EPTCS 398

Proceedings of the

14th International Conference on Automated Deduction in Geometry

Belgrade, Serbia, 20-22th September 2023

Edited by: Pedro Quaresma and Zoltán Kovács

Published: 22nd January 2024
DOI: 10.4204/EPTCS.398
ISSN: 2075-2180
Open Publishing Association

Table of Contents

Preface

Automated Deduction in Geometry (ADG) is a forum to exchange ideas and views, to present research results and progress, and to demonstrate software tools at the intersection between geometry and automated deduction. Relevant topics include (but are not limited to): polynomial algebra, invariant and coordinate-free methods; probabilistic, synthetic, and logic approaches, techniques for automated geometric reasoning from discrete mathematics, combinatorics, and numerics; interactive theorem proving in geometry; symbolic and numeric methods for geometric computation, geometric constraint solving, automated generation/reasoning and manipulation with diagrams; design and implementation of geometry software, automated theorem provers, special-purpose tools, experimental studies; applications in mechanics, geometric modelling, CAGD/CAD, computer vision, robotics, and education.

Traditionally, the conference is held every two years. The previous editions of ADG were held in Hagenberg 2021, postponed from 2020, and held online due to the COVID-19 pandemic, Nanning in 2018, Strasbourg in 2016, Coimbra in 2014, Edinburgh in 2012, Munich in 2010, Shanghai in 2008, Pontevedra in 2006, Gainesville in 2004, Hagenberg in 2002, Zurich in 2000, Beijing in 1998, and Toulouse in 1996.

The 14th edition, ADG 2023, was held in Belgrade, Serbia, in September 20-22, 2023 (`https://adg2023.matf.bg.ac.rs/`). This edition of ADG had an additional special focus topic: Deduction in Education.

The invited speakers at ADG 2023 were:

Julien Narboux, University of Strasbourg, France

Filip Marić, University of Belgrade, Serbia

Zlatan Magajna, University of Ljubljana, Slovenia

The quality of research articles submitted for this proceedings of ADG was very high. Out of seventeen full-papers submitted, fifteen were accepted. Each submission was carefully reviewed by, at least, three reviewers. Therefore this volume consists of fifteen articles, bringing exciting new ideas, spanning various areas of automated deduction in geometry, and showing the current state-of-the-art research in this field.

The conference programme and this volume can be roughly divided into 3 parts:

Automated and interactive theorem proving in geometry The effort of formalise many areas of mathematics using deduction tools such as proof assistants is a huge undertaking, the area of geometry is part of that effort. The two invited talks, by Julien Narbox, *Formalisation, arithmetization and automatisation of geometry* and by Filip Marić, *Automatization, formalization and visualization of hyperbolic geometry* gave a very good account of those efforts with the second also introducing an important subject, somehow less common to see, of the non-Euclidean geometries.

The subject of readability of the proofs when automated or interactive theorem prover are used is addressed in several contributions. Nicolas Magaud addresses the problem by presenting a framework to carry out a posteriori script transformations of *Coq* proof sxcripts. Vesna Marinković et al. demonstrate how their triangle construction solver *ArgoTriCS* can cooperate with automated theorem provers for first order logic and coherent logic so that it generates construction correctness proofs, that are both human-readable and formal. Salwa Tabet Gonzalez et al. present a framework for completing incomplete conjectures and incomplete proofs. The proposed framework can also help in completing a proof sketch into a human-readable and machine-check-able proofs.

The systems combining dynamic geometry and automated deduction are very important, when the aim is to reach a larger audience than the automated deduction experts. A first paper by Zoltán Kovács et al. describes some on-going improvements concerning the automated reasoning tools developed in *GeoGebra Discovery*. A second paper by Zoltán Kovács et al. describes the efforts that are being made to resume the development of *JGEX*, a system that combines the dynamic geometry system with several automated theorem provers and with natural language, with links to visual animations, human-readable proofs.

Milan Banković presents an approach to automated solving of triangle ruler-and-compass construction problems using finite-domain constraint solvers. Pierre Boutry et al. present their progress towards obtaining an independent version of Tarski's System of Geometry, in the form of a variant of Gupta's system. Pedro Quaresma et al. make some considerations on Approaches and Metrics in Automated Theorem Generation/Finding in Geometry, addressing Wos' Problem 31, about the properties that can be identified to permit an automated reasoning program to find new and interesting theorems.

The special focus topic Deduction in Education This special focus topic addresses the several difficulties, or should we say, opportunities, that must be addressed to open the field of automated deduction to a wider audience, namely to the education community. The talks were all around the tools and problems crossing dynamic geometry systems and automated deduction. In the invited talk by Zlatan Magajna, *OK Geometry*, he presents a tool for analysing dynamic geometric constructions, observing invariants of dynamic geometric constructions, and generating conjectures. Philip Todd and Danny Aley present *GXWeb* in the context of proving and mathematical modelling, Zoltán Kovács and other researchers use *GeoGebra*, *GeoGebra Discovery* and *JGEX* to address several problems in mathematical competitions. In two *STEAM* activities, Anna Käferböck and Zoltán Kovács present an activity, finding the locus of a rocking camel and Thierry Dana-Picard et al. present the visualisation of 2D and 3D curves, with various technologies to use the motivational fascination of outer space from students to connect to mathematical modelling.

Algebraic methods in automated reasoning in geometry Philip Todd examine a class of geometric theorems on cyclic 2n-gons, proving that if n disjoint pairs of sides are taken, each pair separated by an even number of polygon sides, then there is a linear combination of the angles between those sides which is constant. Hoon Hong et al. present a piece-wise rational reparameterization of curves obtaining a more uniform angular speed obtaining a better rational parameterization of those curves.

The quality of this proceedings is due to the invited lecturers and the authors of submitted papers, but also to the reviewers, the members of the program committee, and all the organisers of ADG 2023.

General Chair Predrag Janičić (University of Belgrade, Serbia)

Program Committee

Francisco Botana (University of Vigo, Spain)
Xiaoyu Chen (Beihang University, China)
Thierry Dana-Picard (Jerusalem College of Technology, Israel)

We would like to express our gratitude to all participants at ADG 2023 and to all those that helped the organisation of ADG 2023 in some way. We would also like to thank the EPTCS staff for their support and efficiency.

January 2024
Pedro Quaresma, Zoltán Kovács
ADG 2023 Program Committee Chairs

Formalization, Arithmetization and Automatization of Geometry*

Julien Narboux
University of Strasbourg, France
jnarboux@narboux.fr

In this talk we will present an overview of our work (with Michael Beeson, Pierre Boutry, Gabriel Braun and Charly Gries) about formalization and automation of geometry and the GeoCoq library. We will delve into the axiomatic systems of influential mathematicians such as Euclid, Hilbert and Tarski and present formalization issues. A special focus will be placed on the formalization of continuity axioms and parallel postulates.

We will highlight details and issues that can be seen only through the lens of a proof assistant and intuitionist logic. We will present a syntactic proof of the independence of the parallel postulate.

From axiomatic foundations to computer-assisted proofs, we will explore the intricate interplay between synthetic and analytic geometry, and different kinds of automation.

*Slides of the presentation: `https://adg2023.matf.bg.ac.rs/downloads/slides/InvitedTalkNarboux.pdf`

Formalization, Automatization and Visualization of Hyperbolic Geometry[*]

Filip Marić
Faculty of Mathematics, University of Belgrade, Serbia
filip@matf.bg.ac.rs

In this talk we describe our experiences with the formalization, automation and visualization of non-Euclidean geometries.

We start with a formalization of the complex projective line CP(1) (also known as the extended complex plane), its objects (points and circlines) and transformations (Möbius transformations). An algebraic approach is used, where points are described with homogeneous coordinates, circlines are described with Hermitean matrices and Möbius transformations are described using regular matrices. We use the unit disk in CP(1) for the formalization of the Poincaré disk model of hyperbolic geometry and show that it satisfies all Tarski axioms of geometry (with the negated Euclidean axiom).

We also analyze the problem of automatic construction of triangles in absolute and hyperbolic geometry. For this purpose we adapt the software *ArgoTriCS*. For the visualization of the generated constructions we use *ArgoDG*: a lightweight JavaScript library for dynamic geometry.

Finally, we introduce the formalization of gyrogroups and gyrovector spaces introduced by Abraham Ungar, which is an alternative approach to formalize hyperbolic geometry inspired by Einstein's special theory of relativity.

[*]Slides of the presentation: `https://adg2023.matf.bg.ac.rs/downloads/slides/FormalizationFilip.pdf`

P. Quaresma and Z. Kovács (Ed.): Automated Deduction in Geometry 2023 (ADG 2023).
EPTCS 398, 2024, pp. 2–2, doi:10.4204/EPTCS.398.2

OK Geometry[*]

Zlatan Magajna
University of Ljubljana, Slovenia:
Zlatan.Magajna@pef.uni-lj.si

Automated observation of dynamic constructions is based on numerical checks of a variety of geometric properties performed on multiple instances of dynamic constructions, where all free points are continuously dragged. Unlike most dynamic geometry systems that incorporate some elements of observation, *OK Geometry* is a tool developed specifically for the purpose of automated observation of dynamic constructions. The goal of the observations is not to prove facts, but to generate plausible hypotheses about the properties of the observed construction.

In the presentation, we focused on two aspects of observation in addition to the very concept of automated observation. Firstly, we considered the necessary functionalities of software for automated observation. The functionalities range from the possibility of importing dynamic constructions from different dynamic systems, to the management of a database of geometric facts, objects and properties to be considered during the observation, from the possibility of creating implicitly defined geometric objects to the importance of controlling computational errors in numerical checks.

The second focus of the presentation was on the relationship between automated observation, proving and automated provers. Automated observation can be helpful in proving geometric facts as it brings to light properties of a configuration that one may not be aware of. In classroom practice, however, students have difficulty selecting the relevant facts among those automatically observed. It seems that automated observation is particularly useful in combination with automated provers when exploring geometric configurations. Automated observation generates (many) plausible hypotheses about the properties of a geometric configuration that can be proved with a prover.

[*]Slides of the presentation: `https://adg2023.matf.bg.ac.rs/downloads/slides/OKGeometryMagajna.pdf`

P. Quaresma and Z. Kovács (Ed.): Automated Deduction
in Geometry 2023 (ADG 2023).
EPTCS 398, 2024, pp. 3–3, doi:10.4204/EPTCS.398.3

Towards Automatic Transformations of Coq Proof Scripts

Nicolas Magaud
Lab. ICube UMR 7357 CNRS Université de Strasbourg, France
magaud@unistra.fr

Proof assistants like Coq are increasingly popular to help mathematicians carry out proofs of the results they conjecture. However, formal proofs remain highly technical and are especially difficult to reuse. In this paper, we present a framework to carry out *a posteriori* script transformations. These transformations are meant to be applied as an automated post-processing step, once the proof has been completed. As an example, we present a transformation which takes an arbitrary large proof script and produces an equivalent single-line proof script, which can be executed by Coq in one single step. Other applications, such as fully expanding a proof script (for debugging purposes), removing all named hypotheses, etc. could be developed within this framework. We apply our tool to various Coq proof scripts, including some from the GeoCoq library.

1 Motivations

Proof assistants like Coq [1, 4] are increasingly popular to help mathematicians carry out proofs of the results they conjecture. However, formal proofs remain highly technical and are especially difficult to reuse. Once the proof effort is done, the proof scripts are left as they are and they often break when upgrading to a more recent version of the prover. To reduce the burden of maintaining the proof scripts of Coq, we propose a tool to post-process the proof scripts to make them cleaner and easier to reuse. The first transformation that we focused on consists in compacting a several-step proof script into a single-step proof script. Even though our framework can be used to implement other proof script transformations, this one is of special interest to us. Indeed, we recently designed a prover for projective incidence geometry [3, 12] which relies on the concept of rank to carry out proofs of geometric theorems such as Desargues or Dandelin-Gallucci automatically. This prover produces a trace (a large Coq proof script containing several statements and their proofs). We hope to use the proof transformation tool to shape up the automatically generated proofs and make them easier to reuse and integrate in larger proof repositories.

More generally, proof maintenance and reuse tools have been studied extensively by Talia Ringer et al. [11, 10]. Contrary to our approach, their tools aim at fixing the issues when they occur. In our setting, we think it is better to try and improve the proof scripts so that they are less likely to break, even after several years and numerous updates of the components.

Outline of the paper The paper is organized as follows. In Sect. 2, we present a simple example of a proof script transformation. In Sect. 3, we describe the implementation of our tool as well as the future extensions we currently develop. In Sect. 4, we present some concluding remarks and the perspectives of this work.

P. Quaresma and Z. Kovács (Ed.): Automated Deduction
in Geometry 2023 (ADG 2023).
EPTCS 398, 2024, pp. 4–10, doi:10.4204/EPTCS.398.4

```
Lemma foo : forall A B C : Prop, A ∨ (B ∧ C) → (A∨ B)∧ (A∨ C).
```

```
Proof.
  intros; destruct H.
  split.
  left; assumption.
  left; assumption.
  destruct H.
  split.
  right; assumption.
  right; assumption.
  Qed.
```

```
Proof.
intros; destruct H;
  [ split;
    [ left; assumption
    | left; assumption ]
  | destruct H ;
    split;
    [ right; assumption
    | right; assumption ] ].
Qed.
```

Figure 1: A user-written script (left) and the equivalent single-step script (right)

2 Transforming Large Proof Scripts into One-line Scripts

The Coq tactic language [5] features tacticals to execute some tactics in a sequence `tac1;tac2;tac3` or to try and execute different tactics on the same goal `solve [ tac1 | tac2 | tac3 ]`. Moreover these tacticals can be combined. E.g. `tac0 ; [tac1 | tac2 | tac3]` runs the first tactic `tac0` which should yield 3 subgoals. The first one is solved using `tac1`, the second one using `tac2` and the third one using `tac3`. Once a proof script is written (as several steps) by the user, we can use these tacticals to build an equivalent proof script, which can be executed in a single step.

Let us consider a simple example, proving the distributivity of the connective or ($\vee$) over the connective and ($\wedge$) as shown in the statement of figure 1.

The left-hand side presents the proof script that one may expect from a master student, factorizing some parts but still decomposing the reasoning in several steps. On the right-hand side, we propose a one-line script to carry out exactly the same proof.

In Coq, writing directly the right-hand side is almost impossible, whereas it is fairly easy to generate it automatically from the left-hand side. In the Coq standard library, several lemmas are proved using a single one-line tactic. The main advantage is that it provides concise and structured proofs but it has the drawback that, when something goes wrong, it is hard to debug and fix it.

3 Experiments, Limitations and Results

3.1 Implementation

We choose to implement our tool in `OCaml`, using the serialisation mechanism `serapi` [6] developed by Emilio Gallego Arias for communication with the Coq proof assistant. Our tool uses anonymous pipes to communicate with `serapi`, which itself sends requests to Coq and retrieves the answers. Commands are kept as in the input file. Tactics are aggregated using tacticals such as `;`, `[` and `]`. At each step of the proof, we compare the current number of subgoals to the number of subgoals right before the execution of the current tactic. If it is the same, we simply concatenate the tactics with a `;` between them. If the number of goals increases, we open a square bracket `[` and push into the stack the previous number of goals. Each time a goal is solved, we check whether some goals remain to be proved at this level. If yes, we add another `;` and then focus on the next subgoal. If there are no more subgoals at this level, we pop the 0 from the top of the stack, thus closing the current level with a `]` and carry on with subgoals of the previous level.

The source code[1] as well as some examples are freely available online. It is developped using `Coq` 8.17.0 and the corresponding `serapi` version 8.17.0+0.17.0.

3.2 Limitations

So far, commands and tactics are told apart simply by assuming commands start with a capital letter [A-Z] and tactics with a small letter [a-z]. This convention is well-known in Coq, however in some developments (e.g. GeoCoq), some ad-hoc user tactics may start with a capital letter. Handling this properly requires additional developments and is currently under way.

To make the transformation easier, a first phase could be added to our proof script transformer to remove all commands which lay among the proof steps (e.g. `Check`, `Print` or `Locate`) and make sure all tactics names start with a small letter.

Finally, Coq proof scripts can be structured using bullets (`+`, `-`, `*`) as well as curly brackets to identify some subproofs. In addition, one can direct work on a goal which is not the current one using the `2: tac.` notation which performs the tactic `tac` on the second goal of the subgoals. We still need to devise a way to deal properly with such partially-structured proof script.

3.3 Successful Transformations

In addition to our test suite examples, we consider more challenging proof scripts. We successfully transformed a library file from the Standard Library of Coq: Cantor.v[2] from the `Arith` library as well as some large files from the `GeoCoq` library [2, 8] (e.g. `orthocenter.v`[3]). As the tool gets more mature, we plan to transform more files, and we shall especially focus on the `GeoCoq` library which features several different proof styles and thus shall allow us to evaluate the robustness of our tool.

3.4 Refactoring Proof Scripts Automatically Generated by our Prover for Projective Incidence Geometry

We recently developed a new way [3, 12], based on ranks, to automatically prove statements in projective incidence geometry. Our approach works well but produces proof scripts which are very large and often feature several auxiliary lemmas. Figure 2 presents a very simple example `LABC`, which is formally proven by our tool but yields a fairly verbose proof script using one intermediate lemma `LABCD` (see appendix A for details).

We plan to use our script transformation tool to refactor automatically generated proof scripts, inlining auxiliary lemmas and thus making proof scripts more concise and hopefully more readable for humans.

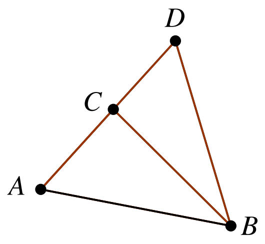
3.5 Next steps

To fully evaluate the tool, we need to handle larger examples, outside of the standard library of Coq. The next step consists in improving Coq options handling (e.g. `-R`) to our script transformation tool to tackle other formal proof libraries.

[1] `https://github.com/magaud/coq-lint`

[2] `https://github.com/coq/coq/blob/master/theories/Arith/Cantor.v`

[3] `https://github.com/GeoCoq/GeoCoq/blob/master/Highschool/orthocenter.v`



- Informal statement
  Assume that ABD is a triangle,
  Assume that C is a point on AD, such that C≠ A and C≠ D,
  Then ABC is a triangle.

- Expressed using ranks

$$\forall A,B,C,D : \texttt{Point}, \\ rk\{A,D,B\} = 3 \rightarrow rk\{A,C,D\} = 2 \rightarrow \\ rk\{C,A\} = 2 \rightarrow rk\{C,D\} = 2 \rightarrow \\ rk\{A,C,B\} = 3.$$

Figure 2: An example of a statement in projective geometry, formalized using ranks

We also plan to propose the reciprocal script transformation, turning a single-step proof script into a more detailed (easier to debug) proof script. This could especially be useful when porting formal proofs from one version of Coq to the next one.

Other applications of interest could be to remove the names of all variables or hypotheses from the scripts, or at least to force them to be explicitly introduced. The script snippet `intros; apply H` could be replaced by a more precise one `intros n p H; apply H`. This way, we could ensure that the proofs are not broken when the names of automatic variables change. From a reliability point of view, it would be even better to use the tactic `intros; assumption`. Although its cost is higher (because we need to search the correct hypothesis among all of them every time we run the tactic), it does not depend on some arbitrary variable names.

Finally, regarding our current implementation, it would be interesting to benchmark the transformation to see whether transforming the whole standard library of Coq into single-step proof scripts could improve the compilation time of this library.

4 Conclusions and perspectives

We build a proof script transformation tool, which transforms an arbitrary large proof script into a single-step *one-Coq-tactic* proof script. This tool has been successfully experimented on some significant library files from the Coq ecosystem.

This first example shows that the approach is sound and we plan to extend it to integrate tactic languages such as ssreflect [7], Ltac2 [9] or Mtac [13] in our framework. In the longer term, we expect to design some new proof script transformations and combine them in order to build more reliable proof developments which can last longer and would be easier to maintain. Among these transformations, we shall start with a mechanism to transform a proof script into a sequence of atomic proof steps (to make debugging easier when the proof breaks). We may also study how to transform proofs carried out automatically by their actual traces, avoiding recomputing the proof search each time the proof is re-runned.

References

[1] Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development, Coq'Art : The Calculus of Inductive Constructions. Texts in Theoretical Computer Science, An EATCS Series, Springer-Verlag, Berlin/Heidelberg (May 2004), 469 pages

[2] Boutry, P., Gries, C., Narboux, J., Schreck, P.: Parallel postulates and continuity axioms: a mechanized study in intuitionistic logic using Coq. Journal of Automated Reasoning p. 68 (2017). https://doi.org/10.1007/s10817-017-9422-8

[3] Braun, D., Magaud, N., Schreck, P.: Two new ways to formally prove dandelin-gallucci's theorem. In: Chyzak, F., Labahn, G. (eds.) ISSAC '21: International Symposium on Symbolic and Algebraic Computation, Virtual Event, Russia, July 18-23, 2021. pp. 59–66. ACM (2021). https://doi.org/10.1145/3452143.3465550

[4] Coq development team: The Coq Proof Assistant Reference Manual, Version 8.13.2. INRIA (2021), `http://coq.inria.fr`

[5] Delahaye, D.: A Tactic Language for the System Coq. In: Parigot, M., Voronkov, A. (eds.) Logic for Programming and Automated Reasoning, 7th International Conference, LPAR 2000, Reunion Island, France, November 11-12, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1955, pp. 85–95. Springer (2000). https://doi.org/10.1007/3-540-44404-1_7

[6] Gallego Arias, E.J.: SerAPI: Machine-Friendly, Data-Centric Serialization for Coq. Tech. rep., MINES ParisTech (Oct 2016), `https://hal-mines-paristech.archives-ouvertes.fr/hal-01384408`

[7] Gonthier, G., Mahboubi, A.: An introduction to small scale reflection in Coq. J. Formaliz. Reason. **3**(2), 95–152 (2010). https://doi.org/10.6092/issn.1972-5787/1979

[8] Narboux, J.: Mechanical Theorem Proving in Tarski's geometry. In: Eugenio Roanes Lozano, F.B. (ed.) Automated Deduction in Geometry 2006. LNCS, vol. 4869, pp. 139–156. Francisco Botana, Springer, Pontevedra, Spain (Aug 2006). https://doi.org/10.1007/978-3-540-77356-6

[9] Pédrot, P.M.: Ltac2: Tactical Warfare. In: Krebbers, R., Sergey, I. (eds.) Proceedings of the CoqPL workshop 2019 (2019)

[10] Ringer, T., Palmskog, K., Sergey, I., Gligoric, M., Tatlock, Z.: QED at large: A survey of engineering of formally verified software. Found. Trends Program. Lang. **5**(2-3), 102–281 (2019). https://doi.org/10.1561/2500000045

[11] Ringer, T., Yazdani, N., Leo, J., Grossman, D.: Adapting proof automation to adapt proofs. In: Andronick, J., Felty, A.P. (eds.) Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, January 8-9, 2018. pp. 115–129. ACM (2018). https://doi.org/10.1145/3167094

[12] Schreck, P., Magaud, N., Braun, D.: Mechanization of incidence projective geometry in higher dimensions, a combinatorial approach. In: Janicic, P., Kovács, Z. (eds.) Proceedings of the 13th International Conference on Automated Deduction in Geometry, ADG 2021, Hagenberg, Austria/virtual, September 15-17, 2021. EPTCS, vol. 352, pp. 77–90 (2021). https://doi.org/10.4204/EPTCS.352.8

[13] Ziliani, B., Dreyer, D., Krishnaswami, N.R., Nanevski, A., Vafeiadis, V.: Mtac: A monad for typed tactic programming in coq. J. Funct. Program. **25** (2015). https://doi.org/10.1017/S0956796815000118

A Proof Script for our basic example

```
Lemma LABCD : forall A B C D ,
rk(A:: C:: nil) = 2 → rk(A:: B:: D:: nil) = 3 →
rk(C:: D:: nil) = 2 → rk(A:: C:: D:: nil) = 2 →
rk(A:: B:: C:: D:: nil) = 3.
```

```
Proof.
intros A B C D
HACeq HABDeq HCDeq HACDeq .
assert(HABCDm2 : rk(A:: B:: C:: D:: nil) >= 2).
{
   assert(HACmtmp : rk(A:: C:: nil) >= 2)
          by (solve_hyps_min HACeq HACm2).
   assert(Hcomp : 2 <= 2) by (repeat constructor).
   assert(Hincl : incl (A:: C:: nil) (A:: B:: C:: D:: nil))
          by (repeat clear_all_rk;my_inO).
   apply (rule_5 (A:: C:: nil) (A:: B:: C:: D:: nil) 2 2 HACmtmp Hcomp Hincl).
}
assert(HABCDm3 : rk(A:: B:: C:: D:: nil) >= 3).
{
   assert(HABDmtmp : rk(A:: B:: D:: nil) >= 3)
          by (solve_hyps_min HABDeq HABDm3).
   assert(Hcomp : 3 <= 3)
          by (repeat constructor).
   assert(Hincl : incl (A:: B:: D:: nil) (A:: B:: C:: D:: nil))
          by (repeat clear_all_rk;my_inO).
   apply (
     rule_5 (A:: B:: D:: nil) (A:: B:: C:: D:: nil) 3 3 HABDmtmp Hcomp Hincl
         ).
}
assert(HABCDM : rk(A:: B:: C:: D:: nil) <= 3)
       by (solve_hyps_max HABCDeq HABCDM3).
assert(HABCDm : rk(A:: B:: C:: D:: nil) >= 1)
       by (solve_hyps_min HABCDeq HABCDm1).
intuition.
Qed.

Lemma LABC : forall A B C D ,
rk(A:: C:: nil) = 2 → rk(A:: B:: D:: nil) = 3 →
rk(C:: D:: nil) = 2 → rk(A:: C:: D:: nil) = 2 →
rk(A:: B:: C:: nil) = 3.
Proof.
intros A B C D
HACeq HABDeq HCDeq HACDeq .

assert(HABCm2 : rk(A:: B:: C:: nil) >= 2).
{
   assert(HACmtmp : rk(A:: C:: nil) >= 2)
         by (solve_hyps_min HACeq HACm2).
   assert(Hcomp : 2 <= 2)
        by (repeat constructor).
   assert(Hincl : incl (A:: C:: nil) (A:: B:: C:: nil))
        by (repeat clear_all_rk;my_inO).
   apply (
     rule_5 (A:: C:: nil) (A:: B:: C:: nil) 2 2 HACmtmp Hcomp Hincl
         ).
}
assert(HABCm3 : rk(A:: B:: C:: nil) >= 3).
```

```
{
   assert(HACDMtmp : rk(A:: C::  D::  nil) <= 2)
          by (solve_hyps_max HACDeq HACDM2).
   assert(HABCDeq : rk(A:: B::  C::  D::  nil) = 3)
          by
          (apply LABCD with (A := A) (B := B) (C := C) (D := D) ; assumption).
   assert(HABCDmtmp : rk(A:: B::  C::  D::  nil) >= 3)
          by (solve_hyps_min HABCDeq HABCDm3).
   assert(HACmtmp : rk(A:: C::  nil) >= 2)
          by (solve_hyps_min HACeq HACm2).
   assert( Hincl :
            incl (A::  C::  nil)
            (list_inter (A:: B::  C::  nil) (A::  C::  D::  nil)))
          by (repeat clear_all_rk;my_inO).
   assert( HT1 :
            equivlist (A:: B::  C::  D::  nil)
            (A::  B::  C::  A::  C::  D::  nil))
          by (clear_all_rk;my_inO).
   assert( HT2 :
           equivlist (A:: B::  C::  A::  C::  D::  nil)
           (( A::  B::  C::  nil) ++ (A::  C::  D::  nil))
         )  by (clear_all_rk;my_inO).
   rewrite HT1 in HABCDmtmp;rewrite HT2 in HABCDmtmp.
   apply (
     rule_2
        (A::  B::  C::  nil) (A::  C::  D::  nil) (A::  C::  nil)
        3  2  2 HABCDmtmp HACmtmp HACDMtmp Hincl
        ).
}
assert(HABCM : rk(A::  B::  C:: nil) <= 3)
       by (solve_hyps_max HABCeq HABCM3).
assert(HABCm : rk(A::  B::  C:: nil) >= 1)
       by (solve_hyps_min HABCeq HABCm1).
intuition.
Qed.
```

Towards Automated Readable Proofs of Ruler and Compass Constructions

Vesna Marinković
Faculty of Mathematics, University of Belgrade
vesna.marinkovic@matf.bg.ac.rs

Tijana Šukilović
Faculty of Mathematics, University of Belgrade
tijana.sukilovic@matf.bg.ac.rs

Filip Marić
Faculty of Mathematics, University of Belgrade
filip.maric@matf.bg.ac.rs

Although there are several systems that successfully generate construction steps for ruler and compass construction problems, none of them provides readable synthetic correctness proofs for generated constructions. In the present work, we demonstrate how our triangle construction solver ArgoTriCS can cooperate with automated theorem provers for first order logic and coherent logic so that it generates construction correctness proofs, that are both human-readable and formal (can be checked by interactive theorem provers such as Coq or Isabelle/HOL). These proofs currently rely on many high-level lemmas and our goal is to have them all formally shown from the basic axioms of geometry.

1 Introduction

Geometry construction problems are usually solved in four phases:

1. *Analysis*: In this phase, the geometric figure to be constructed is analyzed. The specific constraints that apply to this figure and the relationships between its elements are identified. By understanding the requirements and constraints, the steps required to construct the desired figure can be determined.
2. *Construction*: Once the problem is analyzed, the sequence of ruler and compass construction steps used to construct the figure is identified.
3. *Proof*: After the figure is constructed, it should be proved that it satisfies the properties and conditions given by the specification. Proofs in ruler and compass constructions often involve using geometric principles, such as the properties of angles, congruence, or similarity. A formal proof can be used to demonstrate the validity of the construction and ensure that it meets the desired criteria.
4. *Discussion*: The discussion phase involves reflection on the construction, its properties, and the relevant insights. It is often discussed under which condition does the solution exist and whether it is unique. Non-degeneracy conditions are also identified.

In our previous work we have described our system ArgoTriCS that can perform triangle constructions both in Euclidean geometry [6] and in absolute and hyperbolic geometry [8]. Problems from the Wernick's list [10] are analyzed and in Euclidean setting ArgoTriCS manages to solve 66 out of 74 non-isomorphic problems. Essentially it performs the problem analysis based on its internal set of definitions and lemmas, and finds a series of construction steps required to construct a triangle with a given set of significant points (e.g., vertices, orthocenter, centroid, centers of inscribed and circumscribed circles etc.). However it did not generate classic, readable, synthetic construction proofs. In her PhD thesis [7], Marinković describes how theorem provers, based on algebraic methods such as Wu's method [11] and Gröbner basis method [1], and semi-synthetic methods such as area method [4], integrated within GLCL tool [2] and OpenGeoProver [5], could be employed to check the construction correctness. The problem with this approach is that generated proofs are not human-readable. Since the main usage scenario of automated construction solver is in education, it is vital that students understand why some construction is correct. Therefore, obtaining human-readable proofs is of a great importance.

In the current work, we describe how an automated system such as ArgoTriCS can be combined with first-order logic and coherent logic provers so that each generated construction is accompanied by its human-readable proof of correctness. This is a work in progress, and we will describe our approach, prototype implementation, and preliminary results for a small set of selected problems.

2 Examples

Example 2.1. Consider constructing a triangle ABC given its vertex A, altitude foot H_a and circumcenter O. ArgoTriCS finds the following construction, illustrated in Figure 1:

1. Construct the line $l_1 = AH_a$.
2. Construct the line l_2 such that it is perpendicular to the line l_1 and that it contains H_a.
3. Construct the circle c centered at O containing A.
4. Let B and C be the intersections of the line l_2 and the circle c.

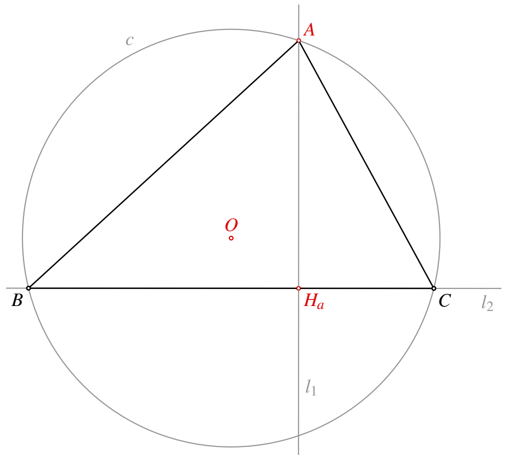


Fig. 1: Construction of the triangle ABC given the points A, O, and H_a.

Proof. We need to show that A is the vertex of the constructed triangle ABC (which is trivial), that H_a is its altitude foot and that O is its circumcenter. This proof is rather straightforward.

By construction, the circle c contains all three vertices A, B, and C, so it must be the circumcircle of the triangle ABC (since the circumcircle of a triangle is unique). The O is the center of c, so it must be the circumcenter (since the center of any circle is unique).

By construction the line l_2 contains the vertices B and C, so it must be equal to the side a of the triangle ABC (since the triangle side through the points B and C is unique). By construction the line l_1 contains A and is perpendicular to $l_2 = a$, so it must be equal to the altitude h_a (since there is a unique altitude from the vertex A). Since by construction H_a belongs both to $l_2 = a$ and $l_1 = h_a$ it must be the altitude foot H_a (since it is the unique intersection of a and h_a). □

If we analyze the previous proof, we see that it essentially relies on several uniqueness lemmas and that it merely reverses the chain of deduction steps used in the analysis phase.

In some cases, however, the proof is very different from the analysis.

Example 2.2. Consider constructing a triangle ABC given its vertex A, circumcenter O and centroid G. The construction that ArgoTriCS finds is the following (see Figure 2):

1. Construct the point P_1 such that $\overrightarrow{AG} : \overrightarrow{AP_1} = 2 : 3$.
2. Construct the point P_2 such that $\overrightarrow{OG} : \overrightarrow{OP_2} = 1 : 3$.
3. Construct the line $l_1 = AP_2$.
4. Construct the line l_2 such that it is perpendicular to the line l_1 and that it contains P_1.
5. Construct the circle c centered at O containing A.
6. Let B and C be the intersections of the line l_2 with the circle c.

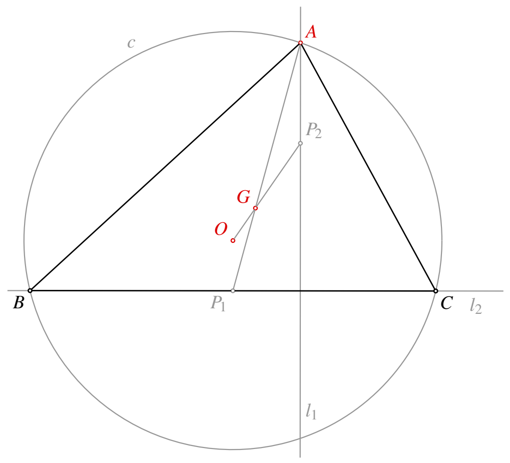


Fig. 2: Construction of the triangle ABC given the points A, O, and G.

Please note that there is a simpler solution to this construction problem, but we wanted to discuss this solution because the proof here is quite different from the construction.

Proof. We need to prove that A is the vertex of the triangle ABC (which is trivial), that G is its centroid and that O is its circumcenter. The latter is very simple (similar to the previous proof), since by construction all points A, B, and C lie on the circle c centered at O.

The line l_2 is equal to the triangle side a, since it contains the vertices B and C (and the triangle side through the points B and C is unique). By construction l_1 contains A and is perpendicular to $l_2 = a$, so it must be equal to the altitude h_a (since the altitude from vertex A is unique).

Consider line $l_3 = OP_1$. We shall prove that it is parallel to the line $l_1 = h_a$. Since by construction it holds that $\overrightarrow{OG} : \overrightarrow{OP_2} = 1 : 3$, by the elementary properties of vector ratio it also holds that $\overrightarrow{OG} : \overrightarrow{GP_2} = 1 : 2$. Similarly, it holds that $\overrightarrow{P_1G} : \overrightarrow{GA} = 1 : 2$. The angles OGP_1 and OGP_2 are opposite and therefore congruent. Hence triangles OGP_1 and P_2GA are similar, and angles OP_1G and GAP_2 are always equal, so the lines $OP_1 = l_3$ and $AP_2 = l_1 = h_a$ are parallel.

Since h_a is perpendicular to $l_2 = a$, so must be $l_3 = OP_1$. Therefore, the line l_3 must be the perpendicular bisector of the segment BC (since it is the unique line containing circumcenter O that is perpendicular to a). Consequently, the point P_1 must be equal to M_a – the midpoint of BC (as it is the unique intersection of the segment with its pependicular bisector). Finally, the point G must be the centroid of ABC since the centroid is the unique point for which it holds that $\overrightarrow{AG} : \overrightarrow{AM_a} = 2 : 3$. $\square$

3 Automation

Our main goal is to obtain proofs such as the previous ones automatically, using coherent logic provers.

3.1 Problem Statement and Lemmas

The first step would be to make ArgoTriCS generate the problem statement, along with the construction steps. For example, the problem statement for the first problem can be given as follows:

$$\begin{aligned}
&\mathrm{inc}(A, l_1) \land \mathrm{inc}(H'_a, l_1) \land \\
&\mathrm{perp}(l_2, l_1) \land \mathrm{inc}(H'_a, l_2) \land \\
&\mathrm{circle}(O', A, c) \land \\
&\mathrm{inc}(B, l_2) \land \mathrm{inc}(C, l_2) \land \mathrm{inc\_c}(B, c) \land \mathrm{inc\_c}(C, c) \land B \neq C \Longrightarrow \\
&H'_a = H_a \land O' = O
\end{aligned}$$

The predicate $\mathrm{inc}(P,l)$ denotes that the point P is incident to the line l i.e., $P \in l$, $\mathrm{inc\_c}(P,c)$ denotes that the point P is incident to the circle c i.e., $P \in c$, $\mathrm{circle}(O,P,c)$ denotes that c is the circle centered at the point O passing through the point P, and $\mathrm{perp}(l_1,l_2)$ denotes that lines l_1 and l_2 are perpendicular. The point O is the real circumcenter of the triangle ABC (this is implicitly given by the lemmas that are given to the prover along with the problem statement), and H_a is the real altitude foot. For simplicity various non-degeneracy conditions are added to the problem statement (e.g., the conditions $H'_a \neq A$, $A \neq B$, $A \neq C$, etc.) before it is given to the automated theorem prover.

Along with the problem statement, automated prover is given a series of carefully chosen lemmas, that are treated as axioms. Most of those lemmas follow from the general geometric knowledge, but are instantiated for the significant points, lines and circles of the triangle ABC. Each significant object is denoted by a constant (e.g., bc for the side BC, O for the circumcenter, M_a for the midpoint of BC, h_a for the altitude from A, H_a for its foot, c° for the circumcircle etc.). Lemmas that encode properties of those objects are used both in analysis (by the ArgoTriCS) and in proofs (by automated theorem provers). Some of those lemmas are:

$$\begin{array}{rcl}
\mathrm{inc}(B,bc) & \wedge & \mathrm{inc}(C,bc) \\
\mathrm{inc}(A,h_a) & \wedge & \mathrm{perp}(h_a,bc) \\
\overrightarrow{AG} : \overrightarrow{AM_a} & = & 2:3 \\
\mathrm{inc\_c}(A,c^\circ) & \wedge & \mathrm{inc\_c}(B,c^\circ) \wedge \mathrm{inc\_c}(C,c^\circ)
\end{array}$$

However, proofs require additional lemmas that guarantee uniqueness of objects. For example:

$$\begin{array}{rcl}
(\forall l)(\mathrm{inc}(A,l) \wedge \mathrm{perp}(l,bc) & \Longrightarrow & l = h_a) \\
(\forall c)(\mathrm{inc\_c}(A,c) \wedge \mathrm{inc\_c}(B,c) \wedge \mathrm{inc\_c}(C,c) & \Longrightarrow & c = c^\circ)
\end{array}$$

Notice that uniqueness lemmas are given in instantiated way, meaning that they hold for some specific objects. This choice was made in order to follow the implementation of ArgoTriCS, where most of the knowledge is given in an instantiated way. However, the uniqueness axioms could be given also in more general way.

Some general lemmas about properties of basic geometric predicates are also needed. For example:

$$\begin{array}{rcl}
(\forall l_1,l_2)(\mathrm{perp}(l_1,l_2) & \Longrightarrow & \mathrm{perp}(l_2,l_1)) \\
(\forall P_1,P_2)(\exists l)(\mathrm{inc}(P_1,l) & \wedge & \mathrm{inc}(P_2,l))
\end{array}$$

All those lemmas are formulated as axioms and the problem statement is formulated as a conjecture in TPTP format.[1] That file is then given to some automated theorem prover. In our experiments we used Vampire [9] and Larus [3]. Vampire is a very efficient, award winning FOL theorem prover. Its main drawback is that it cannot generate readable proofs. We also used Larus [3] that is a coherent-logic prover able to generate readable proofs and also formal proofs that can be checked by interactive theorem provers such as Isabelle/HOL or Coq.

Our second example uses the notion of ratio of vectors. However neither Vampire nor Larus have a native support for arithmetic calculations. Therefore we introduced separate predicates for ratios that frequently occur in geometric constructions (e.g., 1 : 2, 1 : 3, 2 : 3) and added lemmas that connect those ratios. For example:

$$(\forall A,B,C)(\mathrm{ratio13}(A,B,A,C) \Longrightarrow \mathrm{ratio12}(A,B,B,C))$$

The proof uses a result that follows from triangle similarity. We encoded this in the following lemma:

$$\begin{array}{l}
(\forall A,M,B,X,Y,ax,by) \\
(\mathrm{ratio21}(A,M,M,B) \wedge \mathrm{ratio21}(X,M,M,Y) \wedge \\
\mathrm{line}(A,X,ax) \wedge \mathrm{line}(B,Y,by) \Longrightarrow \mathrm{para}(ax,by))
\end{array}$$

Also, in Euclidean geometry there are clear connections between parallel and perpendicular lines.

$$(\forall l1,l2,a)\ (\mathrm{perp}(l_1,a) \wedge \mathrm{para}(l_1,l_2) \Longrightarrow \mathrm{perp}(l_2,a))$$

[1] `https://www.tptp.org/`

3.2 Using Automated Provers

The conjecture of the construction problem considered in Example 2.1 can be formulated in TPTP format in the following way:

```
fof(th_A_Ha_O, conjecture, ( ( inc(pA,ha1) & inc(pHa1,ha1)
    & perp(ha1,a1) & inc(pHa1,a1) & inc_c(pA,cc1) & center(pOc1,cc1)
    & inc_c(pB,cc1) & inc(pB,a1) & inc_c(pC,cc1) & inc(pC,a1) )
    => ( pHa = pHa1 &  pOc = pOc1 ) ) ).
```

where pHa and pOc are defined by the axioms as the foot of the altitude from vertex A to side BC and circumcenter of triangle ABC, respectively.

Larus successfully proved given conjecture as two separate statements, one for each of the facts in the conclusion. Key fragment of generated readable proof is given below (all used geometry axioms are listed, others are the ones implied by equality):

Axioms:

1. bc_unique : $\forall L\ (inc(pB,L) \wedge inc(pC,L) \Rightarrow L = bc\ )$
2. haA : $\forall H\ (perp(H,bc) \wedge inc(pA,H) \Rightarrow ha = H\ )$
3. pHa_def : $\forall H1\ (inc(H1,ha) \wedge inc(H1,bc) \Rightarrow H1 = pHa\ )$
4. cc_unique : $\forall C\ (inc\_c(pA,C) \wedge inc\_c(pB,C) \wedge inc\_c(pC,C) \Rightarrow C = cc\ )$
5. center_unique : $\forall C\ \forall C1\ \forall C2\ (center(C1,C) \wedge center(C2,C) \Rightarrow C1 = C2\ )$

Example 3.1. th_A_Ha_O0 :
$inc(pA,ha1) \wedge inc(pHa1,ha1) \wedge perp(ha1,a1) \wedge inc(pHa1,a1) \wedge inc\_c(pA,cc1)$
$\wedge\ center(pOc1,cc1) \wedge inc\_c(pB,cc1) \wedge inc(pB,a1) \wedge inc\_c(pC,cc1) \wedge inc(pC,a1)$
$\Rightarrow pHa = pHa1$

Proof:

1. $pHa = pHa$ (by MP, using axiom eqnativeEqSub0; instantiation: $A \mapsto pHa$, $B \mapsto pHa$, $X \mapsto pHa$)
2. $a1 = bc$ (by MP, from $inc(pB,a1)$, $inc(pC,a1)$ using axiom bc_unique; instantiation: $L \mapsto a1$)
3. $perp(ha1,bc)$ (by MP, from $perp(ha1,a1)$, $a1 = bc$ using axiom perpEqSub1; instantiation: $A \mapsto ha1$, $B \mapsto a1$, $X \mapsto bc$)
4. $ha = ha1$ (by MP, from $perp(ha1,bc)$, $inc(pA,ha1)$ using axiom haA; instantiation: $H \mapsto ha1$)
5. $inc(pHa1,ha)$ (by MP, from $inc(pHa1,ha1)$, $ha = ha1$ using axiom incEqSub1; instantiation: $A \mapsto pHa1$, $B \mapsto ha1$, $X \mapsto ha$)
6. $inc(pHa1,bc)$ (by MP, from $inc(pHa1,a1)$, $a1 = bc$ using axiom incEqSub1; instantiation: $A \mapsto pHa1$, $B \mapsto a1$, $X \mapsto bc$)
7. $pHa1 = pHa$ (by MP, from $inc(pHa1,ha)$, $inc(pHa1,bc)$ using axiom pHa_def; instantiation: $H1 \mapsto pHa1$)
8. $pHa = pHa1$ (by MP, from $pHa1 = pHa$, $pHa = pHa$ using axiom eqnativeEqSub0; instantiation: $A \mapsto pHa$, $B \mapsto pHa1$, $X \mapsto pHa$)
9. Proved by assumption! (by QEDas)

Example 3.2. th_A_Ha_O1 :
$inc(pA,ha1) \wedge inc(pHa1,ha1) \wedge perp(ha1,a1) \wedge inc(pHa1,a1) \wedge inc\_c(pA,cc1)$
$\wedge\ center(pOc1,cc1) \wedge inc\_c(pB,cc1) \wedge inc(pB,a1) \wedge inc\_c(pC,cc1) \wedge inc(pC,a1)$
$\Rightarrow pOc = pOc1$

Proof:

1. $center(pOc, cc)$ (by MP, using axiom centerEqSub1; instantiation: $A \mapsto pOc$, $B \mapsto cc$, $X \mapsto cc$)
2. $cc1 = cc$ (by MP, from $inc\_c(pA, cc1)$, $inc\_c(pB, cc1)$, $inc\_c(pC, cc1)$ using axiom cc_unique; instantiation: $C \mapsto cc1$)
3. $center(pOc1, cc)$ (by MP, from $center(pOc1, cc1)$, $cc1 = cc$ using axiom centerEqSub1; instantiation: $A \mapsto pOc1$, $B \mapsto cc1$, $X \mapsto cc$)
4. $pOc = pOc1$ (by MP, from $center(pOc, cc)$, $center(pOc1, cc)$ using axiom center_unique; instantiation: $C \mapsto cc$, $C1 \mapsto pOc$, $C2 \mapsto pOc1$)
5. Proved by assumption! (by QEDas)

Correctness proof of the generated construction for the problem considered in Example 2.2 is given in Appendix.

4 Results

We considered the subset of problems from Wernick's corpus, over vertices of the triangle, midpoints of triangle sides, feet of altitudes, centroid, circumcenter and orthocenter of the triangle. It consists of 35 non-isomorphic location triangle problems. For each of these problems, we tried to prove the correctness of constructions found by ArgoTriCS using FOL prover Vampire and coherent logic prover Larus. Vampire succesfully proved 31 of these problems, while Larus proved 20 problems, and for remaining ones it could not prove it in given timelimit.

5 Conclusion

Although this is a work-in-progress, we have managed to show that this approach is plausible and can be used to automatically obtain readable proofs of correctness for geometric constructions. This is very important in the context of mathematical education, where students need to know why a geometric statement holds. In our previous work, we have described ArgoTriCS – a system that is able to perform ruler and compass construction steps for almost all solvable problems in the Wernick's corpus [6, 10]. The main step in the ArgoTriCS implementation was to formulate a good set of lemmas to be used for analysing and finding the construction. This work shows that an identified set of lemmas is not sufficient to generate correctness proofs, and that the proof phase requires an additional set of lemmas (mainly the lemmas that guarantee uniqueness, but also some other equally important lemmas). However, once these lemmas are identified, they can be passed to general-purpose theorem provers, which can then generate fully synthetic proofs of correctness. Although the coherent logic solvers we have tested are not yet as powerful as the FOL solvers such as Vampire, if they succeed in solving the given problem, they provide us with human-readable proofs.

A very important issue is the correctness of the used lemmas. Indeed, if some lemmas are incorrect (e.g., if a precondition or a non-degeneracy condition is missing), a contradiction may arise and the theorem could be proved from this contradiction. We examined all the generated proofs, and all of them were correct. To be completely sure that our lemmas are correct, we formalize them in Isabelle/HOL and prove them using the axioms of geometry. Since Larus can output Isabelle/HOL proofs, we will eventually have a system capable of generating proofs of construction that are fully mechanically verified starting from the axioms.

In the present work we have not considered degenerate cases and the existence of constructed objects (we have simply assumed that everything is non-degenerate and that all constructed objects exist). However, we plan to pay more attention to this issue and extend our tools to perform the final discussion phase where they would automatically identify the necessary non-degeneracy conditions.

Coherent logic prover, Larus, used in this research is currently unable to find all correctness proofs fully automatically. We have worked around this by giving it hints in the form of lemmas. We plan to use other coherent logic provers, and we are in contact with the Larus developers so that they can improve their prover using the feedback they have received from our problems.

A Appendix

Larus cannot currently prove the whole theorem only if no guidance is provided. Therefore, we first derive several lemmas and then use those lemmas to prove the main theorem. The first part of the conjecture is easily proved:

Axioms:

1. cc_unique : $\forall C\ (inc\_c(pA,C) \wedge inc\_c(pB,C) \wedge inc\_c(pC,C) \Rightarrow C = cc\ )$
2. center_unique : $\forall C\ \forall C1\ \forall C2\ (center(C1,C) \wedge center(C2,C) \Rightarrow C1 = C2\ )$
3. bc_unique : $\forall L\ (inc(pB,L) \wedge inc(pC,L) \Rightarrow L = bc\ )$
4. haA : $\forall H\ (perp(H,bc) \wedge inc(pA,H) \Rightarrow ha = H\ )$
5. inc_line : $\forall P1\ \forall P2\ \forall L\ (inc(P1,L) \wedge inc(P2,L) \wedge P1 \neq P2 \Rightarrow line(P1,P2,L)\ )$
6. ex_line : $\forall P1\ \forall P2\ (\exists L\ (line(P1,P2,L))\ )$
7. ratio21_para : $\forall A\ \forall G\ \forall Ma\ \forall H\ \forall Oc\ \forall Lba\ \forall Lha\ (ratio21(A,G,G,Ma) \wedge ratio21(H,G,G,Oc) \wedge line(Oc,Ma,Lba) \wedge line(A,H,Lha) \Rightarrow para(Lba,Lha)\ )$
8. perp_para : $\forall Lba\ \forall Lha\ \forall A\ (perp(Lha,A) \wedge para(Lba,Lha) \Rightarrow perp(Lba,A)\ )$
9. perp_unique : $\forall P\ \forall L\ \forall L1\ \forall L2\ (perp(L1,L) \wedge inc(P,L1) \wedge perp(L2,L) \wedge inc(P,L2) \Rightarrow L1 = L2\ )$
10. pMa_is_interect_bisa_bc : $\forall P\ (inc(P,bc) \wedge inc(P,bisa) \Rightarrow P = pMa\ )$

Example A.1. th_A_O_G_1:
$ratio23(pA,pG1,pA,pMa1) \wedge ratio23(pH1,pG1,pH1,pOc1) \wedge inc(pA,ha1) \wedge inc(pH1,ha1) \wedge inc(pMa1,a1) \wedge perp(a1,ha1) \wedge center(pOc1,cc1) \wedge inc\_c(pA,cc1) \wedge inc\_c(pB,cc1) \wedge inc(pB,a1) \wedge inc\_c(pC,cc1) \wedge inc(pC,a1) \wedge pA \neq pH1 \Longrightarrow pOc1 = pOc$

Proof:

1. $cc1 = cc$ (by MP, from $inc\_c(pA,cc1)$, $inc\_c(pB,cc1)$, $inc\_c(pC,cc1)$ using axiom cc_unique; instantiation: $C \mapsto cc1$)
2. $center(pOc1,cc)$ (by MP, from $center(pOc1,cc1)$, $cc1 = cc$ using axiom centerEqSub1; instantiation: $A \mapsto pOc1$, $B \mapsto cc1$, $X \mapsto cc$)
3. $pOc1 = pOc$ (by MP, from $center(pOc1,cc)$ using axiom center_unique; instantiation: $C \mapsto cc$, $C1 \mapsto pOc1$, $C2 \mapsto pOc$)
4. Proved by assumption! (by QEDas)

Then, the facts `a1 = bc` and `ha1 = ha` can be derived:

Example A.1. *lm_A_O_G_2:*
$ratio23(pA, pG1, pA, pMa1) \wedge ratio23(pH1, pG1, pH1, pOc1) \wedge inc(pA, ha1) \wedge inc(pH1, ha1) \wedge inc(pMa1, a1) \wedge perp(a1, ha1) \wedge center(pOc1, cc1) \wedge inc\_c(pA, cc1) \wedge inc\_c(pB, cc1) \wedge inc(pB, a1) \wedge inc\_c(pC, cc1) \wedge inc(pC, a1) \wedge pA \neq pH1 \Longrightarrow a1 = bc$

Proof:

1. $a1 = bc$ (by MP, from $inc(pB, a1)$, $inc(pC, a1)$ using axiom bc_unique; instantiation: $L \mapsto a1$)
2. Proved by assumption! (by QEDas)

Example A.2. *lm_A_O_G_3:*
$ratio23(pA, pG1, pA, pMa1) \wedge ratio23(pH1, pG1, pH1, pOc1) \wedge inc(pA, ha1) \wedge inc(pH1, ha1) \wedge inc(pMa1, a1) \wedge perp(a1, ha1) \wedge center(pOc1, cc1) \wedge inc\_c(pA, cc1) \wedge inc\_c(pB, cc1) \wedge inc(pB, a1) \wedge inc\_c(pC, cc1) \wedge inc(pC, a1) \wedge pA \neq pH1 \Longrightarrow ha1 = ha$

Proof:

1. $a1 = bc$ (by MP, from $inc(pB, a1)$, $inc(pC, a1)$ using axiom bc_unique; instantiation: $L \mapsto a1$)
2. $perp(bc, ha1)$ (by MP, from $perp(a1, ha1)$, $a1 = bc$ using axiom perpEqSub0; instantiation: $A \mapsto a1$, $B \mapsto ha1$, $X \mapsto bc$)
3. $ha = ha1$ (by MP, from $perp(bc, ha1)$, $inc(pA, ha1)$ using axiom haA; instantiation: $H \mapsto ha1$)
4. $ha1 = ha$ (by MP, from $ha = ha1$ using axiom eq_sym; instantiation: $A \mapsto ha$, $B \mapsto ha1$)
5. Proved by assumption! (by QEDas)

Now the conclusions of these lemmas can be added to the set of premises, and the next lemma can be proved:

Example A.3. *lm_A_O_G_4 :*
$ratio23(pA, pG1, pA, pMa1) \wedge ratio23(pH1, pG1, pH1, pOc1) \wedge inc(pA, ha1) \wedge inc(pH1, ha1) \wedge inc(pMa1, a1) \wedge perp(a1, ha1) \wedge center(pOc1, cc1) \wedge inc\_c(pA, cc1) \wedge inc\_c(pB, cc1) \wedge inc(pB, a1) \wedge inc\_c(pC, cc1) \wedge inc(pC, a1) \wedge pA \neq pH1 \wedge pOc1 = pOc \wedge a1 = bc \wedge ha1 = ha \Longrightarrow line(pOc1, pMa1, bisa)$

Proof:

1. $inc(pOc1, bisa)$ (by MP, from $pOc1 = pOc$ using axiom incEqSub0; instantiation: $A \mapsto pOc$, $B \mapsto bisa$, $X \mapsto pOc1$)
2. Let w be such that $line(pOc1, pMa1, w)$ (by MP, using axiom ex_line; instantiation: $P1 \mapsto pOc1$, $P2 \mapsto pMa1$)
3. $line(pA, pH1, ha1)$ (by MP, from $inc(pA, ha1)$, $inc(pH1, ha1)$, $pA \neq pH1$ using axiom inc_line; instantiation: $P1 \mapsto pA$, $P2 \mapsto pH1$, $L \mapsto ha1$)
4. $para(w, ha1)$ (by MP, from $ratio23(pA, pG1, pA, pMa1)$, $ratio23(pH1, pG1, pH1, pOc1)$, $line(pOc1, pMa1, w)$, $line(pA, pH1, ha1)$ using axiom ratio21_para; instantiation: $A \mapsto pA$, $G \mapsto pG1$, $Ma \mapsto pMa1$, $H \mapsto pH1$, $Oc \mapsto pOc1$, $Lba \mapsto w$, $Lha \mapsto ha1$)
5. $perp(ha1, bc)$ (by MP, from $ha1 = ha$ using axiom perpEqSub0; instantiation: $A \mapsto ha$, $B \mapsto bc$, $X \mapsto ha1$)
6. $perp(w, bc)$ (by MP, from $perp(ha1, bc)$, $para(w, ha1)$ using axiom perp_para; instantiation: $Lba \mapsto w$, $Lha \mapsto ha1$, $A \mapsto bc$)
7. $w = bisa$ (by MP, from $perp(w, bc)$, $line(pOc1, pMa1, w)$, $inc(pOc1, bisa)$ using axiom perp_unique; instantiation: $P \mapsto pOc1$, $L \mapsto bc$, $L1 \mapsto w$, $L2 \mapsto bisa$)
8. $line(pOc1, pMa1, bisa)$ (by MP, from $line(pOc1, pMa1, w)$, $w = bisa$ using axiom lineEqSub2; instantiation: $A \mapsto pOc1$, $B \mapsto pMa1$, $C \mapsto w$, $X \mapsto bisa$)
9. Proved by assumption! (by QEDas)

Finally, with the conclusion of this lemma added to the premises, we can prove the final statament:

Example A.2. th_A_O_G_5:

$ratio23(pA, pG1, pA, pMa1) \wedge ratio23(pH1, pG1, pH1, pOc1) \wedge inc(pA, ha1) \wedge inc(pH1, ha1) \wedge inc(pMa1, a1) \wedge perp(a1, ha1) \wedge center(pOc1, cc1) \wedge inc\_c(pA, cc1) \wedge inc\_c(pB, cc1) \wedge inc(pB, a1) \wedge inc\_c(pC, cc1) \wedge inc(pC, a1) \wedge pA \neq pH1 \wedge pOc1 = pOc \wedge a1 = bc \wedge ha1 = ha \wedge pOc = pOc1 \wedge line(pOc1, pMa1, bisa) \implies pG = pG1$

Proof:

1. $inc(pMa1, bc)$ (by MP, from $inc(pMa1, a1)$, $a1 = bc$ using axiom incEqSub1; instantiation: $A \mapsto pMa1$, $B \mapsto a1$, $X \mapsto bc$)
2. $pMa1 = pMa$ (by MP, from $inc(pMa1, bc)$, $line(pOc1, pMa1, bisa)$ using axiom pMa_is_interect_bisa_bc; instantiation: $P \mapsto pMa1$)
3. $ratio23(pA, pG1, pA, pMa)$ (by MP, from $ratio23(pA, pG1, pA, pMa1)$, $pMa1 = pMa$ using axiom ratio23EqSub3; instantiation: $A \mapsto pA$, $B \mapsto pG1$, $C \mapsto pA$, $D \mapsto pMa1$, $X \mapsto pMa$)
4. $pG = pG1$ (by MP, from $ratio23(pA, pG1, pA, pMa)$ using axiom ratio23_Ma_Gsat0; instantiation: $X \mapsto pG1$)
5. Proved by assumption! (by QEDas)

References

[1] Bruno Buchberger (2006): *An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal*. Journal of Symbolic Computation 41(3), pp. 475–511, doi:10.1016/j.jsc.2005.09.007.

[2] Predrag Janičić (2006): *GCLC – A Tool for Constructive Euclidean Geometry and More than That*. In: Proceedings of International Congress of Mathematical Software (ICMS 2006), Lecture Notes in Computer Science 4151, pp. 58–73, doi:10.1007/11832225_6.

[3] Predrag Janičić & Julien Narboux (2021): *Automated Generation of Illustrations for Synthetic Geometry Proofs*. In: Proceedings of the 13th International Conference on Automated Deduction in Geometry, ADG 2021, EPTCS 352, pp. 91–102, doi:10.4204/EPTCS.352.9.

[4] Predrag Janičić, Julien Narboux & Pedro Quaresma (2012): *The Area Method - a recapitulation*. Journal of Automated Reasoning 48(4), pp. 489–532, doi:10.1007/s10817-010-9209-7.

[5] Filip Marić, Ivan Petrović, Danijela Petrović & Predrag Janičić (2012): *Formalization and Implementation of Algebraic Methods in Geometry*. In: Proceedings First Workshop on CTP Components for Educational Software, Electronic Proceedings in Theoretical Computer Science 79, pp. 63–81, doi:10.4204/EPTCS.79.4.

