the process was re-initialised, taken again the CPU time and kept it in the variable `start` (see Listing 1.4).

[5] The actual values were rounded for better readability.

[6] *GCLC*, GitHub repository, `https://github.com/janicicpredrag/gclc`

```
#include <chrono>
```

Listing 1.2. `chrono`—C++ Library to deal with time

```
start = std::chrono::high_resolution_clock::now();
```

Listing 1.3. `chrono`—C++ Library to deal with time

```
std::cout << "el" << elLema++ << "␣;␣" << "␣El.␣Point␣" + it->arg[0];
stop = std::chrono::high_resolution_clock::now();
duration = std::chrono::duration<double, std::micro> (stop - start);
std::cout  << "␣;␣␣"<<  duration.count() << "\n" ;
start = std::chrono::high_resolution_clock::now();
```

Listing 1.4. `chrono`—C++ Library to deal with time

For simplicity the output was sent to the *standard output*, and, at running time, redirected to a file (see Listing 1.5).

```
gclc GEO0021 > GEO0021_CPU.txt
```

Listing 1.5. Calling *GCLC* and redirecting its output

The format of the output file is CSV (*Comma Separated Values*), for an easy treatment and the construction of the line graph (see Listing 1.4).

5 Proof Simplicity vs Computer Effort

In this section the line charts presenting the coefficients of simplicity for the proof and the computational effort required by the processors executing the theorem proving are presented.

In the line charts that follow all the steps are counting, not like in the previous line chart (see Figure 2) where the algebraic steps were condensed to one position only. In the study of the simplicity coefficients, the algebraic steps were considered trivial, so its weight is always 1. This is something that must be reformulated as shown by the computation effort (CPU time) line, more about this in the conclusions (see § 6).

All the examples were taken from the *TGTP* repository.

Ceva's Theorem TGTP `GEO0001`, (see Theorem 1).

The line chart (or proof trace) is shown in figure 5.

Circumcentre of a Triangle TGTP `GEO0021`. A medium-readability proof.

Theorem 2 (Circumcentre of a Triangle). *The circumcentre of a triangle can be found as the intersection of the three perpendicular bisectors.*

The proof by *GCLC* area method takes 599 steps and 0.091 seconds. By the GRCP criteria it is a medium-readability proof.

See figure 6 for the proof trace (the fist 72 steps of it), with the lines for the geometrography coefficients and CPU times.

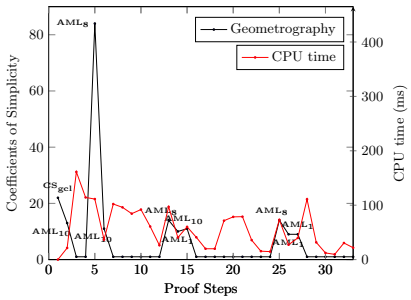


Fig. 5. Ceva's Theorem, Geometrography Proof Trace

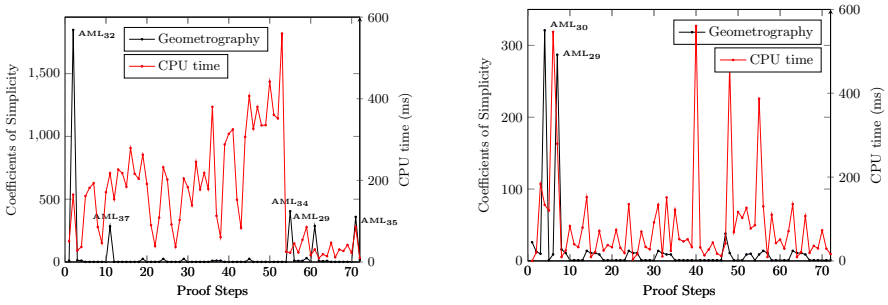


Fig. 6. `GE00021`—the first 72, out of 599, steps & `GE00020`—the first 72, out of 4149, steps

Distance of a line containing the centroid to the vertices TGTP `GEO0020`. A low-readability proof.