[6] Vesna Marinković (2017): *ArgoTriCS – Automated Triangle Construction Solver*. Journal of Experimental & Theoretical Artificial Intelligence 29(2), pp. 247–271, doi:10.1080/0952813X.2015.1132271.

[7] Vesna Marinković (2015): *Automated Solving of Construction Problems in Geometry*. Ph.D. thesis, University of Belgrade.

[8] Vesna Marinković, Tijana Šukilović & Filip Marić (2023): *Automated Triangle Constructions in Hyperbolic Geometry*. to appear in Annals of Mathematics and Artificial Intelligence, doi:10.1007/s10472-023-09850-5.

[9] Alexandre Riazanov & Andrei Voronkov (2002): *The design and implementation of Vampire*. AI Communications 15(2,3), pp. 91–110.

[10] William Wernick (1982): *Triangle Constructions with Three Located Points*. Mathematics Magazine 55(4), pp. 227–230, doi:10.1080/0025570X.1985.11976988.

[11] Wen Tsun Wu (1978): *On the Decision Problem and the Mechnization of Theorem-proving in Elementary Geometry*. Scientia Sinica 21(2), pp. 159–172.

Automated Completion of Statements and Proofs in Synthetic Geometry: an Approach based on Constraint Solving

Salwa Tabet Gonzalez
UMR 7357 CNRS
University of Strasbourg
Pôle API, Bd Sébastien Brant
BP 10413
67412 Illkirch, France
tabetgonzalez@unistra.fr

Predrag Janičić
Department for Computer Science
Faculty of Mathematics
University of Belgrade
Studentski trg 16
11000 Belgrade, Serbia
janicic@matf.bg.ac.rs

Julien Narboux
UMR 7357 CNRS
University of Strasbourg
Pôle API, Bd Sébastien Brant
BP 10413
67412 Illkirch, France
narboux@unistra.fr

Conjecturing and theorem proving are activities at the center of mathematical practice and are difficult to separate. In this paper, we propose a framework for completing incomplete conjectures and incomplete proofs. The framework can turn a conjecture with missing assumptions and with an under-specified goal into a proper theorem. Also, the proposed framework can help in completing a proof sketch into a human-readable and machine-checkable proof. Our approach is focused on synthetic geometry, and uses coherent logic and constraint solving. The proposed approach is uniform for all three kinds of tasks, flexible and, to our knowledge, unique such approach.

1 Introduction

Automated theorem provers take as input the formal statement of a conjecture in a theory described by axioms and lemmas, and try to generate a proof or a counter-example for this conjecture. In the field of geometry, several efficient automated theorem proving approaches have been developed, including algebraic ones such as Wu's method, Gröbner bases method, and semi-synthetic methods such as the area method. In these approaches, typically, the conjecture and the axioms being considered are fixed. However, in mathematical practice, in the context of education and also in mathematical research, the conjecturing and proving activities are not separated but interleaved. The practitioner may try to prove a statement which is valid only assuming some implicit or unknown assumptions, while the list of lemmas and theorem which can be used may not be complete. In education, for some kind of exercises, a precise formulation of the statement to be proved is also left to the student, with questions such as: "What is the nature of the quadrilateral *ABCD*?". Hence, the conjecture can contain unknown assumptions called *abducts*, and the goal may be not completely specified. One may also ask for a proof using a particular theorem or an intermediate fact, i.e., a proof partially specified using constraints specifying some proof steps.

In this paper, we consider the problems of (simultaneously) completing (a) the assumptions of the conjecture; (b) the goal of the conjecture; (c) a proof sketch for the conjecture. The completion process should lead to a proof that is both machine-checkable and human-readable. Because we aim at producing intelligible and readable proofs, with a similar level of granularity as paper-and-pencil proofs, our approach is logic-based, uses a fragment of first-order logic called coherent logic, and is focused on synthetic geometry (in contrast to algebraic methods). Our approach for dealing with partial conjectures and partial proofs is implemented as an extension of the automated theorem prover Larus developed previously [14]. The approach is uniform for all three kinds of completion tasks, flexible and, to our knowledge, unique such approach.

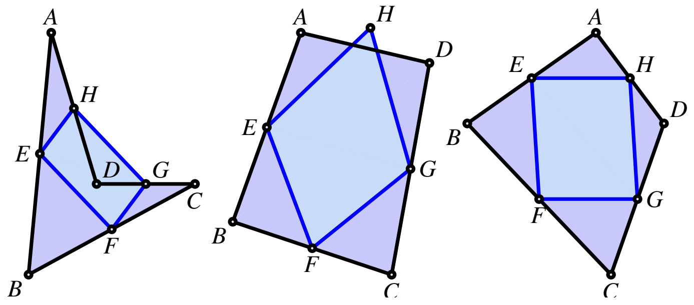


Figure 1: Illustrations for five problems related to Varignon's theorem, respectively: Problem 1; Problem 2; Problem 3.

We list five high-school level synthetic geometry problems related to Varignon's theorem (Figure 1), that we will try to solve using our approach.

Problem 1 (Fully specified statement) Consider a quadrilateral $ABCD$, let E, F, G and H be the midpoints of AB, BC, CD, DA respectively. Prove that the quadrilateral $EFGH$ is a parallelogram (assuming that there are no two sides that are aligned).

Problem 2 (First inverse problem) Consider a quadrilateral $ABCD$, let E, F, and G be the midpoints of AB, BC and CD respectively. Let H be a point. Under which assumption is the quadrilateral $EFGH$ a parallelogram?

Problem 3 (Second inverse problem) Consider a quadrilateral $ABCD$, let E, F, G and H be the midpoints of AB, BC, CD, DA respectively. Under which assumption is the quadrilateral $EFGH$ a rectangle?

Problem 4 (Partially specified goal) Consider a quadrilateral $ABCD$, let E, F, G and H be the midpoints of AB, BC, CD, DA respectively. What is the nature of the quadrilateral $EFGH$?

Problem 5 (Partially specified proof) Consider a quadrilateral $ABCD$, let E, F, G and H be the midpoints of AB, BC, CD, DA respectively. We have that $EG = FH$. Prove that $EFGH$ is a rectangle using the axiom "If the diagonals of a parallelogram are congruent, then it's a rectangle".

The above examples are inspired by exercises given in a teacher training session. A more detailed discussion about how these examples can be used in a didactic context, issues related to the formalization can be found in [11, 19]

2 Background

This section provides some necessary background information on a fragment of first-order logic called coherent logic that our approach uses. There are several automated provers for coherent logic, including Larus, which is based on "theorem proving as constraint solver" paradigm.

2.1 Coherent Logic

A formula of first-order logic is said to be *coherent* if it has the following form:

$$A_0(\vec{x}) \wedge \ldots \wedge A_{n-1}(\vec{x}) \Rightarrow \exists \vec{y}(B_0(\vec{x},\vec{y}) \vee \ldots \vee B_{m-1}(\vec{x},\vec{y}))$$

where universal closure is assumed, and where $\vec{x}$ denotes a sequence of variables $x_0, x_1, \dots, x_{k-1}$; A_i (for $0 \le i \le n-1$) denotes an atomic formula (involving zero or more variables from $\vec{x}$); $\vec{y}$ denotes a sequence of variables $y_0, y_1, \dots, y_{l-1}$; B_j (for $0 \le j \le m-1$) denotes a conjunction of atomic formulae (involving zero or more of the variables from $\vec{x}$ and $\vec{y}$) [14]. If there are no formulae A_i, then the left-hand side of the implication is assumed to be $\top$. If there are no formulae B_j, then the right-hand side of the implication is assumed to be $\bot$. There are no function symbols with arity greater than zero. Coherent formulae do not involve the negation connective. A coherent theory is a set of sentences, axiomatized by coherent formulae, and closed under derivability. A number of theories and theorems can be formulated directly and simply in coherent logic (CL). In addition, any first-order theory can be translated into CL, possibly with additional predicate symbols [12, 21]. Synthetic geometry can be expressed easily using CL. For example, the central part of axioms system of Euclid (as formalized by Beeson et al. [3]), or Hilbert (as formalized by Braun et al. [6]), or Tarski [26] can be expressed in first-order logic without function symbols, and the axioms are mostly in CL form.

Translation of FOL formulae into CL involves elimination of the negation connectives: negations can be kept in place and new predicates symbols for corresponding sub-formula have to be introduced, or negations can be pushed down to atomic formulae [21]. In the latter case, for every predicate symbol R (that appears in negated form), a new symbol $\overline{R}$ is introduced that stands for $\neg R$, and the following axioms are introduced: $\forall\vec{x}(R(\vec{x}) \wedge \overline{R}(\vec{x}) \Rightarrow \bot)$, $\forall\vec{x}(R(\vec{x}) \vee \overline{R}(\vec{x}))$.

In contrast to resolution-based theorem proving, in forward reasoning for CL, the conjecture being proved is kept unchanged and proved without using refutation, Skolemization and clausal form. Thanks to this, CL is suitable for producing human-readable synthetic proofs and also machine verifiable proofs [4, 12]. The problem of provability in CL is semi-decidable. CL admits a simple proof system, a sequent-based variant is as follows [27]:

$$\frac{\Gamma, ax, A_0(\vec{a}), \dots, A_{n-1}(\vec{a}), \underline{B_0(\vec{a},\vec{b})} \vee \dots \vee \underline{B_{m-1}(\vec{a},\vec{b})} \vdash P}{\Gamma, ax, A_0(\vec{a}), \dots, A_{n-1}(\vec{a}) \vdash P}\ \text{MP}$$

$$\frac{\Gamma, \underline{B_0(\vec{c})} \vdash P \quad \dots \quad \Gamma, \underline{B_{m-1}(\vec{c})} \vdash P}{\Gamma, B_0(\vec{c}) \vee \dots \vee B_{m-1}(\vec{c}) \vdash P}\ \text{QEDcs (case split)}$$

$$\frac{}{\Gamma, \underline{B_i(\vec{a},\vec{b})} \vdash \exists\vec{y}(B_0(\vec{a},\vec{y}) \vee \dots \vee B_{m-1}(\vec{a},\vec{y}))}\ \text{QEDas (assumption)}$$

$$\frac{}{\Gamma, \bot \vdash P}\ \text{QEDefq (ex falso quodlibet)}$$

In the rules given above, it is assumed: ax is a formula $A_0(\vec{x}) \wedge \dots \wedge A_{n-1}(\vec{x}) \Rightarrow \exists\vec{y}(B_0(\vec{x},\vec{y}) \vee \dots \vee B_{m-1}(\vec{x},\vec{y}))$;[1] $\vec{a}$, $\vec{b}$, $\vec{c}$ denote sequences of constants (possibly of length zero); in the rule MP (*extended modus ponens*), $\vec{b}$ are fresh constants; $\vec{x}$ and $\vec{y}$ denote sequences of variables (possibly of length zero); $A_i(\vec{x})$ (respectively $B_i(\vec{x},\vec{y})$) have no free variables other than from $\vec{x}$ (respectively $\vec{x}$ and $\vec{y}$); $A_i(\vec{a})$ are ground atomic formulae; $B_i(\vec{a},\vec{b})$ and $B_i(\vec{c})$ are conjunctions of ground atomic formulae; $\underline{\Phi}$ denotes the list of conjuncts in Φ if Φ is conjunction, and otherwise Φ itself. In the proving process, the rules are

[1]Notice the hidden link between the formulae $B_i(\vec{a},\vec{b})$ from the rule MP and the formula ax: the formulae $B_i(\vec{a},\vec{b})$ from the rule are instances of the formulae $B_i(\vec{x},\vec{y})$ from ax.

read from bottom to top, i.e., by a rule application one gets the contents (new sub-goals) above the line.

For a set of coherent axioms AX and the statement $A_0(\vec{x}) \wedge \ldots \wedge A_{n-1}(\vec{x}) \Rightarrow \exists \vec{y}(B_0(\vec{x},\vec{y}) \vee \ldots \vee B_{m-1}(\vec{x},\vec{y}))$ to be proved, within the above proof system one has to derive the following sequent (where $\vec{a}$ denotes a sequence of new symbols of constants): $AX, A_0(\vec{a}), \ldots, A_{n-1}(\vec{a}) \vdash \exists \vec{y}(B_0(\vec{a},\vec{y}) \vee \ldots \vee B_{m-1}(\vec{a},\vec{y}))$.

Notice that, in the above proof system, case split may occur only at the end of a (sub)proof. However, it is not a substantial restriction: any proof with unrestricted use of case split can be transformed to such form.

2.2 Theorem Proving as Constraint Solving and the Larus System

"Theorem proving as constraint solving" is a paradigm for automated theorem proving recently proposed [14]. In contrast to common automated theorem proving approaches, in which the search space is a set of some formulae and what is sought is again a (goal) formula, this new approach is based on searching for a proof (of a given length) as a whole. Namely, a proof of a formula in a fixed logical setting can be encoded as a sequence of natural numbers obeying some constraints. A suitable solver can find such a sequence and from that sequence a sought proof can be reconstructed. This approach is implemented in C++, within an open-source prover Larus,[2] specialized in proofs in coherent logic and using SAT, SMT, and CSP solvers for solving sets of constraints. Larus can generate readable, human understandable proofs in natural language and also machine-verifiable proofs for the interactive provers Coq, Isabelle, and Mizar.

Each CL proof consists of several proof steps, while each of them has one of the following kinds (with obvious meaning): ASSUMPTION, MP, FIRSTCASE, SECONDCASE, QEDBYCASES, QEDBYASSUMPTION, QEDBYEFQ. The information relevant for MP steps include: `AxiomApplied`, `From` (the ordinal numbers of proof steps justifying premises of the axiom applied), `Instantiation` (of the variables in the axiom), `Contents` (the atoms in formula in the proof step), etc. `Nesting` denotes the nesting of the proof step (the nesting of the first step is 1).

The proof can be represented by a sequence of numbers, meeting some constraints (that correspond to definitions of inference steps given in Section 2.1). For instance, if the proof step s is of the kind QEDBYEFQ, then the following conditions must hold (given almost in verbatim as in our C++ code):[3]

1. `StepKind` $(s) =$ QEDBYEFQ;
2. $s > 0$;
3. `Contents` $(s-1)(0) = \bot$;
4. `Goal` (s);
5. `Nesting` $(s) =$ `Nesting` $(s-1)$.

The above conditions can be understood in the following way: if there is a proof of the given conjecture, the proof step s in that proof is of the kind QEDBYEFQ iff the natural number `StepKind` (s) equals the code for QEDBYEFQ, $s > 0$ (since there must be a previous step), the contents of the previous proof step is $\bot$, the contents of the step is the goal itself, and the nesting of the steps $s-1$ and s is the same.

Each proof step has one of the listed kinds and meet corresponding conditions. There are also some additional, global constraints, like that the last proof step has `Nesting` equal 1.

[2]`https://github.com/janicicpredrag/Larus`

[3]The corresponding C++ implementation is an improved version of the implementation presented earlier [14].

Larus works in the following way. If there is a set of axioms, a conjecture, and a proof length, a corresponding proof can be represented as a sequence of natural numbers, still unknown, so they will be represented by variables V. The constraints that have to be met for each proof step and for the proof as a whole can be expressed in terms of these variables V. If a solver can find a model for the constraint, from it the proof in logical terms can be reconstructed. All constraints involved are linear constraints over natural numbers. Since linear arithmetic is decidable, decision procedures for it can decide, for each input constrains, whether or not it has a model. For this purpose, Larus can use SAT, SMT, and CSP solvers. For input, Larus uses axioms and conjectures stored in a file in TPTP/fof format.

3 Abducts and Completing Assumptions

There are three major types of logical inference: induction, deduction, and abduction. The concept of abduction has been introduced by Peirce [20]. In deduction, everything inferred is necessarily true, while it is not the case with the remaining two types of inference. Induction tries to infer general rules based on individual instances. The aim of abduction is to produce additional hypotheses to explain observed facts. Abduction has a wide spectrum of implicit or explicit applications – in everyday life, in education, and in scientific reasoning, including in building mathematical theories, or in software verification. One definition of abduct is given below.

Definition 1 *Given a theory T and a formula G (the goal to be proved), such that $T \not\models G$, an* explanations *or* abduct *is a formula A meeting conditions: $T, A \models G$ and $T, A \not\models \bot$.*

It is clear that some abducts are not *interesting*, so there are often some additional restrictions given. There is no general agreement about such restrictions, but two types are most usual: *syntactical restrictions* (abducts should be of a specific syntactical form) and *minimality restrictions* (for any other abduct A', if $T, A \models A'$ then $A \equiv A'$). It is reasonable to ask that A is not G, as it is trivial. Some authors also add stronger a restriction that $A \not\models G$ (i.e., at least one axiom of T has to be used to prove G).

Approaches for Computing Abducts. Various algorithms to produce different kind of abducts have been developed [1]. In *abductive logic programming*, techniques for abductive reasoning are developed in the context of logic programming. Rules are considered to be Horn clauses [8]. According to Russo et al. [25], some systems assume that predicate symbols appearing in abducts do not appear in the conclusion of any rule and that negation does not appear in the conclusion of any rule. This restriction is not realistic in the context of geometry. In our example, we want to accept geometric predicate symbols both in abducts, and in the assumptions and conclusion of theorems. Some approaches are based on Robinson's resolution algorithm, extended such that when no more clauses can be produced, the atomic clauses are considered as a potential abduct and consistency if checked [17]. There are also approaches developed for the context of SMT solving, dealing with decidable theories like linear arithmetic [10, 23]

In the context of geometry, some algebraic algorithms can generate additional assumptions for the statement to be true. For example, Wu's method [28] can produce non-degeneracy conditions. Algebraic methods can also be used to generate more general abducts [22]. These methods are more efficient than ours, but more specific so cannot be used for arbitrary geometric theories. Also, they cannot generate readable proofs. Moreover, expressing algebraic non-degeneracy conditions in simple geometrical terms is not easy and not always possible [7].

Abduction in Synthetic Euclidean Geometry. In this paper, the theory T from Definition 1 is a synthetic Euclidean geometry. In this context, automated finding of proofs allowing abducts may have several applications. For instance, an automated system may help a student or a researcher who tries to prove (or formalize) a theorem with a missing assumption. Barbosa et al. have proposed such goal (although not for geometry) in the context of interactive proof assistants where conjectures are sent to an SMT solver [2].

Non-degeneracy conditions are often overlooked and missing in informal geometry statements. Abductive reasoning is also a task which can be asked explicitly to students. The answer expected by the teacher for Problem 2 is that H should be the midpoint of AD.

Finding Abducts using Larus. In this paper, we restrict consideration of abduction only to coherent logic and only to abducts that are conjunctions of ground atomic formulae. Larus was not implemented with abduction in mind, yet implementation of support for abduction turned out to be very simple, almost trivial, and took less than 100 lines of C++ code. In order to find abducts using Larus, we treat them as a special case of proof steps, in the main proof branch, just after assumptions. We have to add constraints on what such an abduct can be:

1. the abduct is treated as an assumption;
2. the nesting of the abduct equals 1;
3. the abduct is an atomic formula (no branching);
4. the predicate symbol is one of the predicate symbols in the signature;
5. the arguments are among existing symbols of constants;
6. the abduct is not the goal itself;
7. the abduct is not $\bot$.

The given conditions may be written in the following way, assuming that the abduct is placed in i-th step of the proof:

1. `StepKind` $(i) =$ASSUMPTION
2. `Nesting` $(i) = 1$
3. `Cases` $(i) = \mathit{false}$
4. `ContentsPredicate` $(i,0) < \mathit{sizeof}(\mathit{Signature})$
5. for each argument j (up to maximal arity): `ContentsArgument` $(i,0,j) < \mathit{sizeof}(\mathit{Constants})$
6. `Goal` $(i) = \mathit{false}$
7. `ContentsPredicate` $(i,0) \neq \bot$

One can also choose a number of abducts, each leading to the constraints given above. With such additional constraints for each abduct (for additional proof steps in specific positions in a proof sought), with a given set of axioms and a conjecture, and with a concrete proof length, we run Larus as usual. The solving/proving process is the same as without abducts: the constraint solver finds a way to specify a full proof, including the abducts, i.e., under-specified assumptions.

In the above list of conditions, the last two do not follow the basic definition of abduct. Like in some other variants of the definition, the abduct may not be equal to the goal atom because such abducts are trivial. Also, the abduct may not be equal to $\bot$, since it is inconsistent. It is important to discard other inconsistent abducts early, so we add one more restriction: the proof of $T, A \models G$ should not end with QEDbyEFQ. Some constructed abducts may still be inconsistent with other assumptions, and we use an external, more efficient automatic theorem prover, Vampire [16], to discard such abducts.

Example 1 *For the first inverse problem (Problem 2 from Section 1), Larus produces two consistent, symmetric abducts (the proof obtained with the first abduct is presented in Appendix 7.2):*

- *"H is the midpoint of AD"*
- *"H is the midpoint of DA"*

Example 2 *For the second inverse problem (Problem 3 from Section 1), Larus produces more than 150 consistent abducts, most of which give degenerate cases, hence are less interesting. Apart from such abducts, we obtained the following abducts and their symmetric variants (the proof with the first abduct is presented in Appendix 7.3):*

- *"the diagonals HF and EG are congruent"*
- *"$\angle FGH$ is a right angle"*
- *"$\angle EHG$ is a right angle"*
- *"$\angle HEF$ is a right angle"*
- *"$\angle EFG$ is a right angle"*

4 Deducts and Completing Goals

Non-trivial first-order logic theories have infinite number of theorems. Approaches based on refutation cannot be used with under-specified goals and, hence, cannot be used for completing them. In principle, a controlled forward-reasoning (for instance, based on some kind of breadth-first search) can enumerate all theorems of a theory. However, such a systematic approach can be hardly useful for some practical applications, like looking for possible conjectures of a specific form. Our framework allows (but does not require) specifying partially the form of the goal: for instance, one may specify the dominant predicate symbol in the goal atom, or some of its arguments.

Finding Deducts in Synthetic Euclidean Geometry. In the field of geometry, deduct candidates can also be guessed based on an illustration, giving a concrete model. However, these deduct candidates still have to be verified i.e., proved. Potential deducts could also be listed as large disjunctions of atomic formulae, but this method does not scale when the list of potential deducts is too long.

Finding Deducts using Larus. In Larus, if the goal is given i.e., fully specified, corresponding constraints are added to the full constraint representing a proof sought. Let us assume that the final step of the proof is (some fixed) n and, for simplicity, let us assume that the goal is just a single atom. The corresponding constraint then includes:

1. `Nesting` $(n) = 1$
2. `Cases` $(n) = \mathit{false}$
3. `ContentsPredicate` $(n, 0) =$ the goal predicate symbol

4. for each argument j (except for existentially quantified variables):
   `ContentsArgument` $(n, 0, j) =$ the argument from the given goal instantiated.

If the goal is under-specified, for instance, if the predicate symbol is not given (it is given as _ in the TPTP file), the third condition is just ignored. The same holds for the arguments.[4] During the solving process, if there is a model, these slots are filled-in by some concrete values, giving a concrete goal. Overall, support for finding under-specified deducts is very simple. The current implementation finds one possible deduct, but it can be extended to list all possible deducts, similarly as for abducts (as explained in Section 3).

Example 3 *For Problem 4 from Section 1, Larus produces the deduct "$EF \parallel GH$". The proof obtained with such deduct is presented in Appendix 7.4.*

5 Hints and Completing Proofs

Informal proofs, for instance from textbooks, are often partial and incomplete. They may even provide only a part of a full proof, or some instructions like for filling gaps by analogy. Reconstructing proofs using such hints is very important task, as discussed by Gowers and Hales [13]: "One dream was to develop an automated assistant that would function at the level of a helpful graduate student. The senior mathematician would suggest the main lines of the proof, and the automated grad student would fill in the details."

Completing Proofs in Synthetic Euclidean Geometry. In the context of geometry, completing proofs could be interesting either as a way to render the formalization process simpler (automation would bring in all the details that are overlooked in pen and paper proofs), or as a tool working behind the scene for providing guidance for what could be the next step in the proof. This objective has been studied by Richard et al. [24]. In geometry, hints can also be based on some observations from an associated illustration.

Completing Proofs using Larus. Larus can be instructed to look for a proof of a given conjecture (also possibly only partially specified) meeting some conditions (that we call "hints") [14]. Therefore, Larus can try, for instance, to reconstruct a proof given only in outline (like proofs in textbooks). Larus use hints in a much more general way than just splitting the problem into sub-problems: for instance, some hint may be used in just one proof branch and cannot be proved itself. Hints do not have to be ordered (one can ask for a proof using X and Y in no particular order), they can be vague, imposed only by partial constraints ("find a proof that uses this particular predicate symbol", or "find a proof using some specific axiom", without the way it is instantiated, etc.).

Completing proofs in Larus is supported similarly as for abducts and under-specified goals – by modifying the corresponding constraints. The main difference is that abducts and incomplete goals are under-specified, so some constraints have to be omitted, while partial proof introduce additional constraints, on top of the common constraints that must be met by all proof steps. For expressing hints, we slightly extended the language TPTP/fof to allow a simple but still quite expressible semantics. Some kinds of hints (not all) are illustrated below.

[4]Actually, underspecified arguments can be also handled using existential quantification.

```
fof(hintname0, hint, r(_,_), _ , _).
fof(hintname1, hint, r(_,_), 5 , _).
fof(hintname2, hint, _, 5, ax2(_,_)).
```

The first hint specifies that some proof step will have an atom of the form $r(\dots,\dots)$. The second hint specifies that the 5th proof step will have atom of the form $r(\dots,\dots)$. The third hint specifies that in the 5th proof step the axiom $ax2$ is applied.

Example 4 *For Problem 5 from Section 1, Larus was able to find a proof around 20% faster with a suitable hint presented in Appendix 7.5.*

6 Conclusion and Future Work

In this paper we have shown how a prover using the "theorem proving as constraint solving" paradigm can be extended such that it can complete partially specified conjectures and partially specified proofs. This extension is simple, and the implementation update is very small. The completion algorithm is uniform, since all three completion tasks (completing assumptions, completing goals, completing proofs) are handled in the same spirit – in terms of adding or deleting some constraints. To our knowledge, this approach is new, and we are not aware of other systems that can address all three sorts of completion tasks. The presented approach is flexible as different variations of completion tasks can be supported. The strength of this approach is also that it can generate both proofs that are human-readable and machine-checkable. The proposed framework has two main limitations. First, in current stage, it can deal only with coherent logic, hence the theories cannot involve function symbols, which excludes geometry proof that use (non-trivial) arithmetic. Second, the framework cannot deal with conjectures whose proofs are long (say, longer than 50 proof steps).

To our knowledge, there is only one other approach in which some kind of proof is encoded, and reconstructed from a model for the corresponding set of constraints – the approach in which rigid connection tableaux are encoded as SAT and SMT instances [9, 5, 18]. However, in this line of research, neither machine verifiable or readable proofs, nor any of completion tasks are considered.

The presented work can be extended in several directions. One of our goals is to use our framework to help transfer geometry knowledge from informal sources to proof assistants and between proof assistants, while keeping its high-level structure. In informal sources, statements of theorems may be incomplete, while proofs may be given just in outline. Still, using our approach such contents can be, at least in some cases, completed and turned into a verifiable form. For transferring knowledge from a proof assistant, one would need to go into its specifics, but only to grab (some) proof steps and make hints out of them. We are still to explore these ideas on a larger scale, like one geometry textbook. In the same spirit as the work proposed by Jiang et al. [15], our approach could be combined with large language models to perform automatic formalization by extracting data from natural language proofs. More specific to abduction, we are planning to make an in-depth comparison (both qualitative and quantitative) of our tool to other tools for generating abducts.

Acknowledgement. The work related to this paper has been partially supported by the European Cost project CA20111 EUROProofNet. The second author has been partially supported by the Ministry of Science of Serbia contract 451-03-47/2023-01/200104.

References

[1] Aliseda Atocha (2006): *ABDUCTIVE REASONING*. *Synthese Library* 330, Kluwer Academic Publishers, Dordrecht, doi:10.1007/1-4020-3907-7.

[2] Haniel Barbosa, Chantal Keller, Andrew Reynolds, Arjun Viswanathan, Cesare Tinelli & Clark Barrett (2023): *An Interactive SMT Tactic in Coq using Abductive Reasoning*. In: *EPiC Series in Computing*, 94, EasyChair, pp. 11–22, doi:10.29007/432m. ISSN: 2398-7340.

[3] Michael Beeson, Julien Narboux & Freek Wiedijk (2019): *Proof-checking Euclid*. *Annals of Mathematics and Artificial Intelligence* 85(2-4), pp. 213–257, doi:10.1007/s10472-018-9606-x.

[4] Marc Bezem & Thierry Coquand (2005): *Automating Coherent Logic*. In Geoff Sutcliffe & Andrei Voronkov, editors: *12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning — LPAR 2005*, *Lecture Notes in Computer Science* 3835, Springer, pp. 246–260, doi:10.1007/11591191_18.

[5] Jeremy Bongio, Cyrus Katrak, Hai Lin, Christopher Lynch & Ralph Eric McGregor (2008): *Encoding First Order Proofs in SMT*. *Electron. Notes Theor. Comput. Sci.* 198(2), pp. 71–84, doi:10.1016/j.entcs.2008.04.081.

[6] Gabriel Braun & Julien Narboux (2012): *From Tarski to Hilbert*. In Tetsuo Ida & Jacques Fleuriot, editors: *Post-proceedings of Automated Deduction in Geometry 2012*, *LNCS* 7993, Springer, pp. 89–109, doi:10.1007/978-3-642-40672-0_7.

[7] XueFeng Chen & DingKang Wang (2004): *The Projection of Quasi Variety and Its Application on Geometric Theorem Proving and Formula Deduction*. In *Automated Deduction in Geometry, 4th International Workshop, ADG 2002*, *Lecture Notes in Computer Science* 2930, Springer, pp. 21–30, doi:10.1007/978-3-540-24616-9_2.

[8] Marc Denecker & Antonis C. Kakas (2002): *Abduction in Logic Programming*. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part I*, *Lecture Notes in Computer Science* 2407, Springer, pp. 402–436, doi:10.1007/3-540-45628-7_16.

[9] Todd Deshane, Wenjin Hu, Patty Jablonski, Hai Lin, Christopher Lynch & Ralph Eric McGregor (2007): *Encoding First Order Proofs in SAT*. In *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction*, *Lecture Notes in Computer Science* 4603, Springer, pp. 476–491, doi:10.1007/978-3-540-73595-3_35.

[10] Isil Dillig & Thomas Dillig (2013): *Explain: A Tool for Performing Abductive Inference*. In *Computer Aided Verification*, Lecture Notes in Computer Science, Springer, pp. 684–689, doi:10.1007/978-3-642-39799-8_-46.

[11] Viviane Durand-Guerrier, Paolo Boero, Nadia Douek, Susanna S. Epp & Denis Tanguay (2012): *Examining the Role of Logic in Teaching Proof*. In *Proof and Proving in Mathematics Education*, *New ICMI Study Series* 15, Springer, pp. 369–389, doi:10.1007/978-94-007-2129-6_16.

[12] Roy Dyckhoff & Sara Negri (2015): *Geometrization of first-order logic*. *The Bulletin of Symbolic Logic* 21, pp. 123–163, doi:10.1017/bsl.2015.7.

[13] Thomas Hales (2019): *An argument for controlled natural languages in mathematics*. Available at `https://jiggerwit.wordpress.com/2019/06/20/an-argument-for-controlled-natural-languages-in-mathematics/`.

[14] Predrag Janičić & Julien Narboux (2022): *Theorem Proving as Constraint Solving with Coherent Logic*. *Journal of Automated Reasoning* 66(4), pp. 689–746, doi:10.1007/s10817-022-09629-z.

[15] Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu & Guillaume Lample (2023): *Draft, Sketch, and Prove: Guiding Formal Theorem Provers with Informal Proofs*, doi:10.48550/arXiv.2210.12283. ArXiv:2210.12283 [cs].

[16] Laura Kovács & Andrei Voronkov (2013): *First-Order Theorem Proving and Vampire*. In *Computer Aided Verification - 25th International Conference, CAV 2013*, *Lecture Notes in Computer Science* 8044, Springer, pp. 1–35, doi:10.1007/978-3-642-39799-8_1.

[17] P. Marquis (1991): *Extending abduction from propositional to first-order logic*. In *Fundamentals of Artificial Intelligence Research*, Springer, doi:10.1007/3-540-54507-7_12.

[18] Ralph Eric McGregor (2011): *Automated Theorem Proving Using SAT*. PhD Thesis, Clarkson University. Available at `https://search.proquest.com/openview/b87467cab0987f591010cf19dc554fa3/1?pq-origsite=gscholar&cbl=18750&diss=y`.

[19] Julien Narboux & Viviane Durand-Guerrier (2022): *Combining pencil/paper proofs and formal proofs, a challenge for Artificial Intelligence and mathematics education*. In: *Mathematics Education in the Age of Artificial Intelligence*, *Mathematics Education in the Digital Era* 17, Springer, doi:10.1007/978-3-030-86909-0_8.

[20] Charles Peirce (1932): *Collected papers of Charles Sanders Peirce*. Belknap Press.

[21] Andrew Polonsky (2011): *Proofs, Types and Lambda Calculus*. Ph.D. thesis, University of Bergen.

[22] T. Recio & M. P. Vélez (1999): *Automatic Discovery of Theorems in Elementary Geometry*. *J. Autom. Reason.* 23(1), pp. 63–82, doi:10.1023/A:1006135322108.

[23] Andrew Reynolds, Haniel Barbosa, Daniel Larraz & Cesare Tinelli (2020): *Scalable Algorithms for Abduction via Enumerative Syntax-Guided Synthesis*. In *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Part I*, *Lecture Notes in Computer Science* 12166, Springer, pp. 141–160, doi:10.1007/978-3-030-51074-9_9.

[24] Philippe R. Richard, Josep Maria Fortuny, Michel Gagnon, Nicolas Leduc, Eloi Puertas & Michèle Tessier-Baillargeon (2011): *Didactic and theoretical-based perspectives in the experimental development of an intelligent tutorial system for the learning of geometry*. *ZDM* 43(3), pp. 425–439, doi:10.1007/s11858-011-0320-y.

[25] Alessandra Russo & Bashar Nuseibeh (2001): *On The Use Of Logical Abduction In Software Engineering*. In *Handbook of Software Engineering and Knowledge Engineering*, doi:10.1142/9789812389718_0037.

[26] Wolfram Schwabhäuser, Wanda Szmielew & Alfred Tarski (1983): *Metamathematische Methoden in der Geometrie*. Springer. doi:10.1007/978-3-642-69418-9.

[27] Sana Stojanović, Julien Narboux, Marc Bezem & Predrag Janičić (2014): *A Vernacular for Coherent Logic*. In *Intelligent Computer Mathematics*, *Lecture Notes in Computer Science* 8543, Springer, pp. 388–403, doi:10.1007/978-3-319-08434-3_28.

[28] Wen-Tsun Wu (1978): *On the Decision Problem and the Mechanization of Theorem-Proving in Elementary Geometry*. 21, Scientia Sinica, pp. 157–179.

7 Appendix

In this appendix, we provide a complete list of lemmas and axioms (in coherent logic form) used in our examples, and the results obtained using Larus. The results were obtained on a PC computer with Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz processor running under Linux (the time spent should give just a general picture of the efficiency of the system).

7.1 Problem 1: Varignon's Theorem

The TPTP file used for Problem 1 is the following:

```
fof(triangle_mid_par_strict, axiom, (! [A, B, C, P, Q] : ( ((~ col(A,B,C)) & midpoint(B,P
    ,C) & midpoint(A,Q,C)) => par(A,B,Q,P)))).
fof(lemma_par_trans, axiom, (! [A, B, C, D, E, F] : ((par(A,B,C,D) & par(C,D,E,F) & (~col
    (A,B,E))) => par(A,B,E,F)))).
```

```
fof(defparallelogram2,axiom, (! [A,B,C,D] : ((par(A,B,C,D) & par(A,D,B,C)) => ((pG(A,B,C,
    D)))))).
fof(lemma_parallelNC,axiom, (! [A,B,C,D] : ((par(A,B,C,D)) => ((~ (col(A,B,C)) & ~ (col(A
    ,C,D)) & ~ (col(B,C,D)) & ~ (col(A,B,D))))))).
fof(lemma_parallelflip,axiom, (! [A,B,C,D] : ((par(A,B,C,D)) => ((par(B,A,C,D) & par(A,B,
    D,C) & par(B,A,D,C)))))).
fof(lemma_parallelsymmetric,axiom, (! [A,B,C,D] : ((par(A,B,C,D)) => ((par(C,D,A,B)))))).
fof(midpoint_sym, axiom, (! [A, B, I] : (midpoint(A,I,B) => midpoint(B,I,A)))).
fof(lemma_tP_trans, axiom,  (! [A, B, C, D, E, F] : ((tP(A,B,C,D) & tP(C,D,E,F)) => tP(A,
    B,E,F)))).

fof(th_varignon,conjecture,(! [A,B,C,D,E,F,G,H] : (( (~(col(B,D,A))) & (~(col(B,D,C))) &
    (~(col(A,C,B))) & (~(col(A,C,D))) & (~ (col(E,F,G))) & midpoint(A,E,B) & midpoint(B,F
    ,C) & midpoint(C,G,D) & midpoint(A,H,D)) => pG(E,F,G,H) ))).
```

If Larus is invoked as: `./larus -l100 -m8` (`-l100` means the time limit is 100s, `-m8` means that we look for a proof with 8 or fewer steps), it produces the following proof in 2s:

Consider arbitrary a, b, c, d, e, f, g, h such that:

- $\neg col(b,d,a)$,
- $\neg col(b,d,c)$,
- $\neg col(a,c,b)$,
- $\neg col(a,c,d)$,
- $\neg col(e,f,g)$,
- $b \neq d$,
- $a \neq c$,
- $midpoint(a,e,b)$,
- $midpoint(b,f,c)$,
- $midpoint(c,g,d)$,
- $midpoint(a,h,d)$.

It should be proved that $pG(e,f,g,h)$.

1. $par(a,c,h,g)$ (by MP, from $\neg col(a,c,d)$, $midpoint(c,g,d)$, $midpoint(a,h,d)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto d$, $P \mapsto g$, $Q \mapsto h$)
2. $par(b,d,f,g)$ (by MP, from $\neg col(b,d,c)$, $midpoint(c,g,d)$, $midpoint(b,f,c)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto b$, $B \mapsto d$, $C \mapsto c$, $P \mapsto g$, $Q \mapsto f$)
3. $par(a,c,e,f)$ (by MP, from $\neg col(a,c,b)$, $midpoint(b,f,c)$, $midpoint(a,e,b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto b$, $P \mapsto f$, $Q \mapsto e$)
4. $par(b,d,e,h)$ (by MP, from $\neg col(b,d,a)$, $midpoint(a,h,d)$, $midpoint(a,e,b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto b$, $B \mapsto d$, $C \mapsto a$, $P \mapsto h$, $Q \mapsto e$)
5. $par(e,f,g,h)$ (by MP, from $par(a,c,e,f)$, $par(a,c,h,g)$, $\neg col(e,f,g)$ using axiom lemma_par_trans; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto a$, $D \mapsto c$, $E \mapsto g$, $F \mapsto h$)

6. $par(f,g,h,e)$ (by MP, from $par(b,d,f,g)$, $par(b,d,e,h)$, $par(e,f,g,h)$ using axiom lemma_par_trans; instantiation: $A \mapsto f$, $B \mapsto g$, $C \mapsto d$, $D \mapsto b$, $E \mapsto h$, $F \mapsto e$)
7. $pG(e,f,g,h)$ (by MP, from $par(e,f,g,h)$, $par(f,g,h,e)$ using axiom defparallelogram2; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto g$, $D \mapsto h$)
8. Proved by assumption! (by QEDas)

7.2 Problem 2: First Inverse Problem

The list of axioms used for the first inverse problem (Problem 2) is the same as in Section 7.1. Only the conjecture is different – the assumption `midpoint(A,H,D)` is ommitted:

```
fof(th_varignon,conjecture,(! [A,B,C,D,E,F,G,H] : (( (~(col(B,D,A))) & (~(col(B,D,C))) &
    (~(col(A,C,B))) & (~(col(A,C,D))) & (~ (col(E,F,G))) & (B != D) & (A != C) & midpoint
    (A,E,B) & midpoint(B,F,C) & midpoint(C,G,D)) => pG(E,F,G,H) ))).
```

If Larus is invoked as: `./larus -l100 -m8 -b1` (`-l100` means the time limit is 100s, `-m8` means that we look for a proof with 8 or fewer steps, `-b1` means that we look for one atomic formula as an abduct), it finds a first consistent abduct (after two inconsistent ones) and produces the following human-readable proof in 3.26 seconds (the abduct found is highlighted):

Consider arbitrary a, b, c, d, e, f, g, h such that:

- $\neg col(b,d,a)$,
- $\neg col(b,d,c)$,
- $\neg col(a,c,b)$,
- $\neg col(a,c,d)$,
- $\neg col(e,f,g)$,
- $b \neq d$,
- $a \neq c$,
- $midpoint(a,e,b)$,
- $midpoint(b,f,c)$,
- $midpoint(c,g,d)$.

It should be proved that $pG(e,f,g,h)$.

Abducts found:

- $midpoint(d,h,a)$

1. $par(a,c,e,f)$ (by MP, from $\neg col(a,c,b)$, $midpoint(b,f,c)$, $midpoint(a,e,b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto b$, $P \mapsto f$, $Q \mapsto e$)
2. $par(b,d,f,g)$ (by MP, from $\neg col(b,d,c)$, $midpoint(c,g,d)$, $midpoint(b,f,c)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto b$, $B \mapsto d$, $C \mapsto c$, $P \mapsto g$, $Q \mapsto f$)

3. $par(b,d,e,h)$ (by MP, from $\neg col(b,d,a)$, $midpoint(d,h,a)$, $midpoint(a,e,b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto b$, $B \mapsto d$, $C \mapsto a$, $P \mapsto h$, $Q \mapsto e$)
4. $par(a,c,h,g)$ (by MP, from $\neg col(a,c,d)$, $midpoint(c,g,d)$, $midpoint(d,h,a)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto d$, $P \mapsto g$, $Q \mapsto h$)
5. $par(e,f,g,h)$ (by MP, from $par(a,c,e,f)$, $par(a,c,h,g)$, $\neg col(e,f,g)$ using axiom lemma_par_trans; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto a$, $D \mapsto c$, $E \mapsto g$, $F \mapsto h$)
6. $par(e,h,g,f)$ (by MP, from $par(b,d,e,h)$, $par(b,d,f,g)$, $par(e,f,g,h)$ using axiom lemma_par_trans; instantiation: $A \mapsto e$, $B \mapsto h$, $C \mapsto b$, $D \mapsto d$, $E \mapsto g$, $F \mapsto f$)
7. $pG(e,f,g,h)$ (by MP, from $par(e,f,g,h)$, $par(e,h,g,f)$ using axiom defparallelogram2; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto g$, $D \mapsto h$)
8. Proved by assumption! (by QEDas)

7.3 Problem 3: Second Inverse Problem

The list of axioms used for the second inverse problem (Problem 3) is the same as in section 7.1, extended with the following axioms.

```
fof(defmidpoint,axiom, (! [A,B,C] : ((midpoint(A,B,C)) => ((betS(A,B,C) & cong(A,B,B,C)))
    ))).
fof(defmidpoint2,axiom, (! [A,B,C] : ((betS(A,B,C) & cong(A,B,B,C)) => ((midpoint(A,B,C))
    )))).
fof(midpoint_NC, axiom, (! [A, B, I] : ((midpoint(A,I,B) & (A != B)) => ( (A != I) & ( B
    != I))))).
fof(defrectangle,axiom, (! [A,B,C,D] : ((rectangle(A,B,C,D)) => ((pG(A,B,C,D) & per(A,B,C
    ) & per(B,C,D) & per(C,D,A) & per(D,A,B)))))).
fof(defrectangle2a,axiom, (! [A,B,C,D] : ((pG(A,B,C,D) & per(A,B,C)) => rectangle(A,B,C,D
    )))).
fof(defrectangle2b,axiom, (! [A,B,C,D] : ((pG(A,B,C,D) & per(B,C,D)) => rectangle(A,B,C,D
    )))).
fof(defrectangle2c,axiom, (! [A,B,C,D] : ((pG(A,B,C,D) & per(C,D,A)) => rectangle(A,B,C,D
    )))).
fof(defrectangle2d,axiom, (! [A,B,C,D] : ((pG(A,B,C,D) & per(D,A,B)) => rectangle(A,B,C,D
    )))).
fof(defrectangle2e,axiom, (! [A,B,C,D] : ((per(A,B,C) & per(B,C,D) & per(C,D,A) & per(D,A
    ,B)) => rectangle(A,B,C,D)))).
%fof(defrectangle3a,axiom, (! [A,B,C,D] : (? [X] : ((rectangle(A,B,C,D)) => cong(A,C,B,D)
     & midpoint(A,X,C) & midpoint(B,X,D))))).
fof(defrectangle3b,axiom, (! [A,B,C,D,X] : ((cong(A,C,B,D) & midpoint(A,X,C) & midpoint(B
    ,X,D)) => rectangle(A,B,C,D)))).
fof(defrectangle4a,axiom, (! [A,B,C,D] : ((rectangle(A,B,C,D)) => (pG(A,B,C,D) & cong(A,C
    ,B,D))))).
fof(defrectangle4b,axiom, (! [A,B,C,D] : ((pG(A,B,C,D) & cong(A,C,B,D)) => rectangle(A,B,
    C,D)))).
fof(lemma_8_2,axiom, (! [A,B,C] : ((per(A,B,C)) => ((per(C,B,A)))))).
fof(varignon_th,axiom,(! [A,B,C,D,E,F,G,H] : (( (~(col(B,D,A))) & (~(col(B,D,C))) & (~(
    col(A,C,B))) & (~(col(A,C,D))) & (~ (col(G,F,E))) & (B != D) & (A != C) & midpoint(A,
    E,B) & midpoint(B,F,C) & midpoint(C,G,D) & midpoint(A,H,D)) => pG(E,F,G,H) ))).
```

The conjecture is also different – the goal is to find under which assumption the quadrilateral $EFGH$ is a rectangle.

```
fof(th_varignon_rect,conjecture,(! [A,B,C,D,E,F,G,H] : (( (~(col(B,D,A))) & (~(col(B,D,C)
    )) & (~(col(A,C,B))) & (~(col(A,C,D))) & (~ (col(G,F,E))) & (B != D) & (A != C) &
    midpoint(A,E,B) & midpoint(B,F,C) & midpoint(C,G,D) & midpoint(A,H,D)) => rectangle(E
    ,F,G,H) ))).
```

If Larus is invoked as: `./larus -l100 -m8 -b1`, it produces the following human-readable proof in 14s (the abduct found is highlighted):

Consider arbitrary a, b, c, d, e, f, g, h such that:

- $\neg col(b,d,a)$,
- $\neg col(b,d,c)$,
- $\neg col(a,c,b)$,
- $\neg col(a,c,d)$,
- $\neg col(f,g,e)$,
- $b \neq d$,
- $a \neq c$,
- $midpoint(a,e,b)$,
- $midpoint(b,f,c)$,
- $midpoint(c,g,d)$,
- $midpoint(a,h,d)$.

It should be proved that $rectangle(e,f,g,h)$.

Abducts found:

- $cong(e,g,h,f)$

1. $midpoint(b,e,a)$ (by MP, from $midpoint(a,e,b)$, $midpoint(a,e,b)$ using axiom defmidpoint2; instantiation: $A \mapsto b$, $B \mapsto e$, $C \mapsto a$)
2. $pG(e,f,g,h)$ (by MP, from $\neg col(b,d,a)$, $\neg col(b,d,c)$, $\neg col(a,c,b)$, $\neg col(a,c,d)$, $\neg col(f,g,e)$, $b \neq d$, $a \neq c$, $midpoint(b,e,a)$, $midpoint(b,f,c)$, $midpoint(c,g,d)$, $midpoint(a,h,d)$ using axiom varignon_th; instantiation: $A \mapsto a$, $B \mapsto b$, $C \mapsto c$, $D \mapsto d$, $I \mapsto e$, $J \mapsto f$, $K \mapsto g$, $L \mapsto h$)
3. $rectangle(e,f,g,h)$ (by MP, from $pG(e,f,g,h)$, $cong(e,g,h,f)$ using axiom defrectangle4b; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto g$, $D \mapsto h$)
4. $rectangle(e,f,g,h)$ (by MP, from $rectangle(e,f,g,h)$, $rectangle(e,f,g,h)$, $rectangle(e,f,g,h)$, $rectangle(e,f,g,h)$ using axiom defrectangle2e; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto g$, $D \mapsto h$)
5. Proved by assumption! (by QEDas)

7.4 Problem 4: Partially Specified Goal

The list of axioms used for Problem 4 is the same as presented in Section 7.1. Only the conjecture is different: the goal does not have the predicate symbol specified:

```
fof(th_varignon,conjecture,(! [A,B,C,D,E,F,G,H] : (( (~(col(B,D,A))) & (~(col(B,D,C))) &
    (~(col(A,C,B))) & (~(col(A,C,D))) & (~ (col(E,F,G))) & (B != D) & (A != C) & midpoint
    (A,E,B) & midpoint(B,F,C) & midpoint(C,G,D) & midpoint(A,H,D)) => _(E,F,G,H) ))).
```

If Larus is invoked as `./larus -l100 -m8`, it produces the following human-readable proof (for the goal $par(e, f, g, h)$, highlighted in the proof) in 2s:

Consider arbitrary a, b, c, d, e, f, g, h such that:

- $\neg col(b,d,a)$,
- $\neg col(b,d,c)$,
- $\neg col(a,c,b)$,
- $\neg col(a,c,d)$,
- $\neg col(e,f,g)$,
- $b \neq d$,
- $a \neq c$,
- $midpoint(a,e,b)$,
- $midpoint(b,f,c)$,
- $midpoint(c,g,d)$,
- $midpoint(a,h,d)$.

It should be proved that $\_(e,f,g,h)$.

1. $par(a,c,e,f)$ (by MP, from $\neg col(a,c,b)$, $midpoint(b,f,c)$, $midpoint(a,e,b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto b$, $P \mapsto f$, $Q \mapsto e$)
2. $par(a,c,h,g)$ (by MP, from $\neg col(a,c,d)$, $midpoint(c,g,d)$, $midpoint(a,h,d)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto d$, $P \mapsto g$, $Q \mapsto h$)
3. $par(e,f,g,h)$ (by MP, from $par(a,c,e,f)$, $par(a,c,h,g)$, $\neg col(e,f,g)$ using axiom lemma_par_trans; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto a$, $D \mapsto c$, $E \mapsto g$, $F \mapsto h$)
4. Proved by assumption! (by QEDas)

7.5 Problem 5: Partially Specified Proof

The list of axioms used for Problem 5 is as presented in Section 7.3 (with the axiom `defrectangle3a` deleted). The conjecture is the same plus the abduct as an assumption, but we add the following hint:

```
fof(hint1,hint,_,_,defrectangle4b(4,5,6,7)).
```

If Larus is invoked as `./larus -l100 -m8`, it produces the same proof as in Section 7.3 in 4s, while if the hint is omitted, it takes 5s.

Using GXWeb for Theorem Proving and Mathematical Modelling

Philip Todd
Saltire Software
Portland OR USA
philt@saltire.com

Danny Aley
Saltire Software
Portland OR USA
dannya@saltire.com

GXWeb is the free browser based version of the symbolic geometry software Geometry Expressions. We demonstrate its use in an educational setting with examples from theorem proving, mathematical modelling and loci and envelopes.

1 Introduction

One approach to introducing automated geometrical deduction into an educational environment is to take an existing well accepted Dynamic Geometry System (DGS) and create add-on modules which embody algorithms in automated geometry theorem proving [1]. An advantage of this approach is that new UI can be added incrementally to a familiar DGS, reducing the overhead for the user of adopting a new tool. An advantage of using GeoGebra specifically as a platform is that its open source nature allows researchers to focus on the geometry theorem automation, leaving the broader geometry interface to others.

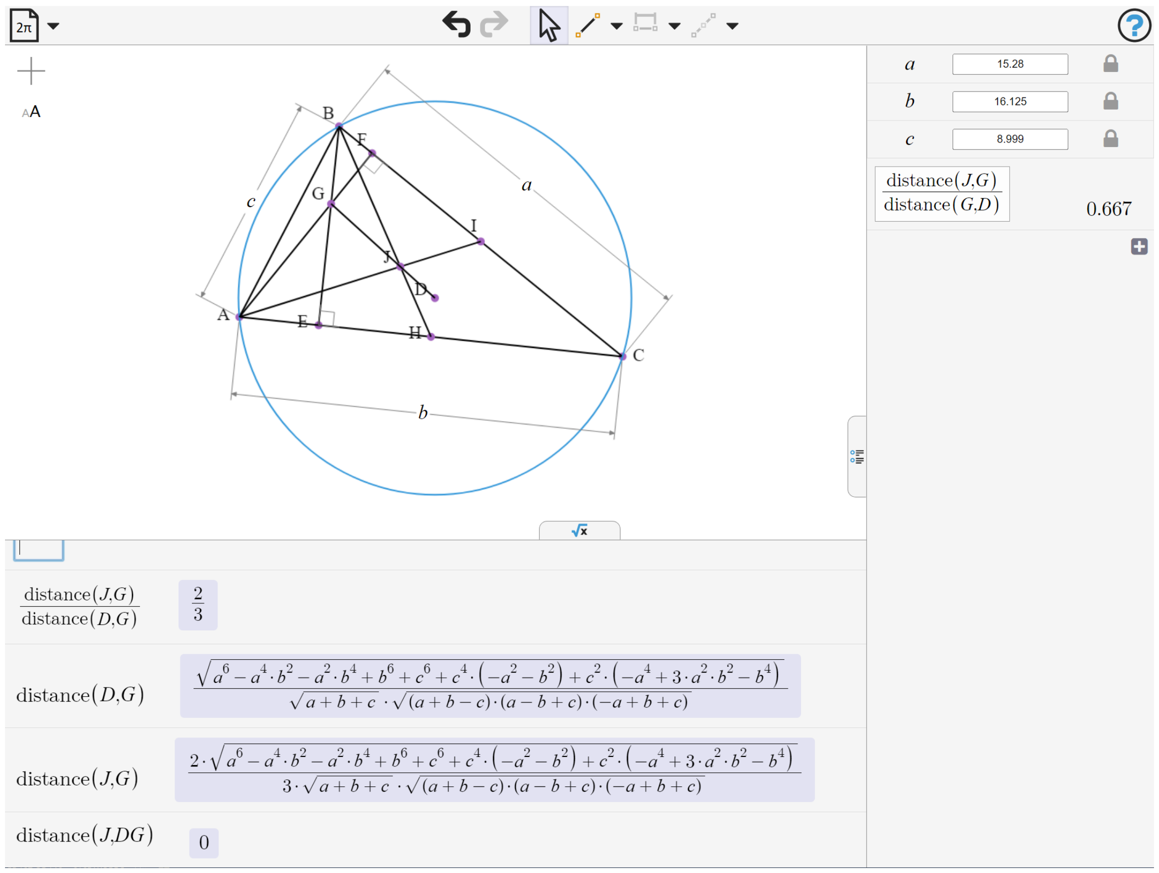
In this presentation, we demonstrate software with a fundamentally different architecture, which achieves many of the same goals, but with its own distinct advantages and disadvantages. We focus on the use of GXWeb in education in the context of geometry theorem proving, and two problems in applied optics.

2 GXWeb

GXWeb [2] is the free browser based version of the symbolic geometry system Geometry Expressions-[3]. It maintains both a numeric model of the geometry (similar to that maintained by a typical DGS), but in parallel it also maintains a symbolic model. User interface is provided (Figure 1) which allows both models to be accessed. Symbolic inputs are facilitated by a constraint based layer which sits on top of the underlying DGS. This layer allows distances and angles to be assigned symbolic values in a very natural way. In parallel to this, a numeric value is maintained for each indeterminate in the symbolic model. These values control the relation between the symbolic model and the diagram. Numeric values assigned to variables may be modified in the numeric panel (Figure 1) and output measurements from the numeric model displayed.

The symbolic panel provides access to the symbolic model. Measurements made in this panel are symbolic rather than numeric. In a theorem proving context, we may use the fact that the distance between a point and a line is symbolically 0 as proof that the point lies on the line. The symbolic panel can do more than verify relations, however, when non trivial formulas are generated, the user can migrate from exploration into interactive theorem discovery.

To create a locus in GXWeb, one needs to select a point and specify which parameter should vary. For an envelope one needs instead to select a line or line segment.



$\dfrac{\text{distance}(J,G)}{\text{distance}(G,D)}$ 0.667

$\dfrac{\text{distance}(J,G)}{\text{distance}(D,G)}$	$\dfrac{2}{3}$
$\text{distance}(D,G)$	$\dfrac{\sqrt{a^6-a^4\cdot b^2-a^2\cdot b^4+b^6+c^6+c^4\cdot\left(-a^2-b^2\right)+c^2\cdot\left(-a^4+3\cdot a^2\cdot b^2-b^4\right)}}{\sqrt{a+b+c}\cdot\sqrt{(a+b-c)\cdot(a-b+c)\cdot(-a+b+c)}}$
$\text{distance}(J,G)$	$\dfrac{2\cdot\sqrt{a^6-a^4\cdot b^2-a^2\cdot b^4+b^6+c^6+c^4\cdot\left(-a^2-b^2\right)+c^2\cdot\left(-a^4+3\cdot a^2\cdot b^2-b^4\right)}}{3\cdot\sqrt{a+b+c}\cdot\sqrt{(a+b-c)\cdot(a-b+c)\cdot(-a+b+c)}}$
$\text{distance}(J,DG)$	0

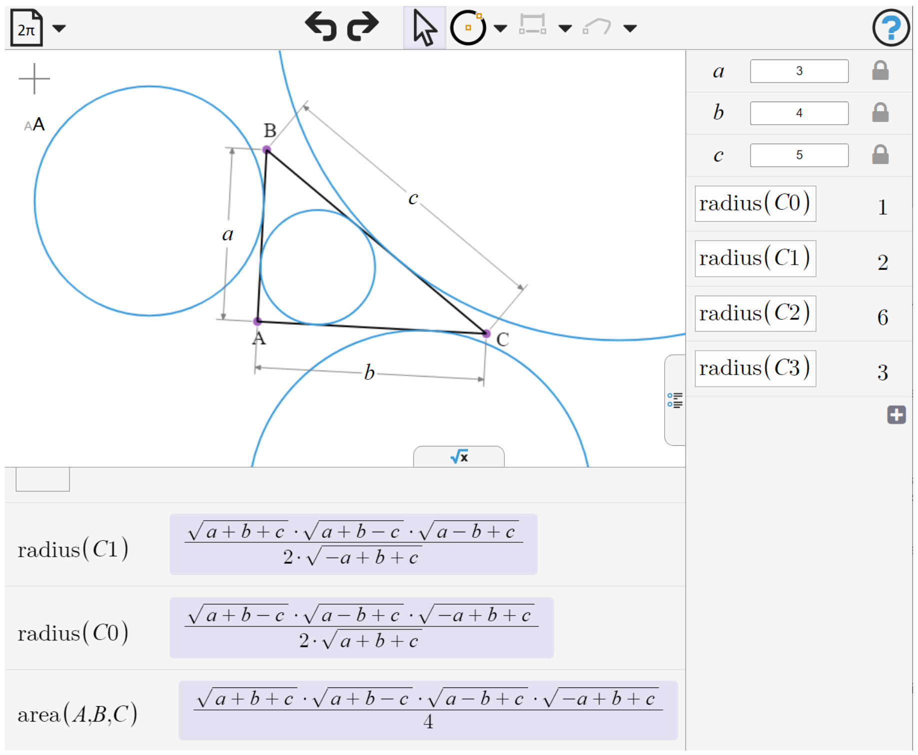
Figure 1: The circumcenter D, incenter J and orthocenter G of triangle ABC are constructed on the diagram. The numeric panel, to the right shows values for lengths a, b, c, and for the ratio $|JG|/|GD|$ The symbolic panel, below shows exact values for the distance between J and GD, along with $|DG|$, $|JG|$ and their ratio.

In this presentation we demonstrate these features in four educational explorations. In the first we look at the classical geometry result involving the relative positions of three triangle centers. We see first how GXWeb can be used in the role of proving a postulated theorem. We then examine how the symbolic features can be used to suggest further avenues of exploration. In the second example, we use the numeric model to make hypotheses about tritangent radii for Pythagorean triangles. These hypotheses can be confirmed in the symbolic panel. In the third example we look at a geometrical model of a box solar cooker and use GXWeb to help us find the best angle to open the lid. Finally we pursue an exploration of the caustic curves caused by reflection in a cylinder (the famous coffee cup caustic).

3 Examples

3.1 Euler Line

In figure 1, the sides of a triangle ABC are specified by symbolic lengths a, b and c, displayed as dimension symbols on the diagram. The numeric values of these variables are displayed in the numeric panel to the right. The orthocenter is constructed as the intersection between two altitudes, the incenter as the



a	3	
b	4	
c	5	
radius($C0$)		1
radius($C1$)		2
radius($C2$)		6
radius($C3$)		3

radius($C1$) $\dfrac{\sqrt{a+b+c}\cdot\sqrt{a+b-c}\cdot\sqrt{a-b+c}}{2\cdot\sqrt{-a+b+c}}$

radius($C0$) $\dfrac{\sqrt{a+b-c}\cdot\sqrt{a-b+c}\cdot\sqrt{-a+b+c}}{2\cdot\sqrt{a+b+c}}$

area(A,B,C) $\dfrac{\sqrt{a+b+c}\cdot\sqrt{a+b-c}\cdot\sqrt{a-b+c}\cdot\sqrt{-a+b+c}}{4}$

Figure 2: Triangle *ABC* has side lengths a, b, c. The incircle and the three excircles are constructed. Numerical radii are shown for these circles where $a = 3$, $b = 4$, $c = 5$. Symbolic values for two of the radii and for the area of the triangle are shown.

intersection between two medians, and the circumcenter as the center of the constructed circumcircle. To prove that the three centers are collinear, the user should join two of the points, and inquire, symbolically, for the distance between the third center and this line. A result of 0 proves the collinearity.

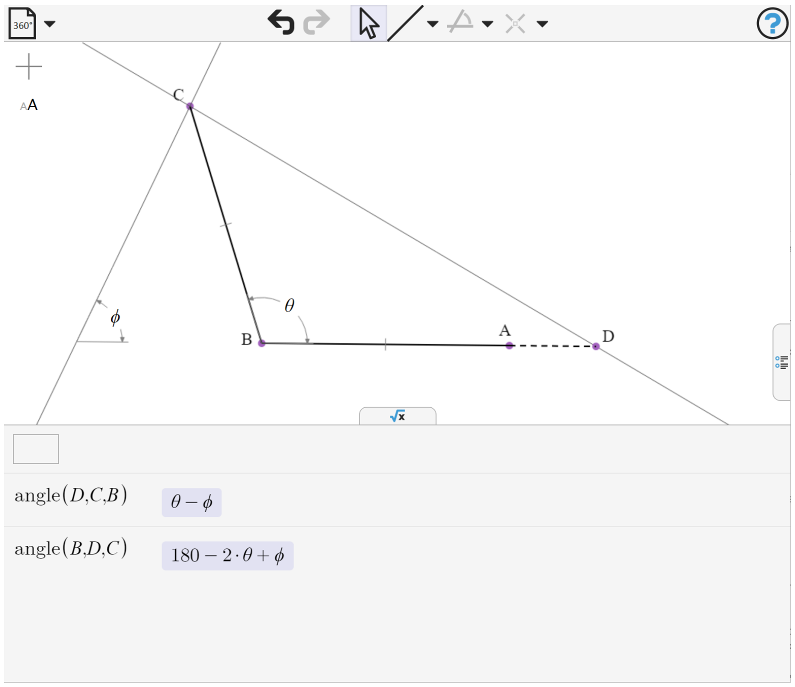
The distances between pairs of centers are complicated formulas involving a, b and c. However, their ratio is simple. Viewed in the symbolic panel, it is a constant. Viewed in the numeric panel, its value does not vary as the numeric values of the indeterminates vary.

3.2 Tritangent Circles and Pythagorean Triples

Numerical experimentation with radii of tritangent circles (Figure 2) leads to a conjecture that for Pythagorean triples, these circles have integer radii. Examination of the formulas for the radii gives a starting point for a proof of this conjecture. It also suggests that multiplying the four radii will provide a significant simplification. This leads to a further conjecture which may be tested numerically, and proved symbolically.

3.3 Box Solar Cooker

A simple solar cooker is a box with a reflective lid [4]. The lid is held open at some angle, and reflected rays are captured in the box. An interesting theoretical question with this simple apparatus is this: for given angle of the sun, and assuming the base of the box is on level ground, what is the best angle for the



angle(D,C,B) $\theta-\phi$

angle(B,D,C) $180-2\cdot\theta+\phi$

Figure 3: A geometric model of a box solar cooker with incident ray at angle ϕ to the horizontal and box lid open at angle θ.

lid.

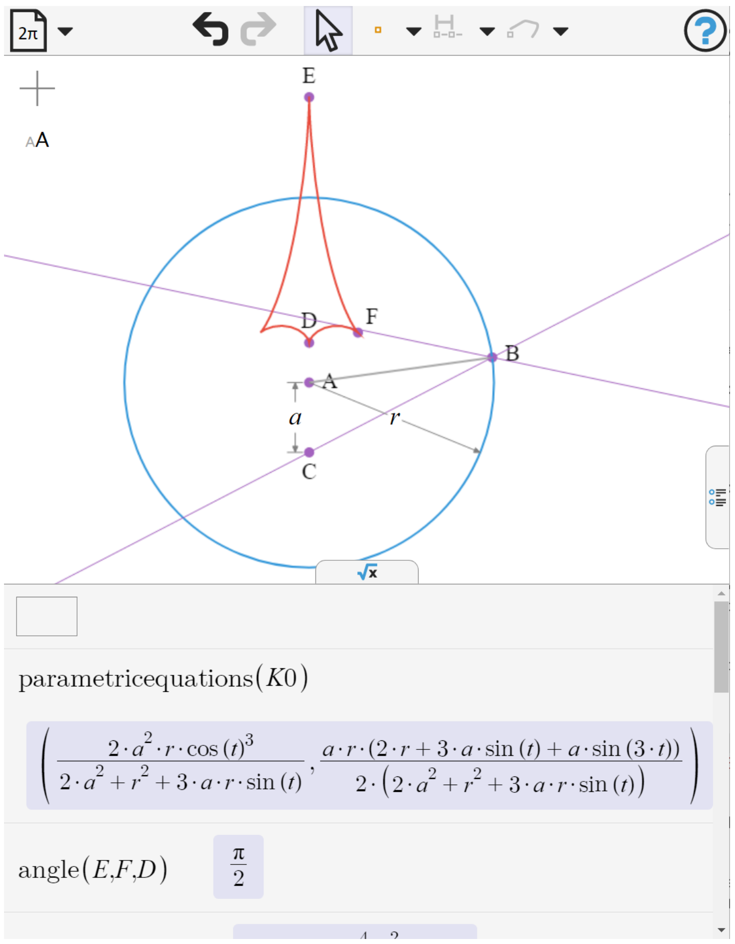
A geometrical model assumes the sun is at angle ϕ to the horizontal and the lid is opened at angle θ. The angle between the ray reflected from the end of the lid to the base can be derived from the model. Some consideration leads to the postulate that the best angle should reflect light from the outer edge of the lid to the outer edge of the box, thus capturing as many rays as possible, but not spilling any. The angle value generated by GXWeb can be used to set up a linear equation whose solution yields the ideal angle.

3.4 Coffee Cup Caustic

A catacaustic of a curve is the envelope of the reflected rays from some point source (or from a family of parallel rays). Figure 4 shows a geometric construction of this curve. Where the light source lies on the circumference, we can derive an implicit equation for the curve. For general position of the curve, the parametric equations are more helpful. By grasping the correspondence between the location on the curve and the parameter of the point of reflection on the circle, students can conjecture the location of points on the cusps of the curve, and discover geometric relations between these points.

4 Conclusion

GXWeb has some distinct features which can be advantageous in an educational setting. The constraint based user interface provides a clean way of setting up and specifying exact geometry problems. The parallel symbolic and numeric model provide flexibility in moving between a numeric and an algebraic



parametricequations($K0$)

$$\left(\frac{2\cdot a^2\cdot r\cdot\cos(t)^3}{2\cdot a^2+r^2+3\cdot a\cdot r\cdot\sin(t)},\frac{a\cdot r\cdot(2\cdot r+3\cdot a\cdot\sin(t)+a\cdot\sin(3\cdot t))}{2\cdot\left(2\cdot a^2+r^2+3\cdot a\cdot r\cdot\sin(t)\right)}\right)$$

angle(E,F,D) $\frac{\pi}{2}$

Figure 4: The caustic curve due to reflection in a circle of light emanating from point C inside the circle.

view of a problem. Loci and envelopes are cleanly defined in terms of a single varying parameter. Both implicit and parametric equations of the curve may be computed. Both have their use in different circumstances.

References

[1] Francisco Botana, Markus Hohenwarter, Predrag Janičić, Zoltán Kovács, Ivan Petrović, Tomás Recio & Simon Weitzhofer (2015): *Automated theorem proving in GeoGebra: Current achievements*. Journal of Automated Reasoning 55(1), pp. 39–59, doi:10.1007/s10817-015-9326-4.

[2] Saltire Software Inc.: *GXWeb*. Available at `https://www.geometryexpressions.com/GXWeb`.

[3] Philip Todd (2020): *A Symbolic Dynamic Geometry System Using the Analytical Geometry Method*. Mathematics in Computer Science 14(4), pp. 693–726, doi:10.1007/s10472-023-09841-6.

[4] Philip Todd, Aleta Kandle & Ariel Chen (2010): *Aligning Solar Cookers: a case study in the use of symbolic geometry and CAS to investigate a real world problem*. Electronic Journal of Mathematics & Technology 4(1).

Showing Proofs, Assessing Difficulty with GeoGebra Discovery[*]

Zoltán Kovács
The Private University College of Education of the Diocese of Linz, Austria
zoltan.kovacs@ph-linz.at

Tomás Recio
Escuela Politécnica Superior, Universidad Antonio de Nebrija, Madrid, Spain
trecio@nebrija.es

M. Pilar Vélez
Escuela Politécnica Superior, Universidad Antonio de Nebrija, Madrid, Spain
pvelez@nebrija.es

In our contribution we describe some on-going improvements concerning the Automated Reasoning Tools developed in GeoGebra Discovery, providing different examples of the performance of these new features. We describe the new `ShowProof` command, that outputs both the sequence of the different steps performed by GeoGebra Discovery to confirm a certain statement, as well as a number intending to grade the difficulty or interest of the assertion. The proposal of this assessment measure, involving the comparison of the expression of the thesis (or conclusion) as a combination of the hypotheses, will be developed.

1 Introduction

In the past years we have been developing and including, both in the standard version of GeoGebra (GG) as well as in the fork version GeoGebra Discovery,[1] different automated reasoning tools. See [1] for a general description and references.

The goal of the current contribution is to present some ongoing work regarding two different, but related, important improvements of GeoGebra Discovery. One, to visualize the different steps that GG Discovery performs with a given geometric statement until it declares its truth (or failure). Two, to test, through different elementary examples, the suitability of an original proposal to evaluate the interest, complexity or difficulty of a given statement. Let us advance that our proposal involves the notion of syzygy of a set of polynomials.

The relevance of showing details about each of the steps performed by our automated reasoning algorithms implemented in GG Discovery is quite evident. In fact, as a consequence of the result in [2], describing the formalization of the arithmetization of Euclidean plane geometry, proofs of geometric statements obtained using algebraic geometry algorithms are also valid on the synthetic geometry realm. Although it might seem obvious that synthetic proofs facilitate human understanding of a geometric statement, as compared with the difficult interpretation associated to algebraic proofs, this assertion could be a matter of discussion if considered for statements (and for human minds) of a certain level,

[*]Authors supported by a grant PID2020-113192GB-I00 (Mathematical Visualization: Foundations, Algorithms and Applications) from the Spanish MICINN.

[1]See project page `https://github.com/kovzol/geogebra-discovery`.

a discussion that it is out of the scope of the present paper, and that could be the subject of a future research, in an educational context.

On the other hand, the evaluation of the *difficulty* of geometric statements is an old subject, regardless of its relation to automated proving. We can mention the work of Lemoine on the number of steps required by a geometric construction (a higher number of steps = a more complicated construction), a proposal that has been thoroughly studied, adapted to the Dynamic Geometry context, and exemplified in different repositories, e.g., in several recent works by Quaresma and collaborators, such as [3, 4].

But the complexity of a geometric construction is not, in general, a good measure to establish the difficulty of a statement involving the same construction: one can make a very complicated figure and then conjecture some obvious property, easy to derive from the construction steps.

Thus, in this paper we make a preliminary report of our current work aiming to establish some *difficulty* criteria, that profits from our algebraic approach to proving geometric statements. Roughly: we propose to label as *more difficult* those statements where:

1. first of all, the polynomial involved in the description of the thesis (or conclusion) is a sum of products of the hypotheses polynomials multiplied by some other polynomials (thus, the statement is just formally *true*). These other polynomials are usually named as syzygies,

2. else, there is a combination of 1 as a sum of products of the hypotheses polynomials and of the negation of the thesis T (expressed as $T \cdot t - 1$) multiplied by some other polynomials (thus, the statement is, by *reductio ad absurdum*, *geometrically true*, meaning that a power of the thesis is a combination of the hypotheses). Again, these other polynomials that multiply the negation of the thesis and the hypotheses, are usually named as syzygies,

3. these expressions of the thesis (or of 1) as a combination of the hypotheses (of the hypotheses and the negation of the thesis) requires higher degree polynomials, i.e. more complicated syzygies.

Of course, although we are aware that *difficulty* and *interest* are not identical concepts, let us recall that this notion, the *interestingness* of theorems, is also subject to current research (see [5, 6, 7, 8]) in different contexts (A.I., Big Data, etc.), sometimes explicitly referring to our particular automated reasoning in geometry approach.

In what follows we will describe, mostly through some examples, our ongoing work on these two subjects.

2 ShowProof Command

Unfortunately, till now, the algebraic geometry nature of the algorithms behind the automatic reasoning tools implemented in GeoGebra or GeoGebra Discovery do not allow providing readable arguments justifying their outputs. Computations are performed in the background, using some embedded Computer Algebra System, such as Giac [10]. The user only gets a kind of yes/no answer.

The `ShowProof` command is a first attempt to enhance the visibility of the proofs achieved by GeoGebra Discovery, by showing the result of the different steps performed by GeoGebra Discovery to confirm a certain statement: algebraic translation of the geometric input construction, numerical specialization of the coordinates of some free points, automatic inclusion of non-degeneracy conditions, and writing – using the concept of syzygy and its computation – the expression of 1 as a combination of the hypotheses and the negation of the thesis (thus proving the statement by *reductio ad absurdum*), or of the thesis as a combination of the hypotheses (direct proof of the statement).

Visualizing the output of most of these steps is just a question of improving the user interface, as it does not involve any new computation. This is so except for the (most important) last two items: the concrete expression of 1 (or of the thesis) as a combination of other polynomials. Indeed, to decide that a statement is generally true [1] GeoGebra Discovery just has to perform some elimination of the ideal of hypotheses plus the negation of the thesis, and to verify that it is not zero. Then, adding to the hypotheses ideal the negation of some of the generators g of this elimination (using Rabinowitsch trick, as $g \cdot t - 1$), it is obvious that the (new hypotheses + negation of thesis) ideal contains 1, since it will include both g and $g \cdot t - 1$.

But the user, who has not any concrete evidence about the result of the previous elimination (for example, expressing the added non-degeneracy condition as a combination of the (given hypotheses + negation of thesis) ideal, probably would appreciate handling an expression of 1 as a combination of the (new hypotheses + negation of thesis) ideal. Or, even more impacting, viewing the thesis as a combination of the hypotheses. Notice that the first possibility just means that *over the complexes* the thesis vanishes over all points satisfying the new hypotheses (the given ones and the added non-degeneracy condition) and, thus, that a power of the thesis is a combination of the hypotheses. We can say that the statement is, in this case, *geometrically true*, while, if the thesis itself is a combination of the hypotheses, we can declare we have a *formally true* statement.

Both issues are now addressed through the `ShowProof` command, using the concept of syzygy (e.g. [9], page 104):

Given any $G = (g_1, \ldots, g_s) \in k[x_1, \ldots, x_n]^s$, we can define a syzygy on G to be an s-tuple $S = (h_1, \ldots, h_s) \in k[x_1, \ldots, x_n]^s$, such that $\sum_i h_i \cdot g_i = 0$.

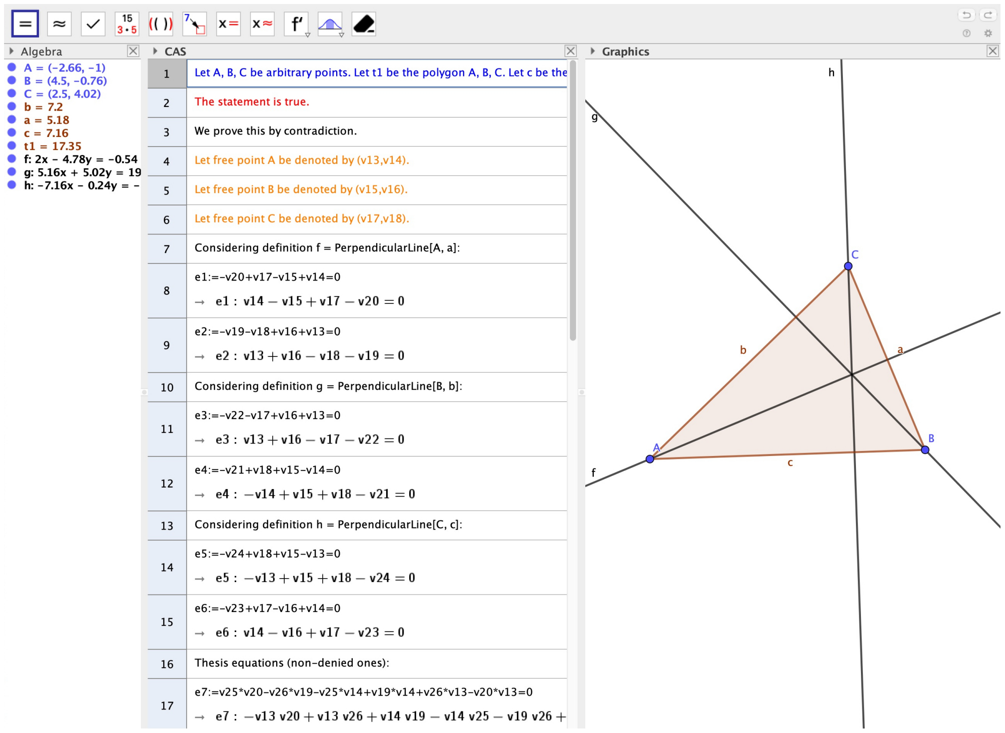
In particular, if we include 1 (respectively, the thesis) as the first element of the collection of polynomials G, and the remaining elements are the generators of the ideal of the new hypotheses plus the negation of the thesis (respectively, of the new hypotheses), then we will get (if the statement is geometrically true, respectively, symbolically true) syzygies of the kind $(1, -h_2, \ldots, -h_s)$, allowing us to output a concrete expression of 1 (respectively, the thesis) as a combination of the new hypotheses plus the negation of the thesis (respectively, of the new hypotheses).

The next figures illustrate the current output of the `ShowProof` command. Figure 1 displays, first, the automatically and internally assigned coordinates of the free vertices of the triangle A, B, C; then, the equations of the different heights; finally, the thesis (the fact that the last height goes through the intersection of the other two).

Next, following our algorithm and without loss of generality, the program automatically specializes the given free coordinates, to reduce the number of variables before starting the computations. This is shown in figure 2, that ends declaring that the statement is *geometrically true* by explicitly showing 1 as a combination of the negation of the thesis and the hypotheses equations (thus, $1 = 0$, since all these equations are equal to zero). Finally, the last line declares that this statement is of difficulty 2, a measure that we will roughly describe in the next section.

3 Interestingness

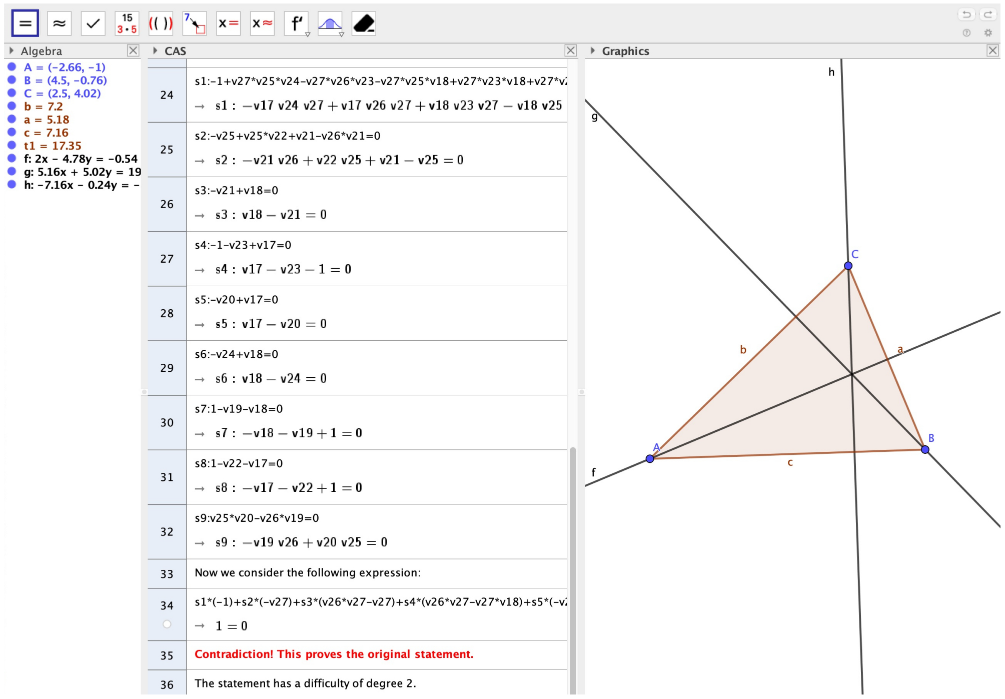
Although the precise formulation of the following reflections require a serious and future research analysis, that is not the goal of the present paper, we dare to consider quite evident that showing arguments for the truth of a geometric statement is important in the scientific and educational context, even more relevant nowadays, in a context of close collaboration (or leadership?) of machine-driven learning. Analogously, under the dominance and ubiquitousness of machine learning environments, very prone to



```
Algebra
A = (-2.66, -1)
B = (4.5, -0.76)
C = (2.5, 4.02)
b = 7.2
a = 5.18
c = 7.16
t1 = 17.35
f: 2x - 4.78y = -0.54
g: 5.16x + 5.02y = 19
h: -7.16x - 0.24y = -

CAS
1   Let A, B, C be arbitrary points. Let t1 be the polygon A, B, C. Let c be the
2   The statement is true.
3   We prove this by contradiction.
4   Let free point A be denoted by (v13,v14).
5   Let free point B be denoted by (v15,v16).
6   Let free point C be denoted by (v17,v18).
7   Considering definition f = PerpendicularLine[A, a]:
8   e1:=-v20+v17-v15+v14=0
    → e1 : v14 − v15 + v17 − v20 = 0
9   e2:=-v19-v18+v16+v13=0
    → e2 : v13 + v16 − v18 − v19 = 0
10  Considering definition g = PerpendicularLine[B, b]:
11  e3:=-v22-v17+v16+v13=0
    → e3 : v13 + v16 − v17 − v22 = 0
12  e4:=-v21+v18+v15-v14=0
    → e4 : −v14 + v15 + v18 − v21 = 0
13  Considering definition h = PerpendicularLine[C, c]:
14  e5:=-v24+v18+v15-v13=0
    → e5 : −v13 + v15 + v18 − v24 = 0
15  e6:=-v23+v17-v16+v14=0
    → e6 : v14 − v16 + v17 − v23 = 0
16  Thesis equations (non-denied ones):
17  e7:=v25*v20-v26*v19-v25*v14+v19*v14+v26*v13-v20*v13=0
    → e7 : −v13 v20 + v13 v26 + v14 v19 − v14 v25 − v19 v26 +
```

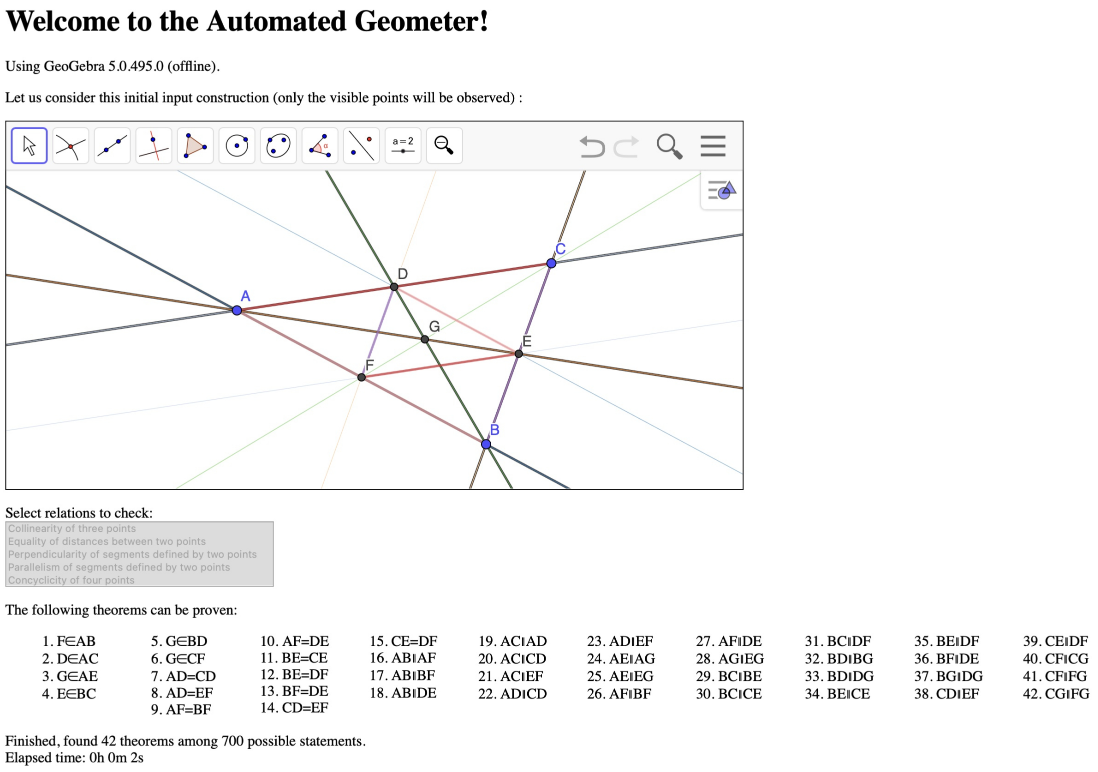
Figure 1: Viewing proof of the *Intersection of heights* theorem through `ShowProof`. Initial steps.



```
Algebra
A = (-2.66, -1)
B = (4.5, -0.76)
C = (2.5, 4.02)
b = 7.2
a = 5.18
c = 7.16
t1 = 17.35
f: 2x - 4.78y = -0.54
g: 5.16x + 5.02y = 19
h: -7.16x - 0.24y = -

CAS
24  s1:-1+v27*v25*v24-v27*v26*v23-v27*v25*v18+v27*v23*v18+v27*v
    → s1 : −v17 v24 v27 + v17 v26 v27 + v18 v23 v27 − v18 v25
25  s2:-v25+v25*v22+v21-v26*v21=0
    → s2 : −v21 v26 + v22 v25 + v21 − v25 = 0
26  s3:-v21+v18=0
    → s3 : v18 − v21 = 0
27  s4:-1-v23+v17=0
    → s4 : v17 − v23 − 1 = 0
28  s5:-v20+v17=0
    → s5 : v17 − v20 = 0
29  s6:-v24+v18=0
    → s6 : v18 − v24 = 0
30  s7:1-v19-v18=0
    → s7 : −v18 − v19 + 1 = 0
31  s8:1-v22-v17=0
    → s8 : −v17 − v22 + 1 = 0
32  s9:v25*v20-v26*v19=0
    → s9 : −v19 v26 + v20 v25 = 0
33  Now we consider the following expression:
34  s1*(-1)+s2*(-v27)+s3*(v26*v27-v27)+s4*(v26*v27-v27*v18)+s5*(-v
    → 1 = 0
35  Contradiction! This proves the original statement.
36  The statement has a difficulty of degree 2.
```

Figure 2: Viewing proof of the *Intersection of heights* theorem through `ShowProof`. Specialized equations, conclusion and difficulty.



Welcome to the Automated Geometer!

Using GeoGebra 5.0.495.0 (offline).

Let us consider this initial input construction (only the visible points will be observed) :

Select relations to check:

Collinearity of three points
Equality of distances between two points
Perpendicularity of segments defined by two points
Parallelism of segments defined by two points
Concyclicity of four points

The following theorems can be proven:

1. F∈AB
2. D∈AC
3. G∈AE
4. E∈BC
5. G∈BD
6. G∈CF
7. AD=CD
8. AD=EF
9. AF=BF
10. AF=DE
11. BE=CE
12. BE=DF
13. BF=DE
14. CD=EF
15. CE=DF
16. AB∥AF
17. AB∥BF
18. AB∥DE
19. AC∥AD
20. AC∥CD
21. AC∥EF
22. AD∥CD
23. AD∥EF
24. AE∥AG
25. AE∥EG
26. AF∥BF
27. AF∥DE
28. AG∥EG
29. BC∥BE
30. BC∥CE
31. BC∥DF
32. BD∥BG
33. BD∥DG
34. BE∥CE
35. BE∥DF
36. BF∥DE
37. BG∥DG
38. CD∥EF
39. CE∥DF
40. CF∥CG
41. CF∥FG
42. CG∥FG

Finished, found 42 theorems among 700 possible statements.
Elapsed time: 0h 0m 2s

Figure 3: Automated Geometer: relevant (e.g. Theorem 7, medians intersect at a common point) and trivial (e.g. Theorem 1, midpoint F of AB in the line AB, or Theorem 9, midpoint F is equidistant from A and B) results.

automatically produce large amounts of geometric information, we think it is very relevant to develop instruments that allow humans to assess the relevance of the obtained results.

For example, GeoGebra Discovery has already a *Discover* command that automatically finds all statements of a certain kind that hold over some element of a construction (and a more general command in a web version, the *Automated Geometer*,[2] automatically finding *all* the statements of a selected kind (collinearity of three points, equality of distances between two points, etc., as declared in the dark box below the geometric figure in figure 3) holding over *all* elements of a figure.) See also the recent `StepwiseDiscovery` command, that discovers automatically all statements involving each of the new elements that the user is adding in each construction step. Now, it happens that a great number of such *discovered* statements are just obvious! See figure 3, for some examples.

Thus, we dare to introduce, as a measure of the complexity of a result, the following definition:

We say that a statement $H \Rightarrow T$ is of complexity d if d is the maximum degree of the syzigies expressing T (or 1) as a combination of the hypotheses (correspondingly, of the hypotheses and the negation of the thesis).

In what follows we will describe different examples, towards analyzing the potential of this proposal for the concept of an *interesting* statement.

[2]`http://autgeo.online/ag/automated-geometer.html?offline=1`

3.1 Example 1

Let us build a point F as the midpoint of AB (see Figure 3), and ask about *Discover(F)*. The program *discovers* (among other, really interesting statements), that the length of segment FA is equal to the length of FB. But, since the definition of F as midpoint of the segment with extremes $A(a_1,a_2),B(b_1,b_2)$ is that the coordinates of F are $((a_1+b_1)/2,(a_2+b_2)/2)$, it immediately follows that $FA=((a_1-b_1)/2,(a_2-b_2)/2)$ and $FB=(b_1-a_1)/2,(b_2-a_2)/2)$. Obviously, from these coordinates, it follows that the length of both segments is identical, it only requires to simplify $((a_1-b_1)/2)^2+((a_2-b_2)/2)^2-(((b_1-a_1)/2)^2+((b_2-a_2)/2)^2)$, yielding 0. Thus, we could roughly declare that this thesis is just the trivial equation $0=0$ and thus it is always a combination of whatever set of hypotheses multiplied by zero. Thus the degree of the zero polynomial 0 (that some algebraist consider as a negative number) could be a measure of the complexity of this highly trivial statement.

3.2 Example 2

On a different example about elements in the same figure, if we consider the coordinates of point $G(g_1,g_2)$, the intersection of the line that goes from A to the midpoint $E(e_1,e_2)$ of side BC, and of the line that goes from B to the midpoint $D(d_1,d_2)$ of side AC, where $C(c_1,c_2)$, we have:

- Coordinates of E,D:

$$H_1,H_2: e_1=(b_1+c_1)/2, e_2=(b_2+c_2)/2, \tag{1}$$

$$H_3,H_4: d_1=(a_1+c_1)/2, d_2=(a_2+c_2)/2. \tag{2}$$

- Equations of lines A,E and B,D

$$\text{line } AE: (x-a_1)\cdot(e_2-a_2)-(y-a_2)\cdot(e_1-a_1)=0, \tag{3}$$

$$\text{line } BD: (x-b_1)\cdot(d_2-b_2)-(y-b_2)\cdot(d_1-b_1)=0. \tag{4}$$

- Thus, when we declare G as the intersection of these two lines, we introduce the equations

$$H_5: \text{ line } AE: (g_1-a_1)\cdot(e_2-a_2)-(g_2-a_2)\cdot(e_1-a_1)=0, \tag{5}$$

$$H_6: \text{ line } BD: (g_1-b_1)\cdot(d_2-b_2)-(g_2-b_2)\cdot(d_1-b_1)=0. \tag{6}$$

- Moreover, let us recall that midpoint F of side AB has coordinates:

$$H_7,H_8: f_1=(a_1+b_1)/2, f_2=(a_2+b_2)/2. \tag{7}$$

- Finally, the Theorem 6 in figure 3, namely, $G\in CF$, means that

$$\textit{Thesis: } G\in \text{ line } CF: (g_1-c_1)\cdot(f_2-c_2)-(g_2-c_2)\cdot(f_1-c_1)=0. \tag{8}$$

Now, after some involved computations on the syzygies of the set

$$(Thesis, H_1, H_2, \ldots, H_8),$$

here we use the same notation for the *Thesis* and the hypotheses H_i, but considering just the involved polynomials, not the equality to zero. We find that we can express the *Thesis* as a combination of $(H_1, H_2, \ldots, H_8)$ multiplied by linear polynomials in the involved variables

$$a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2, e_1, e_2, f_1, f_2, g_1, g_2,$$

namely, the thesis is equal to

$$Thesis = (-g_1 + f_1) \cdot G[1] + (-f_2 + g_2) \cdot G[2] + (1) \cdot G[7] \tag{9}$$

where $G[1], G[2], G[7]$ are elements of the Gröbner basis of the hypotheses ideal with respect to the *plex* order, and can be expressed, respectively, as sums of products of $[0,0,0,-1/2,0,-1/2,0,1/2]$, $[0,0,-1/2,0,-1/2,0,1/2,0]$, $[0,2\cdot g_2-2,-2\cdot g_1,-2\cdot g_2,0,2,2\cdot g_1,-2]$, times $[H_1,H_2,\ldots,H_8]$.

In summary, the thesis is a combination of the hypotheses multiplied by polynomials of degree at most 1. This degree 1, vs. the negative degree of the syzygies in the previous statement shows, it is our proposal, that this second statement, about the coincidence of the intersection of the three medians of a triangle, is *more difficult* than the statement about the equality of lengths of the two segments determined by the mid-point.

Let us remark that this same statement, but specializing vertices $A = (0,0)$, $B = (0,1)$, leads to degree 2 syzygies

$$\begin{aligned} Thesis = (-c_1/2 + g_1/2) \cdot G[1] + (-g_2 + c_2) \cdot G[2] + (-g_1) \cdot G[6] \\ + (g_2 - 1/2) \cdot G[7] - 1 \cdot G[8] \end{aligned} \tag{10}$$

where, again, $G[1], G[2], G[6], G[7], G[8]$ are elements of the Gröbner basis of the (specialized) hypotheses ideal with respect to the *tdeg* order, and can be expressed, respectively, as sums of products of $[0,0,0,0,2,0,0,0]$, $[1,0,0,0,0,0,0,0]$, $[0,0,0,0,0,0,-2,0]$, $[0,2\cdot g_2-2,-2\cdot g_1,-2\cdot g_2,0,2,2\cdot g_1,-2]$, $[0,2\cdot(g_2-1)\cdot g_2,-2\cdot g_1\cdot g_2,-2\cdot(g_2-1)\cdot g_2,0,2\cdot g_2,2\cdot g_1\cdot g_2,-2\cdot g_2+2]$, times $[H_1,H_2,\ldots,H_8]$, notice that only the summand involving $G[7]$ rises the degree to degree two.

3.3 Example 3

As a third example, let us start (see Figure 1) with a triangle with vertices $A(v13, v14)$, $B(v15, v16)$, $C(v17, v18)$. Then, consider the perpendicular line through A to side BC. Considering $P(v19, v20)$ as the coordinates of a generic point P in this line, we obtain the equation

$$H_1 : (v19 - v13) \cdot (v17 - v15) + (v20 - v14) \cdot (v18 - v16) = 0. \tag{11}$$

Similarly for the other two heights.

Thus, the statement *the three heights of a triangle have a common intersection* is a matter of considering two hypotheses (the equation of height from A and from, say, B) with a common generic point P. And the thesis is also one equation, namely, showing that this generic point P satisfies that line PC is perpendicular to line AB. More precisely:

- Hypotheses:

- Height from A:

$$(v19-v13)\cdot(v17-v15)+(v20-v14)\cdot(v18-v16)=0. \tag{12}$$

- Height from B:

$$(v19-v15)\cdot(v17-v13)+(v20-v16)\cdot(v18-v14)=0. \tag{13}$$

- Thesis:

$$(v19-v17)\cdot(v15-v13)+(v20-v18)\cdot(v16-v14)=0. \tag{14}$$

Notice that these equations seem different from those describing this statement in figure 1. There, the height from A is described by considering the coordinates of point $X(v19,v20)$ in this line, and regarding such point as the translation of A by a vector AX perpendicular to BC so that $-v20+v17-v15+v14=0, -v19-v18+v16+v13=0$. Finally, Equation (17) in figure 1, describes a generic point $(v25,v26)$ of this line, verifying $v25\cdot v20-v26\cdot v19-v25\cdot v14+v19\cdot v14+v26\cdot v13-v20\cdot v13=0$.

But let us remark that this equation is practically the same as the one above introduced for the height from A, replacing there generic point coordinates $(v19,v20)$ by $(v25,v26)$ and, in the new equation, $v20=v17-v15+v14, v19=v16+v13-v18$, yielding

$$\begin{aligned} &v25\cdot v20-v26\cdot v19-v25\cdot v14+v19\cdot v14+v26\cdot v13-v20\cdot v13=\\ =&\ v25\cdot(v17-v15+v14)-v26\cdot(v16+v13-v18)-v25\cdot v14+\\ &(v16+v13-v18)\cdot v14+v26\cdot v13-(v17-v15+v14)\cdot v13=\\ =&\ (v15-v17)\cdot(v13-v25)+(v16-v18)\cdot(v14-v26). \end{aligned}$$

Similarly, the thesis is now, with the notation of figure 1:

$$v25\cdot v24-v26\cdot v23-v25\cdot v18+v23\cdot v18+v26\cdot v17-v24\cdot v17=0 \tag{15}$$

that belongs to the new ideal of hypotheses $\langle -v20+v17-v15+v14, -v19-v18+v16+v13, -v22-v17+v16+v13, -v21+v18+v15-v14, -v24+v18+v15-v13, -v23+v17-v16+v14, v25\cdot v20-v26\cdot v19-v25\cdot v14+v19\cdot v14+v26\cdot v13-v20\cdot v13, v25\cdot v22-v26\cdot v21-v25\cdot v16+v21\cdot v16+v26\cdot v15-v22\cdot v15\rangle$.

The interesting point here is that the *complexity* of the statement is different if one considers the previous equations or the ones in figure 1. In the first case, it is immediate to see that the thesis is the difference of the height from B minus the height from A. The same result holds if we do specialize numerically A,B, as it is usual in GeoGebra Discovery, considering $A(0,0), B(0,1)$, so that the above hypotheses turn out to be:

- Height from A:

$$(v19-0)\cdot(v17-0)+(v20-0)\cdot(v18-1)=0. \tag{16}$$

- Height from B:

$$(v19-0)\cdot(v17-0)+(v20-1)\cdot(v18-0)=0. \tag{17}$$

- Thesis:

$$(v19-v17)\cdot(0-0)+(v20-v18)\cdot(1-0)=0. \tag{18}$$

So we could say that here the syzygies are of degree 0, and thus we can declare the complexity of the statement is 0, as the thesis is just a linear combination of the hypotheses (multiplied by constants).

But, with the equations in figure 1, using the same *tdeg* order, one gets the thesis as a combination of the hypothesis multiplied by polynomials of degree at most one, so the complexity rises one element and, if we do specialize $A(0,0), B(0,1)$ in the equations of figure 1, we get that the syzygies have to be of degree 2, as stated in figure 2!

4 Conclusions

The explicit visualization of the algebraic expression that connects hypotheses and thesis (or expresses number 1 as a combination of hypotheses and the negation of the thesis) seems a relevant improvement of GeoGebra Discovery automated reasoning tools, allowing the user not only to be able to personally confirm the truth of the given statement, but also to measure its difficulty.

Of course, the proposal of this measure is just on its initial steps and requires much further research. For example, extending the already considered examples, including others of higher difficulty, such as those concerning Mathematics Olympiad problems. Indeed, we have already verified with some of them that our measure of the computed difficulty agrees that they are much more difficult than the traditional school problems, but analyzing more problems from different sources, comparing our rank and the behavior of the olympic teams on the same problems, is still on-going work [11].

Moreover, the proposed research should not just be restricted to producing benchmarks, but to reflect on several more conceptual issues, such as the role, on the measure of the complexity, of the different ways to express the same statement (as remarked in the previous examples). Ditto, for the specialization of some variables, for choosing different ordering for computing syzygies, etc. We also need to understand the difference, or the connection, between the complexity of expressing a statement and the complexity of deciding its truth, the relation between the complexity of the statement and the complexity of the associated ideal membership problem, etc.

Indeed, we think there is plenty of work in this context and it is our intention to address such issues in a near future.

References

[1] Kovács, Z.; Recio, T.; Vélez, M. P.: Automated reasoning tools with GeoGebra: What are they? What are they good for? In: P. R. Richard, M. P. Vélez, S. van Vaerenbergh (eds): Mathematics Education in the Age of Artificial Intelligence: How Artificial Intelligence can serve mathematical human learning. Series: Mathematics Education in the Digital Era, Vol. 17, 23–44. Springer Cham (2022). doi:10.1007/978-3-030-86909-0_2

[2] Boutry, P.; Braun, G.; Narboux, J.: Formalization of the arithmetization of Euclidean plane geometry and applications. Journal of Symbolic Computation 90, 149–168 (2019). doi:10.1016/j.jsc.2018.04.007

[3] Quaresma, P.; Santos, V.; Graziani, P.; Baeta, N.: Taxonomy of geometric problems. Journal of Symbolic Computation 97, 31–55 (2020). doi:10.1016/j.jsc.2018.12.004

[4] Santos, V.; Baeta, N.; Quaresma, P.: Geometrography in Dynamic Geometry. The International Journal for Technology in Mathematics Education, vol. 26, no. 2, June 2019, 89–96 (2019).

[5] Gao, H.; Goto, Y.; Cheng, J.: A set of metrics for measuring interestingness of theorems in automated theorem finding by forward reasoning: A case study in NBG set theory. Proceedings of the International Conference on Intelligence Science and Big Data Engineering (2015). Part II. Lecture Notes in Computer Science 9243, 508–517 (2015). doi:10.1007/978-3-319-23862-3_50

[6] Gao, H.; Li, J.; Cheng J.: Measuring Interestingness of Theorems in Automated Theorem Finding by Forward Reasoning Based on Strong Relevant Logic. In: 2019 IEEE International Conference on Energy Internet (ICEI), 356–361 (2019). doi:10.1109/ICEI.2019.00069

[7] Puzis, Y.; Gao, Y.; Sutcliffe G.: Automated generation of interesting theorems. In G. Sutcliffe and R. Goebel, editors: Proceedings of the 19th International FLAIRS Conference, AAAI Press, Menlo Park, 49–54 (2006).

[8] Colton, S.; Bundy, A.; Walsh, T.: On the notion of interestingness in automated mathematical discovery. International Journal of Human-Computer Studies Volume 53, Issue 3, September 2000, 351–375 (2000). doi:10.1006/ijhc.2000.0394

[9] Greuel, G-M.; Pfister, G.: A Singular introduction to commutative algebra. Springer Berlin, Heidelberg. Second Edition (2008). doi:10.1007/978-3-540-73542-7

[10] Kovács, Z.; Parisse, B.: Giac and GeoGebra – improved Gröbner basis computations. Presentation at RICAM Special semester on Applications of Algebra and Number Theory, Workshop 3 on Computer Algebra and Polynomials. `https://www.ricam.oeaw.ac.at/specsem/specsem2013/workshop3/slides/parisse-kovacs.pdf`. November (2013).

[11] Ariño-Morera, B.; Recio, T.; Tolmos, P.: Olympic geometry problems: human vs. machine. Communication to the CADGME (Digital Tools in Mathematics Education) 2022 Conference. Abstracts available at `https://drive.google.com/file/d/1qF4ceMg6gNklOPa1JVkgKND1dOqNmyka/view`.

Open Source Prover in the Attic

Zoltán Kovács
The Private University College of Education of the Diocese of Linz, Austria
zoltan.kovacs@ph-linz.at

Alexander Vujic
The Private University College of Education of the Diocese of Linz, Austria
alexander.vujic@ph-linz.at

The well known JGEX program became open source a few years ago, but seemingly, further development of the program can only be done without the original authors. In our project, we are looking at whether it is possible to continue such a large project as a newcomer without the involvement of the original authors. Is there a way to internationalize, fix bugs, improve the code base, add new features? In other words, to save a relic found in the attic and polish it into a useful everyday tool.

Keywords: JGEX, Java development, open source, education, Dynamics Geometry Systems.

1 Introduction

Wide access to computers in education opens a new horizon in applying automated reasoning tools in the classroom. Many software tools exist already, that are of different levels of maturity, and several issues can be found that prevent direct use of them in education. One of the most mature tools is Java Geometry Expert (JGEX, [11–13]), which has its roots in Wichita State University and was developed by the automated geometry reasoning school, led by Chou and Gao, and programmed (mostly) by Ye.

JGEX is a remarkable summary of several decades of pioneering work [2]. It consists of more than 108,000 lines of Java code in more than 230 files. It has been open-sourced, however, just a couple of years ago (in 2016), and it was published on GitHub.[1] Since then, more than 80 forks were created and the project was starred by more than 400 users. These numbers clearly show the popularity of JGEX, even if the original authors discontinued their work: Ye's latest (very minor) change was committed in July 2018. Therefore, in our work, we raise the question, if it is still possible to contribute to this project significantly.

Why are further contributions necessary? First of all, the software tool supports only a couple of languages. For educational use, a translation for native speakers seems unavoidable: young learners may have difficulties with languages. Second, a modern user interface may be more appealing for the new generation of users. Third, science is clearly at a more mature state-of-the-art now, so some updates in the used algorithms seem important after a while. Fourth, it turned out that some minor, annoying bugs still exist in the program – they should be fixed someday: it is very important for a tool that is expected to be error-free in all matters when it is about theorem proving.

In our paper we discuss the possibility of continuing an existing tool for automated proving without the possibility to contact the original authors. Our discussion focuses on translating JGEX into German and Serbian, and on extending its capabilities to provide a better user experience. Also, we discuss how

[1] https://github.com/yezheng1981/Java-Geometry-Expert

P. Quaresma and Z. Kovács (Ed.): Automated Deduction in Geometry 2023 (ADG 2023).
EPTCS 398, 2024, pp. 53–61, doi:10.4204/EPTCS.398.9

difficult it is to build the program with the newest development tools available, and what changes have to be done to be somewhat up-to-date in this concern.

We think this research is important for a whole community, in particular, to save the achieved algorithms and implementations *for the next generation*. In other words, our research is a survival guide which aims at supporting the community for automated geometry reasoning.

2 Expectations

In this section we give an overview of the expected updates and further improvements.

2.1 Translations

JGEX's final mainstream version supports the Chinese, English, Italian and Persian languages. Our research team, after forking the mainstream version in the repository,[2] decided to add further translations: German, Portuguese and Serbian.

It turned out that the translation, technically speaking, is a simple process of creating a text file, similar to the CSV (comma separated values) format, that consists of a text ID integer, the English keyword (because in most cases it is used for the lookup, instead of the ID), the translated text, and (optionally) a tooltip. With minor edits, a new text file can easily be added to the source files and it can be inserted in the menu system.

Unfortunately, it also turned out that the tooltip entries are completely missing for all languages (except for English, but those translations are already given in the Java source code). On the other hand, some translations are completely missing by design: such expressions are short phrases like "by HYP" (which means: "by assumption", "by hypothesis") and "because". We had to add such phrases in the CSV databases, and modify the Java source code to search for the phrases also in the CSV files.

Also, two additional lists of phrases are hardcoded in the original source code. They belong to the Geometry Deduction Database (GDD, [3,12]) and the rule database for the Full Angle method [3]. These are given only in English. It is an ongoing work to move these phrases in the CSV database, instead of using the Java files to store the English version, and, in our fork, the German and Serbian translations as well.

Why are all translations stored in one CSV database? There would be a more efficient way of translating the user interface into different languages. Nevertheless, a variety of languages have already been translated in the CSV database, therefore it was more accessible for the team to continue working with CSV. In our case, the most extensive work was dedicated to the Serbian translation. Our translation team, led by the second author, started to make a copy of the German translation (because of the better knowledge of German than English). After finishing translation of the phrases in the CSV file, it turned out that, in addition, both rule databases need to be translated. Then, some additional phrases were found in the Java source code that had to be translated too. To make a quality work, the translated phrases had to be double-checked by native speakers who were also experts in geometry. Unfortunately, by doing the translation in three different steps, every participant had to work three times: once for the CSV entries, once for the rule databases, and again for the unexpected phrases shown up randomly in the source code. This made the work quite inefficient and unexpectedly long.

Overall, it would have been better to unify the translation system first, and only after then start a concentrated work on doing the real effort in translating the phrases. But this was, actually, not really

[2] `https://github.com/kovzol/Java-Geometry-Expert`

possible: JGEX has no full documentation, and it cannot be therefore assumed that any users (either newcomers or experienced ones) have a complete overview of all phrases used in the program.

We also learned that in many cases, unfortunately, the English translation contains spelling errors. As a consequence, the translation keys have spelling problems too. This makes it difficult to modify all erroneous translation keys in all CSV files and the Java source code at the same time.

In fact, there are sophisticated translation systems like *gettext*. We will consider changing the translation system in a next round of possible updates.

2.2 Modern Interface

First author invited a group of prospective mathematics teachers at the Private University College of Education of the Diocese of Linz (PHDL) to give some feedback on the usability of JGEX. While most feedback was positive, several students mentioned that JGEX had an unusual user interface, and an old-fashioned look. Many features were difficult to find. Among others, to obtain a proof that is easy to understand in the classroom, the user needed to find an appropriate example in the database of showcases. The general method of showing the proof is to fix the objects and to select the relevant one from them, and use the right mouse click to access the proof. In certain cases, such as "08_9point.gex" the students may access the proof by right clicking on "SHOW: CYCLIC D G E F" and selecting "Prove".

To solve such difficulties, several ideas were suggested by the students, and the authors of this paper. One simple possibility is to create tutorials in both textual and video format. On the other hand, a simplified view of the most useful capabilities for classroom use could be helpful. Here we refer to another software package, OK Geometry [7], which comes in different editions (Easy, Basic, Plus modes). Also, a more extensive use of tooltips could tear down some barriers. Another option is to force the user into a workflow: first, she had to create or load a construction, second, point to the searched properties, and third, to obtain the proof.

Today's users are more familiar with web applications and mobile phone apps than native applications. In fact, the original version of JGEX comes with no installer: the user needs to download the source code, and it is her own task to compile and run it with an appropriate Java Development Kit (JDK) and Java Runtime Environment (JRE). First author maintains a ZIP package for Windows and Mac systems (based on the *packr* utility[3]) by providing all required files to run JGEX without any further preparation.[4] Also, a Linux Snap version of JGEX is also available.[5] As of May 2023 there are 175 installations world-wide registered from 48 territories. Anyway, these packages are just one step towards simplifying the access to JGEX – we think it is unavoidable to make JGEX available as an embeddable web application by compiling its codebase to JavaScript.

In fact, Ye's version of JGEX does not even compile automatically when the newest Java version is used. Some minor updates are required on the source code: Either the Java version must be downgraded to 8, or some small changes need to be done to avoid compiler errors. These requirements have been, luckily, also identified by other contributors who forked JGEX. On the other hand, the Java technology, selected by the Chinese experts, has proven to be a good choice, because with just minor modifications, it is easy to import the project into today's favorite Java Integrated Desktop Environment (IDE), *IntelliJ IDEA*.

Finally, we think it is unavoidable to make it possible to import *GeoGebra* [4] figures in JGEX, and to export them in GeoGebra format, since GeoGebra became the de facto standard of dynamic geometry

[3]`https://github.com/libgdx/packr`

[4]`https://github.com/kovzol/Java-Geometry-Expert/releases`

[5]`https://snapcraft.io/jgex`

during the last decade [10]. Even if JGEX supports a wide set of drawing tools, for newcomers, it can be difficult to learn its toolset quickly enough.

2.3 State-of-the-art Mathematics

JGEX implements several mathematical algorithms. One of the supported algorithms is the Gröbner basis method [5] that is known to be slower than the other methods, in a substantial set of input cases. Meanwhile, however, major speedups have been reported in some implementations of computing Gröbner bases. One of the successful implementations is included in GeoGebra's embedded computer algebra system Giac [6].

In fact, the Gröbner basis method, when using elimination, is known to provide better non-degeneracy conditions than the faster algebraic method, Wu's approach. This result could also be incorporated into JGEX. It's important to note that in many cases, the degeneracy conditions output by Wu's method are necessary for the theorems to hold, and they help distinguish the cases when the theorem is true and when it is false. This result could also be incorporated into JGEX.

2.4 Fixing Bugs and Adding Improvements

Even for mature software tools, there is always room to improve minor problems. Among other minor issues, the proof protocol shown for the GDD method, could be improved by reducing the number of output lines and showing the hierarchy of the proof in addition. We show this concept below.

We take the example **1_TOP_TEN/08_9point**: It constructs a figure to illustrate the nine-point circle theorem (Figure 1). We assume a classroom situation to get a readable proof of the fact that the three midpoints *(E, F, G)* of an arbitrary triangle *ABC* and a perpendicular foot point *D* (of vertex *A*, projected on side *BC*) are concyclic.

Now, by issuing some improvements on the source code (further details[6]), we can communicate the proof in a simpler way (Figure 2).

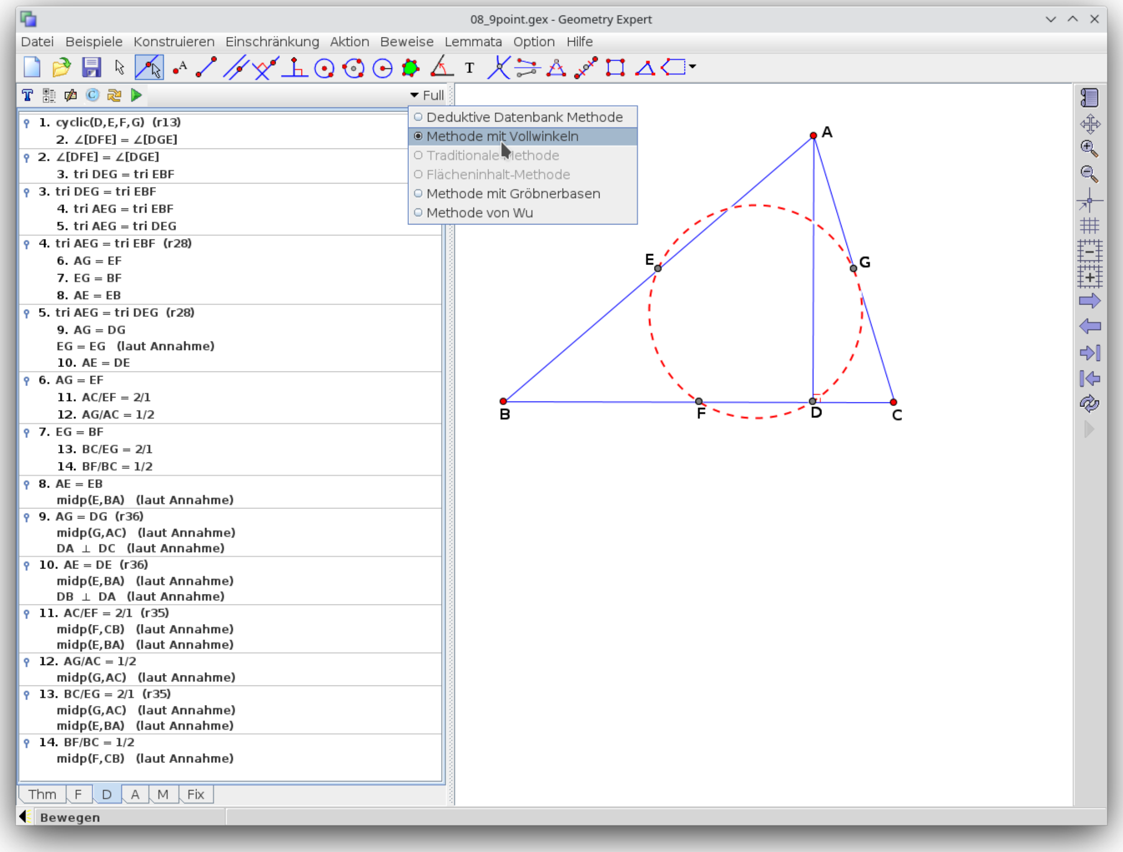
In fact, the change being performed to get this improvement is quite simple. Of course, one needs to have a deeper knowledge in the Java language and eventually in the mathematical background to achieve such changes in a feasible time. But, we can report that it is possible, and it does not require a high amount of time. Thus, a follow-up and continuation of the stopped work seems more than possible

Here we highlight that a tree structure for the obtained GDD proofs cannot always be accomplished. For example, Figure 3 shows the example **1_TOP_TEN/10_5cir** which states the concyclicity of points *M1*, *M2*, *M3* and *M4* in the given figure. Here we can learn that node 20 is used in nodes 13 and 15 too, so the whole subtree of node 20 is repeated in the subtrees of nodes 13 and 15. The presence of this duplication may be inconvenient for some users. Therefore, an option to switch forcing the structure off, could be a solution, or, even better, a different way of visualization might be applied. One possibility to do that is the embedding of the GraphViz library,[7] to provide a professional look of the outline of the proof structure (Figure 4).

Structured proofs may be beneficial when explaining the proof steps directly in the classroom. Of course, proofs that consist of a large number of steps, may be inappropriate for most students, but rather for gifted learners, mostly as preparations or training exercises for mathematical contests. Even so, providing a structured view of the proofs obtained by the GDD algorithm seems to be a substantial improvement for many classroom situations.

[6]`https://github.com/kovzol/Java-Geometry-Expert/commit/bc91b9ec916f97e38a100c08bec5bfda0c49de8d`

[7]`https://graphviz.org/`


08_9point.gex - Geometry Expert
Datei Beispiele Konstruieren Einschränkung Aktion Beweise Lemmata Option Hilfe
Full
Deduktive Datenbank Methode
Methode mit Vollwinkeln
Traditionale Methode
Flächeninhalt-Methode
Methode mit Gröbnerbasen
Methode von Wu
A
B
C
D
E
F
G
Thm F D A M Fix
Bewegen

```
1. cyclic(D,E,F,G)  (r13)
    2. ∠[DFE] = ∠[DGE]
2. ∠[DFE] = ∠[DGE]
    3. tri DEG = tri EBF
3. tri DEG = tri EBF
    4. tri AEG = tri EBF
    5. tri AEG = tri DEG
4. tri AEG = tri EBF  (r28)
    6. AG = EF
    7. EG = BF
    8. AE = EB
5. tri AEG = tri DEG  (r28)
    9. AG = DG
    EG = EG   (laut Annahme)
    10. AE = DE
6. AG = EF
    11. AC/EF = 2/1
    12. AG/AC = 1/2
7. EG = BF
    13. BC/EG = 2/1
    14. BF/BC = 1/2
8. AE = EB
    midp(E,BA)   (laut Annahme)
9. AG = DG  (r36)
    midp(G,AC)   (laut Annahme)
    DA ⊥ DC   (laut Annahme)
10. AE = DE  (r36)
    midp(E,BA)   (laut Annahme)
    DB ⊥ DA   (laut Annahme)
11. AC/EF = 2/1  (r35)
    midp(F,CB)   (laut Annahme)
    midp(E,BA)   (laut Annahme)
12. AG/AC = 1/2
    midp(G,AC)   (laut Annahme)
13. BC/EG = 2/1  (r35)
    midp(G,AC)   (laut Annahme)
    midp(E,BA)   (laut Annahme)
14. BF/BC = 1/2
    midp(F,CB)   (laut Annahme)
```

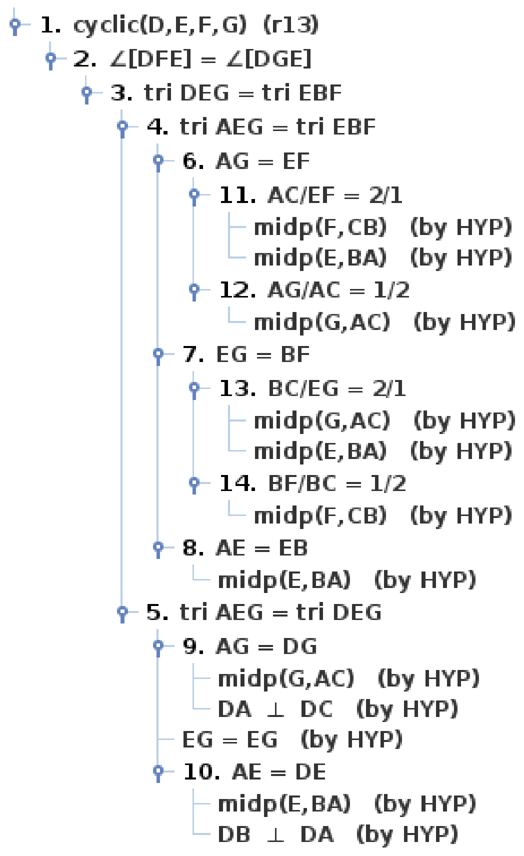
Figure 1: JGEX proves a part of the nine-point theorem by using the GDD method. Its user interface is set to German.