Theorem 3 (Distance of a line containing the centroid to the vertices). *Given a triangle ABC and a point X, the sum of the distances of the line XG, where G is the centroid of ABC, to the two vertices of the triangle situated on the same side of the line is equal to the distance of the line from the third vertex.*

The proof by *GCLC* area method takes 4149 steps and take 15.972 seconds. By the GRCP criteria it is a low-readability proof.

See figure 6 for the proof trace (the fist 72 steps of it), with the lines for the geometrography coefficients and CPU times.

6 Conclusions and Future Work

There are two issues in the coefficients of simplicity proof trace that are quite different form its computer effort counterparts.

The first issue relates to the use of the same lemma several times. It was assumed that the first time a lemma from the area method is applied in a proof, the effort to prove that lemma must be taken into consideration, to all the other times only the effort to check if this step is a correct instance of that lemma is needed. It is considered that, from the second application of a lemma onward, its proof is accepted, so, only its adaptation to the new configuration is needed, i.e., the pattern matching of the lemma configuration to a new setting. For that reason, in any second, third, etc. application of a lemma, only the **GS** coefficient values are considered. This led to a high spike for the first instance and a relative low spike for the next applications of the lemma.

This is not the case for the computational effort. All the lemmas of the area method are part of the program, so apart from the effort of finding the correct pattern (an effort that can be compared to the human effort), every application of the lemmas has the same computational effort, there is no attempt to prove it, before applying it. It can be said that the prover would be the counterpart of an expert in the area method, someone that already have proved all the lemmas and use them, without any extra efforts in its application. On the other hand in Graziani and Quaresma initial view, exposed in [14], a mathematician, not necessary knowledgable of the area method, is considered.

The second issue relates to the treatment of algebraic steps. In previous work, algebraic steps were considered trivial in terms of effort. However, based on the computational effort line, it has become clear that this assumption is inaccurate. There is a need for a more detailed analysis of the algebraic steps, similar to what was done for geometric steps, to develop appropriate simplicity coefficients. This insight underscores the importance of reevaluating algebraic operations to reflect their contribution to the overall effort better.

This study reveals key differences in how humans and machines handle proof readability. Humans can reuse previously proven lemmas with reduced effort

in subsequent applications, while machines treat each lemma application independently, requiring the same computational resources each time. Additionally, humans tend to treat algebraic steps as trivial, whereas machines apply consistent computational effort regardless of the simplicity perceived by humans.

These findings suggest important directions for the future design of automatic proof systems. By recognizing patterns in lemma reuse and adapting proofs to match human readability, developers can create systems that generate more user-friendly proofs. From a computational perspective, minimizing redundant calculations and optimizing the reuse of previously computed steps can significantly reduce the machine's workload.

For example, Figure 7 shows as the proof trace of Ceva's Theorem using Geometrography (see Figure 5) can be manually edited to reflect a human expert's perspective. Specifically, all algebraic steps have been assigned the same minimal CPU value. It is evident that the traces are very similar, with the primary difference being the initial value: for human readability, Graziani and Quaresma [14] assigned the simplicity coefficient based on the geometric construction, whereas the CPU time for this initial step remains to be determined.