3 Conclusion

JGEX is an eminent software tool that summarizes several decades of pioneer work. Unfortunately, its main authors no longer maintain the code base. In our research we studied the question if it is possible to continue their work in some sense, and forward their legacy to the new generations.

According to our case study, we think that such a follow-up is possible, however, it is not straightforward. Several problems, mostly technical ones, can occur. The free availability of the source code is, however, a great help. With enough time and patience, the original version can be extended in a direction that seems fruitful for the long term.

We need to mention the quality of the code base. Unfortunately, the number of comments is very low. On the other hand, the variable names and the naming system for the Java methods are quite straightforward. That is, with little work, it is possible to learn the internals of the Java source code. Some parts of the code, however, require a major restructuring. Among others, the way of how the translations are handled, needs to be improved significantly. Also, some Java coding standards like capitalization of the source files, seem to be ignored. Nevertheless, by using modern refactoring tools, these issues could be solved with minimal efforts.



```
1. cyclic(D,E,F,G)  (r13)
  2. ∠[DFE] = ∠[DGE]
    3. tri DEG = tri EBF
      4. tri AEG = tri EBF
        6. AG = EF
          11. AC/EF = 2/1
            midp(F,CB)  (by HYP)
            midp(E,BA)  (by HYP)
          12. AG/AC = 1/2
            midp(G,AC)  (by HYP)
        7. EG = BF
          13. BC/EG = 2/1
            midp(G,AC)  (by HYP)
            midp(E,BA)  (by HYP)
          14. BF/BC = 1/2
            midp(F,CB)  (by HYP)
        8. AE = EB
          midp(E,BA)  (by HYP)
      5. tri AEG = tri DEG
        9. AG = DG
          midp(G,AC)  (by HYP)
          DA ⊥ DC  (by HYP)
        EG = EG  (by HYP)
        10. AE = DE
          midp(E,BA)  (by HYP)
          DB ⊥ DA  (by HYP)
```

Figure 2: The output of the GDD method is visualized as a tree.

We remark that a general inspection process in IntelliJ IDEA 2023.1.2 found 269 errors, 9645 warnings, 430 weak warnings, 181 grammar errors and 16449 typos in JGEX's code base. Even if many of the items of such reports can be of a matter of taste and coding style, a detailed study of these messages seems unavoidable for the long term.

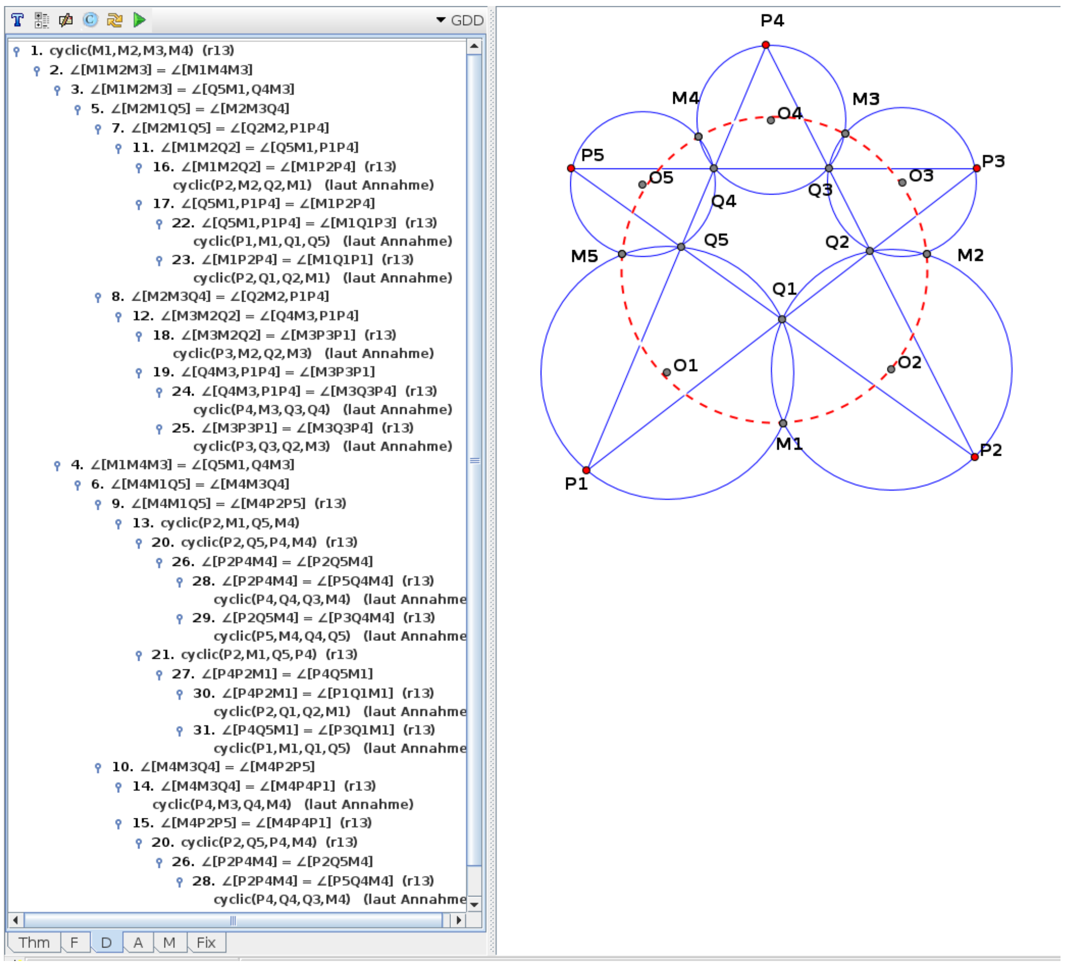
Some recent work [1,8,9] confirm that continuing the pioneer work being put in JGEX is an important step towards the more general use of automated reasoning in the classroom. One possibility is to copy the algorithms and modify them accordingly (this way was chosen by Baeta and Quaresma, using C++), but another option (we chose this) is to use the original source code and do the modifications on it directly.

As a final conclusion, we think it is possible to save the heritage of the Chinese experts, and continue the hard work of popularizing JGEX and extending the user community of automated deduction in geometry.

4 Acknowledgments

The first author was partially supported by a grant PID2020-113192GB-I00 (Mathematical Visualization: Foundations, Algorithms and Applications) from the Spanish MICINN.

We acknowledge the kind support of translators Benedek Kovács, Alexander Thaller, Engelbert Zeintl (German), Jorge Cassio (Portuguese), Amela Hota, Predrag Janičić and Jelena Marković (Ser-



```
1. cyclic(M1,M2,M3,M4)  (r13)
  2. ∠[M1M2M3] = ∠[M1M4M3]
    3. ∠[M1M2M3] = ∠[Q5M1,Q4M3]
      5. ∠[M2M1Q5] = ∠[M2M3Q4]
        7. ∠[M2M1Q5] = ∠[Q2M2,P1P4]
          11. ∠[M1M2Q2] = ∠[Q5M1,P1P4]
            16. ∠[M1M2Q2] = ∠[M1P2P4]  (r13)
                cyclic(P2,M2,Q2,M1)   (laut Annahme)
            17. ∠[Q5M1,P1P4] = ∠[M1P2P4]
              22. ∠[Q5M1,P1P4] = ∠[M1Q1P3]  (r13)
                  cyclic(P1,M1,Q1,Q5)   (laut Annahme)
              23. ∠[M1P2P4] = ∠[M1Q1P1]  (r13)
                  cyclic(P2,Q1,Q2,M1)   (laut Annahme)
        8. ∠[M2M3Q4] = ∠[Q2M2,P1P4]
          12. ∠[M3M2Q2] = ∠[Q4M3,P1P4]
            18. ∠[M3M2Q2] = ∠[M3P3P1]  (r13)
                cyclic(P3,M2,Q2,M3)   (laut Annahme)
            19. ∠[Q4M3,P1P4] = ∠[M3P3P1]
              24. ∠[Q4M3,P1P4] = ∠[M3Q3P4]  (r13)
                  cyclic(P4,M3,Q3,Q4)   (laut Annahme)
              25. ∠[M3P3P1] = ∠[M3Q3P4]  (r13)
                  cyclic(P3,Q3,Q2,M3)   (laut Annahme)
    4. ∠[M1M4M3] = ∠[Q5M1,Q4M3]
      6. ∠[M4M1Q5] = ∠[M4M3Q4]
        9. ∠[M4M1Q5] = ∠[M4P2P5]  (r13)
          13. cyclic(P2,M1,Q5,M4)
            20. cyclic(P2,Q5,P4,M4)  (r13)
              26. ∠[P2P4M4] = ∠[P2Q5M4]
                28. ∠[P2P4M4] = ∠[P5Q4M4]  (r13)
                    cyclic(P4,Q4,Q3,M4)   (laut Annahme
                29. ∠[P2Q5M4] = ∠[P3Q4M4]  (r13)
                    cyclic(P5,M4,Q4,Q5)   (laut Annahme
            21. cyclic(P2,M1,Q5,P4)  (r13)
              27. ∠[P4P2M1] = ∠[P4Q5M1]
                30. ∠[P4P2M1] = ∠[P1Q1M1]  (r13)
                    cyclic(P2,Q1,Q2,M1)   (laut Annahme
                31. ∠[P4Q5M1] = ∠[P3Q1M1]  (r13)
                    cyclic(P1,M1,Q1,Q5)   (laut Annahme
        10. ∠[M4M3Q4] = ∠[M4P2P5]
          14. ∠[M4M3Q4] = ∠[M4P4P1]  (r13)
              cyclic(P4,M3,Q4,M4)   (laut Annahme)
          15. ∠[M4P2P5] = ∠[M4P4P1]  (r13)
            20. cyclic(P2,Q5,P4,M4)  (r13)
              26. ∠[P2P4M4] = ∠[P2Q5M4]
                28. ∠[P2P4M4] = ∠[P5Q4M4]  (r13)
                    cyclic(P4,Q4,Q3,M4)   (laut Annahme
```

Figure 3: The GDD method finds a proof that is not a tree. The interface is set to German and the connecting edges of the graph are not shown.

bian).

We sincerely appreciate Amela Hota for her support in crafting a presentation of a preliminary version of this paper.

References

[1] N. Baeta & P. Quaresma (2023): *Towards a Geometry Deductive Database Prover*. *Annals of Mathematics and Artificial Intelligence*, doi:10.1007/s10472-023-09839-0.

[2] S.C. Chou (1987): *Mechanical Geometry Theorem Proving*. Kluwer Academic Publishers, Norwell, MA, USA, doi:10.1007/978-94-009-4037-6.

[3] S.C. Chou, X.S. Gao & J.Z. Zhang (2000): *A Deductive Database Approach to Automated Geometry Theorem Proving and Discovering*. *Journal of Automated Reasoning* 25, pp. 219–246, doi:10.1023/A:1006171315513.

[4] M. Hohenwarter (2002): *GeoGebra — Ein Softwaresystem für Dynamische Geometrie und Algebra der Ebene*. Ph.D. thesis, University of Salzburg, Austria.

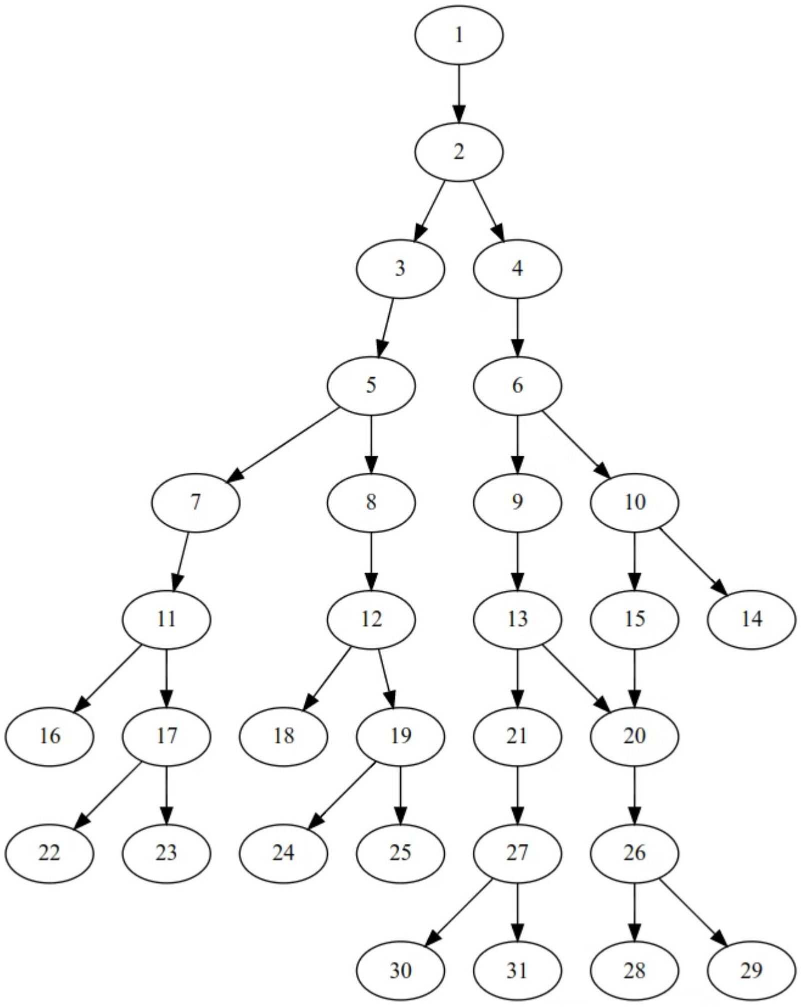


Figure 4: A better visualization of the GDD proof by using GraphViz.

[5] D. Kapur (1986): *Using Gröbner Bases to Reason About Geometry Problems*. Journal of Symbolic Computation 2(4), pp. 399–408, doi:10.1016/S0747-7171(86)80007-4.

[6] Z. Kovács & B. Parisse (2015): *Giac and GeoGebra – Improved Gröbner Basis Computations*. In: Computer Algebra and Polynomials, Lecture Notes in Computer Science, pp. 126–138, doi:10.1007/978-3-319-15081-9_7.

[7] Z. Magajna (2011): *An Observation Tool as an Aid for Building Proofs*. The Electronic Journal of Mathematics and Technology 5(3), pp. 251–260.

[8] P. Quaresma & V. Santos (2022): *Four Geometry Problems to Introduce Automated Deduction in Secondary Schools*. In: EPTCS 354, pp. 27–42, doi:10.4204/EPTCS.354.3.

[9] J. Teles, V. Santos & P. Quaresma (2023): *A Rule Based Theorem Prover: An Introduction to Proofs in Secondary Schools*. In: EPTCS 375, pp. 24–37, doi:10.4204/EPTCS.375.3.

[10] A. Thaller & Z. Kovács (2021): *Online Generation of Proofs Without Words*. In: Automated Deduction in Geometry (ADG 2021) EPTCS 352, pp. 103–105, doi:10.4204/EPTCS.352.10.

[11] Z. Ye, S.C. Chou & X.S. Gao (2010): *Visually Dynamic Presentation of Proofs in Plane Geometry, Part 1: Basic Features and the Manual Input Method*. Journal of Automated Reasoning 45, pp. 213–241, doi:10.1007/s10817-009-9162-5.

[12] Z. Ye, S.C. Chou & X.S. Gao (2010): *Visually Dynamic Presentation of Proofs in Plane Geometry, Part 2: Automated Generation of Visually Dynamic Presentations with the Full-Angle Method and the Deductive Database Method*. Journal of Automated Reasoning 45, pp. 243–266, doi:10.1007/s10817-009-9163-4.

[13] Z. Ye, S.C. Chou & X.S. Gao (2011): *An Introduction to Java Geometry Expert – (Extended Abstract)*. In: *Automated Deduction in Geometry – 7th International Workshop, ADG 2008, Shanghai, China, September 22-24, 2008. Revised Papers. Lecture Notes in Computer Science 6301*, Springer, pp. 189–195, doi:10.1007/978-3-642-21046-4_10.

Automation of Triangle Ruler-and-Compass Constructions Using Constraint Solvers

Milan Banković
Faculty of Mathematics, University of Belgrade, Serbia
milan.bankovic@matf.bg.ac.rs

In this paper, we present an approach to automated solving of triangle ruler-and-compass construction problems using finite-domain constraint solvers. The constraint model is described in the MiniZinc modeling language, and is based on the automated planning. The main benefit of using general constraint solvers for such purpose, instead of developing dedicated tools, is that we can rely on the efficient search that is already implemented within the solver, enabling us to focus on geometric aspects of the problem. We may also use the solver's built-in optimization capabilities to search for the shortest possible constructions. We evaluate our approach on 74 solvable problems from the Wernick's list, and compare it to the dedicated triangle construction solver ArgoTriCS. The results show that our approach is comparable to dedicated tools, while it requires much less effort to implement. Also, our model often finds shorter constructions, thanks to the optimization capabilities offered by the constraint solvers.

1 Introduction

One of the oldest and the most studied classes of problems in geometry is the class of construction problems: given some elements of a figure (such as a triangle), we want to find a sequence of steps to construct the remaining elements of the figure using the available tools – typically a ruler[1] and a compass. The beauty of this class of problems is that each problem is different and requires a specific, often very deep geometric knowledge to be solved. Moreover, many problems are even proven to be unsolvable.

Although geometricians love to deal with such problems by hand, for computer scientists (who also love geometry) it is tempting to try to automate the solving of construction problems. From the algorithmic point of view, the construction problems are *search* problems, and the search space is usually very large. There are two main lines of approaches here: one is to develop a specific search algorithm in some programming language with required geometric knowledge compiled into it, and the other is to use existing artificial intelligence tools that are good in solving search problems in general. In the second case, one should only specify the problem and its constraints using some input language and then leave the search to the tool.

In this paper, we advocate the second approach. More specifically, we show how finite-domain constraint solvers [6] may be used for such purpose. We develop a constraint model in the MiniZinc modeling language [4], based on the automated planning [2]. There are two main benefits of using constraint solvers for this purpose:

- the constraint solvers are very efficient search engines, and by using them we may focus on geometric aspects of the problem and on modeling the geometric knowledge required for its solving, and leave the search to the tool that is good at it.

[1] A more accurate term would be *straightedge*, since a ruler is usually equipped with measuring marks, so it can be used to measure lengths, which is typically not allowed in geometric constructions. Nevertheless, in this paper we use the term *ruler* and consider it as a synonym for a straightedge.

- the constraint solvers are usually equipped with optimization capabilities, enabling us to search for a construction that is the best in some sense (for instance, the shortest possible construction may be required). This can be done with the minimal effort, compared to developing specific search algorithms with optimization capabilities (such as *branch-and-bound* algorithms).

We compare our approach to the state-of-the-art tool for automated generation of triangle constructions ArgoTriCS [3], developed in the Prolog programming language. A detailed evaluation is performed on 74 solvable problems from the Wernick's set of triangle construction problems [7].

The rest of this paper is organized as follows. In Section 2, we introduce needed concepts and notation used in the rest of this paper. In Section 3 we describe our constraint model. Section 4 provides a detailed evaluation of the approach. Finally, in Section 5, we give some conclusions and mention some directions of the further work.

2 Background

2.1 Ruler-and-Compass Constructions

In this paper, we consider ruler-and-compass triangle constructions, where the goal is to construct all vertices of a triangle, assuming that some elements of the triangle (points, lines or angles) are given in advance. A construction consists of a sequence of steps, where in each step some new objects (points, lines, angles or circles) are constructed based on the objects constructed in previous steps. Constructions performed in each of the steps are usually *elementary* ones, such as constructing the line passing through two given points, or the point that is the intersection of two given lines, or the circle centered at a given point that contains another given point. However, in order to simplify the description of a triangle construction, some higher-level construction steps are also considered, such as constructing the tangents to a given circle from a given point, or the line perpendicular or parallel to a given line and passing through a given point, etc. Such higher-level constructions are called *compound* constructions, since they can be easily decomposed into sequences of elementary construction steps.

In this paper, we focus on the Wernick's list of triangle construction problems [7], where the following set of 16 characteristic points of a triangle is considered: the triangle vertices (A, B, C), the circumcenter O, the incenter I, the orthocenter H, the centroid G, the feet of the altitudes (H_a, H_b, H_c), the feet of the internal angles bisectors (T_a, T_b, T_c) and the midpoints of the triangle sides (M_a, M_b, M_c). Each problem from the list assumes that three different points from this set are given, and the goal is to construct all the vertices of the triangle. There are 560 such point triplets, but only 139 among them represent significantly different problems (that is, mutually non-symmetric). Among these, only 74 problems are proven to be solvable by a ruler and a compass (others either contain redundant points, or are undetermined, i.e. may have infinitely many solutions, or are proven to be unsolvable). In our work, we consider only these 74 solvable problems from the Wernick's list.

In further text, we rely on the notation used by Marinković [3]. We also assume the geometric knowledge presented in [3], as well as the set of elementary and compound construction steps used in that work.

2.2 Constraint Solving

In this work, we reduce triangle construction problems to constraint solving [6]. A finite-domain *constraint satisfaction problem* (CSP) consists of a finite set of *variables* $\mathscr{X} = \{x_1, \dots, x_n\}$, each taking values from its given finite domain $D_i = D(x_i)$, and a finite set of *constraints* $\mathscr{C} = \{C_1,\ldots,C_m\}$, which are relations over subsets of these variables. A *solution* of a CSP is an assignment $(x_1 = d_1,\ldots,x_n = d_n)$ of values to variables ($d_i \in D_i$), such that all the constraints of that CSP are satisfied. A CSP is *satisfiable* if it has at least one solution, otherwise is *unsatisfiable*. The optimization version of CSP, known as a *constrained optimization problem* (COP) additionally assumes a function f over the variables of the problem that should be minimized (or maximized), with respect to the constraints from $\mathscr{C}$.

Tools that implement procedures for solving CSPs (and COPs) are called *constraint solvers*. They are usually based on a combination of a backtrack-based search and constraint propagation [6]. Constraint solvers have been successfully used for solving many real-world problems in many fields, such as scheduling, planning, timetabling, combinatorial design, and so on.

An important step in using constraint solvers is *constraint modeling*, that is, representing a real-world problem in terms of variables and constraints. A constraint model is usually described using an appropriate *modeling language*. One such language supported by many modern constraint solvers is *MiniZinc* [4]. This language offers a very flexible high-level environment for modeling different kinds of constraints, enabling a compact and elegant way to represent some very complex problems. Examples of some high level language elements include tuples, multi-dimensional arrays, sets, aggregate functions, finite quantification and so on. Since most of these high level constructs are not supported by backend solvers, each MiniZinc model must be translated into an equivalent *FlatZinc* form, containing only primitive language constructs and constraints supported by a chosen backend solver. MiniZinc supports modeling of both CSPs and COPs.

In MiniZinc, we distinguish *variables* from *parameters*. MiniZinc variables correspond to the variables of a CSP, i.e. we declare their domains and expect from the solver to find their values satisfying the constraints. On the other hand, parameters are just named constants, and their values must be known when the model is translated to the FlatZinc form (i.e. before the solving starts). Parameters are the language's construct that allow us to specify a general model for a class of problems, and then to choose a specific instance of the problem by fixing the values of the model's parameters. Parameter values are usually provided in separate files (called *data files*), so that we can easily combine the same model with different data.

In our work, we use MiniZinc as a modeling language.

2.3 Automated Planning

In our approach, triangle construction problems are considered as problems of *automated planning* [2]. An automated planning problem consists of the following:

- a set $\mathscr{S}$ of possible *states*, which are usually encoded by a set of variables $\mathscr{V}$ and the values assigned to them. One distinguished state $S_0 \in \mathscr{S}$ is the *initial state*.
- a set of operators O, where each operator $o \in O$ consists of a *precondition* C_o describing the conditions (in terms of the variables from $\mathscr{V}$) that must be satisfied in the current state for the operator to be applied, and a set of *effects* E_o (represented as variable-value assignments) describing how the current state is changed when o is applied to it. The state obtained by applying an operator o to some state S is denoted by $o(S)$.
- a goal G, describing the conditions (in terms of the variables from $\mathscr{V}$) that must be satisfied in the final state.

The objective of automated planning is to find *a plan*, that is, a finite sequence of operators $o_1,\ldots,o_n$ from O that can be successively applied to the initial state S_0 (i.e. for each $i \in \{1,\ldots,n\}$, we have

$S_i = o_i(S_{i-1})$, and the state S_{i-1} satisfies the precondition C_{o_i}) producing the final state S_n satisfying the goal G. The number n of operators used in a plan is called the *length* of the plan.

The problem of checking whether a plan (of any finite length) exists is PSPACE-complete in general [1]. For a fixed plan length n, the problem is NP-complete in general, and can be encoded as a CSP [2, 5].

The optimization variant of the planning problem (i.e. finding a plan of the minimal possible length) can be solved by successively checking for existence of plans of lengths $n = 1, 2, 3, \ldots$, that is, by solving the corresponding sequence of CSPs until a satisfiable one is encountered.

3 Model Description

The triangle construction problems that we consider in this paper can be naturally described as problems of automated planning:

- states correspond to the sets of constructed objects, and the initial state is the set consisting of the given elements of the triangle (three points in case of Wernick's problems).
- operators correspond to the construction steps; the precondition for each operator is that objects used in the corresponding construction step are already constructed (i.e. belong to the current state), and that corresponding non-degeneracy and determination conditions are satisfied (e.g. two lines must be distinct and non-parallel in order to construct their intersection); the effect of each operator is the addition of the objects constructed by the corresponding construction step to the current state.
- the goal condition is that vertices A, B and C belong to the final state.

The corresponding planning problem for a fixed plan (construction) length is encoded as a CSP using the MiniZinc language.[2] In the rest of this section, we discuss different aspects of the encoding in more detail.

3.1 Encoding of Geometric Knowledge

Encoding objects. We consider four types of objects: points, lines, circles and angles. Each of these types is encoded as an *enumeration type* in MiniZinc (`Point`, `Line`, `Circle` and `Angle`, respectively), and each object is represented by one enumerator of the corresponding type. The enumerated objects are the only objects that can be constructed. This means that we have to anticipate in advance the set of objects that might be needed during the construction.

Encoding relations. Different relations between the enumerated objects are encoded by the parameters of the model, using MiniZinc's arrays, sets and tuples. These relations are used to statically encode the geometric knowledge used in the constructions. We define the following types of relations:

- *incidence relations*: we define two arrays of sets, `inc_lines` and `inc_circles`, indexed by points. The set `inc_lines`[p] contains the lines incident with the point p, and the set `inc_-circles`[p] contains the circles incident with the point p.
- *relations between lines*: we define two arrays of sets, `perp_lines` and `paralell_lines`, indexed by lines. The set `perp_lines`[l] contains the lines perpendicular to the line l, and the set `parallel_lines`[l] contains the lines parallel to the line l.

[2]The model is available at: `https://github.com/milanbankovic/constructions/`.

- *circle tangents, diameters and centers*: the array of points `circle_center` indexed by circles contains information about circle centers; the array `circle_diameter` of point pairs indexed by circles contains information about circle diameters; the array `tangent_lines` of sets of lines is indexed by circles, and the set `tangent_lines`$[c]$ contains the lines that are tangents of the circle c.
- *vector ratios*: we use the array `known_ratio_triplets` of point triplets to store the information about the triplets of collinear points (X,Y,Z) such that the ratio $\overrightarrow{XY}/\overrightarrow{YZ}$ is known. The exact value of the ratio is not encoded, since it is not important for the search (it is only important to know that we can construct one of the points if the remaining two are already constructed). Similarly, we use the array `known_ratio_quadruplets` to encode quadruplets of points (X,Y,Z,W) such that the ratio $\overrightarrow{XY}/\overrightarrow{ZW}$ is known.
- *angles between the lines*: we use the array `angle_defs` of `Line` $\times$ `Line` $\times$ `Angle` triplets, to encode the information about the angles between the lines. A triplet (p,q,ϕ) means that the angle between the lines p and q is determined by ϕ (e.g. is equal to $\phi/2$ or $\phi+\pi/2$). Such information can be used in two directions: if we have constructed p and q, we can construct the angle ϕ; also if we have constructed p and ϕ, and the intersection point X of p and q, we can then construct the line q.
- *perpendicular bisectors of segments*: we use the array `perp_bisectors` of `Point` $\times$ `Point` $\times$ `Line` triplets to encode the information about the perpendicular bisectors of line segments.
- *harmonic conjugates*: we use the array `harmonic_quadruplets` of point quadruplets, where a quadruplet $(X,Y;Z,W)$ encodes that the points X and Y are harmonic conjugates of each other with respect to the pair (Z,W).
- *loci of points*: we use the array `locus_defs` of `Point` $\times$ `Point` $\times$ `Angle` $\times$ `Circle` tuples, where a tuple (X,Y,ϕ,c) encodes that the locus of points such that the segment XY is seen at an angle determined by ϕ is an arc of the circle c.
- *homothetic images of lines*: we use the array `homothety_triplets` of `Point` $\times$ `Line` $\times$ `Line` triplets, where a triplet (X,p,q) encodes that the line q is the image of the line p by homothety centered in the point X (again, homothety coefficient is not stored in the database).

3.2 Encoding of the Planning Problem

Encoding of states. Let n be the length of a plan that we are searching for, let S_0 be the initial state, and let S_i be the state after the ith step. To encode these states, we introduce arrays of variable sets `known_-points`, `known_lines`, `known_circles` and `known_angles`, where, for instance, `known_points`$[i]$ ($i \in \{0,\ldots,n\}$) denotes the set of points belonging to the state S_i (similarly for other arrays). The initial state S_0 is fixed in advance by appropriate constraints (for instance `known_points`$[0] = \{A,G,O\}$).

Encoding the plan. We define the enumeration type `ConsType`, with one enumerator for each supported type of construction step. We also define the array `construct` of variables of type `ConsType` (with indices in $\{1,\ldots,n\}$) encoding operators used in each step (i.e. the construction step types). For each step, we also need additional information to fully determine the actual construction (for instance, if we choose to construct the intersection of two lines, we must also choose the lines that we want to intersect). For this reason, we also introduce additional two-dimensional arrays of variables: for instance, `points`$[i][j]$ denotes the jth point used in the ith construction step (similarly we have `lines`$[i][j]$, `circles`$[i][j]$ and `angles`$[i][j]$).

Encoding the state transitions. Finally, to glue the whole plan together, we must add the constraints that connect the state variables in the successive states, depending on the chosen operator in the corresponding step. This must be done for each $i \in \{1, \dots, n\}$, and that is where MiniZinc's finite universal quantification comes in handy:

```
constraint forall(i in 1..n)
(
   construct[i] = LineIntersect ->
       % Precondition
       (lines[i,1] in known_lines[i-1] /\
        lines[i,2] in known_lines[i-1] /\
        lines[i,1] != lines[i,2] /\
        not (lines[i,1] in parallel_lines[lines[i,2]]) /\
        lines[i,1] in inc_lines[points[i,1]] /\
        lines[i,2] in inc_lines[points[i,1]] /\
        not (points[i,1] in known_points[i-1]) /\
       % Effects
        known_points[i] = known_points[i-1] union { points[i,1] } /\
        known_lines[i] = known_lines[i-1] /\
        known_circles[i] = known_circles[i-1] /\
        known_angles[i] = known_angles[i - 1]
       )
);
```

That is, for all $i \in \{1, \dots, n\}$, if the chosen operator is `LineIntersect` (constructing the intersection of two lines), then the chosen two lines $\texttt{lines}[i,1]$ and $\texttt{lines}[i,2]$ must belong to the current state S_{i-1} (i.e. they must have been already constructed), they must be distinct and not parallel. Also, the chosen point $\texttt{points}[i,1]$ must belong to both chosen lines (i.e. it must be their intersection), and it must not belong to the current state (we do not want to construct a point that is already constructed). If all these preconditions are met, then the effect is that the set $\texttt{known\_points}[i]$ is obtained by adding the intersection point $\texttt{points}[i,1]$ to the set $\texttt{known\_points}[i-1]$ (the sets of lines, circles and angles remain the same). Similar constraints are defined for all other types of construction steps.

Encoding the goal. The goal is encoded simply by adding the constraints that require that the vertices A, B and C belong to the set $\texttt{known\_points}[n]$:

```
{ A, B, C } subset known_points[n];
```

4 Evaluation

The model described in the previous section is evaluated on 74 solvable instances from Wernick's set [7]. The experiments were performed on a computer with 3.1GHz processor and 8Gb of RAM. We used official MiniZinc distribution[3] for experiments (version 2.7.2). We have experimented with different

[3]https://www.minizinc.org/software.html

backend constraint solvers provided within MiniZinc distribution, and by far the best results were obtained by the `chuffed`[4] solver. Therefore, in the rest of this section, we present only the results obtained by `chuffed`.

We looked for plans of minimal lengths (i.e. constructions with the minimal possible numbers of steps). We used three different setups:

- *linear setup*: for each of the problems, we successively look for plans of length $n = 1, 2, 3, \ldots$, and stop when we encounter a satisfiable CSP, or when some upper limit $maxSteps$ is exceeded. This is the usual way for finding plans of minimal lengths in automated planning [5]. In our experiments, the upper limit for the plan length was set to 11, since our preliminary experiments had shown that all the problems that our model could solve had been solved in at most 11 steps. Note that in this setup the value of $maxSteps$ does not affect the solving time for problems that our model can solve (that is, using a greater value of $maxSteps$ would not slow down the search).
- *minimization setup*: we reformulate our model such that the plan length n is not fixed. Instead, n is a variable with a domain $\{1, \ldots, maxSteps\}$ and we are trying to minimize the value of n (that is, we are solving a constrained optimization problem). The problem with this approach is how to determine the value of $maxSteps$ parameter, since in this setup greater values of this parameter make the model larger and the search becomes slower, even for problems that can be solved in a small number of steps. In our experiments, we used the value $maxSteps = 11$, but this was somewhat artificial choice, since we used the previous knowledge to choose the minimal possible number of steps sufficient to solve all the problems that our model was able to solve.
- *incremental setup*: just like in the previous setup, we reformulate our model such that we are trying to minimize n, but this time the domain for n is some interval $\{l, \ldots, u\}$, where l and u are parameters. Now we successively solve constrained optimization problems for intervals $\{1, \ldots, k\}$, $\{k+1, \ldots, 2k\}$,$\{2k+1, \ldots, 3k\}$,$\ldots$, until some of them turns out to be satisfiable, or until some upper limit $maxSteps$ is exceeded. Like in the first setup, the choice for the value of the parameter $maxSteps$ does not affect the solving time for the problems that are solvable by our model. On the other hand, the number of COPs solved is smaller roughly by the factor k, compared to the first setup. We have experimented with multiple choices for k, and the best results were obtained for $k = 3$.

In Table 1, we provide the main results of our evaluation. We have evaluated all three setups described above. We also compared our approach to the results obtained by the ArgoTriCS dedicated triangle construction solver developed by Marinković [3]. ArgoTriCS is implemented in Prolog programming language, but it uses a very similar knowledge base and an almost identical set of available construction steps.

Setup	**# solved**	**Avg. time**	**Median time**	**Avg. time on solved**	**Avg. length**
linear	63	97.9	22.0	58.5	6.3
minimization	63	43.8	10.8	29.7	6.3
incremental ($k = 3$)	63	66.1	12.0	39.9	6.3
ArgoTriCS	65	54.5	21.6	54.4	7.5

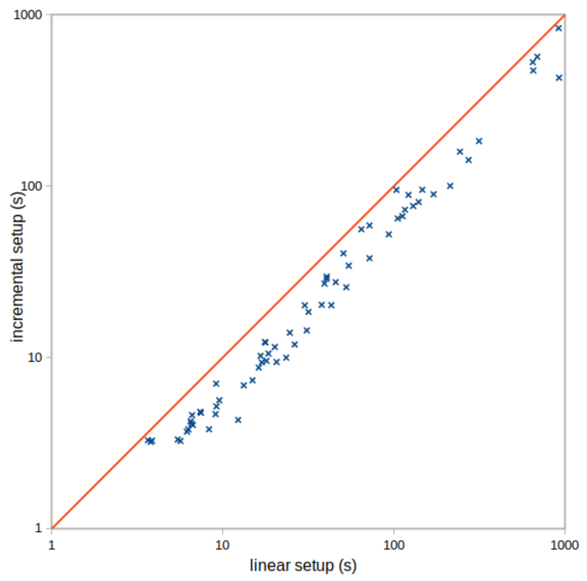
Table 1: Overall results for different setups, compared to ArgoTriCS. Times are given in seconds

[4] `https://github.com/chuffed/chuffed`

Note that the choice of the setup does not affect how many problems from Wernick's list will be solved, since this depends only on the geometric knowledge that is compiled into our model.[5] In total, we managed to solve 63 of 74 problems (for the remaining 11 problems, the constraint solver reported unsatisfiability). On the other hand, ArgoTriCS solved 2 problems more. This is because we missed to incorporate some of the objects and lemmas known to ArgoTriCS to our model.

The best average solving time is obtained by the minimization setup. However, as we mentioned earlier, the average solving time in this setup greatly depends on the choice for the maximal possible value of n. The results shown in Table 1 are obtained for $maxSteps = 11$. We also experimented with some greater values. For instance, for $maxSteps = 20$ the average solving time was over 100 seconds, that is, more than twice greater (of course, the number of solved problems remained the same).

The linear setup has shown the worst performance. This is because in this setup we were solving many unsatisfiable CSPs until we possibly reached some satisfiable CSP. Unsatisfiable CSPs tend to consume more time, especially those that are "almost satisfiable", that is, that are close to some phase transition point. This phenomenon is well-known in automated planning [5].



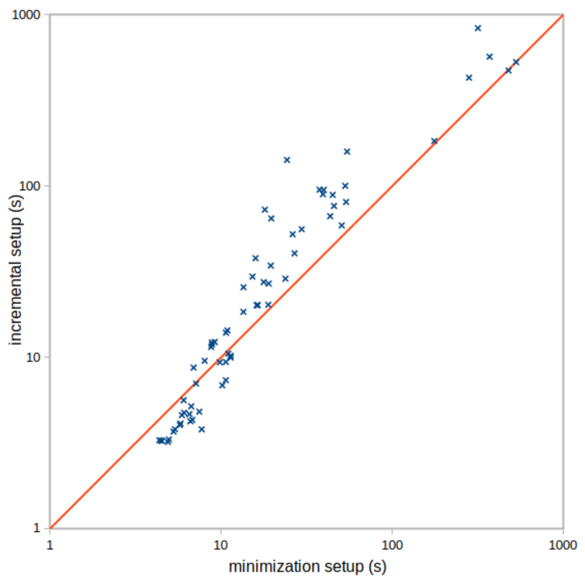


Figure 1: Per-instance comparison of different setups. Times are given in seconds

The performance of the incremental setup was much better on average than in case of the linear setup, and a little worse than in case of the minimization setup, but still comparable. A more detailed, per-instance comparison is shown in Figure 1. We can see that the incremental setup was uniformly better than linear setup, and was also better than the minimization setup on easier instances, but it was outperformed by the minimization setup on harder problems. Overall, the incremental setup seems as a good choice in a realistic context, when we do not know in advance the value of $maxSteps$ parameter.

The overall performance of the ArgoTriCS solver was comparable to our approach, when the average solving time is concerned. However, we may notice that its median solving time was almost twice

[5]This means that we can improve our results by carefully examining the knowledge needed for solving the unsolved problems, and incorporating that knowledge into our model. However, such enrichment of the model enlarges the search space and makes the solving slower even for the problems that are already solvable by our model.

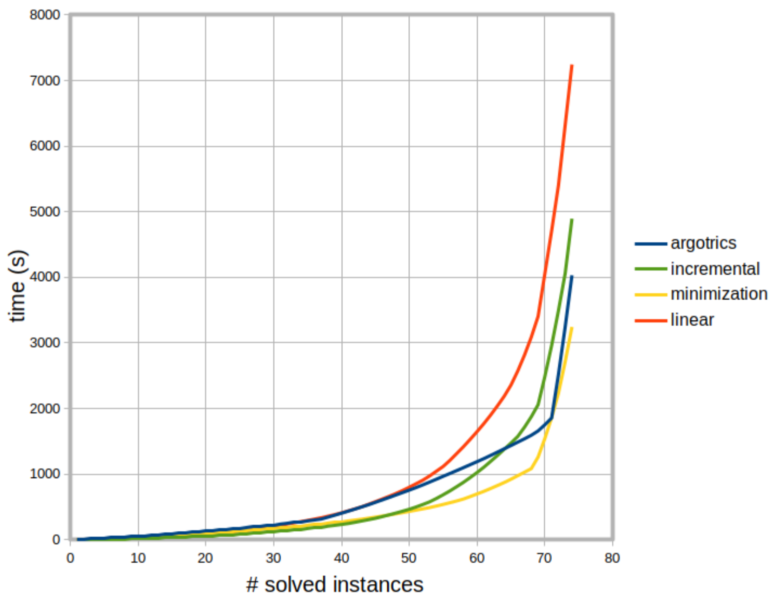


Figure 2: Survival plot for all three setups, compared to ArgoTriCS. Times are given in seconds

greater than in case of our minimization or incremental setup. Also, the average solving time on solved instances was much better in our approach. This suggests that our approach performed better than (or comparable to) ArgoTriCS on problems for which it managed to find a construction plan, especially on easier instances. This is confirmed in Figure 2, which shows the survival plot for all three setups and ArgoTriCS. The minimization setup was clearly the best, while the linear setup was the worst. When compared to ArgoTriCS, the incremental setup was cumulatively better on more than 60 instances, which were roughly all the instances that our model managed to solve. This means that if our model can find a solution, it can do it fast, while its performance is much worse when it comes to the instances that are out of its reach (that is, when the corresponding CSPs are unsatisfiable). On the other hand, the performance of ArgoTriCS had much smaller variance – it performed almost equally solid on all instances (as it can be seen from Table 1, the average solving time on solved instances for ArgoTriCS is almost the same as the average solving time on all instances).

The final comparison between ArgoTriCS and our approach concerns the lengths of the obtained constructions. Table 1 shows that the average plan length in our approach was 6.3 (again, this does not depend on the chosen setup). On the other hand, the average number of steps in ArgoTriCS's constructions was 7.5. Notice that these numbers are comparable, since the sets of available construction steps in both systems are almost identical. A more detailed, per-instance comparison is shown in Figure 3. The plot clearly confirms that our approach is by far superior when finding the shortest constructions is concerned. However, for the sake of fairness, we must stress that ArgoTriCS was not designed with that optimization in mind, that is, it does not even search for the shortest solutions. We guess that such a capability could be integrated in ArgoTriCS, but with much more effort, since it would have to be manually implemented in Prolog (just like the search itself). On the other hand, in our approach, we rely on the built-in capabilities of constraint solvers to solve optimization problems efficiently, imposing the minimal possible effort on our side.

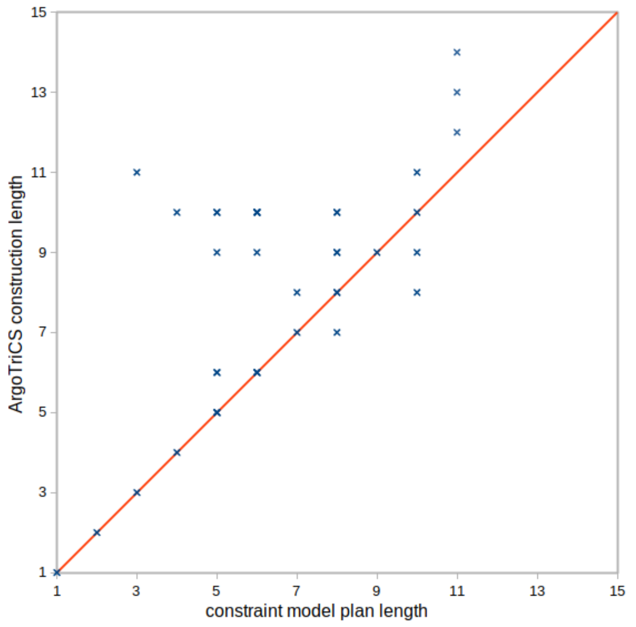


Figure 3: A per-instance comparison of construction (plan) lengths between ArgoTriCS and our approach

5 Conclusions and Further Work

In this paper we presented and evaluated a method for automated triangle construction based on constraint solving. We compared our method to the state-of-the-art dedicated triangle construction solver ArgoTriCS, developed in Prolog programming language. We advocate that our approach has two important advantages. First, our approach is much simpler to implement, since we rely on powerful constraint solvers which can efficiently do the search for us, and we may focus only on modeling. On the other side, in the ArgoTriCS solver the search is implemented by hand, in more than 500 lines of code. Second, we can easily employ the optimization capabilities of modern constraint solvers to search for the shortest possible constructions, while implementing such functionality in ArgoTriCS would require much more effort.

We evaluated our approach on 74 solvable problems from the Wernick's list. The results showed that our approach is comparable to ArgoTriCS when solving time is concerned. On the other hand, our model often finds shorter constructions, due to built-in optimization capability which is missing in ArgoTriCS.

For further work, we plan to extend our model to support construction problems from other sets. This should not be a hard task in the technical sense, since the model is developed such that it can be easily extended (that is, we can easily add new objects, relations and construction step types). The real challenge is to recognize and integrate the geometric knowledge needed for such constructions into the model. Of course, this is a job for geometricians, and our goal was to provide them with (what we hope is) a useful tool that can free them from the tedious task of programming, and let them focus on what they do the best and love the most.

Acknowledgements. This work was partially supported by the Serbian Ministry of Science grant 174021. We are very grateful to the anonymous reviewers whose insightful comments and remarks helped us to make this paper much better.

References

[1] Tom Bylander (1994): *The computational complexity of propositional STRIPS planning*. Artificial Intelligence 69(1-2), pp. 165–204, doi:10.1016/0004-3702(94)90081-7.

[2] Malik Ghallab, Dana Nau & Paolo Traverso (2004): *Automated Planning: theory and practice*. Elsevier.

[3] Vesna Marinković (2017): *ArgoTriCS–automated triangle construction solver*. Journal of Experimental & Theoretical Artificial Intelligence 29(2), pp. 247–271, doi:10.1080/0952813X.2015.1132271.

[4] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck & Guido Tack (2007): *MiniZinc: Towards a standard CP modelling language*. In: Principles and Practice of Constraint Programming–CP 2007: 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007. Proceedings 13, Springer, pp. 529–543, doi:10.1007/978-3-540-74970-7_38.

[5] Jussi Rintanen (2009): *Planning and SAT*. Handbook of Satisfiability 185, pp. 483–504.

[6] Francesca Rossi, Peter Van Beek & Toby Walsh (2006): *Handbook of constraint programming*. Elsevier.

[7] William Wernick (1982): *Triangle constructions with three located points*. Mathematics Magazine 55(4), pp. 227–230, doi:10.1080/0025570X.1985.11976988.

Towards an Independent Version of Tarski's System of Geometry

Pierre Boutry
Centre Inria d'Université Côte d'Azur, Sophia Antipolis, France
pierre.boutry@inria.fr

Stéphane Kastenbaum
No affiliation
stephane.kastenbaum@gmail.com

Clément Saintier
No affiliation
clement.saintier@gmail.com

In 1926–1927, Tarski designed a set of axioms for Euclidean geometry which reached its final form in a manuscript by Schwabhäuser, Szmielew and Tarski in 1983. The differences amount to simplifications obtained by Tarski and Gupta. Gupta presented an independent version of Tarski's system of geometry, thus establishing that his version could not be further simplified without modifying the axioms. To obtain the independence of one of his axioms, namely Pasch's axiom, he proved the independence of one of its consequences: the previously eliminated symmetry of betweenness. However, an independence model for the non-degenerate part of Pasch's axiom was provided by Szczerba for another version of Tarski's system of geometry in which the symmetry of betweenness holds. This independence proof cannot be directly used for Gupta's version as the statements of the parallel postulate differ.

In this paper, we present our progress towards obtaining an independent version of a variant of Gupta's system. Compared to Gupta's version, we split Pasch's axiom into this previously eliminated axiom and its non-degenerate part and change the statement of the parallel postulate. We verified the independence properties by mechanizing counter-models using the Coq proof-assistant.

1 Introduction

The independence[1] of axioms for geometry has often been an important topic in the field of geometry. For centuries, many mathematicians believed that Euclid's fifth postulate was rather a theorem which could be derived from the first four of Euclid's postulates. History is rich with incorrect proofs of Euclid's fifth postulate. In 1763, Klügel provided, in his dissertation, a survey of about 30 attempts to "prove Euclid's parallel postulate" [13]. The question was finally settled in 1832 and 1840, when Bolyai [5] and Lobachevsky [14] exhibited models of hyperbolic geometry, thus establishing that this postulate was independent. Later, Hilbert dedicated the second section of his famous *Grundlagen der Geometrie* [12] to independence properties. Then, when working out the final version [22] of the axioms for *Metamathematische Methoden in der Geometrie* [19], commonly referred to as SST, independence results proved very helpful.

So it is a surprise that, now that we have access to tools like proof assistants which we believe to be perfectly suited for the task, the only independence to be mechanized was the one for Euclid's fifth postulate [18]. To the best of our knowledge, the most recent work on the topic is the formalization of the Poincaré disk model in Isabelle/HOL [20].

[1] We recall that an axiom is said to be independent from a set of axiom if it is not derivable from the axioms in this set.

In this paper, we study independence properties linked to SST [19]. SST has the advantage of being expressed in the first-order language rather than natural language which leaves room for interpretation leading to possible problems [8].[2] There are several ways to prove independence results [4]. Here we focus on independence through counter-model, i.e. constructing a model where the axiom to be proven independent will not hold while all the others will.

In 1965, Gupta presented an independent version of Tarski's system of geometry [11]. To obtain the independence of one of his axioms, namely Pasch's axiom, he proved the independence of one of its consequences: the previously eliminated symmetry of betweenness. However, an independence model for the non-degenerate part of Pasch's axiom was provided by Szczerba for another version of Tarski's system of geometry in which the symmetry of betweenness holds [21]. This independence proof cannot be directly used for Gupta's version as the statements of the parallel postulate differ. This can be remedied by carefully choosing the statement of the parallel postulate amongst the ones known to be equivalent [7]. We aim to verify that splitting Pasch's axiom into its non-degenerate part and the symmetry of betweenness in addition to changing the statement of the parallel postulate allows to obtain a system that is still independent. So we go in the opposite direction of what Makarios did by removing the need of the reflexivity properties for congruence thanks to a modification to the five-segment property [17]. Indeed, our view is that an axiom should capture a limited and well-defined property, instead of trying to minimize the needed number of axioms at any cost.

We remark that a small change in the statement of an axiom can change whether or not it holds in a specific model. This makes a computer very well suited to the verification that an axiom holds in a model. So we chose to mechanize the various counter-models needed for this task in the Coq proof-assistant [23].

The rest of the paper is structured as follows. First, in Sec. 2, we present the system we will be working on throughout the rest of this paper. Then, in Sec. 3, we show how to build a model of Tarski's axiom. Finally, before concluding on the achieved results, we present an example of independence proof in Sec. 4.

2 A Variant of Tarski's System of Geometry

In this section, we start by recalling the axioms of Tarski's system of geometry. Then, we present the modification Gupta made to it to obtain a fully independent system [11]. Finally, we describe how to modify his system to combine his results and the ones from Szczerba [21].

2.1 Tarski's System of Geometry

Tarski's axiom system is based on a single primitive type depicting points and two predicates, namely congruence and betweenness. $AB \equiv CD$ states that the segments $\overline{AB}$ and $\overline{CD}$ have the same length. A–B–C means that A, B and C are collinear and B is between A and C (and B may be equal to A or C). For an explanation of the axioms and their history see [22]. Table 1 lists the axioms for planar Euclidean geometry.

[2]A possible interpretation of Hilbert's axiom could lead to a degenerate model for first two groups of Hilbert's axioms.

A1	Symmetry	$AB \equiv BA$
A2	Pseudo-Transitivity	$AB \equiv CD \wedge AB \equiv EF \Rightarrow CD \equiv EF$
A3	Cong Identity	$AB \equiv CC \Rightarrow A = B$
A4	Segment construction	$\exists E, A\text{—}B\text{—}E \wedge BE \equiv CD$
A5	Five-segment	$AB \equiv A'B' \wedge BC \equiv B'C' \wedge$ $AD \equiv A'D' \wedge BD \equiv B'D' \wedge$ $A\text{—}B\text{—}C \wedge A'\text{—}B'\text{—}C' \wedge A \neq B \Rightarrow CD \equiv C'D'$
A6	Between Identity	$A\text{—}B\text{—}A \Rightarrow A = B$
A7	Inner Pasch	$A\text{—}P\text{—}C \wedge B\text{—}Q\text{—}C \Rightarrow \exists X, P\text{—}X\text{—}B \wedge Q\text{—}X\text{—}A$
A8	Lower Dimension	$\exists ABC, \neg A\text{—}B\text{—}C \wedge \neg B\text{—}C\text{—}A \wedge \neg C\text{—}A\text{—}B$
A9	Upper Dimension	$AP \equiv AQ \wedge BP \equiv BQ \wedge CP \equiv CQ \wedge P \neq Q \Rightarrow$ $A\text{—}B\text{—}C \vee B\text{—}C\text{—}A \vee C\text{—}A\text{—}B$
A10	Euclid	$A\text{—}D\text{—}T \wedge B\text{—}D\text{—}C \wedge A \neq D \Rightarrow$ $\exists XY, A\text{—}B\text{—}X \wedge A\text{—}C\text{—}Y \wedge X\text{—}T\text{—}Y$
A11	Continuity	$(\exists A, (\forall XY, \Xi(X) \wedge \Upsilon(Y) \Rightarrow A\text{—}X\text{—}Y)) \Rightarrow$ $\exists B, (\forall XY, \Xi(X) \wedge \Upsilon(Y) \Rightarrow X\text{—}B\text{—}Y)$

Table 1: Tarski's axiom system for planar Euclidean geometry.

2.2 Gupta's Contribution

The problem of the independence of Tarski's axiom system, as defined in Table 1, remains open. Let us introduce the modifications Gupta made to it to obtain an independent system. He reintroduced the inner transitivity of betweenness A15[3] in Table 2. Having added this axiom, the identity axiom for betweenness A6 became a theorem and could then be removed from the system. Finally A2, A9 and A11 are replaced by A2', A9' and A11'. We omit the details of how to mechanize in Coq that this system, consisting of A1, A2', A3-A5, A7, A8, A9', A10, A11' and A15,[4] and Tarski's system are equivalent. Gupta proves that this system is independent. To prove that A7 is independent in this system, he shows that A14, a consequence of A7 in this system, does not hold. However, Szczerba [21] found that A7 does not hold when A14 and all the other axioms, with the exception of A10, in Gupta's system do. So this would suggest that A7 can be split into A14 and a variant of A7 while still having an independent system.

2.3 An Independent System for Planar Geometry

The system that we want to prove independent is very close to the one Gupta studied in his thesis [11]. We split Pasch's axiom A7 into its non-degenerate part A7' and A14, change the version of the parallel postulate A10 and add one axiom (for reasons explained later). A7' excludes from A7 the degenerate cases where the triangle ABC is flat or when P or Q are respectively not strictly between A and C or B and C. We cannot use A10 as it does not hold in the counter-model found by Szczerba [21]. We chose Proclus postulate,[5] denoted as A10' in Table 2, verified to be equivalent to it when assuming A0-A9, using Coq [7], as it holds in all the counter-models provided by Gupta as well as in the one found by

[3]We number them as in [22].

[4]Actually the statement for A7 differs in [22] but the change is not important here.

[5]Col ABC and $AB \parallel CD$ denotes that A, B and C are collinear and that lines AB and CD are parallel according to the definitions given in SST [19].

A0	Point equality decidability	$X = Y \lor X \neq Y$
A2'	Pseudo-Transitivity	$AB \equiv EF \land CD \equiv EF \Rightarrow AB \equiv CD$
A7'	Inner Pasch	$A\text{—}P\text{—}C \land B\text{—}Q\text{—}C \land$ $A \neq P \land P \neq C \land B \neq Q \land Q \neq C \land$ $\neg(A\text{—}B\text{—}C \lor B\text{—}C\text{—}A \lor C\text{—}A\text{—}B) \Rightarrow$ $\exists X, P\text{—}X\text{—}B \land Q\text{—}X\text{—}A$
A9'	Upper Dimension	$AP \equiv AQ \land BP \equiv BQ \land CP \equiv CQ \land$ $P \neq Q \land A \neq B \land A \neq C \land B \neq C \Rightarrow$ $A\text{—}B\text{—}C \lor B\text{—}C\text{—}A \lor C\text{—}A\text{—}B$
A10'	Proclus	$AB \parallel CD \land \mathrm{Col}\, A\,B\,P \land \neg\mathrm{Col}\, A\,B\,Q \Rightarrow$ $\exists Y, \mathrm{Col}\, C\,D\,Y \land \mathrm{Col}\, P\,Q\,Y$
A11'	Continuity	$(\exists A, (\forall XY, \Xi(X) \land \Upsilon(Y) \Rightarrow A\text{—}X\text{—}Y)) \Rightarrow$ $\exists B, (\forall XY, \Xi(X) \land \Upsilon(Y) \Rightarrow$ $X = B \lor B = Y \lor X\text{—}B\text{—}Y)$
A14	Between Symmetry	$A\text{—}B\text{—}C \Rightarrow C\text{—}B\text{—}A$
A15	Between Inner Transitivity	$A\text{—}B\text{—}D \land B\text{—}C\text{—}D \Rightarrow A\text{—}B\text{—}C$

Table 2: Added axioms to Tarski's system of geometry.

Szczerba, thanks to Theorem 1 in [21]. Finally, the formal development found in SST [19] is essentially classical due to the many case distinctions found in the proofs of its lemmas. However, the decidability of point equality is sufficient to obtain the arithmetization of geometry in an intuitionistic setting [6]. So we add the decidability of point equality A0 so that we can work in an intuitionistic setting. The reader not familiar with the difference between classical and intuitionistic logic may refer to [1]. This system, consisting of A0, A1, A2', A3-A5, A7', A8, A9', A10', A11', A14 and A15, and Tarski's system are equivalent. Again, we do not detail how to mechanize this fact in Coq.

3 A model of Tarski's system of geometry

In this section, we present our proof that Cartesian planes over a Pythagorean ordered field form a model of the variant of Tarski's system of geometry that we have introduced in the previous section. First, we present the structure that we used to define this model. Then we define the model that we used, that is, the way we instantiated the signature of this system. Finally, we detail the proofs of some of the more interesting axioms.

3.1 The *Real Field* Structure

The structure that was used to define this model was built by Cohen [9]. The *real field* structure results of the addition of operators to a discrete[6] field: two boolean comparison functions (for strict and non-strict order) and a norm operator. Elements of this *real field* structure verify the axioms listed in Table 3. Finally, the elements of a *real field* structure are all comparable to zero. We should remark that this field is not necessarily Pythagorean. In fact, there is no defined structure in the *Mathematical Components* library [15] for Pythagorean fields. This can however be added much more easily than before thanks to the

[6]Discrete fields are fields with a decidable equality.

Subadditivity of the norm operator	$\lvert x+y\rvert \leq \lvert x\rvert + \lvert y\rvert$
Compatibility of the addition with the strict comparison	$0 < x \wedge 0 < y \Rightarrow 0 < x+y$
Definiteness of the norm operator	$\lvert x\rvert = 0 \Rightarrow x = 0$
Comparability of positive numbers	$0 \leq x \wedge 0 \leq y \Rightarrow (x \leq y) \vert\vert (y \leq x)$
The norm operator is a morphism for the multiplication	$\lvert x * y\rvert = \lvert x\rvert * \lvert y\rvert$
Large comparison in terms of the norm	$(x \leq y) = (\lvert y-x\rvert == y-x)$[7]
Strict comparison in terms of the large comparison	$(x < y) = (y \mathrel{!}= x) \&\& (x \leq y)$

Table 3: Axioms of the *real field* structure.

recent modification of the *Mathematical Components* library to make use of the *Hierarchy Builder* [10]. However, the Pythagorean property is only required for the proof of the segment construction axiom A4. So we chose to prove that this axiom holds in our model by admitting an extra axiom which was defined in this library: the real closed field axiom. It states that intermediate value property holds for polynomial with coefficients in the field. While it is much stronger than Pythagoras' axiom, we only used it to be able to define the square root of a number which is a sum of squares and would therefore have a square root in a Pythagorean field. Finally, we did not yet prove that $A11$ holds in our model since it would require a much more involved effort. Indeed, this is similar to verifying that Tarski's system of geometry admits a quantifier elimination procedure.

3.2 The Model

Let us now define our model. Being based on a single primitive type and two predicates, the signature of Tarski's system of geometry is rather simple. However, this system has the advantage of having a n-dimensional variant. To obtain this variant, one only needs to change the dimension axioms. So far, we have restricted ourselves to the planar version of this system. With a view to extend the *GeoCoq* library to its n-dimensional variant, we wanted to define a model in which we could prove all but the dimension axioms in an arbitrary dimension to be able to construct a model of the n-dimensional variant by only proving the new dimension axioms. Hence we chose to define `Tpoint` as a vector of dimension $n+1$ with coefficient in the *real field* structure $\mathbb{F}$ (we used the *real field* structure for all the development with the exception of the proof of the segment construction axiom) for a fixed integer n, that is `'rV[R]_(n.+1)`. We adopted Gupta's definition [11] for the congruence `cong`, namely that $AB \equiv CD$ if the squares of the Euclidean norms of $B-A$ and $D-C$ are equal. Actually Gupta also proved that any model of the n-dimensional variant of Tarski's system of geometry is isomorphic to his model. He defined that A—B—C holds if and only if there exists a $k \in \mathbb{F}$ such that $0 \leq k \leq 1$ and $B-A = k(C-A)$. In fact, if such a k exists, it can be computed. By letting $A = (a_i)_{1\leq i\leq n+1}$, $B = (b_i)_{1\leq i\leq n+1}$ and $C = (c_i)_{1\leq i\leq n+1}$, if $A \neq C$ then there exists a $i \in \mathbb{N}$ such that $1 \leq i \leq n+1$ and $a_i \neq c_i$ and in this case we set k to $\frac{b_i-a_i}{c_i-a_i}$ and if $A = C$ we set k to zero. Therefore we defined a function `ratio` that computes the possible value for k, thus allowing us to define the betweenness by the boolean equality test. This was actually important as it permitted to directly manipulate the definition for betweenness by rewriting since we defined it as a boolean test. Finally, as it was often necessary to distinguish whether or not A—B—C holds due to a degeneracy, we split the definition `bet` of the betweenness into two predicates: the first one, `betS`, capturing the general case of k being strictly between 0 and 1 and the second one, `betE`, capturing the three possible degenerate cases, namely either $A = B$, $B = C$ or $A = B$ and $B = C$.

[7]$==$ denotes the boolean equality test for the elements of the field.

Formally, we consider the following model:

```
Variable R : realFieldType.
Variable n : nat.

Implicit Types (a b c d : 'rV[R]_(n.+1)).

Definition cong a b c d := (b - a) *m (b - a)^T == (d - c) *m (d - c)^T.

Definition betE a b c := [ || [ && a == b & b == c ], a == b | b == c ].

Definition ratio v1 v2 :=
  if [pick k : 'I_(n.+1) | v2 0 k != 0] is Some k
  then v1 0 k / v2 0 k else 0.

Definition betR a b c := ratio (b - a) (c - a).

Definition betS a b c (r := betR a b c) :=
  [ && b - a == r *: (c - a), 0 < r & r < 1].

Definition bet a b c := betE a b c || betS a b c.
```

3.3 Proof that the Axioms hold in the Model

Now that we have defined the model, we focus on the proof that the axioms of the system from Sec. 2.3 hold in this model. However, we omit the details of the proofs for axioms A1, A2', A3 and A14 since they are rather straightforward. For the same reason, we do not cover the decidability of point equality A0.

Let us start by focusing on axioms A7' and A15 as the proofs that they hold in our model are quite similar. In the case of axiom A15 we know that A—B—D and B—C—D so let $k_1 \in \mathbb{F}$ be such that $0 < k_1 < 1$ and $B - A = k_1(D - A)$ (the degenerate case of this axiom is trivial so we only consider the general case) and $k_2 \in \mathbb{F}$ be such that $0 < k_2 < 1$ and $C - B = k_2(D - B)$. In order to prove that A—B—C we need to find a $k \in \mathbb{F}$ such that $0 < k < 1$ and $B - A = k(C - A)$. By calculation we find that $k = \frac{k_1}{k_1+k_2-k_1k_2}$ and we can verify that $0 < k < 1$. In a similar way, for axiom A7', we know that A—P—C and B—Q—C so let $k_1 \in \mathbb{F}$ be such that $0 < k_1 < 1$ and $P - A = k_1(C - A)$ (the hypotheses imply that $0 < k_1 < 1$ because $A \neq P$ and $P \neq C$) and $k_2 \in \mathbb{F}$ be such that $0 < k_2 < 1$ and $Q - B = k_2(C - B)$. In order to prove that there exists a point X such that P—X—B and Q—X—A we need to find a $k_3 \in \mathbb{F}$ and a $k_4 \in \mathbb{F}$ such that $0 < k_3 < 1$, $0 < k_4 < 1$ and $k_3(B - P) + P = k_4(A - Q) + Q$. By calculation we find that $k_3 = \frac{k_1(1-k_2)}{k_1+k_2-k_1k_2}$ and $k_4 = \frac{k_2(1-k_1)}{k_1+k_2-k_1k_2}$ and we can verify that $0 < k_3 < 1$ and $0 < k_4 < 1$. In both of these proof, the ratios are almost identical to the point that it suffices to prove the following lemma:

```
Lemma ratio_bet a b c k1 k2 k3 :
  0 < k1 -> 0 < k2 -> k1 < 1 -> 0 < k3 -> k3 < k1+k2-k1*k2 ->
  b - a == ((k1+k2-k1*k2)/k3)^-1 *: (c - a) -> bet a b c.
```

It allows to prove quite easily both of these axioms. For axiom A4, we proceeded in a analogous way: it suffices to set the point E that can be constructed using this axiom to $\frac{\|D-C\|}{\|B-A\|}(B - A) + A$ and to verify this point satisfies the desired properties by calculation.

We now turn to axiom A5. We followed Makarios' approach for the proof that this axiom holds in our model [16]. In his proof he used the cosine rule: in a triangle whose vertices are the vectors A, B and C we have

$$\|C-B\|^2 = \|C-A\|^2 + \|B-A\|^2 - 2(B-A)\cdot(C-A).$$

As noted by Makarios, using the cosine rule allows to avoid defining angles and properties about them. Applying the cosine rule for the triangles BCD and $B'C'D'$ allows to prove that $\|D-C\|^2 = \|D'-C'\|^2$ by showing that

$$(C-B)\cdot(D-B) = (C'-B')\cdot(D'-B')$$

which can be justified, by applying the cosine rule again, this time in the triangles ABD and $A'B'D'$, if

$$\|D-A\| - \|D-B\| - \|A-B\| = \|D'-A'\| - \|D'-B'\| - \|A'-B'\|$$

which we know from the hypotheses and if the ratios corresponding to the betweenness A—B—C and A'—B'—C' are equal which can be obtained by calculation.

Next, let us consider axiom A10.[8] From the hypotheses we have two ratios $k_1 \in \mathbb{F}$ and $k_2 \in \mathbb{F}$ such that $0 < k_1 < 1$, $0 < k_2 < 1$, $D-A = k_1(T-A)$ and $D-B = k_2(C-B)$. Using these ratios, it suffices to define X such that $B-A = k_1(X-A)$ and Y such that $C-A = k_1(Y-A)$. So we know by construction that A—B—X and A—C—Y and we easily get that $T-X = k_2(Y-X)$ by calculation, thus proving that X—T—Y. Since A10 and A10' are equivalent when A0, A1, A2', A3-A5, A7', A8, A9', A11', A14 and A15 hold, this allows to prove that A10' holds in our model.

Finally the remaining two axioms are treated in a slightly different setting since they are the dimension axioms. Formally we fix the value of n to 1. In order to simplify the many rewriting steps needed for these proofs we started by establishing the following two lemmas:

```
Definition sqr_L2_norm_2D a b :=
  (b 0 0 - a 0 0) ^+ 2 + (b 0 1 - a 0 1) ^+ 2.

Lemma congP a b c d :
  reflect (sqr_L2_norm_2D a b = sqr_L2_norm_2D c d) (cong a b c d).

Lemma betSP' a b c (r := betR a b c) :
  reflect ([ /\ b 0 0 - a 0 0 = r * (c 0 0 - a 0 0),
                b 0 1 - a 0 1 = r * (c 0 1 - a 0 1), 0 < r & r < 1])
          (betS a b c).
```

The reader familiar with SSReflect will have recognized the `reflect` predicate, described in [9] for example. In practice, these lemmas allowed to spare many steps that would have been repeated in almost every proof concerning the dimension axioms. It was much more straightforward to prove that axiom A8 holds in our model than for axiom A9'. In fact, it is enough to find three non-collinear points. We simply took the points $(0,0)$, $(0,1)$ and $(1,0)$:

```
Definition row2 {R : ringType} (a b : R) : 'rV[R]_2 :=
  \row_p [eta \0 with 0 |-> a, 1 |-> b] p.
```

[8]Similarly to A7, when we were proving Euclid's axiom, we realized that the same kind of distinctions was also needed. The degenerate cases are implied by the other betweenness axioms so it suffices to show that A10 holds when the angle $\angle BAC$ is non-flat and when D is different from T.

```
Definition a : 'rV[R]_(2) := row2 0 0.
Definition b : 'rV[R]_(2) := row2 0 1.
Definition c : 'rV[R]_(2) := row2 1 0.
```

It was then an easy matter to verify that axiom A8 holds in our model. For axiom A9, the idea of the proof that we formalized was to first show that, by letting M be the midpoint of P and Q, the equation $(x_P - x_M)(x_M - x_X) + (y_P - y_M)(y_M - y_X) = 0$, capturing the property that the points P, M, and X form a right angle with the right angle at vertex M, was verified when X would be equal to A, B or C:

```
Lemma cong_perp (a p q : 'rV[R]_(2)) (m := (1 / (1 + 1)) *: (p + q)) :
  cong a p a q ->
  (p 0 0 - m 0 0) * (m 0 0 - a 0 0) +
  (p 0 1 - m 0 1) * (m 0 1 - a 0 1) = 0.
```

Next, we demonstrated that for three points A, B and C verifying $(x_A - x_B)(y_B - y_C) - (y_A - y_B)(x_B - x_C) = 0$ are collinear in the sense that A–B–$C \vee B$–C–$A \vee C$–A–B:

```
Lemma col_2D a b c :
  (a 0 0 - b 0 0) * (b 0 1 - c 0 1) ==
  (a 0 1 - b 0 1) * (b 0 0 - c 0 0) ->
  (bet a b c \/ bet b c a \/ bet c a b).
```

Using the equations implied by `cong_perp` we could derive that

$$(x_P - x_M)(y_M - y_P)\left((x_A - x_B)(y_B - y_C) - (y_A - y_B)(x_B - x_C)\right) = 0.$$

We were then left with three cases: either the abscissas of P and M are equal in which case the ordinate of A, B and C were equal thus sufficing to complete the proof, or the ordinates of P and M are equal in which case the abscissas of A, B and C were equal thus completing the proof, or $(x_A - x_B)(y_B - y_C) - (y_A - y_B)(x_B - x_C) = 0$ corresponding to the lemma that we had proved and again allowing to conclude.

Putting everything together, we could prove that Cartesian planes over a Pythagorean ordered field form a model of the variant of Tarski's system of geometry, thus proving the satisfiability of the theory.[9]

```
Global Instance Rcf_to_T2D : Tarski_2D Rcf_to_T_PED.

Global Instance Rcf_to_T_euclidean : Tarski_euclidean Rcf_to_T_PED.
```

4 An Example of Independence Proof

To illustrate how we obtain formal proofs of independence we present an example. We start by defining the counter-model we will use to prove the independence of axiom A10'. We then provide the sketch of the formal proof.

[9] `Tarski_euclidean` is the type class that captures the theory consisting of axioms A0-A10.

4.1 Klein's Model

To prove Euclid's Parallel Postulate independent from the other axiom we work in Klein's model as defined in SST [19]:

```
Variable R : realFieldType.
Variable n : nat.

Definition Vector := 'rV[R]_(n.+1).
Definition Point : Type := {p : Vector | (p *m p^T) 0 0 < 1}.
Notation "#" := proj1_sig.

Implicit Types (a b c d : Point).
Implicit Types (v w x y : Vector).

Definition bet' a b c := bet (#a) (#b) (#c).

Definition omd_v v w := (1 - (v *m (w)^T) 0 0).
Definition cong_v v w x y :=
  (omd_v v w)^+2/(omd_v v v * omd_v w w) ==
  (omd_v x y)^+2/(omd_v x x * omd_v y y).
Definition cong' a b c d := cong_v (#a) (#b) (#c) (#d).
```

Here, `Point` is the type of `Vector`, vectors of dimension $n+1$ with coefficient in the *real field* structure, lying inside the unit disk and # the projection allowing to recover the coordinate part of this dependent type. In Klein's model, b is said to be between a and c iff their coordinate parts can be said to be `bet` in the model from Sec. 3 and line-segments $\overline{ab}$ and $\overline{cd}$ are said to be congruent iff

$$\frac{(1-\#a\cdot\#b)^2}{(1-\#a\cdot\#a)(1-\#b\cdot\#b)} = \frac{(1-\#c\cdot\#d)^2}{(1-\#c\cdot\#c)(1-\#d\cdot\#d)}$$

where $\cdot$ denotes the dot product of two vectors.

4.2 Independence of Euclid's Parallel Postulate via Klein's Model

Here we only detail the proof that A10' does not hold in this model. Mechanizing the following proof sketch allows to derive.

```
Lemma euclid : ~ euclidP (@Point R 1) (@bet' R 1).
```

To make sure that we did not introduce any change in the axioms between the various models we relied on predicates such as `euclidP`, which depend on possibly the type for points and the predicate(s) for betweenness and/or congruence.

Theorem 1. *Axiom A10' does not hold in Klein's model.*

Proof. Since Klein's model forms a model of neutral geometry,[10] it suffices to prove that any version of the parallel postulate, proven equivalent to A10' in Coq when assuming A0-A9, does not hold. We

[10]Neutral geometry is defined by the set of axioms of Euclidean geometry from which the parallel postulate has been removed.

choose to work with A10. Picking a, b, c, d and t to be of coordinates $(0,0)$, $(0,\frac{1}{2})$, $(\frac{1}{2},0)$, $(\frac{1}{4},\frac{1}{4})$ and $(\frac{1}{2},\frac{1}{2})$, some computations allow to verify that $\#a$—$\#d$—$\#t$, $\#b$—$\#d$—$\#c$ $b \neq d$, $d \neq c$ and $\neg\mathrm{Col}\ \#a\#b\#c$. So, to prove that this version does not hold, it is enough to show that for any x and y such that x lies inside the unit disk, $\#a$—$\#b$—$\#x$, $\#a$—$\#c$—$\#y$ and $\#x$—$\#t$—$\#y$, it holds that y is not a `Point`, meaning that it lies outside the unit disk. Let us first eliminate the case where $b = x$ as it would lead to a contradiction. Here, we use the algebraic characterization of collinearity[11] to obtain that, if $b = x$, the ordinate of x would need to be equal to both 0 and $\frac{1}{2}$ which is impossible. Now let us pose b' to be the vector $x+a-b$. It is an easy matter to check that $\#a$—$\#b'$—$\#x$ so let us pose k_1 to be the ratio associated to this betweenness. We can verify that $k_1 \leq \frac{1}{2}$ since x is supposed to belong to the unit disk. We can then take d' at ratio k_1 from a to t. Applying what was proven to show that A10' holds in Cartesian planes over a Pythagorean ordered field, we can show that y' at ratio $\frac{1}{k_1}$ from a to c is such that $\#a$—$\#c$—$\#y'$ and $\#x$—$\#t$—$\#y'$. If we can prove that $y = y'$ we will be done as y' lies outside of the unit disk because $k_1 \leq \frac{1}{2}$ so $2 \leq \frac{1}{k_1}$. Finally, to prove that $y = y'$ we can reason by uniqueness of the intersection of lines which is valid in neutral geometry. □

5 Conclusion

We defined ten out of the eleven counter-models present in Gupta's thesis [11], thus obtaining the Coq formal proof of the independence of ten out of the thirteen axioms of the system presented in Sec. 2.3. This seems to indicate that Pasch's axiom could indeed be split into two meaningfully different parts as done in this paper while still having an independent system. However, we will only be sure of this once we will have formalized the missing three counter-models. These can be found in Gupta's thesis [11], Szczerba's paper [21], and Beeson's section *The recursive model* in [2].

Five of the formalized models are finite and the other five are modifications of the model presented in Sec. 3. We highlight that, for the latter five, A11' is not verified for the same reason as for the model from Sec. 3.. All these models are available in the *GeoCoq* library[12] and represent about 4k lines of formal proof.

We are currently extending this work by proving the independence of a more constructive version[13] of the axioms which would also allow to capture n-dimensional geometry. For this extension we could not rely on $\mathrm{A9}^{(n)}$ from [22]. Indeed, we found that it can only be assumed as an upper n-dimensional axiom when $n = 2$ or 3. $\mathrm{A9}^{(n)}$ is stated as follows.

$$\bigwedge_{1\leq i\leq j\leq n} P_i \neq P_j \wedge \bigwedge_{i=2}^{n} AP_1 \equiv AP_i \wedge \bigwedge_{i=2}^{n} BP_1 \equiv BP_i \wedge \bigwedge_{i=2}^{n} CP_1 \equiv CP_i \Rightarrow \mathrm{Col}\,ABC$$

By taking $P_i = (\cos\frac{2i\pi}{n}, \sin\frac{2i\pi}{n}, 0, \ldots, 0)$ for $1 \leq i \leq n$ then $(0,0,x_3,x_4,\ldots,x_n)$ satisfies the premises for any $x_3,x_4,\ldots,x_n$ in the standard n-dimensional model while triplets of points of this form are not necessarily collinear. The various modifications did not allow to reuse some of the counter-models already mechanized, so new ones are necessary.

We are convinced that using a proof-assistant is crucial when proving the independence of a system, where small changes in a statement are critical. Actually, there was a typo in Gupta's counter-model for

[11]Here we use the converse of `col_2D`.

[12]`http://geocoq.github.io/GeoCoq/`

[13]We replace point equality decidability by point equality "stability", namely $\forall XY, \neg\neg X = Y \Rightarrow X = Y$, which allows to prove equality of points by contradiction but does not allow case distinctions. We do not go as far as in [3] where not even "stability" is assumed. We also apply the same modifications made to obtain what is called "continuous Tarski geometry" in [1].

A2 and we just exhibited a problem with axiom A9$^{(n)}$ from [22]. The *GeoCoq* library also proved very useful as it allowed us to combine the algebraic and geometric[14] reasoning.

Acknowledgments: We would like to thank Marius Hinge for his contribution to the early stage of this work.

References

[1] Beeson, M.: A Constructive Version of Tarski's Geometry. Annals of Pure and Applied Logic **166**(11), 1199–1273 (2015). doi:10.1016/j.apal.2015.07.006

[2] Beeson, M.: Constructive Geometry and the Parallel Postulate. Bulletin of Symbolic Logic **22**(1), 1–104 (2016). doi:10.1017/bsl.2015.41

[3] Beeson, M.: Brouwer and Euclid. Indagationes Mathematicae **29**(1), 483–533 (2018). doi:10.1016/j.indag.2017.06.002

[4] Beeson, M., Boutry, P., Narboux, J.: Herbrand's theorem and non-Euclidean geometry. The Bulletin of Symbolic Logic **21**(2), 111–122 (2015). doi:10.1017/bsl.2015.6

[5] Bolyai, J.: Appendix, Scientiam Spatii absolute veram exhibens: a veritate aut falsitate Axiomatis XI. Euclidei (a priori haud unquam decidenda) independentem; adjecta ad casum falsitatis, quadratura circuli geometrica. Auctore Johanne Bolyai de eadem, Geometrarum in Exercitu Caesareo Regio Austriaco Castrensium Capitaneo. Coll. Ref. (1832)

[6] Boutry, P., Braun, G., Narboux, J.: Formalization of the Arithmetization of Euclidean Plane Geometry and Applications. Journal of Symbolic Computation (2018). doi:10.1016/j.jsc.2018.04.007

[7] Boutry, P., Gries, C., Narboux, J., Schreck, P.: Parallel Postulates and Continuity Axioms: A Mechanized Study in Intuitionistic Logic Using Coq. Journal of Automated Reasoning (2017), doi:10.1007/s10817-017-9422-8

[8] Braun, G., Boutry, P., Narboux, J.: From Hilbert to Tarski. In: Narboux, J., Schreck, P., Streinu, I. (eds.) Proceedings of the Eleventh International Workshop on Automated Deduction in Geometry. pp. 78–96 (2016), `https://hal.inria.fr/hal-01332044`

[9] Cohen, C.: Formalized algebraic numbers: construction and first-order theory. Theses, Ecole Polytechnique X (Nov 2012), `https://pastel.archives-ouvertes.fr/pastel-00780446`

[10] Cohen, C., Sakaguchi, K., Tassi, E.: Hierarchy Builder: Algebraic hierarchies Made Easy in Coq with Elpi (System Description). In: 5th International Conference on Formal Structures for Computation and Deduction (FSCD 2020). vol. 167, pp. 34:1–34:21. Paris, France (Jun 2020). doi:10.4230/LIPIcs.FSCD.2020.34

[11] Gupta, H.N.: Contributions to the Axiomatic Foundations of Geometry. Ph.D. thesis, University of California, Berkeley (1965)

[12] Hilbert, D.: Les fondements de la géométrie. Dunod, Jacques Gabay edn. (1971), edition critique avec introduction et compléments préparée par Paul Rossier

[13] Klugel, G.S.: Conatuum praecipuorum theoriam parallelarum demonstrandi recensio. Ph.D. thesis, Schultz, Göttingen (1763), German translation available: `http://www2.math.uni-wuppertal.de/~volkert/versuch.html`

[14] Lobatschewsky, N.: Geometrische Untersuchungen zur Theorie der Parallellinien, pp. 159–223. Springer Vienna, Vienna (1985). doi:10.1007/978-3-7091-9511-6_4

[15] Mahboubi, A., Tassi, E.: Mathematical Components. Zenodo (Sep 2022), doi:10.5281/zenodo.7118596

[16] Makarios, T.J.M.: A Mechanical Verification of the Independence of Tarski's Euclidean Axiom. Master's thesis, Victoria University of Wellington (2012)

[14]Such as the use of the uniqueness of the intersection in our proof of Theorem 1.

[17] Makarios, T.J.M.: A further simplification of Tarski's axioms of geometry. arXiv: Logic (2013). doi:10.1285/i15900932v33n2p123.

[18] Narboux, J., Janicic, P., Fleuriot, J.: Computer-assisted Theorem Proving in Synthetic Geometry. In: Sitharam, M., John, A.S., Sidman, J. (eds.) Handbook of Geometric Constraint Systems Principles. Discrete Mathematics and Its Applications, Chapman and Hall/CRC (2018). doi:10.1201/9781315121116-2, `https://inria.hal.science/hal-01779452`

[19] Schwabhäuser, W., Szmielew, W., Tarski, A.: Metamathematische Methoden in der Geometrie. Springer-Verlag (1983)

[20] Simić, D., Marić, F., Boutry, P.: Formalization of the Poincaré Disc Model of Hyperbolic Geometry. Journal of Automated Reasoning **65**(1), 31–73 (Apr 2020). doi:10.1007/s10817-020-09551-2

[21] Szczerba, L.W.: Independence of Pasch's axiom. Bull. Acad. Polon. Sci. Sér. Sci. Math. Astronom. Phys. **18**, 491–498 (1970)

[22] Tarski, A., Givant, S.: Tarski's System of Geometry. The Bulletin of Symbolic Logic **5**(2), 175–214 (1999). doi:10.2307/421089

[23] Team, T.C.D.: The Coq Proof Assistant, version 8.15 (2022), doi:10.5281/zenodo.5846982

Considerations on Approaches and Metrics in Automated Theorem Generation/Finding in Geometry

Pedro Quaresma[*]
CISUC / Department of Mathematics,
University of Coimbra, Portugal
pedro@mat.uc.pt

Pierluigi Graziani[†]
Department of Pure and Applied Sciences, University of Urbino, Italy
pierluigi.graziani@uniurb.it

Stefano M. Nicoletti[‡]
Formal Methods and Tools, University of Twente, Enschede, the Netherlands
s.m.nicoletti@utwente.nl

The pursue of what are properties that can be identified to permit an automated reasoning program to generate and find new and interesting theorems is an interesting research goal (pun intended). The automatic discovery of new theorems is a goal in itself, and it has been addressed in specific areas, with different methods. The separation of the "weeds", uninteresting, trivial facts, from the "wheat", new and interesting facts, is much harder, but is also being addressed by different authors using different approaches. In this paper we will focus on geometry. We present and discuss different approaches for the automatic discovery of geometric theorems (and properties), and different metrics to find the interesting theorems among all those that were generated. After this description we will introduce the first result of this article: an undecidability result proving that having an algorithmic procedure that decides for every possible Turing Machine that produces theorems, whether it is able to produce also interesting theorems, is an undecidable problem. Consequently, we will argue that judging whether a theorem prover is able to produce interesting theorems remains a non deterministic task, at best a task to be addressed by program based in an algorithm guided by heuristics criteria. Therefore, as a human, to satisfy this task two things are necessary: an expert survey that sheds light on what a theorem prover/finder of interesting geometric theorems is, and—to enable this analysis—other surveys that clarify metrics and approaches related to the interestingness of geometric theorems. In the conclusion of this article we will introduce the structure of two of these surveys —the second result of this article— and we will discuss some future work.

1 Introduction

In *Automated Reasoning: 33 Basic Research Problems*, Larry Wos, wrote about the problems that computer programs that reason face. Problem 31 is still open and object of active research [56, 57]:

> *Wos' Problem 31*—What properties can be identified to permit an automated reasoning program to find new and interesting theorems, as opposed to proving conjectured theorems?

Two problems in a single sentence: *new* and *interesting* theorems. The automatic discovery of new theorems is a goal in itself, it has been addressed in specific areas, with different methods. The separation

[*]Partially supported by FCT – Foundation for Science and Technology, I.P., within the scope of the project CISUC – UID/CEC/00326/2020 and by European Social Fund, through the Regional Operational Program Centro 2020.

[†]Partially supported by Italian Ministry of Education, University and Research through the PRIN 2017 project "The Manifest Image and the Scientific Image" prot. 2017ZNWW7F_004.

[‡]Funded by ERC Consolidator Grant 864075 (*CAESAR*).

of the “weeds”, uninteresting, trivial facts, from the “wheat”, new and interesting facts, is much harder, but is being addressed also, by different authors using different approaches.

Paraphrasing, again, Wos, “since a reasoning program can be instructed to draw some (possible large) set of conclusions” what should be the “criteria that permit the program to select from those the ones (if any) that correspond to interesting results.”

Different fields have come across the finding of new and interesting theorems’ questions.

Regarding the novelty side: there are different views of approaching new mathematical results. One of those approaches is the systematic exploration of a given broad area of mathematical knowledge, generating, by different means, new theorems and expecting to find interesting ones among those generated (that will be analysed in section 4) [14, 15, 19, 25, 30, 35, 36, 46]. Another approach is given by the pursue of mathematical discovery in specific areas, e.g. *Computing Locus Equations* [1, 9], *Automated Discovery of Angle Theorems* [54], *Automated Discovery of Geometric Theorems Based on Vector Equations* [42], *Automated Generation of Geometric Theorems from Images of Diagrams* [13], *Automatic Discovery of Theorems in Elementary Geometry* [48]. These approaches do not aim to address the problem of automated theorem finding in itself but, for example, to find complementary hypotheses for a given geometric statements to become true [48] i.e. automatic discovery for specific areas.[1]

Regarding the interestingness side we are aware that relevant literature can be found in different areas. For example in automated theorem proving [19, 25, 30, 46] and in sociological studies on the concept of proving [20, 39, 40], in cognitive and educational science studies on the concept of proving [2, 11, 21, 33, 45, 52] and in semiotics and epistemology of mathematics [3, 4, 5, 8, 12].

Despite the cited studies, the Wos’ problem is still on the table. On the contrary, a new result of undecidability can be added to the problem, i.e. having an algorithmic procedure that decides for every possible Turing Machine that produces theorems, whether it is able to produce also interesting theorems, is an undecidable problem. Consequently, we can argue that judging whether a theorem prover is able to produce interesting theorems remains a non deterministic task, at best a task to be addressed by program based in an algorithm guided by heuristics criteria. Therefore, as a human, to satisfy this task we need expert survey that sheds light on what a theorem prover/finder of interesting geometric theorems is, and—to enable this analysis—other surveys that clarify metrics and approaches related to the interestingness of geometric theorems.

Structure of the paper. In section 2 the issue of Automated Theorem Generation (ATG) is discussed. In section 3 we discuss the deductive approach in ATG. In section 4 the issue of Automated Theorem Finding (ATF) is analysed. In section 5 we present an undecidability result concerning the problem of finding interesting theorems and its conceptual consequences. In section 6 we will introduce the structure of two surveys to empirically explore the interestingness of theorems in geometry and its potential application in theorem proving/finding (a third survey). Finally, we will discuss some future work.

2 Automated Theorem Generation

Automated theorem generation, independently of being interesting, or not, can be addressed in several ways [46].

[1]We left aside the notion of discovery in education, given that, in that area, the goal is the student’s discovery of “new” (for them) theorems, giving the student the possibility of freely making conjectures and having an interactive/automatic deduction support in the exploration of those “new” theorems [10, 32, 37, 38].

The Inductive approach, is a natural approach. Conclusions are drawn by going from the specific to the general. Exploring a given domain, seeking for properties that emerge from a set of particular cases and making a conjecture about the general case.

Dynamic Geometry Software (DGS) can be seen as software environments to inductively explore new knowledge. Making a geometric construction, constrained by a given set of geometric properties, and then moving the free point around will show all the fix-points, conjecturing if those new fixed relations between objects are true in all cases, or not. For example the Pappus' Theorem, in this case, a well-known theorem: are the intersection points (see Figure 1) G, H and I, collinear? By moving, in the DGS, the free-points it seems that they are, it remains to prove it.

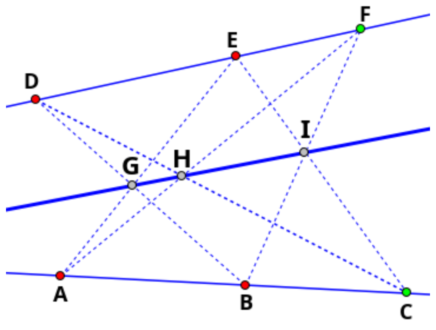


Figure 1: Pappus' Theorem

The inductive approach has the advantage of being stimulated by observations in the domain. but has the disadvantage that induction is unsound. A famous example of such unsound inductive approach can be seen in the Euclid Parallel lines Postulate, that nevertheless was very fruitful, giving raise to different geometries.

The Generative approach, i.e. the generation of conjectures, testing them for theorem-hood. The simplest form of generation is syntactic, in which conjectures are created by mechanical manipulation of symbols, e.g. [44]. The *MCS* program generates conjectures syntactically and filters them against models of the domain [59]. A stronger semantically based approach is taken by the *HR* program, which generates conjectures based on examples of concepts in the domain [18]. A theory exploration system called *QuickSpec*, works by interleaving term generation with random testing to form candidate conjectures [34]. In [34] the conjecture generation approaches are classified into three categories: heuristic rule-based systems, term generation-and-testing and neural network-based systems. The *RoughSpec* system adds to *QuickSpec* the notion of *shapes* of theorems, specifying the shapes of conjectures the user is interested in, and thus limiting the search [22].

Like induction, generation is unsound. However, if the rules by which the generation is performed are sufficiently conservative then this approach may generate a higher fraction of theorems than the inductive approach.

The Manipulative Approach, conjectures are generated from existing theorems. An existing theorem is modified by operations such as generalisation, specialisation, combination, etc. This approach is used in abstraction mapping, which converts a theorem to a simpler theorem, and uses a solution to the simpler theorem to help find a solution to the original theorem [43]. Manipulation of ATP theorems has also been used to produce new theorems for testing the robustness of ATP systems' performances [55].

An advantage of the manipulative approach is that, if the manipulations are satisfiability preserving, then theorems, rather than conjectures, are produced from existing theorems. However, the conjectures produced by the manipulative approach are typically artificial in nature, and thus uninteresting.

The Deductive Approach, consequences are generated by application of sound inference rules to the axioms and previously generated logical consequences. This can be done by an appropriately configured saturation-based ATP system.

The advantage of this approach is that only logical consequences are ever generated. The challenge of this approach is to avoid the many uninteresting logical consequences that can be generated.

3 The Deductive Approach

Some systems addresses, explicitly, the generation of new geometric results using different approaches. In the following some of these approaches are described.

3.1 Strong Relevant Logic-based Forward Deduction Approach

In [27] the authors argue for the fundamental difference between the Automated Theorem Proving (ATP) and the Automated Theorem Finding (ATF). ATP is the process of finding a justification for an explicitly specified statement from given premises which are already known facts or previously assumed hypotheses. ATF is the process to find out or bring to light that which was previously unknown. Where ATP is all about known (old) facts, ATF is about previously unknown conclusions from given premises. Jingde Cheng [15] claims that classical mathematical logic, its various classical conservative extensions, and traditional (weak) relevant logics cannot satisfactorily underlie epistemic processes in scientific discovery, presenting an approach based on strong relevant logic. Hongbiao Gao et al. have followed this approach applying it for several domains such as NBG set theory, Tarski's Geometry and Peano's Arithmetic [26, 27, 29, 30]

3.2 Rule Based Systems

The rule-based automated deduction system are often used when the proof itself is an object of interest (and not only the end result), given that the proofs are developed from the hypothesis and sets of axioms, to the conclusion by application of the inference rules, the proofs are "readable".

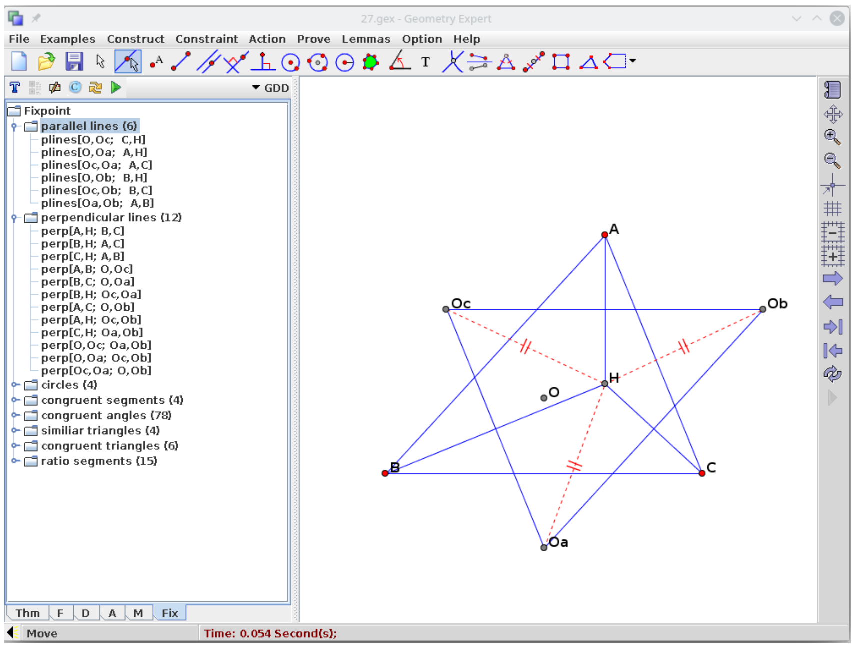
Example of such approaches can be seen in systems like *QED-Tutrix* [23, 24] and *JGEx* [58], both for geometry. In the tutorial system *QED-Tutrix*, the rule based automated theorem prover goal is to find the many possible branches of the proof tree, in order to be able to help the student approaching the proof of a geometric conjecture. In the *JGEx* system we can have the proof in a "readable" and "visual" renderings and also the set of all properties that can be deduced from the construction.

One of the ATP built-in in *JGEx* is an implementation of the geometry deductive database method [16, 58]. Using a breadth-first forward chaining a fix-point for the conjecture at hand is reached. For that geometric construction and the rules of the method, the fix-point gives us all the properties that can be deduced, some already known facts, but also new facts (not necessary interesting ones).

The geometry deductive database method proceeds by using a simple algorithm where, starting from the geometric construction D_0, the rules, R, are applied over and over till a fix-point, D_k is reached:

$$\boxed{D_0} \overset{R}{\subset} \boxed{D_1} \overset{R}{\subset} \cdots \overset{R}{\subset} \boxed{D_k} \quad \text{(fix-point)} \tag{1}$$

In figure 2 an example, using *JGEx*, is shown. On the right, the geometric construction, on the left, the fix-point, with all the facts that were found for that construction.



```
Fixpoint
  parallel lines (6)
    plines[O,Oc;  C,H]
    plines[O,Oa;  A,H]
    plines[Oc,Oa;  A,C]
    plines[O,Ob;  B,H]
    plines[Oc,Ob;  B,C]
    plines[Oa,Ob;  A,B]
  perpendicular lines (12)
    perp[A,H; B,C]
    perp[B,H; A,C]
    perp[C,H; A,B]
    perp[A,B; O,Oc]
    perp[B,C; O,Oa]
    perp[B,H; Oc,Oa]
    perp[A,C; O,Ob]
    perp[A,H; Oc,Ob]
    perp[C,H; Oa,Ob]
    perp[O,Oc; Oa,Ob]
    perp[O,Oa; Oc,Ob]
    perp[Oc,Oa; O,Ob]
  circles (4)
  congruent segments (4)
  congruent angles (78)
  similiar triangles (4)
  congruent triangles (6)
  ratio segments (15)

Time: 0.054 Second(s);
```

Figure 2: Fix-point in *JGEx*

A new open source implementation of this method, *OGP-GDDM*,[2] is described in [7]. It will be integrated in the *Open Geometry Prover Community Project* (OGPCP) [6]. One of the medium-term goals of the *OGP-GDDM* project, is to develop a meta-prover, a program capable to receive different sets of rules and synthesise a specific ATP for those rules.

3.3 Algebraic Approaches

A similar approach is taken in the well-known dynamic geometry system *GeoGebra*.[3] The *GeoGebra Discovery* version[4] has the capability to find, from a user defined geometric construction, properties about that construction. *GeoGebra Discovery* reports some facts that were systematically checked from a list of possible features including identical points, parallel or perpendicular lines, equal long segments, collinearity or concyclicity. This is not a deductive method so the generation process must be externally verified, *GeoGebra Discovery* do that by recurring to a built-in algebraic automated theorem prover based in the Gröbner bases method [35, 36].

[2] `https://github.com/opengeometryprover/OpenGeometryProver`
[3] `https://www.geogebra.org/`
[4] `https://github.com/kovzol/geogebra-discovery`

4 Automated Theorem Finding

Apart from our research goal of finding the interesting geometric theorems among all those that were automatically generated, the pursue of measures of interestingness has applicability in the interactive and automated theorem proving area. In that area a common use of interestingness is to improve the efficiency of the programs, tailoring the search space, making the search depth limited and guaranteeing that only comprehensible concepts are produced [19].

A goal, pursued with different approaches by many researchers, is the creation of strong AI methods capable of complex research-level proofs, mathematical discovery, and automated formalisation of today's vast body of mathematics [47]. The *MATHsAiD* (Mechanically Ascertaining Theorems from Hypotheses, Axioms and Definitions) project aimed to build a tool for automated theorem-discovery, from a set of user-supplied axioms and definitions. In the words of its authors, *MATHsAiD 2.0* can conjecture and prove interesting Theorems in high-level theories, including Theorems of current mathematical significance, without generating an unacceptable number of uninteresting theorems [41]. The *TacticToe* system, combines reinforcement-learning with Monte-Carlo proof search on the level of *HOL4* tactics [31]. The *ENIGMA-NG* system uses efficient neural and gradient-boosted inference guidance for the ATP *E*, improving its efficiency [17]. This two systems, one for interactive provers and the other to automatic provers, are examples of systems that uses discovery and filtering for improving the efficiency of automated deduction systems.

4.1 The Deductive Approach Algorithm

The different approaches found in the literature [18, 27, 46] share, in their general lines, the same algorithm: for a given logical fragment, select a initial set of facts and then a cycle of generation/filtering is applied until some stopping condition is matched (see Fig. 4.1).

4.2 Filtering Interesting Theorems

A first level of filtering (run-time filter) should discard the obvious tautologies and also conjectures proved false by empirical evidence.

The filtering for interesting theorems or for uninteresting conjectures, two sides of the same coin, is done by application of a series of filters. These filters are still to be validated, being of speculative nature [19, 29, 30, 46].

Obviousness: the number of inferences in its derivation. Obviousness estimates the difficulty of proving a formula, it can be given by the number of inferences in its derivation.

Weight: the effort required to read a formula. The weight score of a formula is the number of symbols it contains.

Complexity: the effort required to understand a formula, the number of distinct function and predicate symbols it contains.

Surprisingness: measures new relationships between concepts and properties.

Intensity: measures how much a formula summarises information from the leaf ancestors in its derivation tree.

Adaptivity: measures how tightly the universally quantified variables of a formula are constrained (for formulae in clause normal form).

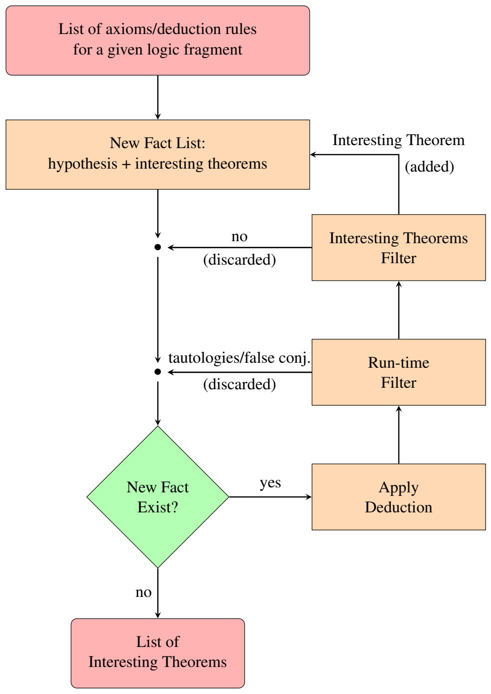

List of axioms/deduction rules for a given logic fragment
New Fact List: hypothesis + interesting theorems
Interesting Theorem
(added)
no
(discarded)
Interesting Theorems Filter
tautologies/false conj.
(discarded)
Run-time Filter
New Fact Exist?
yes
Apply Deduction
no
List of Interesting Theorems

Figure 3: New and Interesting Theorems Algorithm

Focus: measures the extent to which a formula is making a positive or negative statement about the domain of application.

Usefulness: measures how much an interesting theorem has contributed to proofs of further interesting theorems.

In spite of the relevance of these metrics, it would be appropriate to construct an expert survey with which we could validate them by referring to a significant public of experts. We believe this kind of survey would be relevant not only to face Wos' problem, but also to better understand how to construct and evaluate software that generates/finds interesting theorems. Despite having only relevant metrics and approaches regarding Wos' problem, while not yet having formal results, we can prove a relevant result that concerns the second issue, i.e., the question regarding *Interesting Turing Machines*, i.e., programs capable of generating interesting new geometric results.

5 Undecidability Result

In section 4.2 the application of filters was discussed, these filters are based on some measures of interestingness that are still to be validated and that are applied in an heuristic way. Is it possible to have a deterministic approach, i.e., is it possible to write a computer program that in a deterministic way, find interesting theorems? We show, as an application of the Rice's theorem [49, 50, 51] (see Lemma 1), that it is undecidable to determine, for a given Turing Machine, whether the language recognised by it has the (non-trivial) property of finding interesting theorems.

Definition 1 (Non-Trivial Property)**.** *A property p of a formal language is* non-trivial *if:*

- *there exist a recursively enumerable language having the property p;*
- *there exist a recursively enumerable language not having the property p.*

Lemma 1 (Rice's Theorem)**.** *Let p be any non-trivial property of the language of a Turing machine. The problem of determining whether a given Turing machine's language has property p is undecidable.*

Theorem 1 (Undecidability Result)**.** *For any given Turing Machine, it is undecidable to determine, whether the language recognised by it has the property of finding interesting theorems.*

Proof. All programs (Turing machines) capable of automated theorem proving and by extension generating/finding geometric theorems rely on a formal language to describe the geometric constructions, conjectures and proofs. For example we can consider the (full) *First-Order Form* (FOF)[5] of *TPTP* [53] and the formal axiomatic theories for geometry based on that language.[6]

Let p be the property of that language that says that theorem t is interesting, for any conceivable definition of interestingness, then there exist a recursively enumerable language having the property p. It will be enough to restrict the language in such a way that the theorem t, and only this, would be recognised. But, it also exist a recursively enumerable language not having the property p. It would be enough to restrict that language in such a way that only tautologies would be recognised. Tautologies are, for any conceivable definition of interestingness, uninteresting. We have proved that p, the property that can establish if a given theorem is interesting, is a non-trivial property.

Having establish that the property p is non-trivial, then, by application of Rice's theorem, it is undecidable to determine for any given Turing machine M, whether the language recognised by M has the property p. $\square$

In other words, it is undecidable to have a deterministic program that can find interesting problems. At best this is a task to be addressed by programs based on algorithms guided by heuristics criteria.

6 Designing Interesting Surveys

In light of our undecidability result, to understand what experts mean by, "a program that is able to also prove interesting theorems", must be done referring to empirical data, via the formulation of an expert survey. However, for it to be fulfilled, one has to first reach a minimal degree of agreement on the definition of interestingness of theorems. How could one speak about programs that produce such theorems? In order to achieve this agreement, an empirical exploration of the notion of interestingness and of what it concretely entails is paramount. This exploration requires to situate the notion of interestingness historically and socio-culturally, considering logical, epistemological, sociological, cognitive, semiotic and pedagogical aspects of the issue. Probably—and as Wos already implies—interestingness entails different tangible properties, which differ in given centuries, geographical locations and societies. Moreover, in some cases we say that a theorem is interesting for what we can call *global reasons* e.g., Euclid's theorem on the infinitude of the set of prime numbers, Zorn's lemma and Gödel's Theorems are interesting due to their role in mathematics, logic and computer science. Other times for *local reasons* e.g., in relation to what we are teaching our students at that moment. In order to assess which tangible properties—both global and local—interestingness entails today, we are proposing to conduct two *expert surveys* with two statistically significant pools of participants.

[5]`http://tptp.cs.miami.edu/TPTP/QuickGuide/`

[6]TPTP Axioms Files for geometry, `https://www.tptp.org/cgi-bin/SeeTPTP?Category=Axioms`, e.g. *Tarski geometry axiom*, `GEO001` and `GEO002`, *Deductive Databases Method in Geometry*, `GEO012`.

Influencing factors. Gao et al. performed an extensive analysis of areas like Set Theory, Peano's arithmetic and Tarski's Geometry, looking for the relevance of structural factors, such as the degree of logical connectives in the theorem, the propositional schema of the formula formalising the theorem, the abstract level of predicates and functions in the theorem and the deduction distance of a theorem [19, 28, 30, 46]. Some of these structural aspects might be related to our cognitive dynamics. But also the epistemological role of a theorem with respect to other theorems might be a relevant feature; or the educational role that some theorems have with respect to some notions might influence their interestingness. Finally, the history of a theorem—e.g. Fermat's last theorem—could add points to its interestingness, which, in the case of Fermat's last theorem, might be already caused by the technicalities of the proof itself.

Designing the surveys. Taking all these factors into consideration, we would propose to design three surveys that question experts from different fields.

Before describing the surveys below some clarifications are necessary. We will use the term "expert" to mean mathematics teachers at primary, middle, and high schools, and professors or researchers in pure and applied mathematics at universities or at research centres. Furthermore, we will focus on the case study of geometry, hence interesting theorems in geometry. The reasons for this restriction to geometry are as follows: on the one hand, considering all fields of research in mathematics might require a too large number of experts and could produce too many divergent ideas. On the other hand, having in mind an application of the results in automatic theorem proving as a target, it seems appropriate to move into an area were there are many different methods and many automated provers implementing those methods. Finally, geometry is a kind of language common to many areas of mathematics and has been a domain for reflection since the early years of mathematics teaching.

Finally, these surveys are intended to involve mathematics teachers, but their outcome does not target mathematics education. Of course, this is a possible target, but it is not the primary goal of these starting surveys.

6.1 Three Surveys

In the first survey, we will ask the experts both to indicate some situations in which they remember to have used the adjective interesting concerning a theorem, and to explain the use of this expression. In addition, we will ask experts to list several geometric theorems they find interesting, and to list several geometric theorems they find not interesting, both from elementary and higher geometry, explaining the reasons for their answers (see Appendix A). This first survey is already under way, the steering committee is already approaching it and the authors of accepted papers in the conference, *14th International Conference on Automated Deduction in Geometry* (ADG 2023)[7], were invited to participate. We are planning to enlarge it to our network of contacts and we invite the interested reader to also participate, answering it.[8] We are planning to begin collect and analyse the answers in February, 2024.

We will use the information from this survey to define a list of characteristics (A,B,C, ...) of a theorem that offer sufficient reasons to attribute interestingness to it. We will assign weights to the various characteristics by considering the answers to this first survey.

After the first survey, we will implement a second one. This second survey will consider a list of theorems that, in different percentages, have the characteristics inferred from the first survey. We will

[7]ADG 2023, 14th International Conference on Automated Deduction in Geometry, Belgrade, Serbia, September 20-22, 2023.

[8]`https://docs.google.com/forms/d/e/1FAIpQLScIXZbLPBHTLvmQ28P30Cm_-lkOrM7e6rab7hoOWrAFwf_mbQ/viewform?usp=sf_link`

submit the second survey to a set of experts different from those used in the first survey. We will ask these experts whether they find the theorems listed interesting or not. We will ask them to rate, using a Likert scale,[9] the degree of impact that having certain characteristics plays in their attribution of interestingness (see Appendix B).

This second group will allow us to understand whether the characteristics isolated through the first survey are sufficient conditions to affirm that a theorem is interesting.

With an agreement on what an interesting theorem is, based on empirical research, we could query experts in theorem generators/finders design, with another survey (the third survey) asking how to design software able to produce these interesting theorems.

After that, we will focus our empirical inquiry on programs, driven by heuristics based on our findings, able to find interesting theorems.

We have established a *steering committee* to design the surveys and who will oversee the submission of the surveys to experts around the world.

The steering committee consists of the following scholars:

- Thierry Dana-Picard, Jerusalem College of Technology, Jerusalem, Israel;
- James Davenport, University of Bath, United Kingdom;
- Pierluigi Graziani, University of Urbino, Urbino, Italy;
- Pedro Quaresma, University of Coimbra, Coimbra, Portugal;
- Tomás Recio, University Antonio de Nebrija, Madrid, Spain.

7 Conclusions

The pursuit of new and interesting theorems in geometry, by automatic means is an interesting open problem. From the point of view of generating new information the deductive approach seems the most appropriated, given that: only logical consequences are ever generated and also the paths to those new theorems can be analysed from the point of view of the geometric theory used, i.e. in the process of generating new facts, geometric proofs of their validity are produced. Already existing implementations, e.g. the deductive databases method (DDM) implemented in *JGEx*, and new implementations, e.g. the *GeoGebra Discovery* and the new implementation of the DDM, the *OGPCP-GDDM* prover, can be used. The separation of the uninteresting, trivial facts, from the new and interesting facts is much harder. The current approaches are based in ad-hoc measures, proposed by experts from the field, but nevertheless, not substantiated by any study approaching that problem. Our goal is to fulfil that gap, to produce a comprehensive survey, supported in a large set of mathematicians, in order to be able to return to that question and to develop filters supported by the findings of that survey.

Acknowledgements The authors wish to thank Francisco Botana, Thierry Dana-Picard, James Davenport and Tomás Recio for their support in the pursue of this long term project.

[9]A Likert scale is a question which is a five-point or seven-point scale. The choices range from Strongly Agree to Strongly Disagree so the survey maker can get a holistic view of people's opinions. It was developed in 1932 by the social psychologist Rensis Likert.

References

[1] Miguel Á. Abánades, Francisco Botana, Antonio Montes & Tomás Recio (2014): *An algebraic taxonomy for locus computation in dynamic geometry*. Computer-Aided Design 56, pp. 22–33, doi:10.1016/j.cad.2014.06.008.

[2] Gilles Aldon, Pierre-Yves Cahuet, Viviane Durand-Guerrier, Mathias Front, Didier Krieger, Michel Mizony & Claire Tardy C (2010): *Expérimenter des problèmes innovants en mathématiques à l'école*. INRP. Available at `https://hal.archives-ouvertes.fr/hal-00989132`.

[3] Ferdinando Arzarello & Cristina Sabena (2010): *Semiotic and theoretic control in argumentation and proof activities*. Educational Studies in Mathematics 77(2-3), pp. 189–206, doi:10.1007/s10649-010-9280-3.

[4] Michael Aschbacher (2005): *Highly complex proofs and implications of such proofs*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 363(1835), pp. 2401–2406, doi:10.1098/rsta.2005.1655.

[5] Jeremy Avigad (2006): *Mathematical Method and Proof*. Synthese 153(1), pp. 105–159, doi:10.1007/s11229-005-4064-5. Available at `http://www.jstor.org/stable/27653412`.

[6] Nuno Baeta & Pedro Quaresma (2021): *Open Geometry Prover Community Project*. Electronic Proceedings in Theoretical Computer Science 352, pp. 129–138, doi:10.4204/EPTCS.352.14.

[7] Nuno Baeta & Pedro Quaresma (2023): *Towards a Geometry Deductive Database Prover*. Annals of Mathematics and Artificial Intelligence 91(6), pp. 851–863, doi:10.1007/s10472-023-09839-0.

[8] Henk Barendregt & Freek Wiedijk (2005): *The challenge of computer mathematics*. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 363(1835), pp. 2351–2375, doi:10.1098/rsta.2005.1650.

[9] Francisco Botana, Miguel A. Abánades & Jesús Escribano (2007): *Computing Locus Equations for Standard Dynamic Geometry Environments*. In Yong Shi, G. Dick van Albada, Jack Dongarra & Peter M. A. Sloot, editors: International Conference on Computational Science, Lecture Notes in Computer Science 4488, Springer Berlin Heidelberg, pp. 227–234, doi:10.1007/978-3-540-72586-2_32.

[10] Francisco Botana & José L. Valcarce (2002): *A dynamic-symbolic interface for geometric theorem discovery*. Computers and Education 38(1-3), pp. 21–35, doi:10.1016/S0360-1315(01)00089-6.

[11] Alan Bundy, Mateja Jamnik & Andrew Fugard (2005): *What is a proof?* Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 363(1835), pp. 2377–2391, doi:10.1098/rsta.2005.1651.

[12] Tyler Burge (1998): *Computer proof, apriori knowledge, and other minds: The sixth philosophical perspectives lecture*. Philosophical perspectives 12, pp. 1–37, doi:10.1093/acprof:oso/9780199672028.003.0013.

[13] Xiaoyu Chen, Dan Song & Dongming Wang (2014): *Automated generation of geometric theorems from images of diagrams*. Annals of Mathematics and Artificial Intelligence 74(3-4), pp. 1–26, doi:10.1007/s10472-014-9433-7.

[14] Jingde Cheng (1995): *Entailment calculus as the logical basis of automated theorem finding in scientific discovery*. In: Systematic Methods of Scientific Discovery: Papers from the 1995 Spring Symposium, AAAI Press, pp. 105–110.

[15] Jingde Cheng (2000): *A Strong Relevant Logic Model of Epistemic Processes in Scientific Discovery*. In E. Kawaguchi, H. Kangassalo, H. Jaakkola & I.A. Hamid, editors: Information Modelling and Knowledge Bases XI, Frontiers in Artificial Intelligence and Applications 61, IOS Press, pp. 136–159, doi:10.1007/3-540-49292-5_42.

[16] Shang-Ching Chou, Xiao-Shan Gao & Jing-Zhong Zhang (2000): *A Deductive Database Approach to Automated Geometry Theorem Proving and Discovering*. Journal of Automated Reasoning 25(3), pp. 219–246, doi:10.1023/A:1006171315513.

[17] Karel Chvalovský, Jan Jakubův, Martin Suda & Josef Urban (2019): *ENIGMA-NG: Efficient Neural and Gradient-Boosted Inference Guidance for E*. In: *Lecture Notes in Computer Science*, Springer International Publishing, pp. 197–215, doi:10.1007/978-3-030-29436-6_12.

[18] Simon Colton (2002): *The HR Program for Theorem Generation*. In Andrei Voronkov, editor: *Automated Deduction—CADE-18*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 285–289, doi:10.1007/3-540-45620-1_24.

[19] Simon Colton, Alan Bundy & Toby Walsh (2000): *On the notion of interestingness in automated mathematical discovery*. *International Journal of Human-Computer Studies* 53(3), pp. 351–375, doi:10.1006/ijhc.2000.0394.

[20] Richard A De Millo, Richard J Lipton & Alan J Perlis (1979): *Social processes and proofs of theorems and programs*. *Communications of the ACM* 22(5), pp. 271–280, doi:10.1145/359104.359106.

[21] Bruno D'Amore & MI Fandiño Pinilla (2016): *Una formula per la misurazione oggettiva della difficoltà di comprensione di un testo di matematica da parte degli studenti. Uso valutativo e uso didattico*. *La matematica e la sua didattica, 1* 2, pp. 59–78.

[22] Sólrún Halla Einarsdóttir, Nicholas Smallbone & Moa Johansson (2020): *Template-based Theory Exploration: Discovering Properties of Functional Programs by Testing*. In: *IFL 2020: Proceedings of the 32nd Symposium on Implementation and Application of Functional Languages*, ACM, doi:10.1145/3462172.3462192.

[23] Ludovic Font (2021): *Génération automatique de preuves pour un logiciel tuteur en géométrie*. phdthesis, Polytechnique Montréal. Available at `https://publications.polymtl.ca/9090/`.

[24] Ludovic Font, Philippe R. Richard & Michel Gagnon (2018): *Improving QED-Tutrix by Automating the Generation of Proofs*. In Pedro Quaresma & Walther Neuper, editors: *Proceedings 6th International Workshop on Theorem proving components for Educational software, Gothenburg, Sweden, 6 Aug 2017*, *Electronic Proceedings in Theoretical Computer Science* 267, Open Publishing Association, pp. 38–58, doi:10.4204/EPTCS.267.3.

[25] Hongbiao Gao & Jingde Cheng (2015): *An epistemic programming approach for automated theorem finding*. In: *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCIxCC)*, IEEE, doi:10.1109/ICCI-CC.2015.7259365.

[26] Hongbiao Gao & Jingde Cheng (2017): *Measuring Interestingness of Theorems in Automated Theorem Finding by Forward Reasoning: A Case Study in Peano's Arithmetic*. In Ngoc Thanh Nguyen, Satoshi Tojo, Le Minh Nguyen & Bogdan Trawinski, editors: *Intelligent Information and Database Systems*, *Lecture Notes in Computer Science* 10192, Springer International Publishing, pp. 115–124, doi:10.1007/978-3-319-54430-4_12.

[27] Hongbiao Gao, Yuichi Goto & Jingde Cheng (2014): *A systematic methodology for automated theorem finding*. *Theoretical Computer Science* 554, pp. 2–21, doi:10.1016/j.tcs.2014.06.028.

[28] Hongbiao Gao, Yuichi Goto & Jingde Cheng (2015): *A Set of Metrics for Measuring Interestingness of Theorems in Automated Theorem Finding by Forward Reasoning: A Case Study in NBG Set Theory*. In: *Intelligence Science and Big Data Engineering. Big Data and Machine Learning Techniques*, Springer International Publishing, pp. 508–517, doi:10.1007/978-3-319-23862-3_50.

[29] Hongbiao Gao, Jianbin Li & Jingde Cheng (2018): *Measuring Interestingness of Theorems in Automated Theorem Finding by Forward Reasoning: A Case Study in Tarski's Geometry*. In: *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, IEEE, doi:10.1109/SmartWorld.2018.00064.

[30] Hongbiao Gao, Jianbin Li & Jingde Cheng (2019): *Measuring Interestingness of Theorems in Automated Theorem Finding by Forward Reasoning Based on Strong Relevant Logic*. In: *2019 IEEE International Conference on Energy Internet (ICEI)*, IEEE, pp. 356–361, doi:10.1109/ICEI.2019.00069.

[31] Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar & Michael Norrish (2021): *TacticToe: Learning to Prove with Tactics*. Journal of Automated Reasoning 65(2), pp. 257–286, doi:10.1007/s10817-020-09580-x.

[32] Gila Hanna, David Reid & Michael de Villiers, editors (2019): *Proof Technology in Mathematics Research and Teaching*. Springer, doi:10.1007/978-3-030-28483-1.

[33] Kirsti Hemmi, Erika Julin & Ray Pörn (2017): *Misconceptions and developmental proof*. In: CERME 10.

[34] Moa Johansson & Nicholas Smallbone (2021): *Automated Conjecturing in QuickSpec*. In: 1st Mathematical Reasoning in General Artificial Intelligence Workshop, ICLR 2021.

[35] Zoltán Kovács, Tomás Recio & M. Pilar Vélez (2021): *Automated reasoning tools in GeoGebra discovery*. ACM Communications in Computer Algebra 55(2), pp. 39–43, doi:10.1145/3493492.3493495.

[36] Zoltán Kovács & Jonathan H. Yu (2022): *Automated Discovery of Geometrical Theorems in GeoGebra*. In João Marcos, Walther Neuper & Pedro Quaresma, editors: Proceedings 10th International Workshop on Theorem Proving Components for Educational Software, Electronic Proceedings in Theoretical Computer Science 354, pp. 1–12, doi:10.4204/EPTCS.354.1.

[37] Fou-Lai Lin, Feng-Jui Hsieh, Gila Hanna & Michael de Villiers, editors (2009): *Proceedings of the ICMI Study 19 conference: Proof and Proving in Mathematics Education*. 1, The Department of Mathematics, National Taiwan Normal University.

[38] Fou-Lai Lin, Feng-Jui Hsieh, Gila Hanna & Michael de Villiers, editors (2009): *Proceedings of the ICMI Study 19 conference: Proof and Proving in Mathematics Education*. 2, The Department of Mathematics, National Taiwan Normal University.