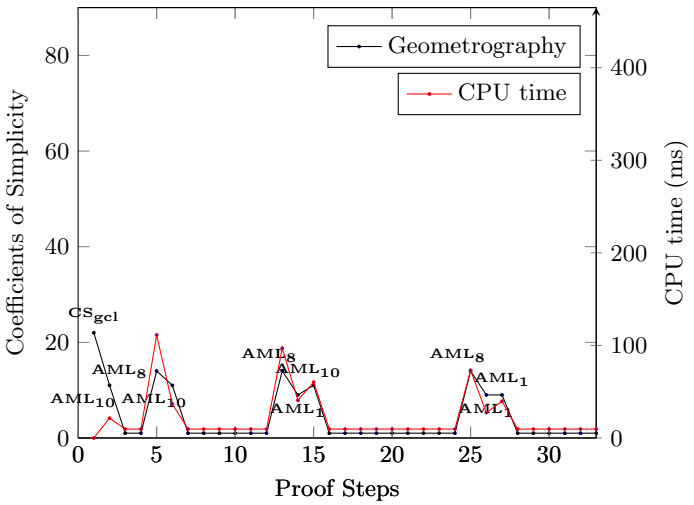


Fig. 7. Ceva's Theorem, Geometrography Proof Trace — edited

Looking forward, this study provides a foundation for research into hybrid-proof systems that balance human comprehension with machine efficiency. Future work could focus on exploring these concepts using different provers and/or methods different from GCLC/area methods, as well as developing algorithms that dynamically adjust the structure of proofs to accommodate both human-readable formats and machine-efficient processing. This dual approach has the potential to improve tools for teaching and learning mathematics, as well as

enhance the efficiency of automated proof verification in advanced research contexts. Such advancements could make formal proofs more accessible, both by reducing the cognitive load on human users and optimizing the computational resources required for their verification.

References

1. Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants, I. multiple and shortest proof generation. *Journal of Automated Reasoning*, 17:325–347, 1996.
2. Davida Hilbert. Mathematical problems. *Bulletin of the American Mathematical Society*, 8:437—-479, 1902.
3. Inês Hipolito and Reinhard Kahle. Discussing Hilbert's 24th problem. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 377(2140):20180040, jan 2019.
4. Predrag Janičić. GCLC — A tool for constructive Euclidean geometry and more than that. In Andrés Iglesias and Nobuki Takayama, editors, *Mathematical Software - ICMS 2006*, volume 4151 of *Lecture Notes in Computer Science*, pages 58–73. Springer, 2006.
5. Predrag Janičić. Geometry constructions language. *Journal of Automated Reasoning*, 44:3–24, 2010.
6. Predrag Janičić, Julien Narboux, and Pedro Quaresma. The Area Method: A Recapitulation. *Journal of Automated Reasoning*, 48(4):489–532, 2012.
7. Predrag Janičić and Pedro Quaresma. System description: GCLCprover + GeoThms. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 145–150. Springer, 2006.
8. Predrag Janičić and Pedro Quaresma. Automatic verification of regular constructions in dynamic geometry systems. In Francisco Botana and Tomás Recio, editors, *Automated Deduction in Geometry*, volume 4869 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2007.
9. Émile Lemoine. *Géométrographie ou Art des constructions géométriques*. Number 18 in Scientia, Série Physico-Mathématique. C. Naud, Éditeur, Paris, Février 1902.
10. John Sturgeon Mackay. The geometrography of Euclid's problems. *Proceedings of the Edinburgh Mathematical Society*, 12:2–16, 1893.
11. Jorma K. Merikoski and Timo Tossavainen. Two approaches to geometrography. *Journal for Geometry and Graphics*, 13(1):15–28, 2010.
12. Pedro Quaresma. Thousands of Geometric problems for geometric Theorem Provers (TGTP). In Pascal Schreck, Julien Narboux, and Jürgen Richter-Gebert, editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*, pages 169–181. Springer, 2011.
13. Pedro Quaresma and Pierluigi Graziani. The geometrography's simplicity coeficient for the axioms and lemma of the area method. Technical Report TR 2021-001, Center for Informatics and Systems of the University of Coimbra, 2021.
14. Pedro Quaresma and Pierluigi Graziani. Measuring the readability of geometric proofs—the area method case. *Journal of Automated Reasoning*, 67(1), jan 2023. https://rdcu.be/c3abe.

15. Pedro Quaresma, Vanda Santos, Pierluigi Graziani, and Nuno Baeta. Taxonomy of geometric problems. *Journal of Symbolic Computation*, 97:31–55, 2020.
16. Vanda Santos, Nuno Baeta, and Pedro Quaresma. Geometrography in dynamic geometry. *International Journal for Technology in Mathematics Education*, 26(2):89–96, 2019.
17. Rüdger Thiele. Hilbert's twenty-fourth problem. *The American Mathematical Monthly*, 110(1):1, jan 2003.
18. Freek Wiedijk. The de Bruijn factor. Poster at International Conference on Theorem Proving in Higher Order Logics (TPHOL2000), 2000. Portland, Oregon, USA, 14-18 August 2000.