[39] Donald MacKenzie (1995): *The automation of proof: A historical and sociological exploration*. IEEE Annals of the History of Computing 17(3), pp. 7–29, doi:10.1109/85.397057.

[40] Donald MacKenzie (1999): *Slaying the Kraken: The sociohistory of a mathematical proof*. Social studies of science 29(1), pp. 7–60, doi:10.1177/030631299029001002.

[41] R. L. McCasland, A. Bundy & P. F. Smith (2017): *MATHsAiD: Automated mathematical theory exploration*. Applied Intelligence 47(3), pp. 585–606, doi:10.1007/s10489-017-0954-8.

[42] Xicheng Peng, Qihang Chen, Jingzhong Zhang & Mao Chen (2021): *Automated Discovery of Geometric Theorems Based on Vector Equations*. Journal of Automated Reasoning 65(6), pp. 711–726, doi:10.1007/S10817-021-09591-2.

[43] D. A. Plaisted (1980): *Abstraction mappings in mechanical theorem proving*. In: 5th Conference on Automated Deduction Les Arcs, France, July 8–11, 1980, Springer Berlin Heidelberg, pp. 264–280, doi:10.1007/3-540-10009-1_21.

[44] David A. Plaisted (1994): *The search efficiency of theorem proving strategies*. In: Automated Deduction — CADE-12, Springer Berlin Heidelberg, pp. 57–71, doi:10.1007/3-540-58156-1_5.

[45] George Polya (2004): *How to solve it: A new aspect of mathematical method*. 246, Princeton university press.

[46] Yury Puzis, Yi Gao & G. Sutcliffe (2006): *Automated Generation of Interesting Theorems*. In: FLAIRS Conference.

[47] Markus N. Rabe & Christian Szegedy (2021): *Towards the Automatic Mathematician*. In André Platzer & Geoff Sutcliffe, editors: Automated Deduction – CADE 28, Springer International Publishing, pp. 25–37, doi:10.1007/978-3-030-79876-5_2.

[48] T. Recio & M. P. Vélez (1999): *Automatic Discovery of Theorems in Elementary Geometry*. J. Autom. Reason. 23, pp. 63–82, doi:10.1023/A:1006135322108. Available at `http://dl.acm.org/citation.cfm?id=594128.594243`.

[49] H. G. Rice (1953): *Classes of recursively enumerable sets and their decision problems*. Transactions of the American Mathematical Society 74(2), pp. 358–366, doi:10.2307/1990888.

[50] Hartley Rogers Jr (1987): *Theory of recursive functions and effective computability*. MIT press.

[51] Michael Sipser (1997): *Introduction to the Theory of Computation*. PWS Publishing Company.

[52] Andreas J Stylianides & Guershon Harel (2018): *Advances in mathematics education research on proof and proving: An international perspective*. Springer, doi:10.1007/978-3-319-70996-3.

[53] Geoff Sutcliffe (2017): *The TPTP Problem Library and Associated Infrastructure*. *Journal of Automated Reasoning* 59(4), pp. 483–502, doi:10.1007/s10817-017-9407-7.

[54] Philip Todd (2021): *A Method for the Automated Discovery of Angle Theorems*. In: *Proceedings of ADG 2021*, 352, Open Publishing Association, pp. 148–155, doi:10.4204/EPTCS.352.17.

[55] Andrei Voronkov (2000): *CASC-16-1/2*. Preprint Series CSSPP-4, The University of Manchester. Available at `http://www.s.man.a.uk/preprints/index.html`.

[56] Larry Wos (1988): *Automated Reasoning: 33 Basic Research Problems*. Prentice-Hall.

[57] Larry Wos (1993): *The problem of automated theorem finding*. *Journal of Automated Reasoning* 10(1), pp. 137–138, doi:10.1007/BF00881868.

[58] Zheng Ye, Shang-Ching Chou & Xiao-Shan Gao (2011): *An Introduction to Java Geometry Expert*. In Thomas Sturm & Christoph Zengler, editors: *Automated Deduction in Geometry*, *Lecture Notes in Computer Science* 6301, Springer Berlin Heidelberg, pp. 189–195, doi:10.1007/978-3-642-21046-4_10.

[59] Jian Zhang (1999): *System Description: MCS: Model-Based Conjecture Searching*. In: *Automated Deduction — CADE-16*, Springer Berlin Heidelberg, pp. 393–397, doi:10.1007/3-540-48660-7_37.

A First Survey—Interesting Theorems

With this survey the goal will be to find the characteristics that make a theorem interesting, or not. A list of questions about geometric theorems found to be interesting, or not interesting.

For an initial pool of expert on the area it is our intention to use the network created for the submission of the COST proposal, *iGEOMXXI*.[10] This survey will be available online, based on an online survey tool.[11]

A.1 Interesting and Why?

A list of situations/explanations about interesting theorems.

[10]OC-2020-1-24509, Building a Networked Environment for Geometric Reasoning (iGEOMXXI), The submitted Action (not funded) focused on the exploration of new paradigms and methodologies for supporting formal reasoning in the field of Geometry. A network of 49 experts from 19 countries.

[11]e.g. *LimeSurvey*, `https://www.limesurvey.org/`

Can you describe in detail a situation (during classes or lectures) in which you have used the adjective interesting applied to a theorem in geometry?

***n*th Situation** ______________________________

____

____

____

Can you explain in detail the reasons why you used the adjective interesting in the first situation?

***n*th Explanation** ______________________________

____

____

A.2 Five Interesting Theorems in Geometry

A list of 5 questions, each about an interesting theorem.

Can you list at least five theorems in geometry that you consider interesting?

Theorem *n* ______________________________

____

____

Can you explain in detail the reason for your choice by listing at least five adjectives that describe characteristics of the previous theorem making it interesting?

____

____

____

A.3 Five Not Interesting Theorems in Geometry

A list of 5 questions, each about a not interesting theorem.

Can you list at least five theorems in geometry that you consider NOT interesting?

Theorem *n* ____

____

____

Can you explain in detail the reason for your choice by listing at least five adjectives that describe characteristics of the previous theorem making it NOT interesting?

____

____

____

B Second Survey—Characteristics of Interesting Theorems

This survey will only be designed after studying the results of the first survey. The second survey will propose theorems (taken from the first survey) and will provide characteristics (taken from the first survey) for each of them. The survey will ask the participants to express their opinion on characteristics that (presumably) make the theorems interesting or not interesting.

This survey will be available online, based on an online survey tool.[9]

Please express whether you consider the following theorems interesting or not, and why?

Is Theorem *n* interesting?
☐ YES ☐ NO

Why? Because it has the characteristic A.
☐ Strongly disagree ☐ Disagree ☐ Neutral ☐ Agree ☐ Strongly Agree

Why? Because it has the characteristic B.
☐ Strongly disagree ☐ Disagree ☐ Neutral ☐ Agree ☐ Strongly Agree

Why? Because it has the characteristic C.
☐ Strongly disagree ☐ Disagree ☐ Neutral ☐ Agree ☐ Strongly Agree

Why? Because it has the characteristic D.
☐ Strongly disagree ☐ Disagree ☐ Neutral ☐ Agree ☐ Strongly Agree

Solving with GeoGebra Discovery an Austrian Mathematics Olympiad Problem: Lessons Learned

Belén Ariño-Morera
Departamento de Economía Financiera y Contabilidad, Universidad Rey Juan Carlos, Madrid, Spain
belen.arino@urjc.es

Zoltán Kovács
The Private University College of Education of the Diocese of Linz, Austria[*]
zoltan.kovacs@ph-linz.at

Tomás Recio
Escuela Politécnica Superior, Universidad Antonio de Nebrija, Madrid, Spain
trecio@nebrija.es

Piedad Tolmos
Departamento de Economía Financiera y Contabilidad, Universidad Rey Juan Carlos, Madrid, Spain
piedad.tolmos@urjc.es

We address, through the automated reasoning tools in GeoGebra Discovery, a problem from a regional phase of the Austrian Mathematics Olympiad 2023. Trying to solve this problem gives rise to four different kind of feedback: the almost instantaneous, automated solution of the proposed problem; the measure of its complexity, according to some recent proposals; the automated discovery of a generalization of the given assertion, showing that the same statement is true over more general polygons than those mentioned in the problem; and the difficulties associated to the analysis of the surprising and involved high number of degenerate cases that appear when using the `LocusEquation` command in this problem. In our communication we will describe and reflect on these diverse issues, enhancing its exemplar role for showing some of the advantages, problems, and current fields of development of GeoGebra Discovery.

1 Introduction

In the past years we have been developing and including, both in the standard version of GeoGebra[1] as well as in the fork version GeoGebra Discovery[2], different automated reasoning tools (ART) [1]. Mathematics Olympiads problems provide an interesting benchmark for testing the performance of such instruments. Interesting, from multiple perspectives: by itself, as a source of challenging mathematical questions that our ART should be able to deal with; as a test to measure and compare the performance of humans and machines on the same problems [2],[3]; as a test on the suitability of the recently proposed intrinsic measure of complexity of a geometric statement [4].

In this communication we will focus on how our ART behave concerning some of these issues, when addressing an interesting problem recently proposed in a regional phase of the Austrian Mathematics

[*]The work was partially supported by the grant PID2020-113192GB-I00 from the Spanish MICINN.
[1]`www.geogebra.org`
[2]`https://kovzol.github.io/geogebra-discovery/`

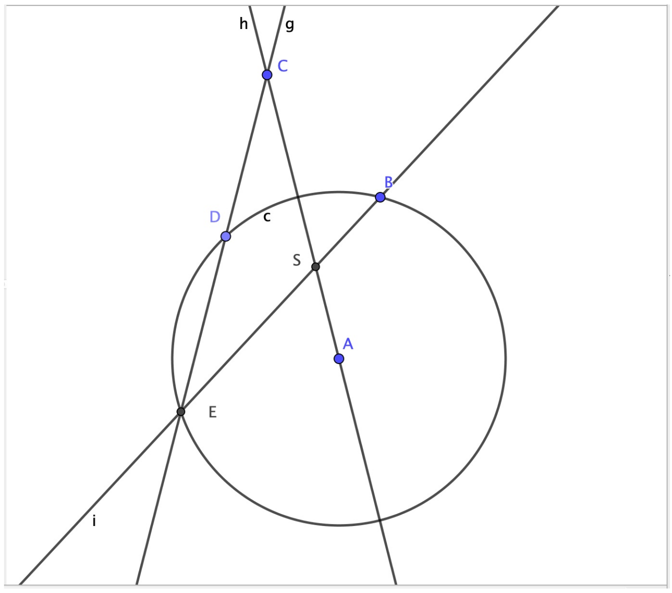


Figure 1: Problem 2. Österreichische Mathematik-Olympiade Regionalwettbewerb für Fortgeschrittene 30. März 2023.

Olympiad 2023. Namely, Problem 2 at the 54. Austrian Mathematics Olympiad 2023,[3] stated as follows (see Figure 1):

> Sei $ABCD$ eine Raute mit $\angle BAD < 90°$. Der Kreis durch D mit Mittelpunkt A schneide die Gerade CD ein zweites MaI im Punkt E. Der Schnittpunkt der Geraden BE und AC sei S.
>
> Man beweise, dass die Punkte A, S, D und E auf einem Kreis liegen.

that is

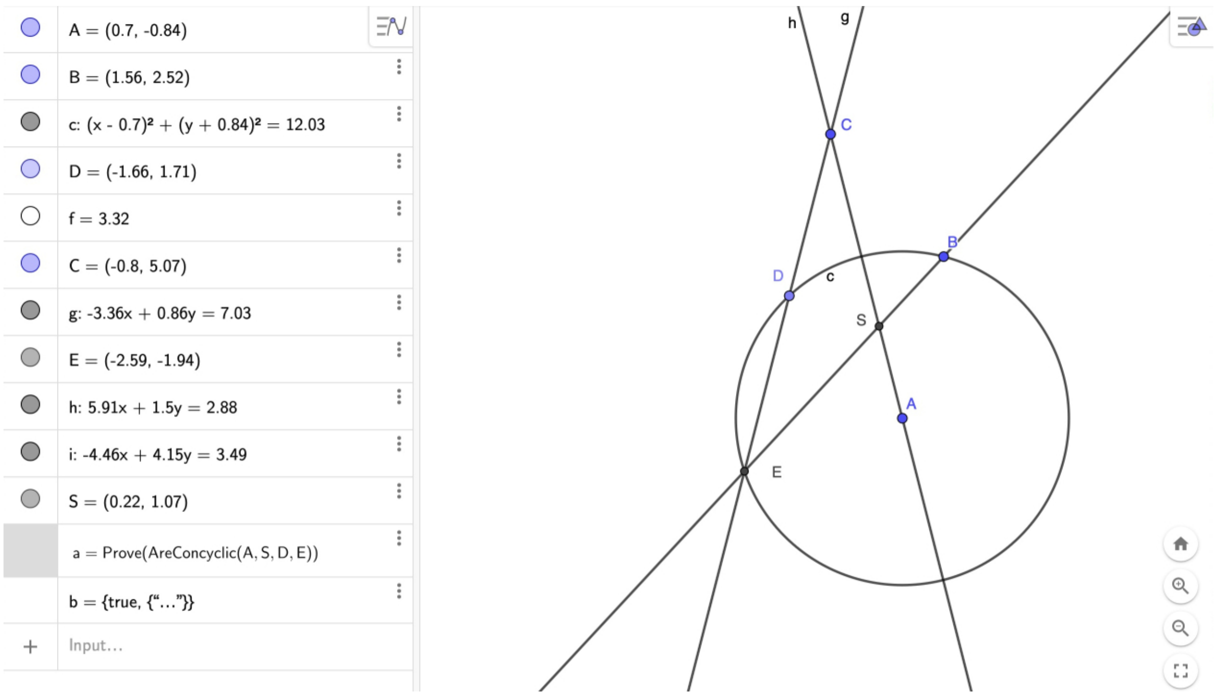
> Let $ABCD$ be a rhombus with $\angle BAD < 90°$. The circle through D with center A intersects straight line CD a second time at point E. The intersection of the lines BE and AC is S.
>
> Prove that the points A, S, D and E lie on a circle.

In the next section we will show how GeoGebra Discovery is able to confirm (by proving internally, in a mathematically rigorous way) the truth of the proposed problem.

2 Solving the Problem

In what follows, to exemplify how GeoGebra Discovery handles this problem, we will use in different moments, the web version GeoGebra Discovery 6 or the app GeoGebra Discovery 5. To start with, Figure 2 shows how GeoGebra Discovery version 6.0.641.0-2023Apr22 confirms the truth of the given statement.

[3]Österreichische Mathematik-Olympiade Regionalwettbewerb für Fortgeschrittene, 30. März 2023. `https://oemo.at/OeMO/Termine/2023/`



```
A = (0.7, -0.84)
B = (1.56, 2.52)
c: (x - 0.7)² + (y + 0.84)² = 12.03
D = (-1.66, 1.71)
f = 3.32
C = (-0.8, 5.07)
g: -3.36x + 0.86y = 7.03
E = (-2.59, -1.94)
h: 5.91x + 1.5y = 2.88
i: -4.46x + 4.15y = 3.49
S = (0.22, 1.07)
a = Prove(AreConcyclic(A, S, D, E))
b = {true, {"..."}}
Input...
```

Figure 2: Confirming the truth of the statement in Problem 2.

First, we have chosen some free points A,B, then the circle c centered at A through B, then another point D on this circle, such that $\angle BAD < 90^\circ$. Next, we have built the (hidden) segment $f = BD$, and point C as the symmetrical of A with respect to f. Thus, $ABCD$ is a rhombus. Finally, points E,S are displayed, following the hypotheses, as the intersection of line CD and c (ditto, as the intersection of line BE and AC).

Then we have introduced the commands

```
Prove(AreConcyclic(A,S,D,E))
```

and

```
ProveDetails(AreConcyclic(A,S,D,E)),
```

yielding in both cases just the declaration that the statement is true. See items a,b in the input column in the Figure 2 or in Figure 3. Notice that Figure 3 displays the circle through A,S,D,E, even if the angle $\angle BAD > 90^\circ$. Indeed, we have not made any formal implementation of such restriction, and this implies the statement holds even without such conditions. Thus, we can say we have already proved an extended version of the given problem.

Let us conclude by remarking two more things: one, that the internal proof is mathematically rigorous, dealing with symbolic computation algorithms (not through a numerical or probabilistic approach); two, that the complexity of the involved computation has made impossible to output the list of associated degeneracy conditions (i.e. geometric situations that must be avoided for the statement to hold true, like having $ABCD$ aligned) that could have been displayed (in simpler cases) on the output of the `ProveDetails` command.

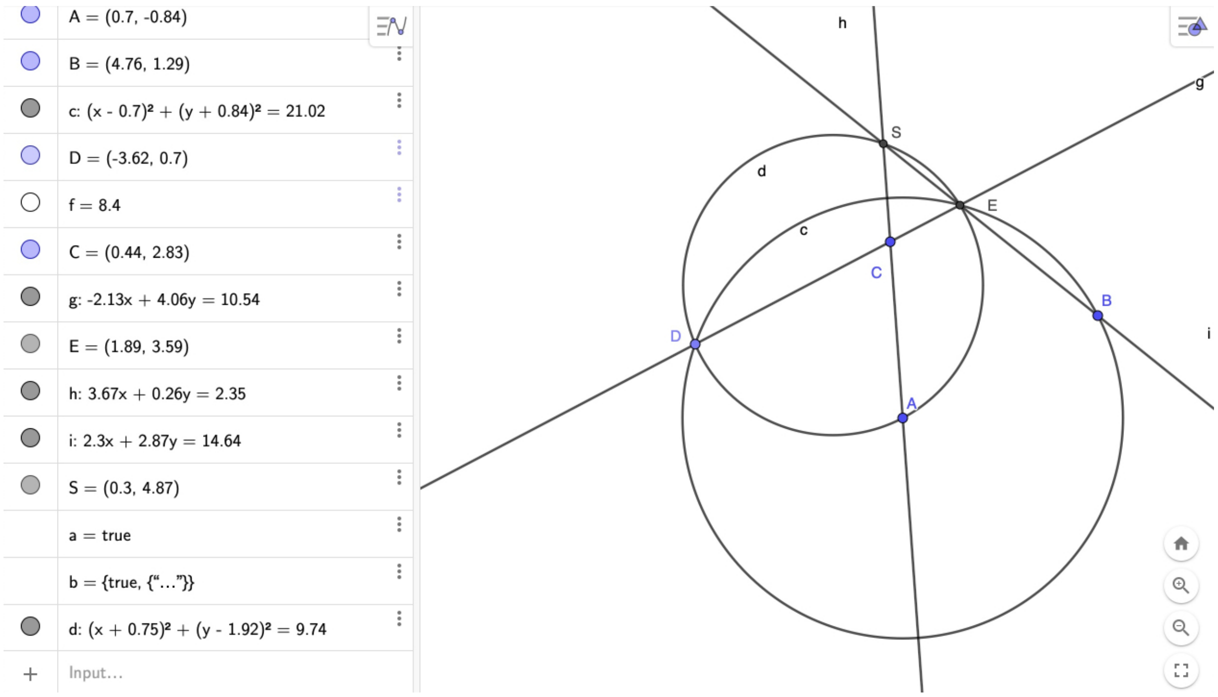

A = (0.7, -0.84)
B = (4.76, 1.29)
c: (x - 0.7)² + (y + 0.84)² = 21.02
D = (-3.62, 0.7)
f = 8.4
C = (0.44, 2.83)
g: -2.13x + 4.06y = 10.54
E = (1.89, 3.59)
h: 3.67x + 0.26y = 2.35
i: 2.3x + 2.87y = 14.64
S = (0.3, 4.87)
a = true
b = {true, {"..."}}
d: (x + 0.75)² + (y - 1.92)² = 9.74
Input...

Figure 3: Confirming the truth of the statement in Problem 2 and showing the circle through A,S,D,E.

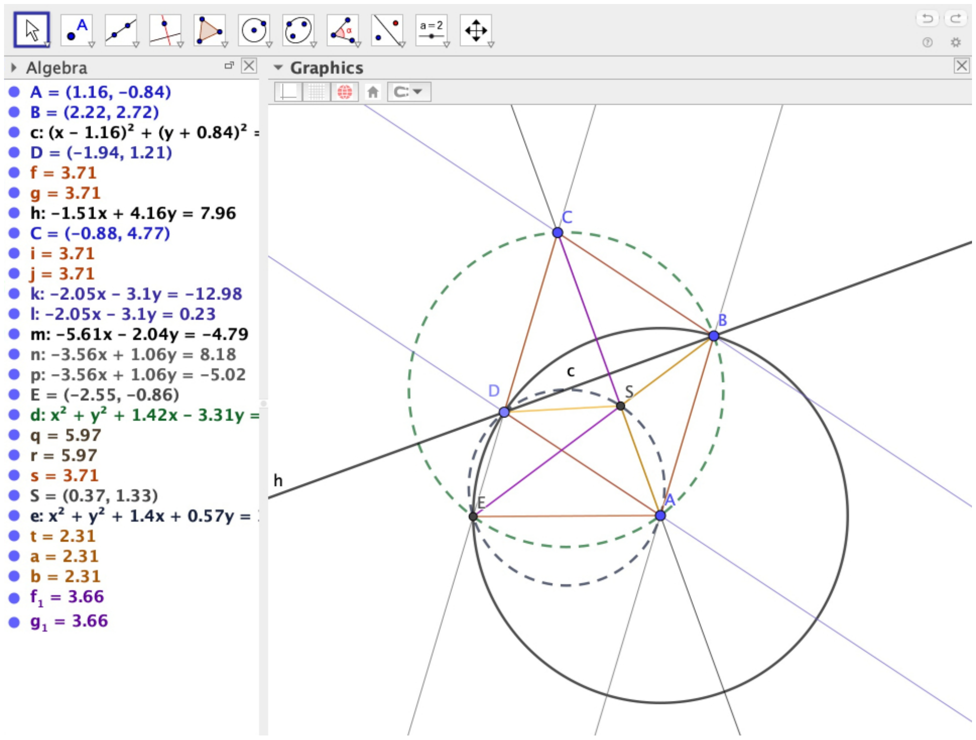
3 Generalizing the Problem

Conceptually speaking, GeoGebra Discovery is prone to offer tools that help, not only to check the truth/falsity of a given assertion, but to automatically test the verification of a large collection of properties among the elements that the user is adding in the construction of the geometric figure. That is, to help the user *discovering* different properties holding under the given hypotheses. Figure 4 shows, in colors, different properties that the program has discovered, after enabling the `StepwiseDiscovery` command, along each of the steps towards the construction of the rhombus, of the points E,S, etc.

For instance, just after introducing free points A,B and point D in the circle c, GeoGebra Discovery tells the user that segments $f = DA$, $t = BA$ have equal length, a trivial result, since A is the center of circle c and B,D are points on c. Perhaps more interesting is to learn that circle d contains points A,B,C,E, i.e. that these points are concyclic; or that $t = BS$, $a = AS$, $b = DS$ have equal length, a couple of non-trivial results that GeoGebra Discovery outputs automatically towards the end of the construction, without being asked to consider such specific relations.

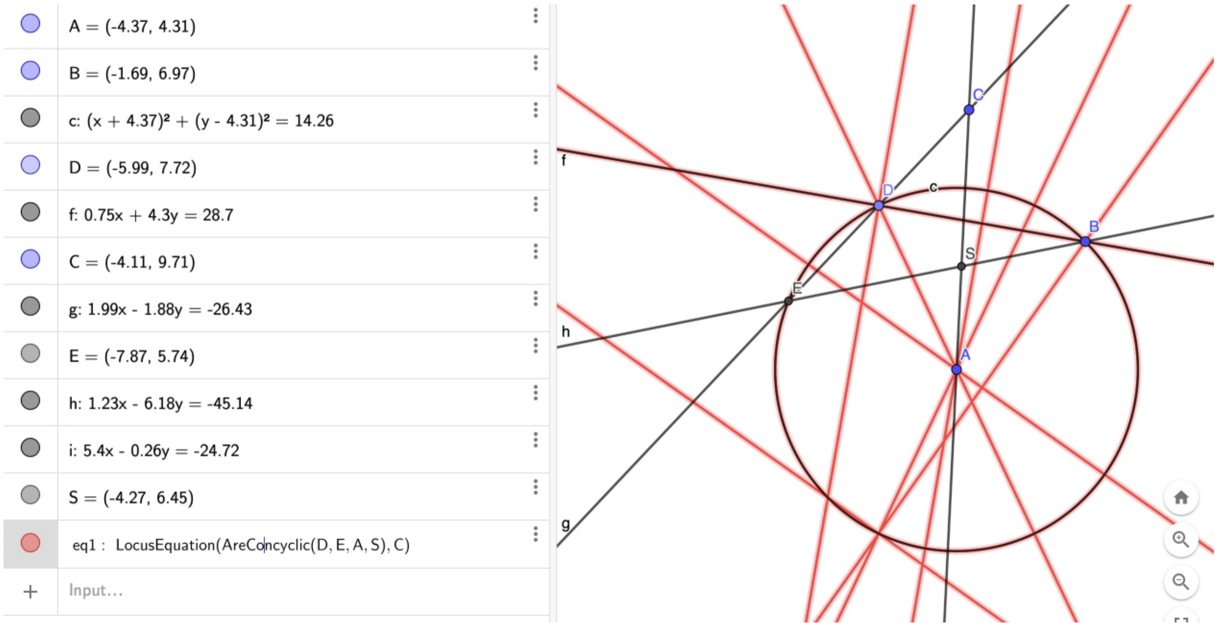
We would like to focus on some other way, perhaps less automatic but more relevant for the user, of *discovering* results with GeoGebra Discovery. Namely, let us assume the user is interested in a precise question: finding necessary conditions for the converse of the given Problem 2. This can be addressed through the `LocusEquation(AreConcyclic(D,E,A,S),C)` command. Figure 5 shows the output of this command, a very complicated degree-10-equation that (seems) to be the product of eight lines and the circle c. The locus described by this equation includes all positions of a free point C such that A,S,D,E are concyclic (where A,S,D,E are defined repeating the construction of the previous figures, except for point C, that here is just a free point, not the mirror of A with respect to the line BD).

Let us remark that the output is a numerical equation, with coefficients depending on the coordinates



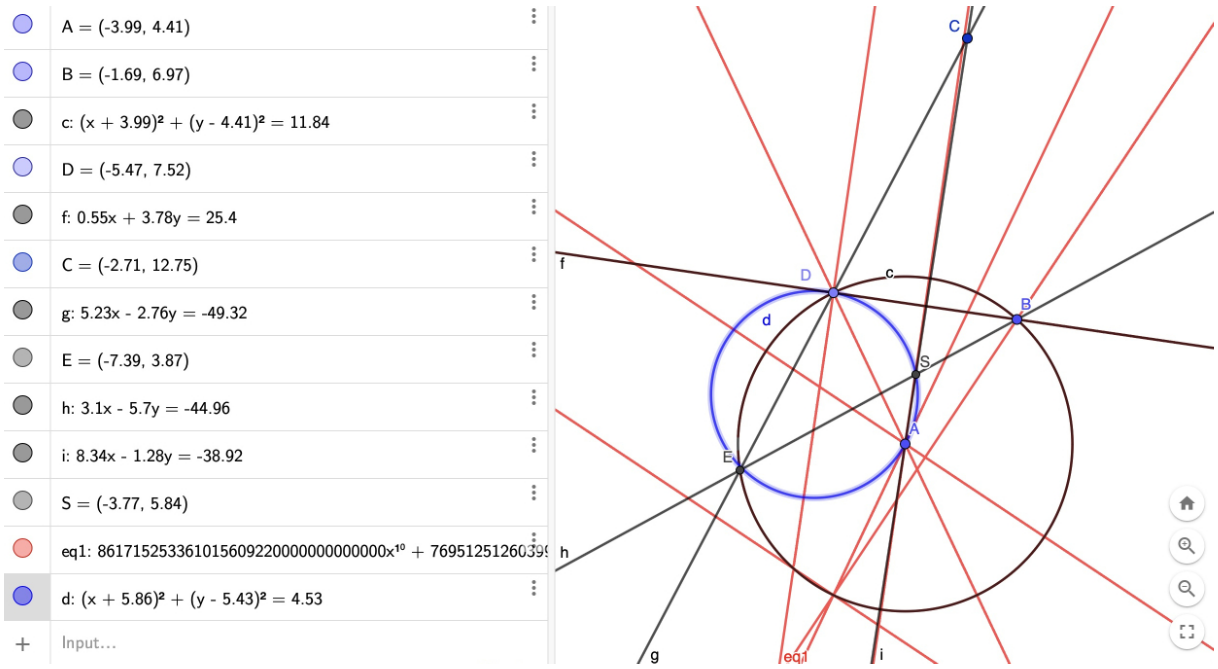
A = (1.16, −0.84)
B = (2.22, 2.72)
c: (x − 1.16)² + (y + 0.84)² =
D = (−1.94, 1.21)
f = 3.71
g = 3.71
h: −1.51x + 4.16y = 7.96
C = (−0.88, 4.77)
i = 3.71
j = 3.71
k: −2.05x − 3.1y = −12.98
l: −2.05x − 3.1y = 0.23
m: −5.61x − 2.04y = −4.79
n: −3.56x + 1.06y = 8.18
p: −3.56x + 1.06y = −5.02
E = (−2.55, −0.86)
d: x² + y² + 1.42x − 3.31y =
q = 5.97
r = 5.97
s = 3.71
S = (0.37, 1.33)
e: x² + y² + 1.4x + 0.57y =
t = 2.31
a = 2.31
b = 2.31
f₁ = 3.66
g₁ = 3.66

Figure 4: Confirming the truth of the statement in Problem 2, and many others, through the *Stepwise-discover* command.



A = (-4.37, 4.31)
B = (-1.69, 6.97)
c: (x + 4.37)² + (y - 4.31)² = 14.26
D = (-5.99, 7.72)
f: 0.75x + 4.3y = 28.7
C = (-4.11, 9.71)
g: 1.99x - 1.88y = -26.43
E = (-7.87, 5.74)
h: 1.23x - 6.18y = -45.14
i: 5.4x - 0.26y = -24.72
S = (-4.27, 6.45)
eq1 : LocusEquation(AreConcyclic(D, E, A, S), C)
+ Input...

Figure 5: Locus of C (in red color) for the concyclicity of A,S,D,E, assuming only that B,D are in a circle centered at A.



$A = (-3.99, 4.41)$

$B = (-1.69, 6.97)$

c: $(x + 3.99)^2 + (y - 4.41)^2 = 11.84$

$D = (-5.47, 7.52)$

f: $0.55x + 3.78y = 25.4$

$C = (-2.71, 12.75)$

g: $5.23x - 2.76y = -49.32$

$E = (-7.39, 3.87)$

h: $3.1x - 5.7y = -44.96$

i: $8.34x - 1.28y = -38.92$

$S = (-3.77, 5.84)$

eq1: $861715253361015609220000000000000x^{10} + 769512512603 9$

d: $(x + 5.86)^2 + (y - 5.43)^2 = 4.53$

Input...

Figure 6: Visually checking the validity of the locus of C for the bisector line of BD.

of the free points A, B, D in the construction. So, it is neither a symbolic geometric object, nor an object we could enter in the ART to check if placing C over some of the components implies A, S, D, E are concyclic. We would have to learn how to build the locus starting from A, B, D, we would have to find some intrinsic geometric description of the locus, say, the analysis and discussion of the different lines.

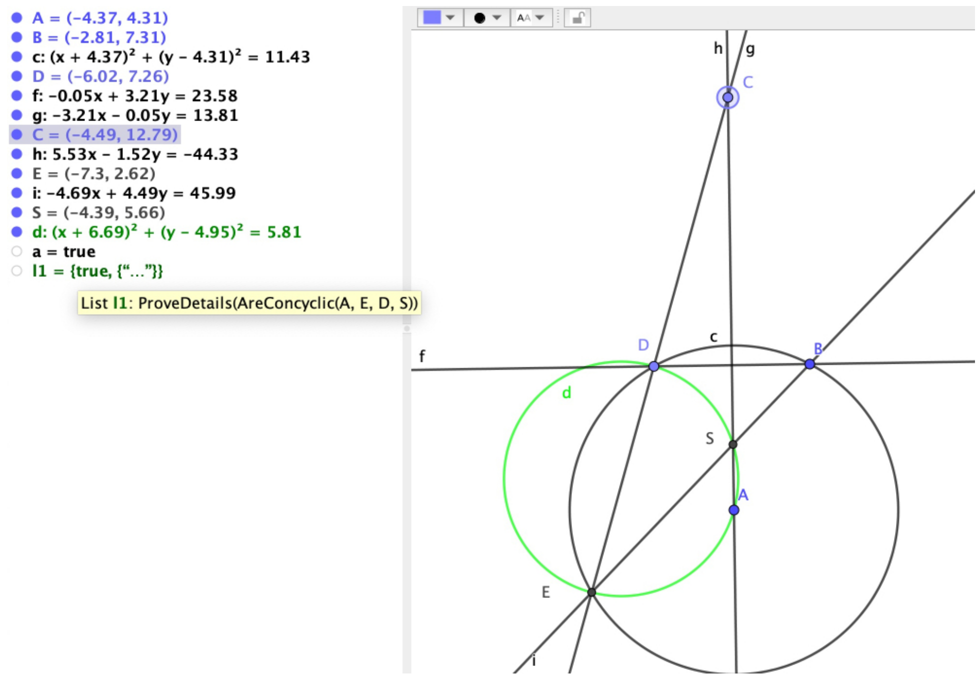
In this particular case is not difficult to state that placing C in the circle yields a degenerate case (since then $C = E = S$). And the same happens for several of the lines, except for the line that is perpendicular to BD, see Figure 6 and, for a rigorous verification, Figure 7, displaying the original construction, but now $ABCD$ forms a *kite*, since C is placed anywhere on the bisector line of BD.

Leaving aside the curious fact that, using GeoGebra Discovery ART we have been able to solve, and to generalize, an Olympiad Problem, we consider it is important to reflect on some other consequences—on the educational context—of the sequence of facts we have described in this section.

Indeed, it is well known the use of locus computation as a relevant methodology in mathematics education. For example, let us refer to the recent chapter [5], where the authors describe an experience involving over 200 secondary education students from Sicily (Italy), analyzing in detail the impact of GeoGebra on the performance of the students that had to find and to describe different geometric loci.

We have realized (see [6]) that the use of GeoGebra Discovery ART for accomplishing the same tasks, would imply a substantial methodological change: looking for most of the considered loci would turn out to be quite trivial, if it is just required finding the equation or displaying the graph of the locus. On the other hand, we think that the considered rich learning environment remains if the proposed activities focus, instead, on the description of the intrinsic geometric features of the obtained locus, as well as on their construction, two tasks that GeoGebra Discovery is not able to address automatically, but can contribute to its achievement.

In this context, we consider that the extended version of Problem 2 we have considered in this sec-



```
A = (−4.37, 4.31)
B = (−2.81, 7.31)
c: (x + 4.37)² + (y − 4.31)² = 11.43
D = (−6.02, 7.26)
f: −0.05x + 3.21y = 23.58
g: −3.21x − 0.05y = 13.81
C = (−4.49, 12.79)
h: 5.53x − 1.52y = −44.33
E = (−7.3, 2.62)
i: −4.69x + 4.49y = 45.99
S = (−4.39, 5.66)
d: (x + 6.69)² + (y − 4.95)² = 5.81
a = true
l1 = {true, {"..."}}
List l1: ProveDetails(AreConcyclic(A, E, D, S))
```

Figure 7: Proving the validity of Problem 2 extended by placing C just on the bisector line of BD.

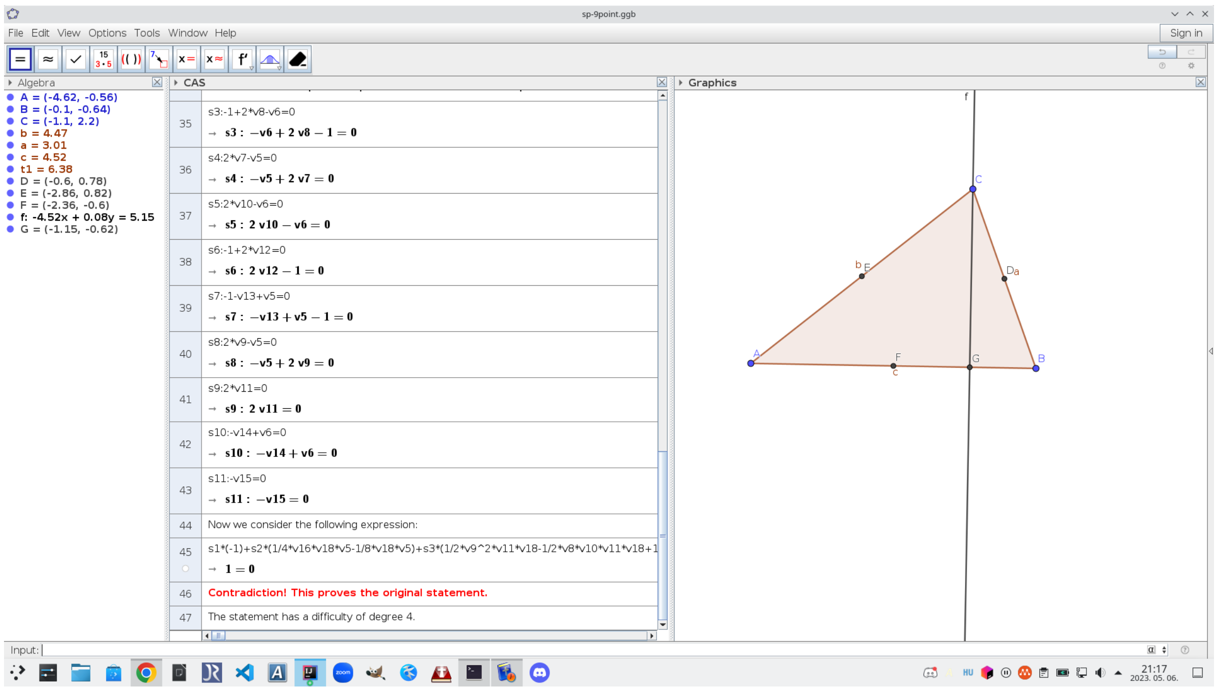
tion, provides another excellent environment for exploring geometric locus with the help of automated reasoning tools, showing both their limitations as well as their useful features.

4 Grading Problem 2

Very recently we have proposed an algorithmic way to associate, to each geometry statement, a grade that intends to estimate its difficulty or interest. It is yet a proposal in a very initial state—although already implemented in the last version of GeoGebra Discovery, through the `ShowProof` command—that, roughly speaking, computes (a bound on) the degree of the polynomials g_i expressing the thesis polynomial T as a combination of the hypotheses polynomials h_i, i.e. $T = \sum_i g_i \cdot h_i$.

We refer to [4] for further details and examples. Let us just mention here that most classical, elementary theorems (e.g. Pythagoras, Intersection of Medians, Intersection of Heights, etc.) get grades 1 or 2. Or a partial formulation of the 9-point circle theorem gets complexity 4 (see Figure 8) while this Problem 2 of the Austrian Mathematical Olympiad has got grade 10! See Figure 9.

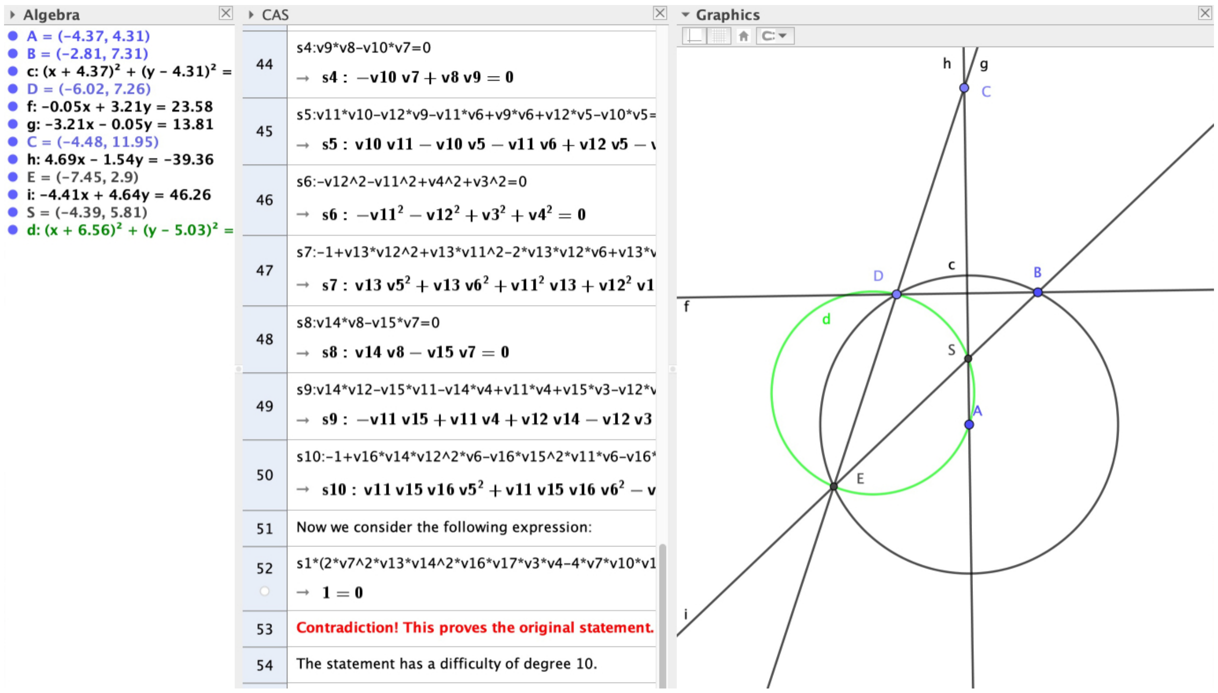
To obtain this grade, GeoGebra Discovery internally computes a Gröbner basis of the hypotheses ideal (with respect to a certain degree ordering: the impact of the choice of order is still under study), with 13 elements. This computation outputs, as well, the expression of each element of the basis in terms of the hypotheses, with coefficients of degree bounded by 8. Then the expression of the thesis as a combination of the elements in the Gröbner basis is computed. Let us remark that in this formula, the degree of the multiplier polynomials is bounded by 3, but considering more precisely the sum of the involved degrees (of the polynomial multiplying a certain element of the Gröbner basis times the maximum of the degrees of the polynomials expressing this element in terms of the hypotheses), the bound is limited to 10, a number that we think is adequate to be associated to a Mathematical Olympiad



```
A = (-4.62, -0.56)
B = (-0.1, -0.64)
C = (-1.1, 2.2)
b = 4.47
a = 3.01
c = 4.52
t1 = 6.38
D = (-0.6, 0.78)
E = (-2.86, 0.82)
F = (-2.36, -0.6)
f: -4.52x + 0.08y = 5.15
G = (-1.15, -0.62)
```

```
35  s3:-1+2*v8-v6=0
    → s3 : −v6 + 2 v8 − 1 = 0
36  s4:2*v7-v5=0
    → s4 : −v5 + 2 v7 = 0
37  s5:2*v10-v6=0
    → s5 : 2 v10 − v6 = 0
38  s6:-1+2*v12=0
    → s6 : 2 v12 − 1 = 0
39  s7:-1-v13+v5=0
    → s7 : −v13 + v5 − 1 = 0
40  s8:2*v9-v5=0
    → s8 : −v5 + 2 v9 = 0
41  s9:2*v11=0
    → s9 : 2 v11 = 0
42  s10:-v14+v6=0
    → s10 : −v14 + v6 = 0
43  s11:-v15=0
    → s11 : −v15 = 0
44  Now we consider the following expression:
45  s1*(-1)+s2*(1/4*v16*v18*v5-1/8*v18*v5)+s3*(1/2*v9^2*v11*v18-1/2*v8*v10*v11*v18+1
    → 1 = 0
46  Contradiction! This proves the original statement.
47  The statement has a difficulty of degree 4.
```

Figure 8: Estimating the complexity of proving the concyclicity of the midpoints of the sides and the feet of a height in a triangle.



```
A = (−4.37, 4.31)
B = (−2.81, 7.31)
c: (x + 4.37)² + (y − 4.31)² =
D = (−6.02, 7.26)
f: −0.05x + 3.21y = 23.58
g: −3.21x − 0.05y = 13.81
C = (−4.48, 11.95)
h: 4.69x − 1.54y = −39.36
E = (−7.45, 2.9)
i: −4.41x + 4.64y = 46.26
S = (−4.39, 5.81)
d: (x + 6.56)² + (y − 5.03)² =
```

```
44  s4:v9*v8−v10*v7=0
    → s4 : −v10 v7 + v8 v9 = 0
45  s5:v11*v10−v12*v9−v11*v6+v9*v6+v12*v5−v10*v5=
    → s5 : v10 v11 − v10 v5 − v11 v6 + v12 v5 −
46  s6:−v12^2−v11^2+v4^2+v3^2=0
    → s6 : −v11² − v12² + v3² + v4² = 0
47  s7:−1+v13*v12^2+v13*v11^2−2*v13*v12*v6+v13*v
    → s7 : v13 v5² + v13 v6² + v11² v13 + v12² v1
48  s8:v14*v8−v15*v7=0
    → s8 : v14 v8 − v15 v7 = 0
49  s9:v14*v12−v15*v11−v14*v4+v11*v4+v15*v3−v12*v
    → s9 : −v11 v15 + v11 v4 + v12 v14 − v12 v3
50  s10:−1+v16*v14*v12^2*v6−v16*v15^2*v11*v6−v16*
    → s10 : v11 v15 v16 v5² + v11 v15 v16 v6² − v
51  Now we consider the following expression:
52  s1*(2*v7^2*v13*v14^2*v16*v17*v3*v4−4*v7*v10*v1
    → 1 = 0
53  Contradiction! This proves the original statement.
54  The statement has a difficulty of degree 10.
```

Figure 9: Estimating the complexity of Problem 2.

Problem of an intermediate level of difficulty.

5 Conclusions

In our communication we have illustrated, considering just a single problem from the Austrian Mathematical Olympiad 2023, several facts concerning the use of GeoGebra Discovery in the classroom:

1. the ability of GeoGebra Discovery Automated Reasoning Tools (ART) to immediately solve a problem presented at a regional Mathematics Olympiad, that the recent GeoGebra ART complexity measure ranks quite highly,
2. the use of GeoGebra Discovery as a decisive auxiliary tool to develop and confirm new, non-trivial, conjectures, such as the generalization of the proposed problem,
3. the need to change the methodological focus when working with locus computation in the classroom with Dynamic Geometry programs, from finding equations and displaying its graph, to analyzing and obtaining the geometric characteristics of the involved locus, and its construction, by using GeoGebra Discovery ART.

The opportunity to consider simultaneously all these items around a single problem, is probably the most relevant contribution of this communication.

References

[1] Kovács, Z.; Recio, T.; Vélez, M. P.: Automated reasoning tools with GeoGebra: What are they? What are they good for? In: P. R. Richard, M. P. Vélez, S. van Vaerenbergh (eds): Mathematics Education in the Age of Artificial Intelligence: How Artificial Intelligence can serve mathematical human learning. Series: Mathematics Education in the Digital Era, Vol. 17, pp. 23–44. Springer Cham (2022). doi:10.1007/978-3-030-86909-0_2

[2] Ariño-Morera, B.; Recio, T.; Tolmos, P.: Olympic geometry problems: human vs. machine. Communication to the CADGME (Digital Tools in Mathematics Education) 2022 Conference. Abstracts available at `https://drive.google.com/file/d/1qF4ceMg6gNklOPa1JVkgKND1dOqNmyka/viewpp`

[3] Ariño-Morera, M. B.: GeoGebra Discovery at EGMO 2022. Revista Do Instituto GeoGebra Internacional De São Paulo, 11(2), 005-016.2022. doi:10.23925/2237-9657.2022.v11i2p005-016

[4] Kovács, Z.; Recio, T.; Vélez, M. P.: Showing proofs, assessing difficulty with GeoGebra Discovery. Communication to the ADG (Automated Deduction in Geometry) Conference, Belgrade, 2023. `https://adg2023.matf.bg.ac.rs/downloads/slides/ShowingProofsZoltanRecioVelez.pdf` doi:10.13140/RG.2.2.31885.31205

[5] Mammana, M.F.; Pennisi, M.; Taranto, E.: Teaching Intriguing Geometric Loci with DGS. In: Aldon, G., Hitt, F., Bazzini, L., Gellert, U. (Eds). Mathematics and Technology. Advances in Mathematics Education. Springer, Cham (2017). doi:10.1007/978-3-319-51380-5_26

[6] Ariño-Morera, B.; Recio, T.; Tolmos, P.: Teaching Intriguing Geometric Loci with GeoGebra Discovery. Communication to the CADGME (Digital Tools in Mathematics Education) 2023 Conference. `https://sites.google.com/view/cadgme2023/program`

Solving Some Geometry Problems of the Náboj 2023 Contest with Automated Deduction in GeoGebra Discovery

Amela Hota
The Private University College of Education of the Diocese of Linz, Austria
amela.hota@ph-linz.at

Zoltán Kovács
The Private University College of Education of the Diocese of Linz, Austria
zoltan.kovacs@ph-linz.at

Alexander Vujic
The Private University College of Education of the Diocese of Linz, Austria
alexander.vujic@ph-linz.at

In this article, we solve some of the geometry problems of the Náboj 2023 competition with the help of a computer, using examples that the software tool GeoGebra Discovery can calculate. In each case, the calculation requires symbolic computations. We analyze the difficulty of feeding the problem into the machine and set further goals to make the problems of this type of contests even more tractable in the future.

1 Introduction

With the everyday rise of Artificial Intelligence (AI), the power of computers has become tangible for the masses. Yes, it can do your homework (not just in maths), but it can also pass your A-level exams.[1] A long series of ad-hoc studies have shed light on what the present can offer: often instant and perfect answers to questions that take years of learning to solve by human means. This raises a number of research questions, such as whether the current school system is still needed, whether teachers are still needed, or whether it is enough to have AI.[2] Of course, alongside the praise, there are also many criticisms: AI sometimes makes mistakes, especially in textual problem settings where the question is formulated in a challenging way.

Automatic geometrical derivations, on the other hand, are perfect and, as such, there is no such a major possibility of error. The answer is not derived from a statistically computed result (as is so often the case with AI-based algorithms), but a verifiable derivation is given in each case. We do not claim that the two directions cannot meet once, and indeed, ultimately, AI should refer to, i.e. use, the ADG algorithm as a subroutine. There are already prototypes working in this direction, e.g. the WolframAlpha system has been successfully coupled with an AI frontend.[3]

[1]See https://telex.hu/tech/2023/05/09/chatgpt-bing-mesterseges-intelligencia-matematika-erettsegi.

[2]See Bill Gates' notes at https://www.gatesnotes.com/ASU-and-GSV?WT.mc¡d=20230419100000$_A$SU-GSV-2023$_B$G-EM.

[3]See Stephen Wolfram's notes at https://writings.stephenwolfram.com/2023/03/chatgpt-gets-its-wolfram-superpowers/.

In this contribution, we aim for less. We are just trying to solve competitive problems with an ADG algorithm in the background. However, we leave the exact task setting to the user. This means that it is up to the user to provide the exact flow of the editing task with concrete steps. This must be done in GeoGebra Discovery[4]. However, for the inference, which requires a symbolic calculation in the tasks, the ADG algorithm steps in and, as we will see, gives the correct result in all cases. In the second half of the paper, we propose how the range of problems that can be solved in this way can be further extended.

2 The Náboj Contest

According to **naboj.org**, Náboj is an international mathematical competition designed for teams of five high-school students that represent their schools, which lasts 120 minutes and where they are trying to solve as many given problems as possible. As soon as the team correctly solves any of the problems, they receive new ones. The solutions of the problems are usually numerical. The team that solves most problems correctly in the given time limit wins. The Náboj problems in contrast to the most school exercises require a certain amount of inventiveness and ingenuity.

Traditionally, many geometric tasks require proving. Checking if a proof is correct may be a difficult process for the organizers, so it is usually avoided to set proof related problems during contests like Náboj. Instead, problem settings require computing fractions, or better, providing a non-trivial algebraic number. As a consequence, geometric problems in Náboj are mostly non-geometric, or if still so, they are set in a way to require a numerical result.

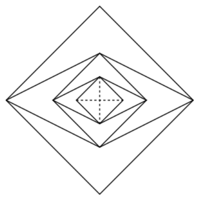
All the problems we discuss in this paper will have exact answers, non-trivial fractions or some root expressions. It is clear that the exact definition of the latter requires symbolic computation. By default, software that allows a geometric problem to be well visualized (GeoGebra in particular) provides only numerical support for measuring the quantity in question. However, the software presented here, the GeoGebra fork GeoGebra Discovery, is capable of making measurements symbolically. This also means that a full proof has already been created in the background, but the user is not informed about this.

The use of electronic aids in the Náboj competition has recently been restricted since the competition is on-site again. In the long term, electronic assistance will certainly not be eliminated. It is a fact that students are turning to AI for quick help, and the tasks set must take this into account. It may seem like fun, but the rapid pace of the modern age also poses a huge challenge for assignment writers: is the task set difficult enough to prevent the AI from giving a quick, accurate, complete answer? But the task setter is not only fighting the AI, but also the ADG algorithm: cannot the task be solved in a flash if the right data is entered into the right software in the right way and the right button is pressed?

Overall, we conclude that the tasks set should be tested with different software before they are announced, in order to avoid embarrassing surprises. Even if we manage to keep the students working with paper and pencil only for the 2 hours of the competition, i.e. to exclude electronic assistance, it raises serious questions about what the AI and ADG algorithms can achieve with the tasks set. In the preliminary analysis of problems, but also in retrospect, when we return to the correct solution of problems in mathematics class or in a specialised course, it may be useful to use the electronic method.

[4]GeoGebra Discovery is freely available at `https://kovzol.github.io/geogebra-discovery`.

Problem 6. The rhombus flower grows according to the following pattern: In the middle there is a square blossom with two diagonals of length 1. In the first step the horizontal diagonal is doubled creating a new quadrilateral. In the next step the vertical diagonal is doubled and again a new quadrilateral blossom is generated. This procedure is continued until there is a flower with five quadrilateral blossoms. Find the perimeter of the outer (i.e. the fifth) blossom.



Result. $8\sqrt{2}$

Solution. The fifth blossom is a square with diagonals of length 4, hence the length of its side is $2\sqrt{2}$ and the perimeter equals $8\sqrt{2}$.

Figure 1: Problem setting 6 and the official solution of Náboj 2023

3 Mathematical Background

The method used by GeoGebra Discovery is essentially the Recio-Vélez method [1], complemented by the algorithm given in [2]. For the problems of the Náboj competition we are discovering a ratio of two lengths, it is therefore worth using an elimination method.

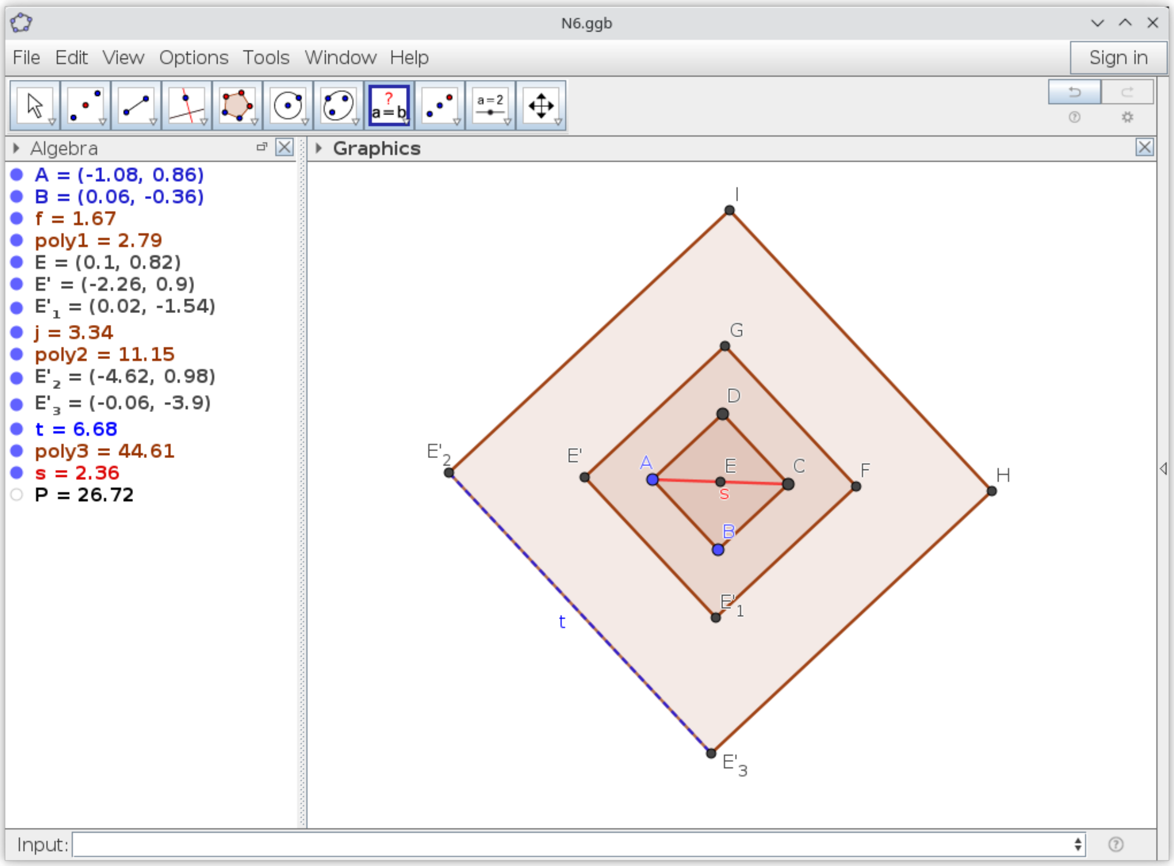
As an illustration, we show how the program solves Problem 6 of Náboj 2023 (Fig. 1).

When drawing the figure in GeoGebra Discovery (Fig. 2), we learn that the problem can be simplified to three squares. First, an arbitrary square $ABCD$ is drawn. Then midpoint E of AC is defined. This point will be then reflected about A to get E' and about B to get E'_1. Now a second square $E'E'_1FG$ is drawn. Finally, by reflecting E about E' and E'_1 we get points E'_2 and E'_3, respectively, and create the square $E'_2E'_3HI$ as well. By defining $s = AC$, $t = E'_2E'_3$ and $P = 4t$, we can use GeoGebra Discovery's Relation tool to compare s and P and we learn (after pressing the button "More..." to obtain a symbolic analysis) that $P = 8\sqrt{2} \cdot s$. The report of the symbolic analysis of Problem 6 in GeoGebra Discovery shows that "It is generally true that: $P = (8\sqrt{2}) \cdot s$ under the condition: the construction is not degenerate" (see Fig. 3). The construction steps in GeoGebra Discovery for Problem 6 can be taken from Table 1. (These extra explanations are listed in the Appendix.)

Here we highlight that the problem setting could be further simplified by skipping the construction of the two latter squares. In fact, only the reflection points matter. Also, we used the fact that the perimeter of a square equals to four times the length of a side, but this piece of information could have been ignored and asked the program to learn this on its own.

At this point, we will jump to the last steps and assuming that it is possible to do that without loss of generality, GeoGebra Discovery, in the background (in an invisible way for the normal user, but in a verifiable way via its debug messages), decides to substitute $A = (0,0)$. This will simplify the computations and the final question is if $e_{21} : m \cdot v_{24} = 4v_{23}$ holds for a given value of m. This can be answered by eliminating all variables but m from all of the equations $e_1, e_2, \ldots, e_{14}, e_{18}, \ldots, e_{21}$, and we learn that $m^2 = 128$. That is, by assuming that $m > 0$, we obtain that $m = 8\sqrt{2}$.

The algebraic solution is, of course, quite complicated. Also, the way used in GeoGebra Discovery by constructing all the required objects, may still be very complicated when compared to the quick official solution.



A = (-1.08, 0.86)
B = (0.06, -0.36)
f = 1.67
poly1 = 2.79
E = (0.1, 0.82)
E' = (-2.26, 0.9)
E'$_1$ = (0.02, -1.54)
j = 3.34
poly2 = 11.15
E'$_2$ = (-4.62, 0.98)
E'$_3$ = (-0.06, -3.9)
t = 6.68
poly3 = 44.61
s = 2.36
P = 26.72

Figure 2: Sketching Problem 6 in GeoGebra Discovery

4 Problems that can be Solved with GeoGebra Discovery

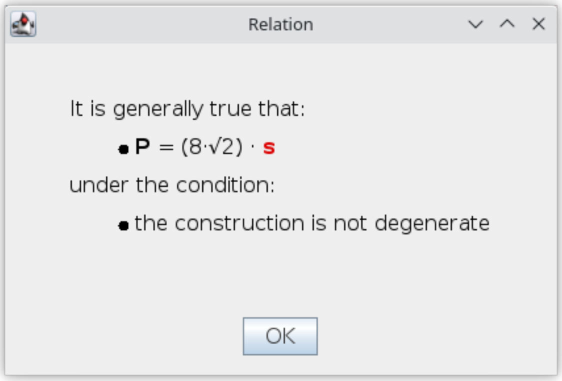
In this section we list four additional problems that can be solved with GeoGebra Discovery, assuming some effort. In fact, some other Náboj 2023 problems can be supported as well, but they may require some additional steps. See the next section for more details.

4.1 Pentominos (Problem 15)

The problem setting can be seen in Fig. 4.

At a first look, it seems complicated to draw a figure that describes the problem setting adequately. Some attempts may lead to Fig. 5: lines $g_1 = EK$ and (after extending the large pentomine with square $DINF$) $l_1 = HN$ help finding point O. Then, segment $m_1 = EO$ will be one side of the small X-pentomino, and it will be possible to compare it to one of the sides of the large pentomino. For more details about the construction steps have a look at Table 2.

Now, asking the relation between m_1 and f we obtain that $f = \frac{1}{2} \cdot \sqrt{10} \cdot m_1$. Even if GeoGebra Discovery cannot compare areas symbolically in a direct way, we can still conclude that the ratio of the



It is generally true that:

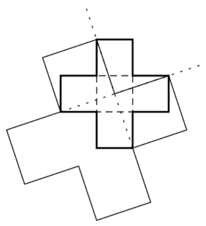
- $P = (8\cdot\sqrt{2})\cdot s$

under the condition:

- the construction is not degenerate

Figure 3: Report of a symbolic analysis of Problem 6 in GeoGebra Discovery

Problem 15. Antonia drew a small X-pentomino made of 5 congruent squares. Then she drew two perpendicular diagonals of this pentomino with dotted lines. Finally she constructed a bigger X-pentomino with some of the sides lying on the diagonals of the small pentomino as in the figure. Find the ratio of the area of Antonia's big pentomino to the area of the small one.



Result. 5 : 2.

Figure 4: Problem setting 15

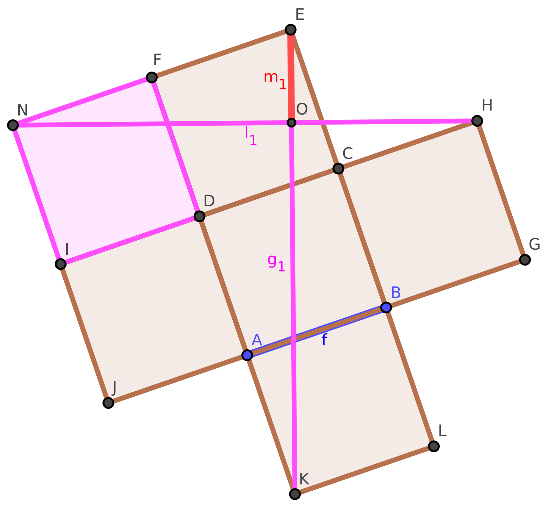
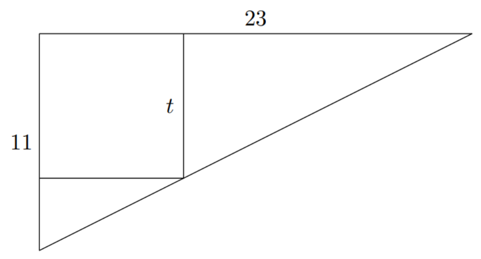


Figure 5: Problem setting 15 in GeoGebra Discovery

Problem 23. The legs of a right-angled triangle have lengths 11 and 23. A square of side length t has two of its sides lying on the legs of the triangle and one vertex on its hypotenuse as in the picture. Find t.



Result. $\frac{253}{34}$

Figure 6: Problem setting 23

areas must be

$$f^2 : {m_1}^2 = 10 : 4 = 5 : 2.$$

We remark here that a sophisticated way to do the construction may give a quicker result than the official solution. It may be, however, not trivial to find this alternative solution.

4.2 A Right Triangle (Problem 23)

The problem setting can be seen in Fig. 6.

We use some recent features of GeoGebra Discovery to solve this problem. Most importantly, a square is created based on free points $A = (0,0)$ and $B = (1,0)$. They must not be defined with the help of any axes, because in that case the background proof will fail and no output will be obtained. Here, instead of copying the initial square several times, we use the Dilate tool to stretch the segment AB and AC to get B' and C' accordingly. Another trick is to create the diagonal $j = AD$ of the initial square. Now the intersection E of $k = B'C'$ and j is the searched point. Finally, projecting E on $n = AB'$ and obtaining intersection point F of perpendicular l and line n, comparison of $m = EF$ and $f = AB$ is to be done. And, indeed $m = 253/34$, as expected. (See the sketch in Fig. 7 in GeoGebra Discovery.)

Here we remark that finding the rational value $253/34$ (it is approximately 7.44) seems very difficult unless one does not solve the problem explicitly (as shown in the official solution, by using an equation). If a user has some routine in GeoGebra, sketching the problem may take a shorter time than finding the required equation (even if it is a linear one). The construction steps can be found in Table 3.

4.3 A Triangle and a Circle (Problem 47)

Problem 47 was not even accessible during the contest for most of the teams because it was almost the last problem in the list and they were not that fast.

The problem setting can be seen in Fig. 8.

The official solution of Problem 47 required some non-trivial ideas. When using GeoGebra Discovery, we may face the question how the problem setting can be constructed, which is shown on Table 4. Since the lengths $AD = 8$ and $BD = 3$ are given, it seems reasonable to create AB arbitrarily and create D by using the command `Dilate(B, 8/11, A)`. Now, we create another point A' in a similar way, by placing A' on AB and letting $AA' = 7$. This helps us restricting the position of O because it must be

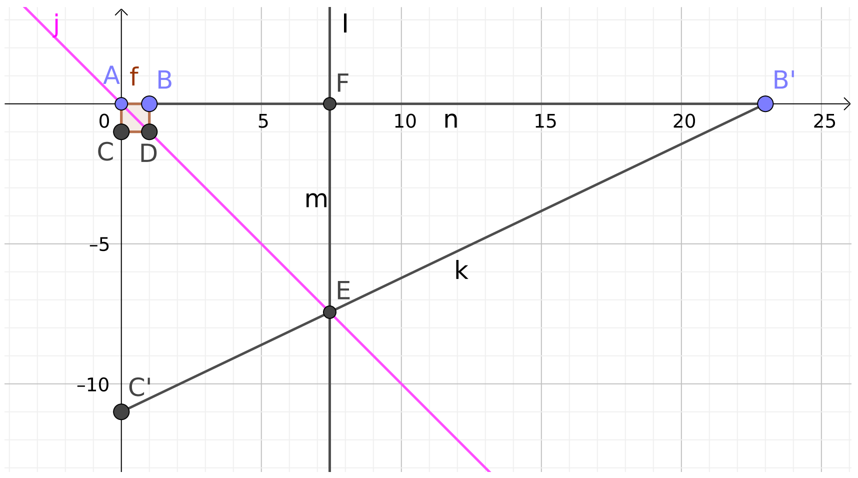
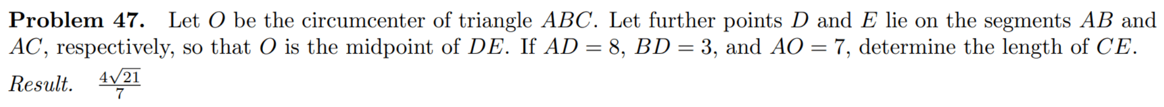


Figure 7: Sketch for Problem 23



Problem 47. Let O be the circumcenter of triangle ABC. Let further points D and E lie on the segments AB and AC, respectively, so that O is the midpoint of DE. If $AD = 8$, $BD = 3$, and $AO = 7$, determine the length of CE.

Result. $\frac{4\sqrt{21}}{7}$

Figure 8: Problem setting 47

on the circumcircle of the circle c with center A and radius AA'. On the other hand, O must lie on the perpendicular bisector g of AB. At this point we already know the position of $O = c \cap g$. (In fact, there may be two solutions here, but they are identical in the sense of symmetry.)

Now, by reflecting D about O we obtain E. By having E, we already know the line BC. To get the point C we only have to intersect this line with the circumcircle c. (Again, there are two solutions, but the other one C' leads to a degenerate case because it yields $A = C'$. To force getting the non-degenerate case we need to click near the intersection point with the mouse. Otherwise GeoGebra Discovery will compute with *both* cases at the same time.)

As a final step, we designate the unit length. Luckily, DA' is exactly 1. So we just have to compare $j = AE$ and $i = DA'$. As expected, the result is $j = 4/7 \cdot \sqrt{21} \cdot i$. Thus, $CE = j = \frac{4\sqrt{21}}{7}$.

The sketch can be seen in Fig. 9.

5 Problems that Require Further Improvements

In this section we take an overview of other examples that cannot be fully solved in simple steps in GeoGebra Discovery. Some hints may be, however, obtained. Instead of getting such hints, we summarize how the software tool could be extended to be able to give full solutions for such problems.

A first set of problems (4, 25 and 58, see Fig. 10, 12 and 16) are related with angles, another set (Problems 18, 30, 34, 43 and 58, see Fig. 11, 13, 14, 15 and 16) is about areas. Angle support (via symbolic computation) is very poor in GeoGebra Discovery: this has roots in a non-bijective relationship between angles and their algebraic counterparts. Area support is also somewhat minimal, because it is restricted to triangles, and the expected way of use is not polished yet in the software tool.

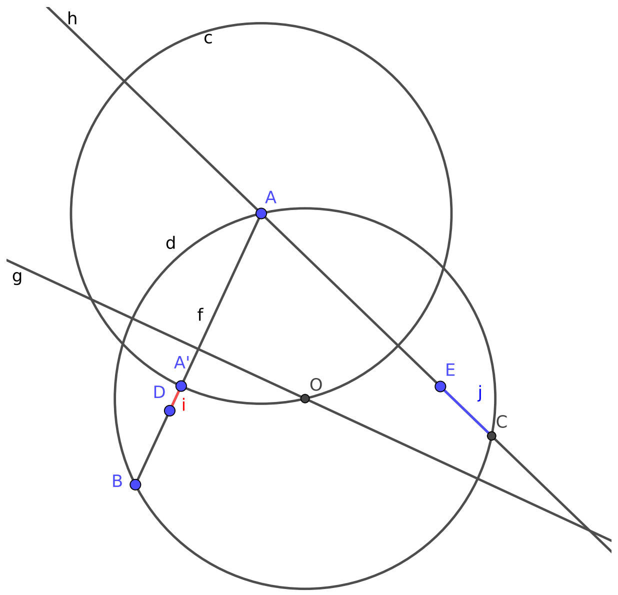


Figure 9: Sketch for Problem 47

Problem 4. A square and a regular pentagon are as in the picture below. Find the angle α in degrees.

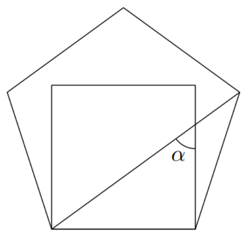


Figure 10: Problem 4

Problem 18. A rectangle with sides of length 3 and 4 is inscribed into a circle. Moreover, four half-circles are glued to its sides from outside as in the picture. What is the area of the shaded region, which consists of points of the half-circles not lying inside the circle?

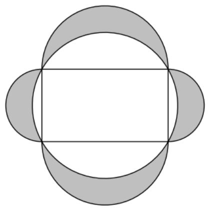


Figure 11: Problem 18

Problem 25. Consider a semi-circle with centre C and diameter AB. A point P on AB satisfies the following. A laser beam leaves P in a direction perpendicular to AB, bounces off the semicircle at points D and E following the rule of reflection, that is, $\angle PDC = \angle EDC$ and $\angle DEC = \angle BEC$, and then it hits the point B. Determine $\angle DCP$ in degrees.

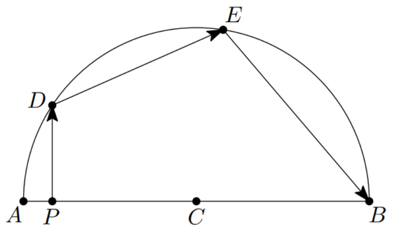


Figure 12: Problem 25

Problem 30. Giuseppe bought an ice cream. It had the shape of a ball of radius 4 cm in an ice cream cone. Giuseppe noticed that the ice cream ball fitted in the cone in the following way: The centre of the ball was precisely 2 cm above the base of the cone and the cone surface ended exactly where it touched the ball tangentially. What was the volume of the cone?

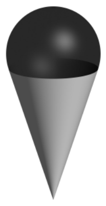


Figure 13: Problem 30

Problem 34. Given a triangle ABC of area 1, extend its sides BC, CA, AB to points D, E, and F respectively, as in the figure, so that $BD = 2BC$, $CE = 3CA$ and $AF = 4AB$. Find the area of the triangle DEF.

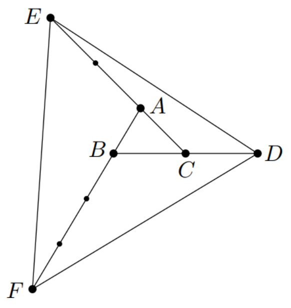
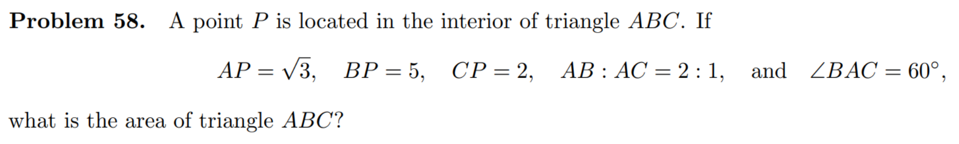


Figure 14: Problem 34

Problem 43. The medians of triangle ABC dissect it into six sub-triangles. The centroids of these sub-triangles are vertices of hexagon $DEFGHI$. Find the area ratio between the hexagon $DEFGHI$ and the triangle ABC.

Figure 15: Problem 43


Problem 58. A point P is located in the interior of triangle ABC. If

$$AP = \sqrt{3}, \quad BP = 5, \quad CP = 2, \quad AB : AC = 2 : 1, \quad \text{and} \quad \angle BAC = 60°,$$

what is the area of triangle ABC?

Figure 16: Problem 58

One can find that Problems 25 and 58 have some common roots. They can be formulated with "implicit assumptions".

We have a closer look at Problem 25. Let point F be the next bounce, when we assume that a laser beam starts from point P. Now, a command like `LocusEquation(`F`==`B`,`$\angle DCP$`)` could address the question (but having angles in the second parameter is not implemented). In fact, we may already get the exact position of P when applying consecutive reflections. According to Fig. 17, when reflecting P about CD, and intersecting the line connecting D and the mirror image P' with the semicircle, we can obtain E. Another reflection can yield P'' and the final visualization can be achieved with `LocusEquation(AreCollinear(`E`,`P''`,`B`),`P`)`. Since GeoGebra Discovery shows 5 isolated points, one can conjecture that there is something to do with a regular pentagon. Here, unfortunately, the factorization of the obtained polynomial does not help, because the interesting quadratic numbers are appearing just approximately. A deeper symbolic study shows that (by assuming $A = (0,0)$ and $B = (1,0)$) for the x-coordinate of P is one of the roots of the polynomial $64x^5 - 128x^4 + 80x^3 - 17x^2 + x$, and they are

$$0, \frac{1}{16} \cdot (-2 \cdot \sqrt{5} + 6), \frac{1}{4}, \frac{1}{16} \cdot (2 \cdot \sqrt{5} + 6), 1,$$

and to these values belong the α values

$$0^{\mathrm{o}}, 36^{\mathrm{o}}, 60^{\mathrm{o}}, 288^{\mathrm{o}}, 360^{\mathrm{o}},$$

the latter two ones without real geometrical meaning. Finally we can conclude that $\alpha = 36^{\mathrm{o}}$, this is the only meaningful solution. But, all of this derivation requires some additional steps, GeoGebra Discovery alone does not bring a satisfactory final answer.

Finally, we show a *wrong* conjecture for Problem 58, based on GeoGebra Discovery. Like Problem 25, an implicit locus equation seems here helpful. Let us, first, create a regular triangle $AB'C$ with $A = (0,0)$, $B = (1,0)$, and reflect A about B' to get B. Clearly, these preparations are sufficient to ensure assumptions $AB : AC = 2 : 1$ and $\angle BAC = 60^{\mathrm{o}}$ (See Fig. 18). This dummy triangle has the area $\frac{\sqrt{3}}{2}$. Now, we create an arbitrary point P and connect it with points A, B and C, to get segments i, j and k, respectively. We create two locus equations with the commands `LocusEquation(`j/k`==5/2,`P`)` and `LocusEquation(`i/k`==sqrt(3)/2,`P`)`. Now, we want to find the correct position for P, so we consider the intersection of the two locus curves visually. After zooming in, we learn that for $P = (0.4739140532, 0.24828147621)$ we obtain $k = 0.618294458$ which seems to be close enough to the well known number $f = \frac{\sqrt{5}-1}{2}$. If so, the triangle must be enlarged by a factor $1/f \cdot 2$ which is twice the golden ratio, $2\varphi = \sqrt{5} + 1$. Finally, the triangle will have the area $\frac{\sqrt{3}}{2} \cdot (\sqrt{5} + 1) \approx 9.06913\ldots$

Assuming that the golden ratio plays a role here is, however, incorrect. The correct solution is $\frac{6+7\cdot\sqrt{3}}{2} \approx 9.06217\ldots$[5] Note that the ratio between the two values is $1.000767\ldots$ which is remarkably small difference.

[5]See `https://math.old.naboj.org/archive/problems/pdf/math/2023`$_{\mathrm{e}}$`n`$_{\mathrm{s}}$`ol.pdf` for a full computation.

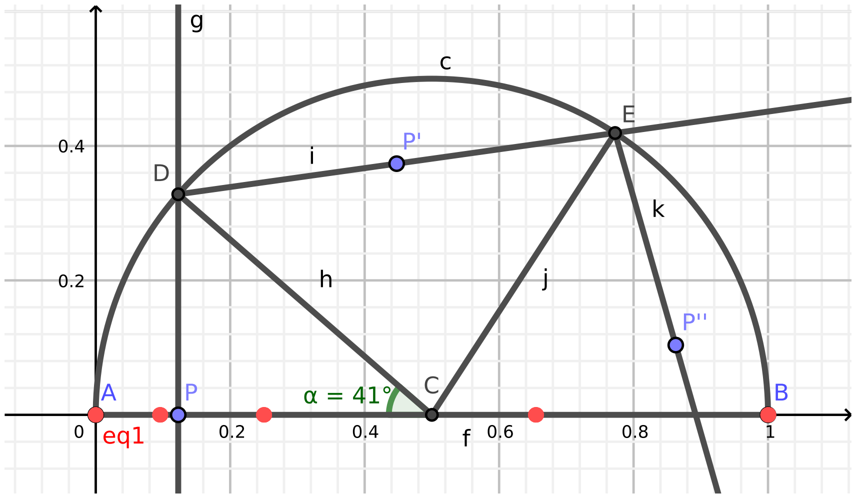


Figure 17: A possible approach to solve Problem 25

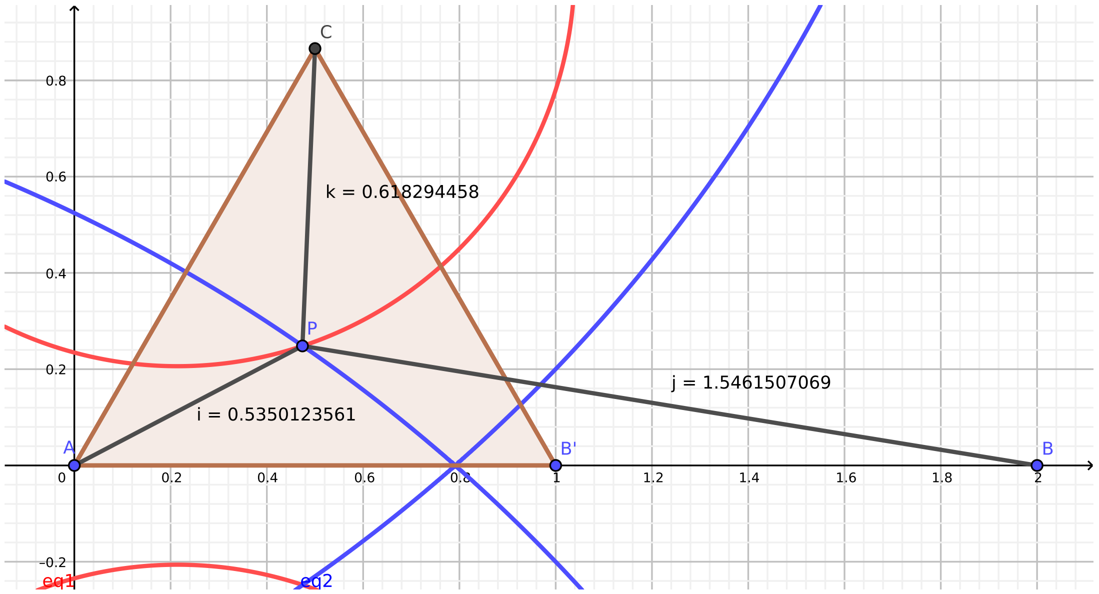


Figure 18: Obtaining an incorrect conjecture to solve Problem 58

6 Conclusion

The last section showed that GeoGebra Discovery can be a useful tool to get a correct conjecture if the right steps are taken to finish the solution, but it can also be misleading in some delicate situations. On the other hand, several contest problems can be handled and solved with minimal effort by using this tool. We need to admit that a good knowledge of the software is unavoidable. However, experienced users may need just a couple of steps to achieve the solution.

For a future improvement, full support of computing angles and areas seems to be a great step forward. Some sophisticated problems, however, may need further developments towards symbolic computations that are based on implicit assumptions.

7 Acknowledgements

The second author was partially supported by a grant PID2020-113192GB-I00 (Mathematical Visualization: Foundations, Algorithms and Applications) from the Spanish MICINN.

References

[1] Recio, T., Vélez, M.P.: Automatic discovery of theorems in elementary geometry. Journal of Automated Reasoning **23**, 63–82 (1999). doi:10.1023/A:1006135322108

[2] Vajda, R., Kovács, Z.: GeoGebra and the *realgeom* reasoning tool. In: Fontaine, P., Korovin, K., Kotsireas, I.S., Rümmer, P., Tourret, S. (eds.) PAAR+SC-Square 2020. Workshop on Practical Aspects of Automated Reasoning and Satisfiability Checking and Symbolic Computation Workshop 2020. pp. 204–219 (6 2020), `https://ceur-ws.org/Vol-2752/paper15.pdf`

8 Appendix

In this Appendix, we provide the construction protocols in GeoGebra Discovery for the problems presented above.

Table 1: Construction protocol in GeoGebra Discovery for Problem 6

No.	Name	Toolbar Icon	Description
1	Polygon poly1	⬠	Polygon(A, B, 4)
2	Point E	⋰	Midpoint of A, C
3	Point E'	⋰	E mirrored at A
4	Point E'_1	⋰	E mirrored at B
5	Polygon poly2	⬠	Polygon(E', E'_1, 4)
6	Point E'_2	⋰	E mirrored at E'
7	Point E'_3	⋰	E mirrored at E'_1
8	Polygon poly3	⬠	Polygon(E'_2, E'_3, 4)
9	Segment t	⟋	Segment E'_2, E'_3
10	Segment s	⟋	Segment A, C

Table 1: Construction protocol in GeoGebra Discovery for Problem 6

No.	Name	Toolbar Icon	Description
11	Number P		4t

Table 2: Construction protocol for Problem 15

No.	Name	Toolbar Icon	Description	Value
1	Polygon poly1		Polygon(A, B, 4)	$poly1 = 7.33$
2	Segment f		Segment A, B	$f = 2.71$
3	Polygon poly2		Polygon(D, C, 4)	$poly2 = 7.33$
4	Polygon poly3		Polygon(C, B, 4)	$poly3 = 7.33$
5	Polygon poly4		Polygon(A, D, 4)	$poly4 = 7.33$
6	Polygon poly5		Polygon(B, A, 4)	$poly5 = 7.33$
7	Segment g_1		Segment E, K	$g_1 = 8.56$
8	Polygon poly6		Polygon(I, D, 4)	$poly6 = 7.33$
9	Segment l_1		Segment N, H	$l_1 = 8.56$
10	Point O		Intersection of g_1 and l_1	$O = (1.18, 3.29)$
11	Segment m_1		Segment O, E	$m_1 = 1.71$

Table 3: Construction protocol for Problem 23

No.	Name	T. Icon	Description	Value
1	Point A			$A = (0, 0)$
2	Point B			$B = (1, 0)$
3	Point B’		B dilated by factor 23 from A	$B' = (23, 0)$
4	Segment n		Segment A, B’	$n = 23$
5	Polygon poly1		Polygon(B, A, 4)	$poly1 = 1$
6	Segment f		Segment B, A	$f = 1$
7	Line j		Line D, A	$j : -x - 1y = 0$
8	Point C’		C dilated by factor 11 from A	$C' = (0, -11)$
9	Segment k		Segment C’, B’	$k = 25.5$
10	Point E		Intersection of j and k	$E = (7.44, -7.44)$
11	Line l		Line through E perpendicular to n	$l : x = 7.44$
12	Point F		Intersection of l and n	$F = (7.44, 0)$

Table 3: Construction protocol for Problem 23

No.	Name	T. Icon	Description	Value
13	Segment m		Segment F, E	$m = 7.44$

Table 4: Construction protocol for Problem 47

No.	Name	T. Icon	Description	Value
1	Point B			$B = (-5.41, -5.8)$
2	Point A			$A = (-0.78, 4.18)$
3	Line g		Perpendicular Bisector of AB	$g : 4.63x + 9.98y = -22.4$
4	Point D		B dilated by factor 8/11 from A	$D = (-4.15, -3.08)$
5	Point A’		A dilated by factor 1/8 from D	$A' = (-3.73, -2.17)$
6	Circle c		Circle through A’ with center A	$c : (x + 0.78)^2 + (y - 4.18)^2 = 49.04$
7	Point O		Intersection of c and g	$O = (0.84, -2.63)$
8	Circle d		Circle through A with center O	$d : (x - 0.84)^2 + (y + 2.63)^2 = 49.04$
9	Point E		D mirrored at O	$E = (5.82, -2.19)$
10	Line h		Line A, E	$h : 6.37x + 6.6y = 22.63$
11	Segment f		Segment A, B	$f = 11$
12	Segment i		Segment D, A’	$i = 1$
13	Point C		Intersection of d and h	$C = (7.7, -4.01)$
14	Segment j		Segment C, E	$j = 2.62$

Using Java Geometry Expert as Guide in the Preparations for Math Contests

Ines Ganglmayr
The Private University College of Education of the Diocese of Linz, Austria
ines.ganglmayr@ph-linz.at

Zoltán Kovács
The Private University College of Education of the Diocese of Linz, Austria
zoltan.kovacs@ph-linz.at

We give an insight into Java Geometry Expert (JGEX) in use in a school context, focusing on the Austrian school system. JGEX can offer great support in some classroom situations, especially for solving mathematical competition tasks. Also, we discuss some limitations of the program.

1 Introduction

The use of technical media in Austrian mathematics lessons is largely limited to the GeoGebra medium. GeoGebra [5] proved to be a great tool to visualize and analyze classroom problems, but certain tasks like proving geometric facts rigorously by using a visual explanation is not supported by GeoGebra. As an alternative approach, we focus on introducing JGEX [6] as opposed to GeoGebra, specifically in the area of competition tasks.

Geometric proofs are no longer an important part of secondary school curriculum in Austria and many other countries. Formerly, however, Euclidean plane geometry and proving more complicated facts was a part of the expected knowledge of secondary level. There are, however, some initiatives, that call for rethinking school curriculum, by focusing on structured thinking again.

According to the Ministry of Education and Science in Portugal, for example, structured thinking is one of the main goals for teaching mathematics. It is also anchored in Austrian curricula that the qualitative development of tasks requires various dimensions of content, dimensions of action and dimensions of complexity, which hierarchically structure the understanding and learning of students. Analytically consistent thinking and the acquisition of mathematical skills are in the foreground. A central concern of mathematics is learning processes and techniques to acquire connections as well as insights and to solve problems [3].[1]

One method to achieve this can be done by examining properties of certain structures. To achieve this, a concept of hierarchical structures is developed, and a systematic investigation is attempted. This form of working out is part of hypothetical-deductive thinking, which can be equated with mathematical thinking. But beyond that, inductive reasoning is also part of mathematical understanding, since it enables assumptions and conclusions [2].

According to Duval, three cognitive processes are involved in learning geometry. These are visualization, construction and reasoning. Those can be activated separately but also together. They may or

[1]New version of the curriculum from Portugal: `https://eurydice.eacea.ec.europa.eu/national-education-systems/portugal/national-reforms-school-education`.

may not be interdependent. Indeed, Duval emphasizes that these three processes necessary for mathematical understanding and the mastery of geometry are closely related. JGEX can contribute to this and support children and young people in these three processes [4].

In the context of a university course organized for prospective mathematics teachers at the Private University College of Education of the Diocese of Linz during the winter semester of 2022, some preliminary work was conducted to assess whether JGEX could be used to aid in solving geometric proof problems. This university course, under the guidance of the second author, primarily emphasized solving a broad spectrum of contest problems. As we will later demonstrate, JGEX proved to be an outstanding software tool for conducting such experiments. Additionally, participants in this university course attended a preparatory session for the Mathematics Olympiad in February 2023. During this session, they had the opportunity to gain firsthand experience and interact with young learners and their instructors, enabling them to derive conclusions regarding the feasibility of preparing for contest problems without technological assistance.

On the preparation day for the Mathematics Olympiad at Johannes Kepler University of Linz, Austria (JKU), 70 young learners geared up for a mathematics competition. The teaching staff focused on prime numbers and the pigeonhole principle as part of their preparations. The day commenced with an introduction to the content, followed by a 90-minute group exercise session facilitated by students [1].

Even if the topic of prime numbers and the pigeonhole principle have not much to do with geometry, the tasks sometimes required a geometric overview and therefore structured thinking. The learners were sometimes overwhelmed with the processing of the geometric tasks. This observation was concluded by the participating prospective mathematics teachers, including the first author of this paper. A deficit in the three categories of visualization, construction and reasoning when solving the competition tasks with geometrical connections, was recognizable. The students often asked for help or verbally communicated their helplessness. While this is not yet covered with our research, it seems a plausible reason that structured thinking may require further support at all levels of school mathematics education.

In this paper and the related research, we assumed that JGEX can offer support in improving structured thinking. This geometry program may offer some missing steps in the visualization. Also, it can lead to constructions and above all it may provide various possibilities for arguments. Java Geometry Expert often guides students to solve more complex problems.

When solving competition tasks, the focus is on recognizing connections and grouping arguments for solutions. For this reason, JGEX is ideal in combination with competitive tasks.

We highlight that this paper is just a first report on our experiments, and it requires further research. However, we think that the first experiences are already promising.

2 Testing of Competition Tasks Using JGEX

JGEX is a complex software system that supports geometric proofs on "equational" properties of a planar figure. This means that ratios of quantities (like lengths of segments or size of angles) can be proven. Also, perpendicularity, parallelism, collinearity, concyclicity and related properties can be checked and proved. This limitation arises from the methods employed within JGEX, which encompass polynomial techniques such as Wu's method, the Gröbner basis method, as well as the Geometry Deductive Database method. Consequently, properties involving inequalities are not accommodated.

We show two examples. Below is a competition task that can be solved using JGEX and another task that is "equational" but JGEX does not provide any useful information on how to solve it. The tasks are preparatory materials for the mathematical competition in the field of geometry, collected by teacher

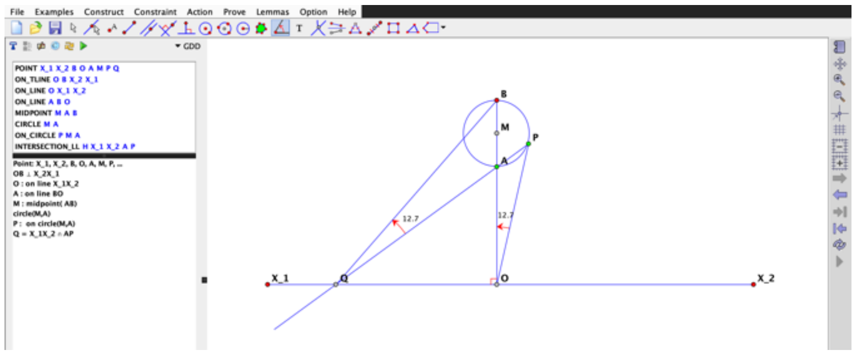
Ralf Roupec from the Bundesrealgymnasium Freistadt, a well-known expert and team member for the preparations for math contests in Upper Austria.

2.1 A Solvable Problem

We consider a Problem of BAMO (Bay Area Mathematical Olympiad[2]) 1999:

> Set $O = (0,0)$, $A = (0,a)$, $B = (0,b)$, where $0 < a < b$. Let k be the circle with diameter AB, and let P be an arbitrary point on k. The line PA intersects the x-axis at point Q.
>
> Show that $\angle BQP = \angle BOP$.

A solution via JGEX. Here JGEX supports step-by-step solving through important properties that lead to the proof process. By collecting the various insights, students are supported in solving the problem. See figures 1–5.

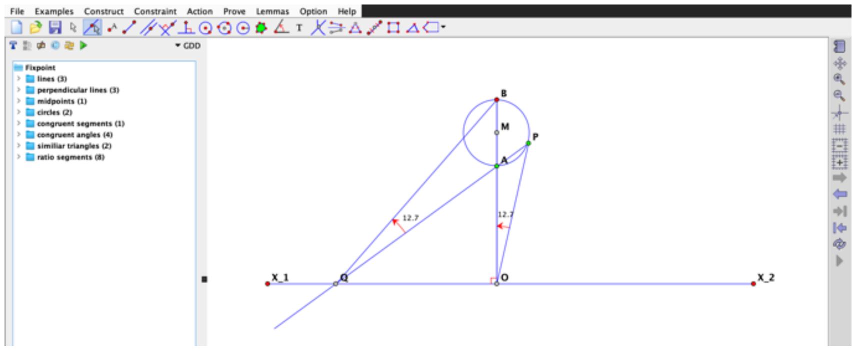


```
POINT X_1 X_2 B O A M P Q
ON_TLINE O B X_2 X_1
ON_LINE O X_1 X_2
ON_LINE A B O
MIDPOINT M A B
CIRCLE M A
ON_CIRCLE P M A
INTERSECTION_LL H X_1 X_2 A P
```

```
Point: X_1, X_2, B, O, A, M, P, ...
OB ⊥ X_2X_1
O : on line X_1X_2
A : on line BO
M : midpoint( AB)
circle(M,A)
P :  on circle(M,A)
Q = X_1X_2 ∩ AP
```

Draw the construction and then press *Fix* (below).

Figure 1: A step-by-step solution provided by JGEX, step 1



```
Fixpoint
  lines (3)
  perpendicular lines (3)
  midpoints (1)
  circles (2)
  congruent segments (1)
  congruent angles (4)
  similiar triangles (2)
  ratio segments (8)
```

Open the folder: *Congruent angles*.

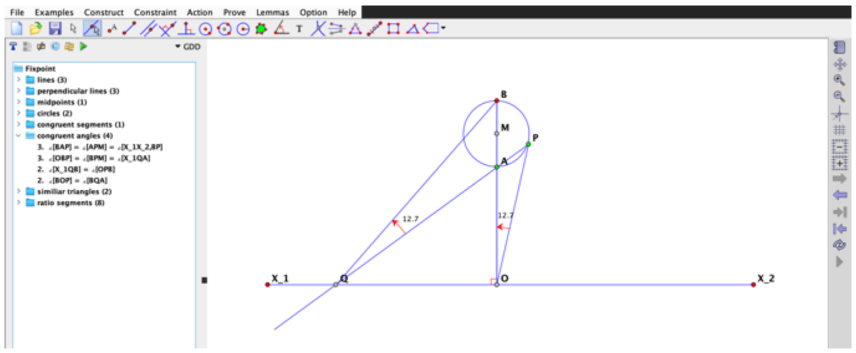
Figure 2: A step-by-step solution provided by JGEX, step 2

When performing further observations, JGEX can provide detailed information. See figures 6–8.

2.2 A Problem that is Difficult to Solve Automatically

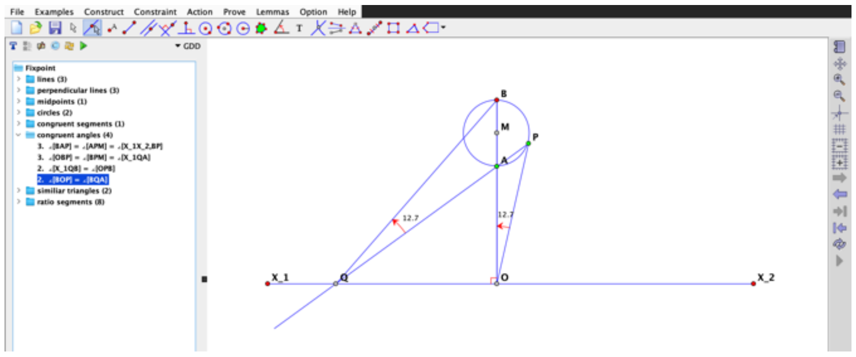
We consider another problem from Roupec's collection:

[2]The web page of the Bay Area Mathematical Olympiad is at `https://www.bamo.org`.



Fixpoint
- lines (3)
- perpendicular lines (3)
- midpoints (1)
- circles (2)
- congruent segments (1)
- congruent angles (4)
  - 3. ∠[BAP] = ∠[APM] = ∠[X_1X_2,BP]
  - 3. ∠[OBP] = ∠[BPM] = ∠[X_1QA]
  - 2. ∠[X_1QB] = ∠[OPB]
  - 2. ∠[BOP] = ∠[BQA]
- similiar triangles (2)
- ratio segments (8)

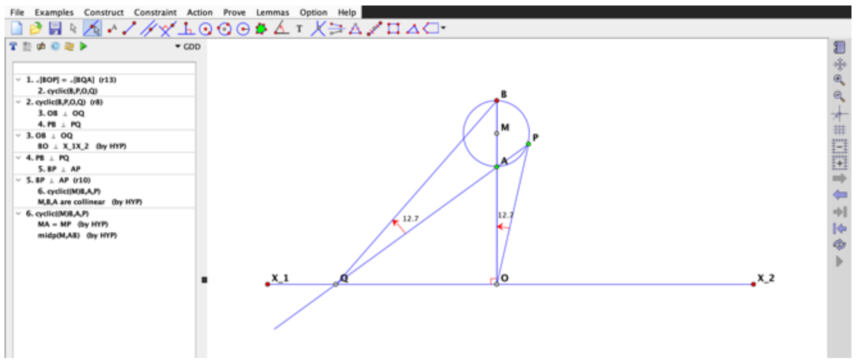
Figure 3: A step-by-step solution provided by JGEX, step 3



Fixpoint
- lines (3)
- perpendicular lines (3)
- midpoints (1)
- circles (2)
- congruent segments (1)
- congruent angles (4)
  - 3. ∠[BAP] = ∠[APM] = ∠[X_1X_2,BP]
  - 3. ∠[OBP] = ∠[BPM] = ∠[X_1QA]
  - 2. ∠[X_1QB] = ∠[OPB]
  - 2. ∠[BOP] = ∠[BQA]
- similiar triangles (2)
- ratio segments (8)

Select $\angle[BQP] = \angle[BQA]$ and then right click *Prove*.

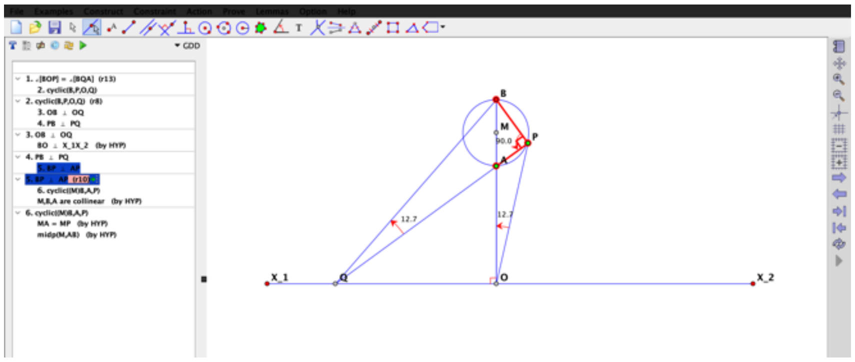
Figure 4: A step-by-step solution provided by JGEX, step 4



```
1. ∠[BOP] = ∠[BQA]  (r13)
    2. cyclic(B,P,O,Q)
2. cyclic(B,P,O,Q)  (r8)
    3. OB ⊥ OQ
    4. PB ⊥ PQ
3. OB ⊥ OQ
    BO ⊥ X_1X_2   (by HYP)
4. PB ⊥ PQ
    5. BP ⊥ AP
5. BP ⊥ AP  (r10)
    6. cyclic((M)B,A,P)
    M,B,A are collinear   (by HYP)
6. cyclic((M)B,A,P)
    MA = MP   (by HYP)
    midp(M,AB)   (by HYP)
```

The steps of proof are shown.

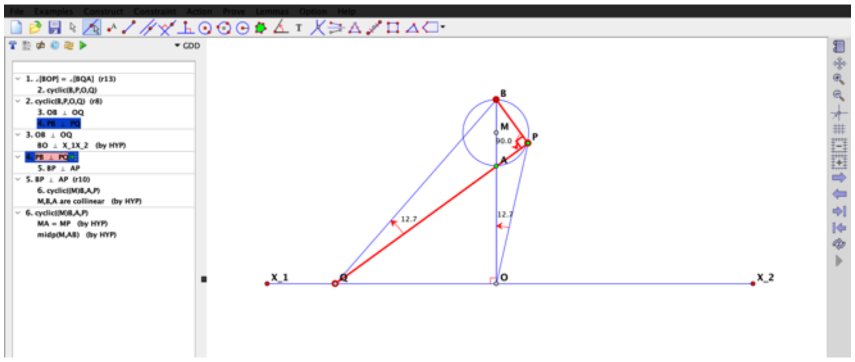
Figure 5: A step-by-step solution provided by JGEX, step 5


File Examples Construct Constraint Action Prove Lemmas Option Help
GDD
B
M
90.0
P
A
12.7
12.7
X_1
Q
O
X_2

```
1. ∠[BOP] = ∠[BQA]  (r13)
    2. cyclic(B,P,O,Q)
2. cyclic(B,P,O,Q)  (r8)
    3. OB ⊥ OQ
    4. PB ⊥ PQ
3. OB ⊥ OQ
    BO ⊥ X_1X_2   (by HYP)
4. PB ⊥ PQ
    5. BP ⊥ AP
5. BP ⊥ AP  (r10)
    6. cyclic((M)B,A,P)
    M,B,A are collinear   (by HYP)
6. cyclic((M)B,A,P)
    MA = MP   (by HYP)
    midp(M,AB)   (by HYP)
```

Observation: Right angle in triangle *BPA*.

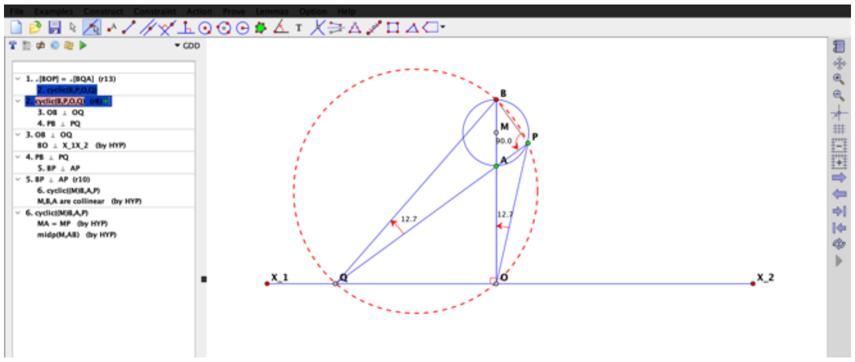
Figure 6: Detailed information provided by JGEX, step 1


File Examples Construct Constraint Action Prove Lemmas Option Help
GDD
B
M
90.0
P
A
12.7
12.7
X_1
Q
O
X_2

```
1. ∠[BOP] = ∠[BQA]  (r13)
    2. cyclic(B,P,O,Q)
2. cyclic(B,P,O,Q)  (r8)
    3. OB ⊥ OQ
    4. PB ⊥ PQ
3. OB ⊥ OQ
    BO ⊥ X_1X_2   (by HYP)
4. PB ⊥ PQ
    5. BP ⊥ AP
5. BP ⊥ AP  (r10)
    6. cyclic((M)B,A,P)
    M,B,A are collinear   (by HYP)
6. cyclic((M)B,A,P)
    MA = MP   (by HYP)
    midp(M,AB)   (by HYP)
```

Observation: persistence of the right angle. (Right angle also in triangle *BQP*.)

Figure 7: Detailed information provided by JGEX, step 2


File Examples Construct Constraint Action Prove Lemmas Option Help
GDD
B
M
90.0
P
A
12.7
12.7
X_1
Q
O
X_2

```
1. ∠[BOP] = ∠[BQA]  (r13)
    2. cyclic(B,P,O,Q)
2. cyclic(B,P,O,Q)  (r8)
    3. OB ⊥ OQ
    4. PB ⊥ PQ
3. OB ⊥ OQ
    BO ⊥ X_1X_2   (by HYP)
4. PB ⊥ PQ
    5. BP ⊥ AP
5. BP ⊥ AP  (r10)
    6. cyclic((M)B,A,P)
    M,B,A are collinear   (by HYP)
6. cyclic((M)B,A,P)
    MA = MP   (by HYP)
    midp(M,AB)   (by HYP)
```

Observation: Circle through the points *B*, *P*, *Q*, *O*.
Proof using the peripheral angle theorem:
The red circle shows the existence of the peripheral angle theorem.
This is made possible by the chord *BP* and concludes that the $\angle BQP = \angle BOP$.

Figure 8: Detailed information provided by JGEX, step 3

Let ABC be an acute triangle with circumcircle k. Let be X the midpoint of the arc BC, that does not contain A. The points Y and Z are defined analogously.

Show that the orthocenter of XYZ is center of the incircle of ABC.

See figure 9 for a graphical explanation.

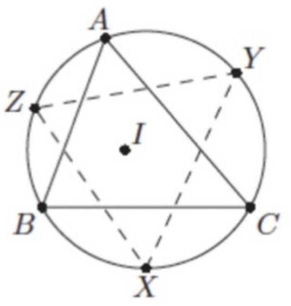


Figure 9: A sketch of problem setting.

An attempt for getting the solution via JGEX. Figure 10 shows that, after creating the figure in JGEX, there is a list provided by some fixed properties of the figure. There is, however, no automation provided to detect the property asked by the problem setting. One can try, however, other ways to formulate the problem, but an intuitive and quick way to obtain the required property (and its proof) does not seem to be accessible.

Here we highlight that JGEX provides several ways to construct the figure, we chose an intuitive one among the possible methods. For young learners, however, it may be challenging or even impossible to find the required formulation to achieve the expected property.

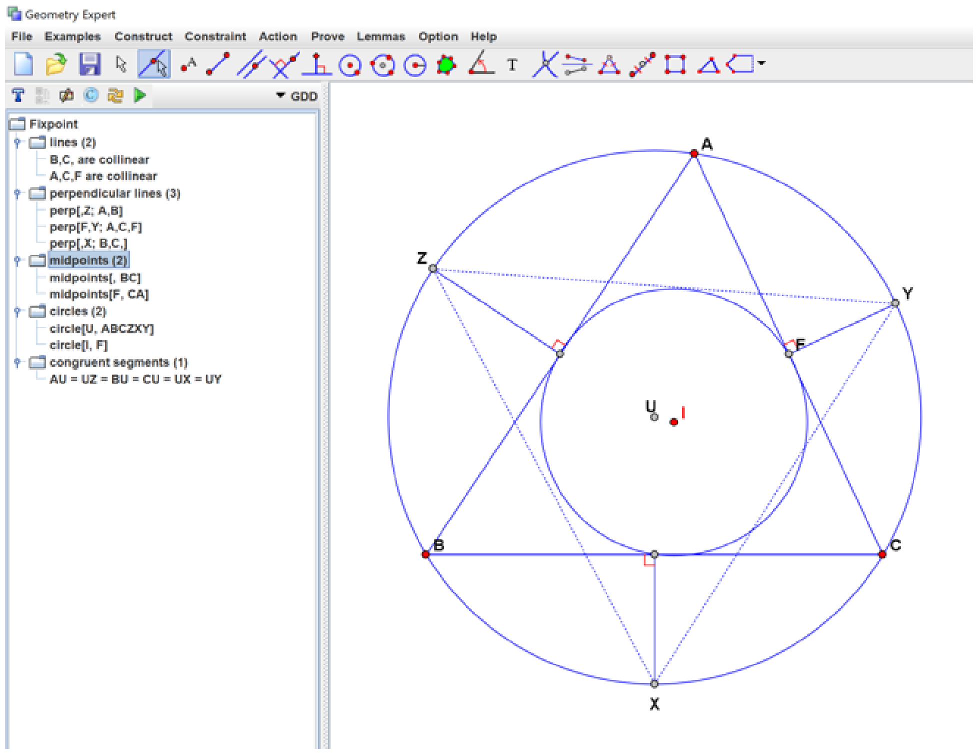
3 Conclusion

JGEX comes with a sophisticated user interface, however, the lack of support of languages (currently English, Chinese, German, Portuguese, Persian and Serbian are supported) and some other difficulties of its intuitive usage may make JGEX challenging to be used in classroom situations out-of-the-box. Among others, beginners may find difficult to look for the fixed properties of the given construction.

In our discussions during the university course activities, it turned out that JGEX can be indeed useful for the introduction of geometric topics in the classroom, including demonstrating the steps of visual proofs. After some technical introduction, students could construct simple figures and show connections with a mouse click.

Even though, it remains a challenging task to formulate the problem setting in a way that JGEX can provide a step-by-step proof. In our presentation we will show further examples in which JGEX can be of help, and in which it cannot provide a proof. For young learners, avoiding such limitations seems to be crucial for the everyday use.

An outlook for further research activities would be further work into how this program could be used in regular classes in a compliant manner.


Geometry Expert
File Examples Construct Constraint Action Prove Lemmas Option Help
GDD
A
Z
Y
F
U
I
B
C
X

```
Fixpoint
  lines (2)
    B,C, are collinear
    A,C,F are collinear
  perpendicular lines (3)
    perp[,Z; A,B]
    perp[F,Y; A,C,F]
    perp[,X; B,C,]
  midpoints (2)
    midpoints[, BC]
    midpoints[F, CA]
  circles (2)
    circle[U, ABCZXY]
    circle[I, F]
  congruent segments (1)
    AU = UZ = BU = CU = UX = UY
```

Figure 10: An attempt to collect some fixed properties in JGEX.

4 Acknowledgements

The second author was partially supported by a grant PID2020-113192GB-I00 (Mathematical Visualization: Foundations, Algorithms and Applications) from the Spanish MICINN.

References

[1] F. Bitter & F. Baksa (2023): *Rechnen macht Spaß: Vorbereitungstag für die Mathematik-Olympiade*. `https://www.jku.at/news-events/news/detail/news/rechnen-macht-spass-vorbereitungstag-fuer-die-mathematik-olympiade`.

[2] Bundesministerium für Finanzen (2023): *Bundesrecht konsolidiert: Gesamte Rechtsvorschrift für Lehrpläne – allgemeinbildende höhere Schulen, Fassung vom 08.06.2023*. `https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10008568`.

[3] E. Básico (2013): *Programa e Metas Curriculares Matemática*. `https://www.dge.mec.pt/sites/default/files/Basico/Metas/Matematica/programa_matematica_basico.pdf`.

[4] R. Duval (1998): *Geometry from a cognitive point of view*. In C. Mammana & V. Villani, editors: *Perspectives on the Teaching of Geometry for the 21st Century*, Kluwer Academic Publishers, Dordrecht, pp. 37–52, doi:10.1007/978-94-011-5226-6_3.

[5] M. Hohenwarter (2002): *GeoGebra – ein Softwaresystem für dynamische Geometrie und Algebra der Ebene*.

[6] Z. Ye, S.C. Chou & X.S Gao (2011): *An introduction to Java Geometry Expert*. In: *Automated Deduction in Geometry, 7th International Workshop, ADG 2008, Shanghai, China, September 22-24, 2008, Revised*

Papers, Lecture Notes in Computer Science. Volume 6301, Springer-Verlag, p. 189–195, doi:10.1007/978-3-642-21046-4_10.

The Locus Story of a Rocking Camel in a Medical Center in the City of Freistadt

Anna Käferböck
The Private University College of Education of the Diocese of Linz, Austria
anna.kaeferboeck@gmx.at

Zoltán Kovács
The Private University College of Education of the Diocese of Linz, Austria
zoltan.kovacs@ph-linz.at

We give an example of automated geometry reasoning for an imaginary classroom project by using the free software package *GeoGebra Discovery*. The project is motivated by a publicly available toy, a rocking camel, installed at a medical center in Upper Austria. We explain how the process of a false conjecture, experimenting, modeling, a precise mathematical setup, and then a proof by automated reasoning could help extend mathematical knowledge at secondary school level and above.

1 Introduction

Automated reasoning in geometry is available in various software tools for several years, mostly in prover packages. In this paper we pay our attention to a non-trivial presence of a geometry prover in the software tool GeoGebra Discovery [4, 5] that aims at reaching secondary schools with its intuitive user interface.

Most importantly, we give a report on a STEM/STEAM project that was discussed in a group of prospective mathematics teachers at the Private University College of Education of the Diocese of Linz in Upper Austria during the winter semester 2022/23, in the frame of a course that focuses on exploiting technology in mathematics education (36 students in 2 working groups). This project consisted of several other experiments that were already communicated by the second author. The discussed activity, a detailed study of the movement of a rocking camel, is however, completely new. Also, some major improvements in the underlying software tool (implemented by the second author with a substantial help of the students' feedback), makes it much easier to model similar project setups and conclude mathematical knowledge in an automated way.

2 GeoGebra Discovery and its Automated Reasoning Tools

GeoGebra Discovery is a fork of *GeoGebra*,[1][2] a de facto standard tool that supports mathematics education at various levels of learners. GeoGebra 5.0 and above come with a built-in automated reasoning subsystem. The supported commands: *Prove*, *ProveDetails*, *LocusEquation* and *Envelope* are further developed in GeoGebra Discovery by an addition of various other commands like *Discover*, *Compare* and *RealQuantifierElimination*. Also, several improvements of the existing commands are included.

[1]GeoGebra is an interactive geometry, algebra, statistics and calculus application, intended for learning and teaching mathematics and science from primary school to university level, available at `https://geogebra.org`.

[2]GeoGebra Discovery is available at `https://kovzol.github.io/geogebra-discovery`.

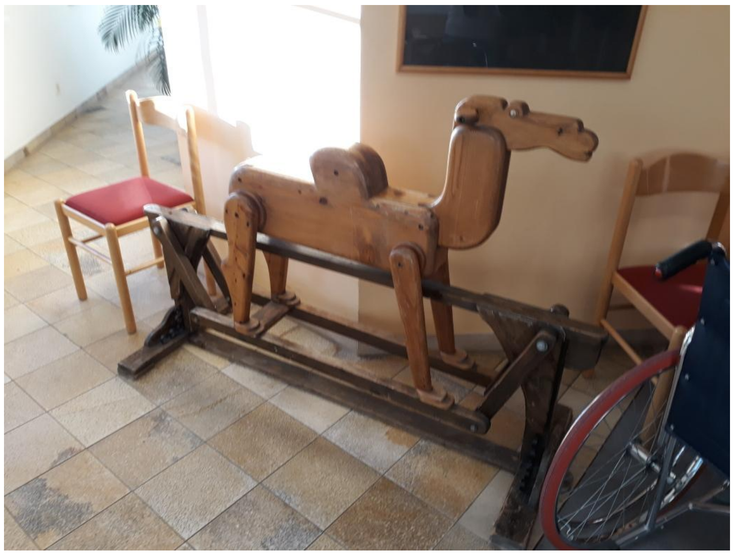


Figure 1: The exhibited toy in the medical center of the city of Freistadt, Upper Austria.

Most importantly, we focus on the symbolic support of the *Dilate* command and tool that was added in GeoGebra Discovery version 2023Feb01.[3] It can be used to dilate an object from a point (which is the dilation center point), using a given factor, a rational number. This makes it easy to divide a line segment in a given ratio. Formerly, for such constructions the intercept theorem, or a consecutive use of midpoints, reflections or rotations had to be used. As a further result, the user can use the slider feature of GeoGebra (which is a numerical tool) and at the same time precise discovery and proofs (which are symbolic tools) can be automatically obtained now.

Our paper argues for the possible classroom use of GeoGebra Discovery on the one hand, and also for activities that combine real-life applications and automated geometry reasoning.

3 A Rocking Camel

The toy shown in figure 1 is exhibited in a medical center in Freistadt, Upper Austria. It is installed for amusement purposes for children who are waiting for medical treatments. An obvious question is, from the mathematical point of view, to identify the movement of certain points of the camel. Clearly, some points of the camel move on circular paths. For example, the suspension points, close to the legs, move along a circle. When asking for a general point of the camel, however, some non-trivial movements can show up. For example, the movement of the hump of the camel seems to move on an elliptical path, after the first experiments are performed by using a former version of the GeoGebra applet at `https://geogebra.org/m/b8mbjxcz` (Fig. 2).

Here we can also note that points E and H (they are the above-mentioned suspension points close to the legs of the camel) indeed move on a circular path. In fact, here the lengths of the quadrilateral $ABHE$ are given with the segments f and g, and point M was constructed by consecutive use of midpoints and a rotation by 90 degrees. At a later point of our paper we will use a better approach, based on the improvements on the Dilate command.

[3]See `https://github.com/kovzol/geogebra/releases/tag/v5.0.641.0-2023Feb01`.

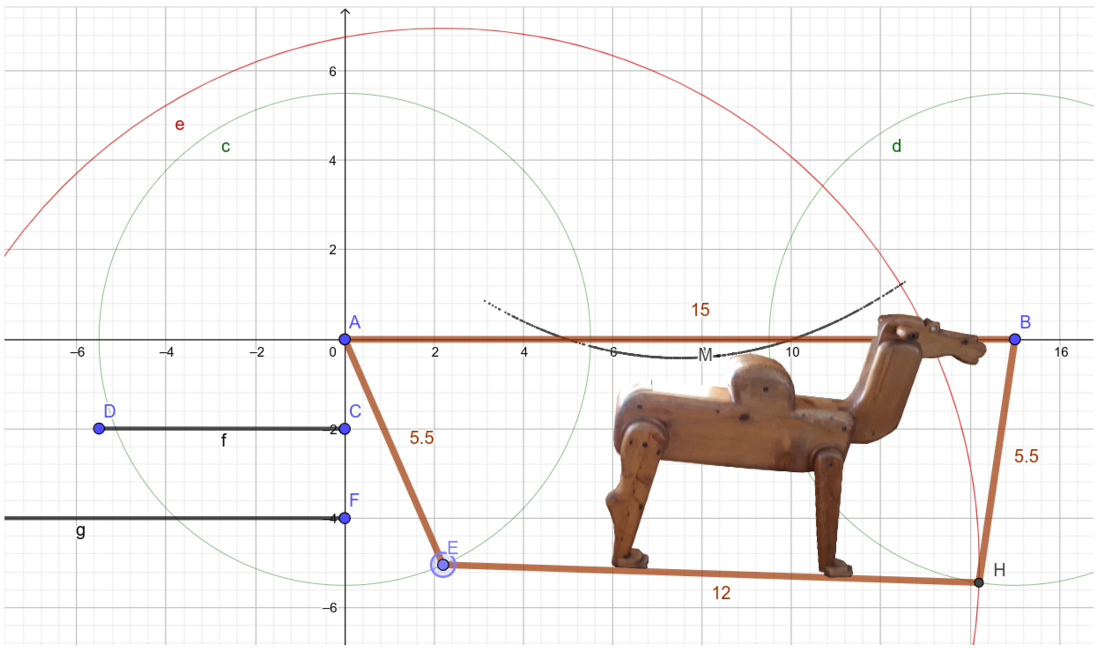


Figure 2: A former version of a GeoGebra applet that suggests that the motion of the hump of the camel is a part of an ellipse.

3.1 The History of the Rocking Camel

As a part of our project we researched after on how the camel got to the medical center. In fact, the camel was in the attic for many decades and no one knew what it was all about. It was probably passed down from generation to generation. Unfortunately, this much is known about it.

How many more toys are there in the attic that have a nice mathematical background but we have forgotten about them? How many forgotten mathematical books, writings and ideas are there that the modern age has put aside and not even superficially exploited their excitement?

We believe that our contribution will help dust off these forgotten gems and put them at the service of education today.

4 A STEM Activity

The abbreviation STEM stands for “science, technology, engineering and mathematics” [8]. It is a popular approach to teach mathematics via real-life applications. Sometimes STEM is extended with an “A” (“arts”) and it becomes the abbreviation “STEAM”. Later we will learn how this engineering experiment can be extended to an artistic activity.

Now, the main question of the activity, raised to the prospective mathematics students, was to describe the movement of the hump of the camel by following the steps below:

1. Make an exact measurement of the toy and its parts. (As a first approach, this was prepared by a student by providing photos. Later, with another student, more exact data was collected in the medical center, by using measuring tapes, a camera and some graphical analysis with technology.)

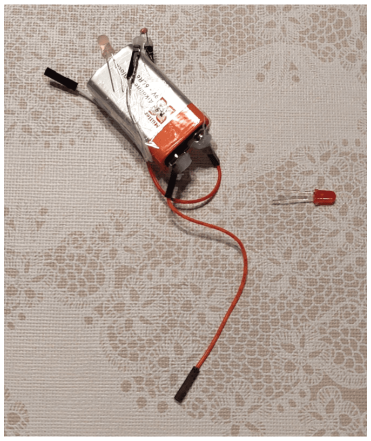


Figure 3: A small lamp mounted on a battery with cables. It can be attached to a moving object by a glue tape.

2. Model the toy in GeoGebra and trace the movement of the hump. (The students already had an acceptable background of GeoGebra knowledge to make it possible to do experiments on their own.)
3. Make a conjecture. (Here most students conjectured that the movement was an ellipse.)
4. Show the locus of the trace points. (We will see later that the conjecture was wrong, because the trace shows a different curve, namely, something like a form “8”).
5. Make a second conjecture. (This was a very difficult question, since an 8-formed curve is not present in the curriculum, neither at secondary nor university level.)
6. Compute the mathematical equation of the locus. (This is easy by using the command or tool LocusEquation. Without this step, no satisfactory conjecture can be done.)
7. Check the conjectures. (This is possible by setting up an equation system by using pencil and paper, and then compute the locus curve by using technological means. For this problem, however, the students skipped this step. It was used to check a different problem, publicized in the LEGO 4094 set as the “moving monkey” [7].)
8. Generalize the problem with different inputs. (In general, we have a 4-bar linkage problem [2] that leads to a sextic movement.)

In the next subsections we give some details on the steps described above.

4.1 Exact Measurements

After measuring the distances among the most significant parts of the camel we mounted a small lamp with a battery on the camel (Fig. 3). Then we switched the light off and recorded the movement with a camera of a mobile phone. When attaching the lamp to the top of the camel, we can get a motion like shown in figure 4. These pictures were created after saving individual frames (25 images) with

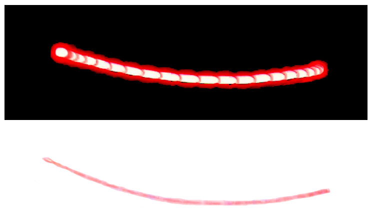


Figure 4: Motion of the lamp, by using two steps of preprocessing.

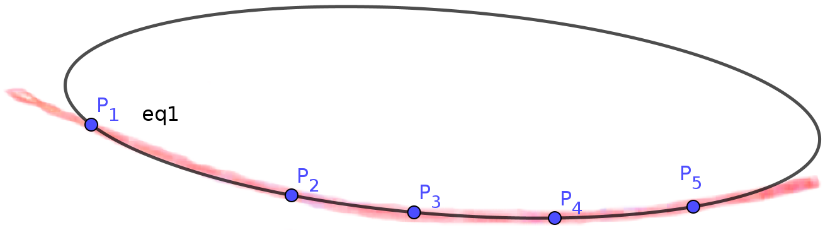


Figure 5: An attempt to identify the motion as an ellipse.

the *VLC media player*[4] and then opening them in *GIMP*.[5] Then the individual layers were edited with the "Exposure" function by changing the value of Black level to 0.1 (instead of 0.0). Furthermore the background of all layers was removed with the help of the function "Color by Alpha", so that only the red light was left and the single layers did not cover each other anymore.

4.2 Modeling in GeoGebra

An option to continue with is to try to fit a curve on the output. This is well-supported in GeoGebra by the possibility to insert a transparent figure, making it as a background picture, and then create some free points by hand that approximately cover the curve. GeoGebra's *ImplicitCurve* command can find the best fitting implicit polynomial (see figure 5 or `https://geogebra.org/m/c93pegab` for an online applet): for a curve of degree n one needs to enter $\frac{n\cdot(n+3)}{2}$ input points. That is, if we expect that the motion follows an ellipse (which is of degree 2), then 5 points are required. During the university course, however, we followed a different path. In GeoGebra we constructed the drawing as in figure 2 by creating free points $A = (0,0)$, $B = (15,0)$, then creating a segment CD with length $f = 5.5$, drawing a circle c with center A and radius f, and another circle d with center B and radius f. Then we attached point E on c, and after this step we created another segment FG with length $g = 12$. Next, we drew a third circle e with center E and radius g. One of the intersection points of d and e was designated to be point H. Then, as mentioned above, point M was created with some further steps by halving and rotating some additional points.

[4]VLC media player is a free and open-source, portable, cross-platform media player software and streaming media server developed by the VideoLAN project, available at `https://www.videolan.org/vlc`.

[5]GIMP (GNU Image Manipulation Program) is a free and open-source raster graphics editor used for image manipulation (retouching) and image editing, free-form drawing, transcoding between different image file formats, and more specialized tasks, available at `https://www.gimp.org`.

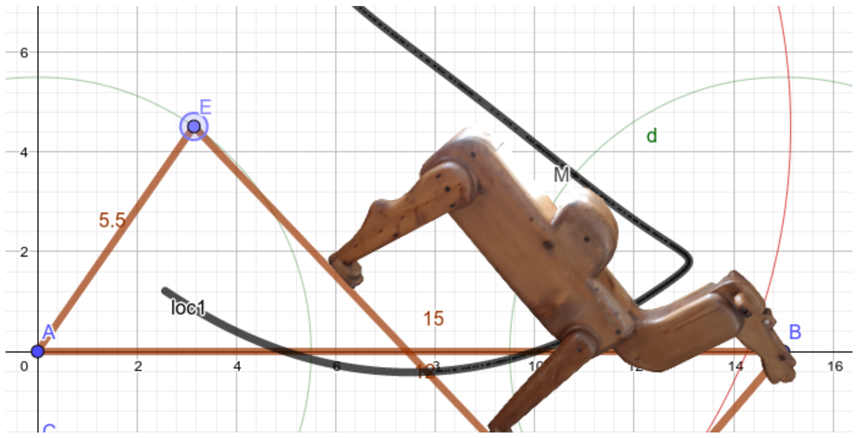


Figure 6: Dragging point E in an unrealistic position to disprove that the searched motion is elliptical.

We remark that this construction is a special case of a planar 4-bar linkage, which is well-known in the study of mechanisms, and has important applications like Watt's steam engine or a pumpjack.

An exact GeoGebra model helped the students to make experiments with the linkage without visiting the medical center and making their own measurements.

4.3 A First Conjecture

The students had one week of working time to make a conjecture. Several learners made a false conjecture, however, because they had no idea that there could be a solution other than the ellipse. This also raises the general question of the pedagogical consequences of oversimplifying the mathematical modeling of world problems.

4.4 A Numerical Locus

Some students, however, continued dragging point E to unrealistic positions and they obtained visual evidence that the searched curve is clearly not an ellipse (Fig. 6). This can also be checked in the above mentioned applet by enabling the "Locus" checkbox.

4.5 A Second Conjecture

At this point, a second conjecture could be made, but due to the lack of ideas, we more or less skipped this step. In fact, if you do not know the concept of higher degree curves, there is no chance to have a conjecture that the output is a polynomial curve.

4.6 A Symbolic Locus

This step can be reproduced by enabling the "LocusEquation" checkbox in the above mentioned applet. We obtain, by using some computer algebra (which is not further explained in this step) a sextic polynomial equation,

$$256 \cdot 10^{14}x^6 - 1152 \cdot 10^{17}x^5y^2 - 768 \cdot 10^{14}x^4y^2 - 312 \cdot 10^{15}x^4y + \ldots = 0$$

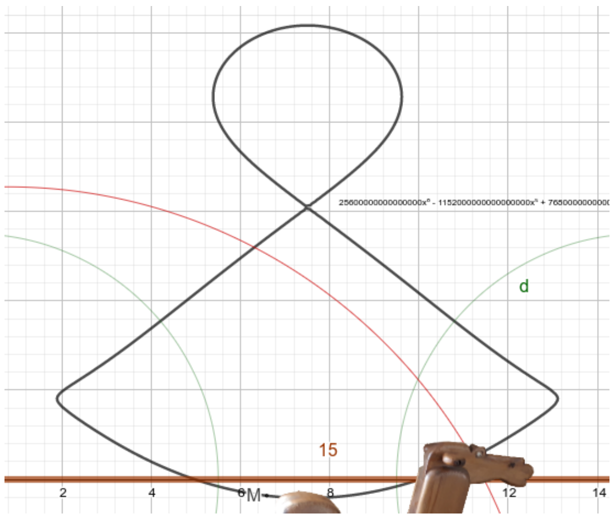


Figure 7: Obtaining a symbolic locus equation.

(Fig. 7). Here the students can only rely on the underlying computer algebra system, it is just a black box, but the coincidence of the numerical and symbolic loci can confirm, at least, partially, that the computations are hopefully correct.

4.7 A Proof

Now we need to prove that the obtained curve is indeed a sextic. To achieve this, we can set up an equation system with equations $a^2+b^2=5.5^2$ (here $E=(a,b)$), $(c-15)^2+d^2=5.5^2$ (here $H=(c,d)$), $(a-c)^2+(b-d)^2=12^2$, and for obtaining the coordinates M we might compute the coordinates of the midpoint I of segment EH and then rotate E around H by -90 degrees to get E'. Having E', the midpoint J of IE' can help to create the midpoint K of JE', and midpoint L of KE'. Finally, M is the midpoint of LM. This process is, of course, quite complicated, but it shows how we can be arbitrarily close to any point of the camel, by using just simple geometric operations. Later, by using dilations, this will be easier.

Now, by using elimination from algebraic geometry we can obtain the locus equation by using GeoGebra's *Eliminate* command. This is still a black box operation, but at least the students can have an idea what the exact input is, and the teacher can argue that by using the first three basic operations (addition, subtraction and multiplication), there is a finite algorithm [1] that indeed produces the result.

And this is actually a proof, in the deepest sense of the notion. Even if the atomic steps of the computation remain hidden, a reliable computer algebra system on reliable hardware will indeed compute the expected equation of the searched curve.

Let us highlight this fact even more. In classical geometry we are used to proofs that give arguments why the studied outputs are certain curves like lines, circles or maybe ellipses. The argumentation is sometimes purely synthetic, but sometimes analytic. Here we cannot really give a synthetic argumentation why a sextic curve appears. Only an analytic proof is applicable. But, because of the technical difficulty of the proof there is no way to check each step in a manual way. Therefore, a computer assisted proof is required, and as such, the automated way of elimination is satisfactory.

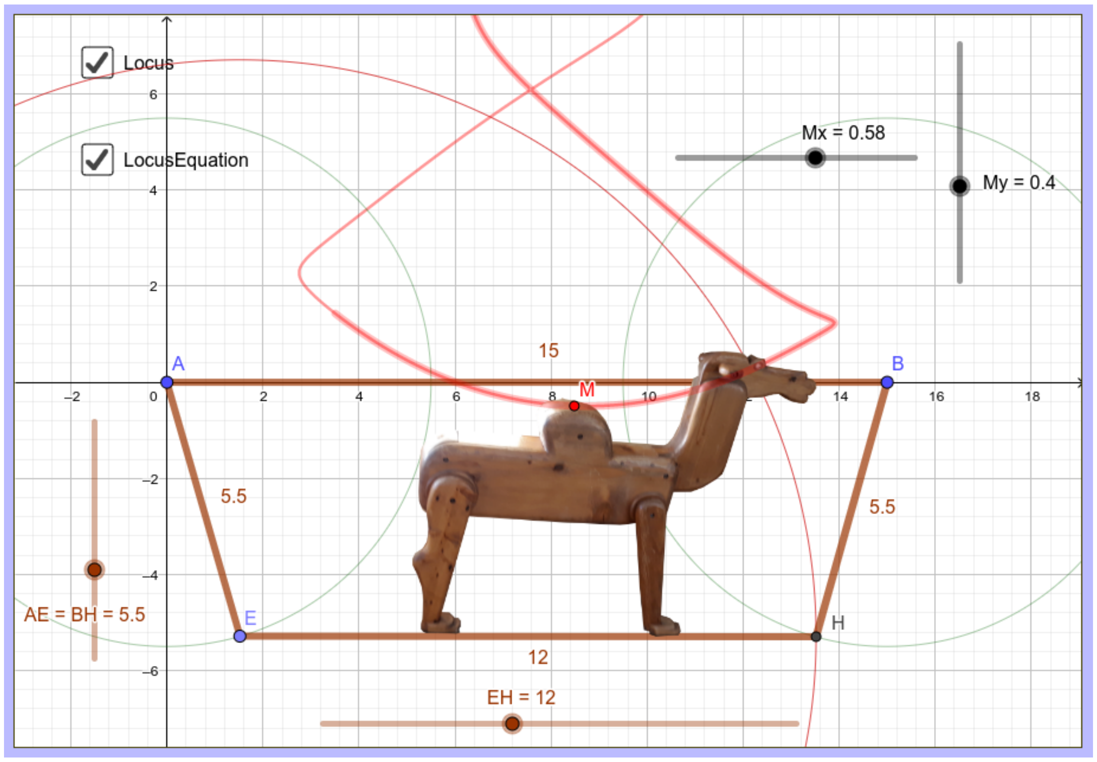


Figure 8: Generalization with sliders via the Dilate command.

4.8 Generalization

With some feedback from the students it was possible to improve GeoGebra Discovery to support generalizing the problem setting in the following way: *How does the output curve change when the lamp has a different position than the hump of the camel?*

To achieve this, the Dilate command in GeoGebra required symbolic support. The applet at `https://matek.hu/zoltan/camel.php` (see figure 8) allows the user to conveniently change the length of the bars AE and BH (they are still equally long) and the bar EH. By using dilation and sliders, the background computation requires less variables, because instead of 4 free variables just one needs to be used. This speeds up the computation substantially. To avoid the difficult way of defining M we introduced two sliders Mx and My that help find the position of M in an intuitive way. In addition, the user is notified immediately when the locus equation changes by using GeoGebra's JavaScript API[6] (Fig. 9). This applet was created by the use of the Dilate command. Dilation allows the user to create an arbitrary linear combination of two vectors. The coefficients of the linear combination can usually be rational numbers. Using one direct step to define ratios of certain quantities, instead of using the intercept theorem or utilizing midpoints, helps simplify the construction and avoid slow computation because of the high amount of variables. As well-known, elimination may be double exponentially slow in the number of variables in its worst case [6]. Therefore each optimization step may be crucial.

As a conclusion, the students can have a general conjecture after some further experiments, that 4-bar linkages usually yield sextic curves [3]. Of course, such experiments are insufficient to get a general

[6]Available at `https://wiki.geogebra.org/en/Reference:GeoGebra_Apps_API`.

$$\begin{gathered}625000000x^6 - 29625000000x^5 + 1875000000x^4y^2 - 7500000000x^4y + 513916750000x^4 - 59250000000x^3\\ y^2 + 225000000000x^3y - 3894242700000x^3 + 1875000000x^2y^4 - 15000000000x^2y^3 + 701583500000x^2\\ y^2 - 2494326000000x^2y + 12634068729100x^2 - 29625000000xy^4 + 225000000000xy^3 - 3894242700000xy^2+\\ 14694390000000xy - 26440635548340x + 625000000y^6 - 7500000000y^5 + 187666750000y^4 - 2494326000000y^3+\\ 23089046979100y^2 - 75203840809200y = -80422746144129\end{gathered}$$

Figure 9: A particular sextic equation, the corresponding curve is plotted in figure 8.

proof for all possible parameters. And, in fact, in some degenerate cases these results are actually not true, for example if the construction collapses into one point.

5 Final Thoughts and Conclusion

Automated geometric proofs may play an even more important role as before at secondary school level and above. The concept of analytic proofs (instead of synthetic ones) can already be familiar with algebraization of the geometric setup. For example, the well-known theorem by Thales that highlights a connection between right triangles and their circumcircles, can be easily translated into an algebraic setup and proven without difficulty. Indeed, let $A = (-1,0)$, $B = (1,0)$, $C = (x,y)$, and assume that $x^2 + y^2 = 1$, that is, C lies on a circle whose diameter is segment AB. Now, checking if AC is perpendicular to BC means exactly that $(x-(-1)) \cdot (x-1) + (y-0) \cdot (y-0) = 0$, and this is equivalent with our assumption on the sum of squares. That is, after making sure that the algebraization is performed correctly and generally enough, some algebraic manipulation will give the required argumentation.

Such an easy derivation is, unfortunately, not always possible. But we can learn that it is possible to formulate also the converse of the statement, that is, to ask: What is the geometric locus of points (x,y) such that AC is perpendicular to BC, when A and B are fixed? And here we conclude that the searched equation is $x^2 + y^2 = 1$, a *quadratic* one, in particular, the equation of a circle. In general, however, we may obtain *non-linear* and *non-quadratic results* as well. In our example in this paper we obtained a sextic equation, with huge coefficients. And this can happen in many other situations. Real life examples (of study of mechanisms, or optics) are full of higher degree polynomial curves. Here we mention conchoids, cissoids, strophoids (of degree 3) or cardioids, deltoids or lemniscates (of degree 4), many of them already well-known by the ancient Greek mathematicians.

In such higher degree cases, a proof that a certain curve is the expected result is nothing else than a long elimination process. Even if the computations are hidden, we expect that each step of the derivation is performed correctly, and therefore the result is correct.

That is, STEM/STEAM education cannot avoid such proofs in the long term. But, luckily, the existing tools are already safe and rich enough to support the learners in both the exploration and verification.

Acknowledgments

We are grateful to students Eva Erhart and Engelbert Zeintl for their help in many aspects of this paper. Benedek Kovács kindly helped us in preparing the photo of the rocking camel for further work in GeoGebra. He also helped in the preparation of the mounted small lamp. The second author was partially supported by a grant PID2020-113192GB-I00 (Mathematical Visualization: Foundations, Algorithms and Applications) from the Spanish MICINN.

References

[1] Bruno Buchberger (2006): *Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal*. *Journal of Symbolic Computation* 41, pp. 475–511, doi:10.1016/j.jsc.2005.09.007.

[2] Kenneth Henderson Hunt (1990): *Kinematic Geometry of Mechanisms*, 2 edition. 7, Oxford Engineering Science Series.

[3] Zoltán Kovács, Tomás Recio & M. Pilar Vélez (2020): *Reasoning about linkages with dynamic geometry*. *Journal of Symbolic Computation* 97, pp. 16–30, doi:10.1016/j.jsc.2018.12.003.

[4] Zoltán Kovács, Tomás Recio & M. Pilar Vélez (2021): *Automated reasoning tools in GeoGebra Discovery*. *ACM Communications in Computer Algebra* 55(2), pp. 39–43, doi:10.1145/3493492.3493495.

[5] Zoltán Kovács, Tomás Recio & M. Pilar Vélez (2021): *GeoGebra Discovery in Context*. In Predrag Janičić & Zoltán Kovács, editors: Proceedings of the 13th International Conference on *Automated Deduction in Geometry*, Hagenberg, Austria/virtual, September 15-17, 2021, *Electronic Proceedings in Theoretical Computer Science* 352, Open Publishing Association, pp. 141–147, doi:10.4204/EPTCS.352.16.

[6] Ernst W. Mayr & Albert R. Meyer (1982): *The Complexity of the Word Problem for Commutative Semigroups and Polynomial Ideals*. *Advances in Mathematics* 46, pp. 305–329, doi:10.1016/0001-8708(82)90048-2.

[7] Reinhard Oldenburg (2008): *FeliX – mit Algebra Geometrie machen*. *Computeralgebra Rundbrief, Sonderheft zum Jahr der Mathematik*. Available at `http://www.fachgruppe-computeralgebra.de/data/JdM-2008/Sonderheft.pdf`.

[8] Bryan Edward Penprase (2020): *STEM Education for the 21st Century*. Springer Cham, doi:10.1007/978-3-030-41633-1.

3D Space Trajectories and beyond: Abstract Art Creation with 3D Printing

Thierry Dana-Picard
Jerusalem College of Technology
Jerusalem, Israel
ndp@jct.ac.il

Matias Tejera
Eva Ulbrich
Johannes Kepler University
Linz, Austria
Mathias.Tejera@jku.at
Eva.Ulbrich@jku.at

We present simple models of trajectories in space, both in 2D and in 3D. The first examples, which model bicircular moves in the same direction, are classical curves (epicycloids, etc.). Then, we explore bicircular moves in reverse direction and tricircular moves in 2D and 3D, to explore complex visualisations of extraplanetary movements. These moves are studied in a plane setting. Then, adding increasing complexity, we explore them in a non planar setting (which is a closer model of the real situation). The exploration is followed by using these approaches for creating mathematical art in 2D and 3D printed objects, providing new ways of mathematical representations. Students' activities are organized around this exploration.

1 Introduction

All over the world, newspapers and TV news are full of reports about launching satellites, the International Space Station, the Chinese space station, Mars exploration and the Artemis project to establish a permanent human presence on the Moon. Nowadays, the NASA offers the public to send their names on a probe to be launched in 2024 and arrive to Encelade, an icy moon of Jupiter, in 2030. With such an ubiquitous topic, students asked a lot of questions, about spacecrafts, their trajectories, their trajectories, why these are curved and sometimes complicated, etc. Numerous dedicated websites are freely accessible, showing representations of trajectories of extraplanetary objects. These are connected to the students' cultural background, on which it is worth to rely in order to attract students to mathematics, and to show applications in real world [4]. This paper explores mathematical situations with a STEAM approach visualising curves in 2D and 3D with various technologies to use the motivational fascination of outer space from students to connect to mathematical modelling.

When asking about spacecrafts, they wish to understand the trajectories. Not all the news items include graphs and maps of the trajectories, but they frequently do so and can be the source of questions, whence of mathematical activities. These are good reasons for mathematics educators to be part of this atmosphere, showing complex real world applications of mathematics. Examples could be curves describing trajectories or calculating the speed of objects in an accessible way by interactive visualizations and explanatory animations. Students have the opportunity to create and explore these trajectories themselves by visualising them using mathematical modelling and certain technologies and we present possible approaches in 2D and 3D.

According to the 1$^{\text{st}}$ Kepler's law (see [8], p. 127), the orbit of a planet around the Sun is an ellipse, with the Sun at one of the foci. As the foci are very close, actually both inside the Sun,[1] in order to make

[1] Actually, in a system of two objects, both orbit their common center of gravity. The system Sun-Earth's center of gravity is inside the Sun, therefore considering the Earth as orbiting the Sun is acceptable. Of course, every other pair Sun-Planet presents the same situation.

the example as simple as possible, we consider an approximation of the orbits as coplanar concentric circles. Kepler's 2nd law is illustrated by figure 1, taken from [8] p. 129: the areas of the shaded sectors, covered by the radius in equal times (i.e. it takes equal times to travel distances AB, CD and EF), are equal.

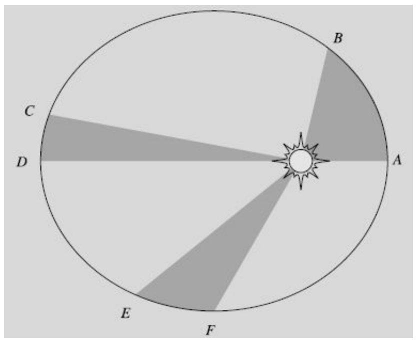


Figure 1: Kepler's 2nd law of planetary motion

In our simplified model, we consider motion with constant angular velocity on circular orbits. We compute the velocities according to the year length of the planet, with Earth year equal to 1. Note that Table 1 displays just the eight official planets acknowledged by the international astronomical organization. According to the 3rd Kepler's law, the orbital velocity is a function of the distance to the Sun.

Planet	**Distance to the Sun (km)**	**Period (1=terrestrial year)**
Mercury	$57.91\ 10^6$ km	0.2408
Venus	$108.2\ 10^6$ km	0.6152
Earth	$149.6\ 10^6$ km	1
Mars	$227.9\ 10^6$ km	1.8808
Jupiter	$778.5\ 10^6$ km	11.862
Saturn	$1.434\ 10^9$ km	29.457
Uranus	$2.871\ 10^9$ km	84.018
Neptune	$4.495\ 10^9$ km	164.78

Table 1: Some orbital data

Because of the huge differences between the distances and the hardware constraints[2] (we mean mostly the size of the screen and the number of available pixels), we will consider examples with Earth and Mars only. The same activities can be done with the pair Venus-Earth, they will produce the same family of curves. Note that in order to make the first examples easy, we use approximations less precise than in Table 1.

The visualisations we explore in this paper are created by two softwares called GeoGebra and Maple to utilise their respective strengths. We use GeoGebra,[3] whose main characteristic is devoted to Dynamic Geometry. For some applications, including automated determination of loci and envelopes, it can be supplemented with the package GeoGebra-Discovery.[4] A general analysis of the automated methods for

[2]A general study of constraints, either of the hardware or of the software can be found in [9], with some extension in [2].

[3]Freely downloadable from `http://www.geogebra.org`.

[4]Look for the last version, freely downloadable from `https://github.com/kovzol/geogebra-discovery`.

loci and envelopes is given in [6]. We will also use the Computer Algebra System Maple for its specific animated affordances, which are different from those of GeoGebra.

Exploration of curves obtained as trajectories of points modeling moves in space, such as midpoint or center of gravity of two planets, is described and analyzed in [3, 5]. The present paper is a new contribution, with more complex constructions. Its goal is to present mathematical situations with a STEAM[5] approach, where plane curves, either algebraic or not, are presented and some of their properties explored using technology. We see this kind of study as an opportunity to connect to the classical families of plane curves in a motivating manner for students and can be used as a unifying frame for cases previously seen as separate cases. Later, space curves given by similar parametric equations are explored, also modelling spatial phenomenon. The ratio of the mean radii of two neighboring planets (such as Venus-Earth, or Earth-Mars) is huge, and still more the ratio between the mean radius of the Earth orbit around the Sun (about 149 Mkm) and the radius of the Moon's orbit around the Earth (about 360,000 km), it is impossible to represent both on a computer screen. Therefore, we chose to work with arbitrary[6] coefficients, whose variations provide different curves. We explore the composition of two circular movements in the same direction (the general case in the Solar System). Figure 2 shows a simplified model of a spacecraft flying to Mars, without explicit presentation of the orbiting direction around Mars.[7]

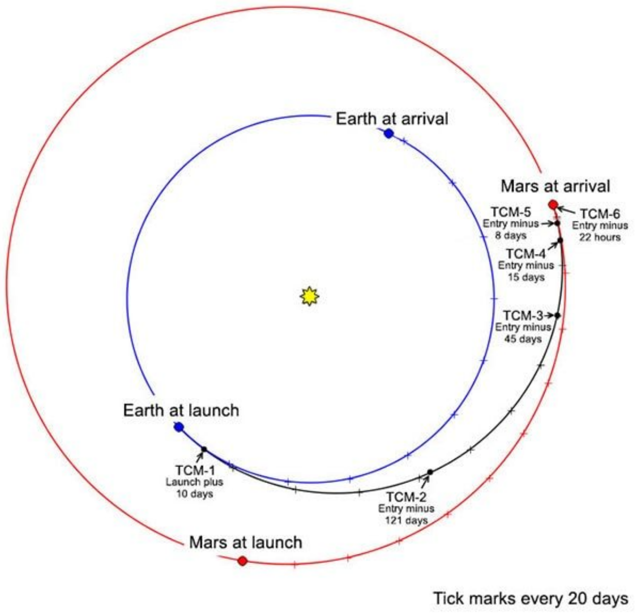

Earth at arrival
Mars at arrival
TCM-5 Entry minus 8 days
TCM-6 Entry minus 22 hours
TCM-4 Entry minus 15 days
TCM-3 Entry minus 45 days
Earth at launch
TCM-1 Launch plus 10 days
TCM-2 Entry minus 121 days
Mars at launch
Tick marks every 20 days

Figure 2: Trajectory from the Earth to the Mars orbit

We also consider the composition of 3 circular movements, inspired by lunar orbiters and observed also from the Sun: they orbit the Moon, which orbits the Earth, which in its turn orbits the Sun. In Subsection 2.2, we explore the composition of 3 circular movements, all in the same direction. In Subsection 2.3, we explore also models of a composition of movements, two in one direction and the 3rd

[5]STEAM = acronym for Science, Technology, Engineering, Arts and Mathematics.

[6]By arbitrary, we mean coefficients enabling a representation on the screen, not taken from the orbital data.

[7]Credit: NASA/JPL, `https://marspedia.org/File:InSight_Trajectory.jpg`

in reversed direction. The motivation for this is provided by the trajectories of spacecrafts to the Moon; figure 3 shows a diagram of the trajectory of the Artemis 1 spacecraft, elliptic around the Earth, followed by a transfer orbit made of arcs of ellipses, then elliptic around the Moon in reversed direction.[8] The geometric locus of the moving object around the Moon, when observed from the Sun, may be an epicycloid, a hypocycloid or another already known curve. Here, more "exotic" curves are also explored and plotted; in particular, rotational symmetries of order 7, 11, 13, etc. may be discovered. This provides an opportunity for an interactive exploration of such symmetries.

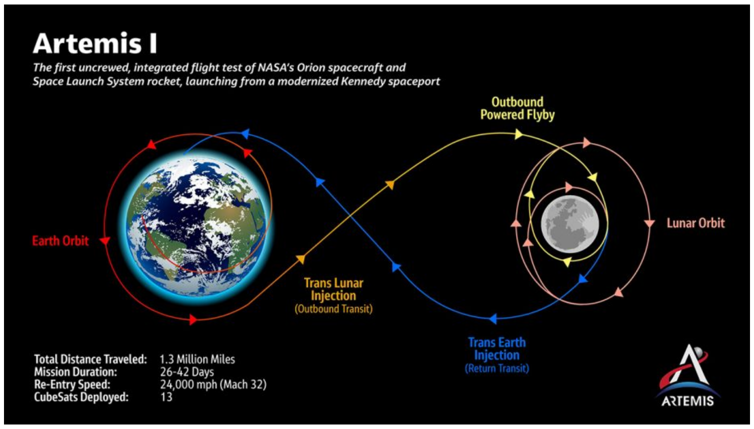

Artemis I
The first uncrewed, integrated flight test of NASA's Orion spacecraft and Space Launch System rocket, launching from a modernized Kennedy spaceport
Outbound Powered Flyby
Earth Orbit
Lunar Orbit
Trans Lunar Injection (Outbound Transit)
Trans Earth Injection (Return Transit)
Total Distance Traveled: 1.3 Million Miles
Mission Duration: 26-42 Days
Re-Entry Speed: 24,000 mph (Mach 32)
CubeSats Deployed: 13
ARTEMIS

Figure 3: Artemis orbit to the Moon and around (Credit: NASA)

Jablonski [7] says that "Mathematical modelling is characterized through its interplay of reality and mathematics. It offers a way to integrate references to reality into the classroom and shows students where in everyday life their mathematical knowledge can be applied." Therefore, we started from real world situations utilizing the amazement created by media reports. The first examples provide some understanding of how the orbit of the Moon around the Sun looks like, but quickly we explored compositions of movements without a connection to reality. Changing the parameters (either ratio of radii or ratio of angular velocities) induces important changes on the shape and topology of the curves. Tricircular moves are inspired by, for example, lunar orbiters (which orbit the Moon, which orbits the Earth, which in its turn orbits the Sun), or Mars orbiters. As already mentioned, figure 2 shows the trajectory of Mars Orbiter, from start to arrival: at first ellipses around the Earth, then a transfer orbit (made of arcs of ellipses), then elliptic orbits around Mars. This can be explained to students.

Instead of returning from models to the real world situation, which had to be understood, numerous new directions are possible. As an example, curves of degree 8, obtained from a construction disconnected from the physical data, have been explored recently; see [6].

Finally, we explore artistic creation using these mathematical models. We obtain curves presenting non usual symmetries and explore them using our software.

In the real world of the software may change. The exploration of the curves is an important incentive to 3D print them. We quote once again Jablonski [7]: "The idea of involving real objects in mathematical modelling leads to the question of how much the way in which a real object is introduced might influence the modelling processes of students. Despite its actual physical presence in reality, a real object could be introduced through different representations and provided artefacts, e.g., newspaper articles,

[8]The trajectories of future Artemis missions will be different from this one, but based on the same principle.

photographs, videos, 3D print replications or combinations. Potentially, the different representations of the real object might lead to differences in the modelling activities of students and motivate a comparison of them."

2 Classical Curves and beyond

In all the examples we consider a planet (let us call it the Earth, orbiting the Sun at distance 1 (a reference to 1 astronomical unit, 1 AU) at constant velocity, and completing 1 orbit in 1 year. The other coefficients describe the mean radius of another planet and the length of its own year. The is described by the following parametric presentation:

$$(x,y) = (\cos u,\ \sin u),\ u \in \mathbb{R}. \tag{1}$$

For the animations with software, $u \in [-0, 2\pi]$ is enough with repetitive animation. We denote the parameter by u, as in GeoGebra t has a special role. The second planet is described by

$$(x,y) = r\left(\cos\frac{u}{h},\ \sin\frac{u}{h}\right),\ u \in \mathbb{R}, \tag{2}$$

where $r > 0$ denotes the radius of the planet's orbit and h encodes the length of its year.

2.1 Epicycloids in 2D and Extension towards 3D

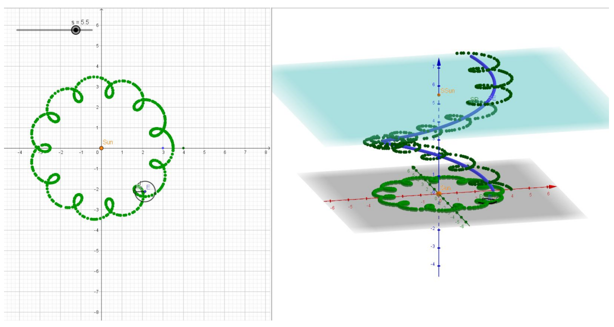
Figure 4 shows a screenshot of a dedicated GeoGebra applet.[9] The parameters can be changed with slider bars. The figure on the left shows the trajectories in the plane containing the Sun, the planet and its satellite. Here the satellite orbits the planet 12 times a year, almost modelling the Moon around the Earth. The figure on the right shows a simulation when the Sun travels on a straight line; note that the 3 objects remain all the time in a plane which moves according to the Sun. For the 2D representation, the orbits can be either plotted in a non-animated way using GeoGebra's **Locus** command or to be animated using the corresponding option of the slider bar. Nite that other way to animate the constructions are available.

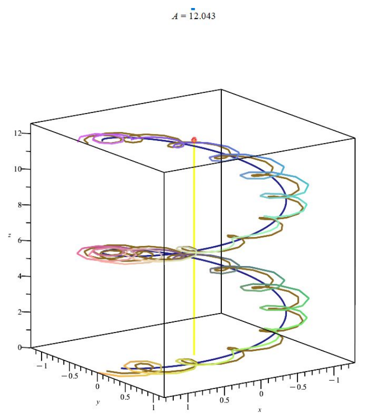
As already mentioned, the Sun is also mobile, it has its own orbit. Figure 4(a) shows a model where the Sun moves along a segment of line. The planet and its satellite move in a plane containing the 3 objects (this plane is visible in blue). The commands are similar in 3D as in 2D. Note that the projection on the plane is on display in the adjacent window. This is due to the total synchronisation of the 3D and 2D windows in GeoGebra. This task was an incitement to go to 3D printing.

A similar animation can be programmed with Maple. The code is easy: each object is defined in a separate **plot[animate]** command, them all together they are displayed using the display command. The Sun has two commands: one for plotting a large dot, the other one to plot the trajectory. An animated gif can be obtained with a right-click on the output of the **display** command. A screenshot is shown in figure 4(b).

```
c1 := spacecurve([cos(t) + 1/5*cos(12*t), sin(t) + 1/5*sin(12*t), t],
      t = 0 .. 4*Pi, thickness = 3, labels = [x, y, z]):
sun := plots[animate](spacecurve, [[0, 0, t], t = 0 .. A], A = 0 .. 4*Pi,
      thickness = 3, color = yellow)
sunplo := plots[animate](pointplot3d, [[0, 0, A]], A = 0 .. 4*Pi,
```

[9]See https://www.geogebra.org/m/ksyd6hat.


(a) GeoGebra



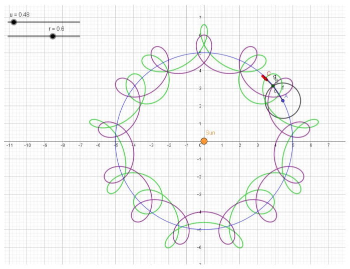
(b) Maple

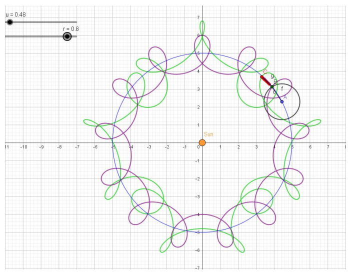
Figure 4: A satellite around a planet orbiting the Sun

```
      color = orange, symbol = sphere)
planet := plots[animate](spacecurve, [[cos(t), sin(t), t], t = 0 .. A],
A = 0 .. 4*Pi, thickness = 3, color = navy)
sat := plots[animate](spacecurve, [[cos(t) + 1/5*cos(12*t), sin(t)
       + 1/5*sin(12*t), t], t = 0 .. A], A = 0 .. 4*Pi,
        color = sienna, labels = [x, y, z]):
display(sun, planet, sat, sunplo)
```

2.2 Three Circular Movements with Constant Angular Velocity – Same Direction

Figure 5 shows snapshots of GeoGebra sessions based on the **Locus** command. Subfigure (c) is a snapshot of a GeoGebra applet[10] with 2 parameters encoding the distances. A further step consists in adding parameters to change the ratios of angular velocities.


(a)


(b)

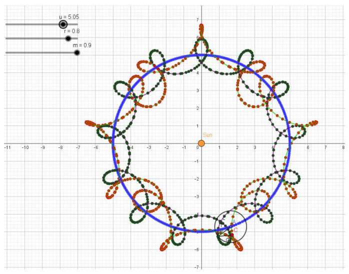
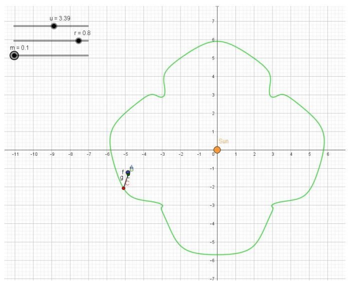

(c)

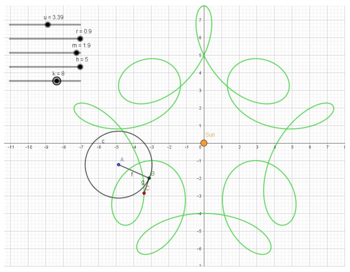
Figure 5: Screenshots of a tricircular motion in the same direction

[10]See https://www.geogebra.org/m/sagpjzzb.

2.3 Three Circular Movements with Constant Angular Velocity – One in Reverse Direction

Figure 6 displays 3 curves obtained with the **Locus** command, in an applet[11] where all the parameters can vary. In what follows, we explore the symmetries of the obtained curves. These symmetries are often of odd order, a situation which is not frequent in classroom.


(a) A 4-star


(b) A strange star

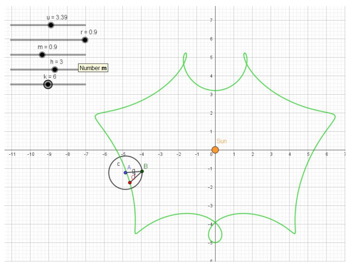

(c) A bat curve

Figure 6: Screenshots of a tricircular motion with the middle in reversed direction

We consider now the family of curves whose parametric equations are as follows:

$$(x,y,z) = (\cos u, \sin u) + \frac{1}{3}(\cos au, \sin au) + \frac{1}{2}(\sin bu, c\cos bu), \tag{3}$$

where a, b encode the ratios of circular velocities. In the applet `https://www.geogebra.org/m/jugrcbx5`, their increment is defined to be 1.

For $a = b = 1$, the curve is an ellipse. But there are other cases, maybe more interesting. Figure 7(a) has been obtained for $(a,b) = (6,14)$ with the **Curve** command. It presents a 5-fold rotational symmetry, i.e. it is invariant under a rotation whose center is the origin and of angle $2\pi/5$. This has been checked with a plot of the parametric equations for $u \in [0, 2\pi/5]$, the applying the automated command for rotations. The colors have to be manually adapted to create visualisations that are easier to interpret where each curve has a unique style. Part of GeoGebra's algebraic display can be seen in figure 7(a) to illustrate what has been done. The curve can also be plotted defining a variable point depending on the parameter u, then applying the **Locus** command. The definition of a variable point provides a dynamic plot of the curve, but both in this case and with the **Locus** command, the output is not a geometric object on which a plane transformation can be applied.

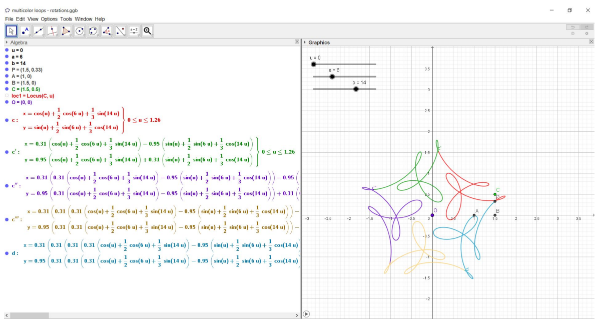
Other cases have to be cautiously explored for symmetries. For example, the case $(a,b) = (10,14)$ shows a 3-fold symmetry (see figure 7(b). Experomentation will show that this also true for $(a,b) = (7,14)$ and $(a,b) = (7,17)$.

2.4 Math Art Creation

The applet mentioned in the previous subsection has been opened, running animations for the parameters a and b separately. Exploration has been preformed according the following steps:

- The entire curve is plotted, using Trace On;
- Analyzing the graphical display, the existence of rotational symmetry is conjectured;

[11]See `https://www.geogebra.org/m/xgrx7ntx`.

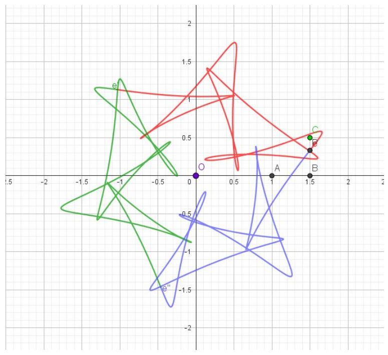

multicolor loops - rotations.ggb

File Edit View Options Tools Window Help

Algebra

- u = 0
- a = 6
- b = 14
- P = (1.5, 0.33)
- A = (1, 0)
- B = (1.5, 0)
- C = (1.5, 0.5)
- loc1 = Locus(C, u)
- O = (0, 0)
- c : $\left.\begin{aligned} x &= \cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u) \\ y &= \sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u) \end{aligned}\right\} 0 \le u \le 1.26$
- c′ : $\left.\begin{aligned} x &= 0.31\left(\cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u)\right) - 0.95\left(\sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u)\right) \\ y &= 0.95\left(\cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u)\right) + 0.31\left(\sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u)\right) \end{aligned}\right\} 0 \le u \le 1.26$
- c″ : $x = 0.31\left(0.31\left(\cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u)\right) - 0.95\left(\sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u)\right)\right) - 0.95$
  $y = 0.95\left(0.31\left(\cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u)\right) - 0.95\left(\sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u)\right)\right) + 0.31$
- c‴ : $x = 0.31\left(0.31\left(0.31\left(\cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u)\right) - 0.95\left(\sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u)\right)\right) -$
  $y = 0.95\left(0.31\left(0.31\left(\cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u)\right) - 0.95\left(\sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u)\right)\right) -$
- d : $x = 0.31\left(0.31\left(0.31\left(0.31\left(\cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u)\right) - 0.95\left(\sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u)\right)\right.\right.\right.$
  $y = 0.95\left(0.31\left(0.31\left(0.31\left(\cos(u) + \frac{1}{2}\cos(6u) + \frac{1}{3}\sin(14u)\right) - 0.95\left(\sin(u) + \frac{1}{2}\sin(6u) + \frac{1}{3}\cos(14u)\right)\right.\right.\right.$

(a) 5-fold

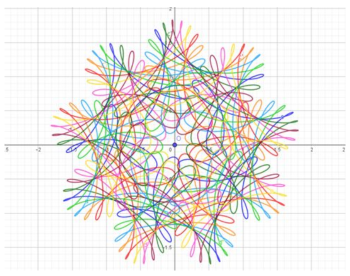


(b) 3-fold

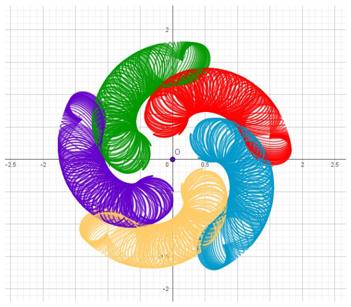
Figure 7: Tricircular moves creating multicolor curves with rotational symmetries

- The rotational symmetry is checked by first reducing the plot to a subset of the interval chosen for the parameter; we mean taking an interval of the form $[0, 2\pi/m]$, where m is te order of the conjectured symmetry, and then using the automated command for a rotation about the origin with angle $2\pi/m$.
- Of course, this has to be checked afterwards by symbolic means, using a substitution.

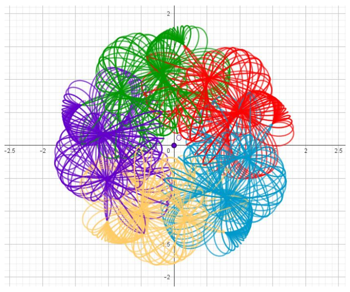
Later, an experiment has been made, choosing an arbitrary number m, not the order of the rotational symmetry which has been discovered. The obtained multicolor plot does not describe a specific mathematical situation. Some of the results are displayed in figure 8.



(a)



(b)



(c)

Figure 8: Some random math art creations

Discovering such creations has been greeted with enthusiasm by the audience of lectures delivered by the authors, whose topics was linked to curves and math art.

3 Some More Remarks

The starting point of the study is STEAM oriented, namely using a scientific model from an item in the news. Students may have prior interest in the domain, without having a strong knowledge. The present topic offers an opportunity to collaborate between educators, between man and machine, of course between students. The study output is multiple, and among the “rewards” we have:

- Acquisition of new mathematical knowledge: classical curves (epicycloids, epitrochoids, etc.), which are not part of the regular curriculum, have been discovered and studied. Epitrochoids are members of a larger family of curves, which involves roses, epicycloids, etc. Activities as in this work may be a nice incitement to explore other situations and to broaden horizons. The literature describes generally the epitrochoids for integer values of the parameters, and our experimentations showed also more general settings.
- Discovery of new curves; we mean curves which do not appear in the catalogues such as [10].
- Emphasis on the importance of the data precision (in space, contrary to most classrooms, nothing is measured by integers) and of rounding. We considered non integer ratios of radii of orbits, and of orbital angular velocity, approximations and rounding became an important issue. We could discover that different precisions in the approximation yield very different output. This is probably a central outcome of this work: students do not always believe that mastering errors is important, and they believe that the answers provided by a numerical calculator are always accurate. Asking them which answer is true among the cases that we studied with different rounding should lead at least to some questioning.
- Development on new technological skills, which are part of the new mathematical knowledge [1].
- Emphasis on multidisciplinary tasks, whence development of STEAM skills.

Note that generally, modeling is intended to construct mathematical descriptions of a concrete situation. Then, the model is applied to enhance more understanding of the concrete situation. The process is summarized in figure 9.

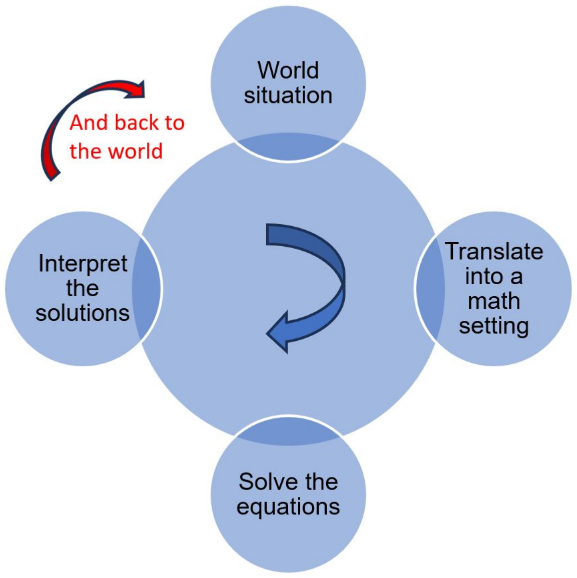


Figure 9: A classical diagram for a modelling process

In the present paper, we go in a totally different way in this case: modeling a concrete astronomical situation (orbits), the activities provide more abstract curves without a physical meaning. Finally 3D printing could provide both outcomes: a concrete object modeling planets and trajectories, and also some pieces of visual art, either in 2D or in 3D to apply constructivist as well as constructionist ideas. This is summarized in figure 10.

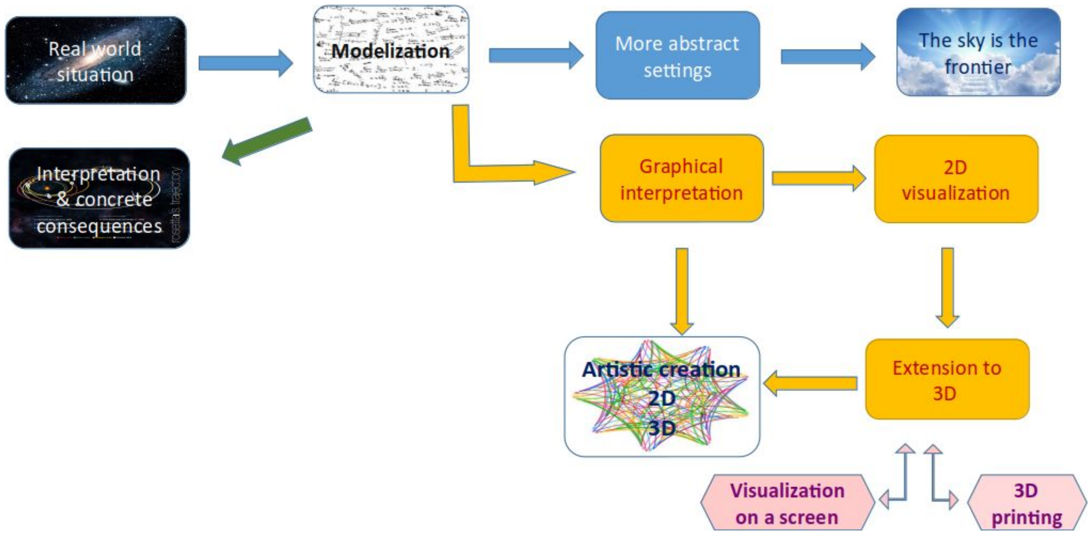


Figure 10: A modelling process leading in other directions

We performed the same experiments and constructions using Maple. The characteristics of the work CAS is slightly different.

- After a command line to define a parametric curve, an **animate** command has to be entered. Its output is not immediately visible.
- A left-click on the graphical window is necessary, and it switches automatically to the row of graphical buttons.
- Here too, the relevant values for the parameters (number of frames, speed, etc.), in order to obtain a significant graphical output have to be experimentally looked for, using the buttons.
- Other modifications of the output may require changes in the written commands.

After having presented some of the applets to a certain audience, the authors decided to 3D print part of them, together with some other cases. In parallel, tasks have been defined for groups of students, either gifted High-School students having benefit of an extension of the curriculum, or undergraduates. These students belong to two different countries. The tasks include the 3D printing of some examples. The transfer of the CAS output to a 3D printer requested the translation of this output into a language that the 3D printer understands. In our presentation, we will report on the math part and on the outcome of the activities with students.

4 Acknowledgements

The first author was partially supported by the CEMJ Chair at JCT.

References

[1] M. Artigue (2002): *Learning Mathematics in a CAS Environment: The Genesis of a Reflection about Instrumentation and the Dialectics between Technical and Conceptual Work*. International Journal of Computers for Mathematical Learning 7(3), pp. 245–274, doi:10.1023/A:1022103903080.

[2] T. Dana-Picard (2007): *Motivating constraints of a pedagogy embedded Computer Algebra System*. International Journal of Science and Mathematics Education 5(2), pp. 217–235, doi:10.1007/s10763-006-9052-9.

[3] T. Dana-Picard (2022): *The loci of virtual points constructed with elementary models of planetary orbits*. In: Electronic Proceedings of the Asian Conference on Technology in Mathematics ACTM 2021, Mathematics and Technology.

[4] T. Dana-Picard & S. Hershkovitz (2022): *STEAM Education: technological skills, students' cultural background and Covid-19 crisis*. Open Education Studies 2(1), pp. 171–179, doi:10.1515/edu-2020-0121.

[5] T. Dana-Picard & S. Hershkovitz (2024): *From Space to Maths and to Arts: Virtual Art in Space with Planetary Orbits*. to appear in Electronic Journal of Mathematics & Technology.

[6] T. Dana-Picard & T. Recio (2023): *Dynamic construction of a family of octic curves as geometric loci*. AIMS Mathematics 8(8), pp. 19461–19476, doi:10.3934/math.2023993.

[7] S. Jablonski (2023): *Is it all about the setting? – A comparison of mathematical modelling with real objects and their representation*. Educational Studies in Mathematics 113(2), doi:10.1007/s10649-023-10215-2.

[8] H. Karttunen, P. Kröger, H. Oja, M. Poutanen & K.J. Donner, editors (2008): *Fundamental Astronomy*. Springer, doi:10.1007/978-3-662-53045-0.

[9] L. Trouche (2000): *La parabole du gaucher et de la casserole à bec verseur: étude des processus d'apprentissages dans un environnement de calculatrices symboliques*. Educational Studies in Mathematics 41(3), pp. 239–264, doi:10.1023/A:1003939314034.

[10] R. Yates (1947): *A Handbook on Curves and their Properties*. J.W. Edwards, MI: Ann Arbor.

Theorem Discovery Amongst Cyclic Polygons

Philip Todd
Saltire Software
Portland OR USA
philt@saltire.com

We examine a class of geometric theorems on cyclic $2n$-gons. We prove that if we take n disjoint pairs of sides, each pair separated by an even number of polygon sides, then there is a linear combination of the angles between those sides which is constant. We present a formula for the linear combination, which provides a theorem statement in terms of those angles. We describe a program which uses this result to generate new geometry proof problems and their solutions.

1 Introduction

In [8], a characterization is made of linear systems involving angle bisection conditions which are not full rank. In such a system, one of the conditions is implied by the remainder, and, if the angle bisections are interpreted geometrically, this dependence may be stated in a number of different ways as a geometry theorem. The characterization leads to a catalog of such linear systems. An approach to theorem discovery is proposed wherein a linear system is initially selected, and then interpreted geometrically as a theorem. In [7], a program is described which applies this approach, constructing a particular geometry theorem corresponding to a randomly selected linear system from the catalog. In order to reduce diagram complexity, the program is biased in favor of constructing cyclic polygons wherever possible. In the case where it is able to construct a cyclic polygon using all the rows of the linear system, the theorem which is produced has the following form. Given a cyclic $2n$-gon, where $n-1$ specified pairs of sides are parallel, then a final specified pair of sides is also parallel. For example, in a cyclic hexagon, with two pairs of opposite sides parallel, the third pair of sides is also parallel. (In passing, we note thet this theorem is not true for a cyclic octagon, but is for a cyclic decagon.)

In this presentation, we consider a generalization of the above class of theorems, where instead of making line pairs parallel, we allow line pairs to be given non zero, but determined angles. While there is no geometric meaning to a parallel relationship between consecutive sides, replacing the parallelism by a defined non-zero angle permits adjacent sides of the polygon to be related. We examine the case of a $2n$-gon, with the angles between n pairs of sides named. We show that if the pairs share no side, and if the sides in each pair are either adjacent or seperated by an even number of polygon sides, then the named angles satisfy a particular linear relation.

An approach to theorem discovery in this context mirrors and illustrates that described in [8] and [7]. As a first step, a value of n is chosen, and a set of line pairs conforming to our criterion selected from a catalog containing all such sets. A diagram is produced directly which allows the coefficients of the constant linear combination to be computed.

2 Cyclic Polygons and Angle Bisectors

Let P be a polygon whose vertices lie on the unit circle centered at the origin whose vertices $p_1, \ldots, p_n$ have position vectors $u_1, \ldots, u_n$.

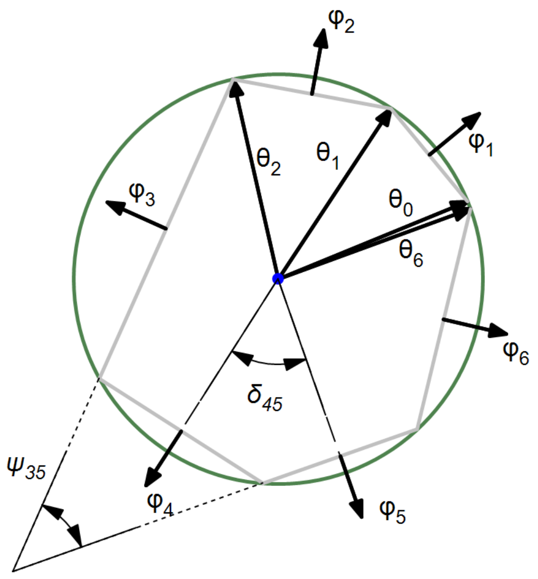


Figure 1: $\theta_0 \dots \theta_6$ are angles of position vectors. $\theta_6 = \theta_1 + 2\pi$. $\phi_i = \frac{1}{2}(\theta_{i-1} + \theta_i)$

We define $u_0 = u_n$.

Let $\alpha(u,v)$ be the directed angle between vector u and vector v. We define $\theta_0 = 0$ and for i from 1 to n:

$$\theta_i = \theta_{i-1} + \alpha(u_{i-1}, u_i)$$

As $u_0 = u_n$, $\theta_n = \theta_0 + 2\pi W$ where W is the winding number of the polygon about the origin. For i from 1 to n we define

$$\phi_i = \frac{1}{2}(\theta_i + \theta_{i-1})$$

For i and j from 1 to n we define

$$\delta_{ij} = \phi_j - \phi_i$$

Define L_i to be the line passing through points p_{i-1} and p_i . We will define q_{ij} to be the intersection of L_i and L_j. We define the angle

$$\psi_{ij} = \angle p_{i-1} q_{ij} p_j$$

2.1 Cyclic Quadrilateral

We first examine the cyclic quadrilateral (Figure 2).

In the figure, the quadrilateral has winding number 1 about the circle center, hence $\theta_4 = \theta_0 + 2\pi$. The two indicated opposite angles of the quadrilateral have values $\pi - \delta_{12}$ and $\pi - \delta_{34}$.

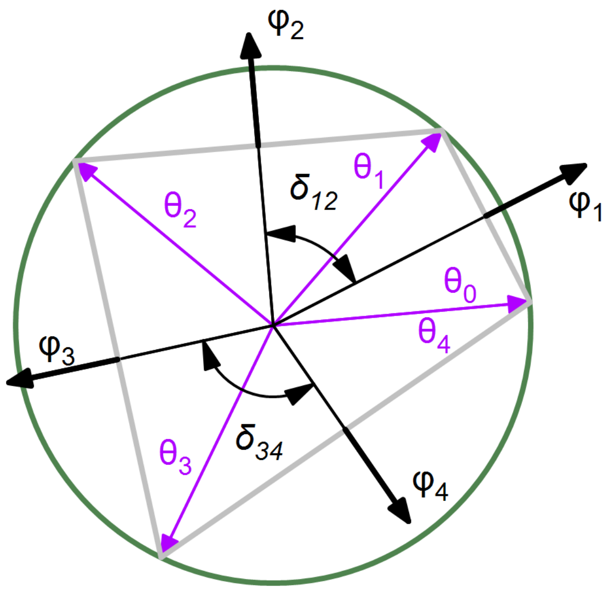


Figure 2: A cyclic quadrilateral with winding number 1.

The figure may be expressed by the following matrix equation:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -2 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2\pi \\ \delta_{12} \\ \delta_{34} \end{pmatrix} \tag{1}$$

We transform the matrix equation by performing the following row operations:

$R1 \leftarrow R1 + R5$, $R2 \leftarrow R2 + 2 \times R6$, and $R3 \leftarrow R3 + 2 \times R7$, which, after eliminating the zero columns, gives this matrix equation:

$$\begin{pmatrix} 1 & 0 & 0 & 1 & -2 & 0 \\ 1 & 1 & 0 & 0 & -2 & 0 \\ 0 & 1 & 1 & 0 & 0 & -2 \\ 0 & 0 & 1 & 1 & 0 & -2 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \phi_1 \\ \phi_3 \end{pmatrix} = \begin{pmatrix} 2\pi \\ 2\delta_{12} \\ 0 \\ 2\delta_{34} \end{pmatrix} \tag{2}$$

The matrix can be triangularized using the algorithm of [2]. Let R_i be the i'th row of the original matrix, and T_i the i'th row of the triangularized matrix, then $T_1 = R_1$, and for i from 2 to n:

$$T_i = R_i - T_{i-1} \tag{3}$$

Using this algorithm, our triangularized matrix equation is:

$$\begin{pmatrix} 1 & 0 & 0 & 1 & -2 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \phi_1 \\ \phi_3 \end{pmatrix} = \begin{pmatrix} 2\pi \\ 2\delta_{12} - 2\pi \\ -2\delta_{12} + 2\pi \\ 2\delta_{34} + 2\delta_{12} - 2\pi \end{pmatrix} \tag{4}$$

Consistency of this system requires

$$2\delta_{34} + 2\delta_{12} - 2\pi = 0 \tag{5}$$

or

$$\delta_{34} + \delta_{12} = \pi \tag{6}$$

In terms of ψ_{12} and ψ_{34}

$$\pi - \psi_{12} + \pi - \psi_{34} = \pi \tag{7}$$

Hence $\psi_{12} + \psi_{34} = \pi$, which is the familiar result that the opposite angles of a cyclic quadrilateral are supplementary.

2.2 General Theorem

The following general theorem may be proved by an analogous approach to that employed in the above section.

Theorem 2.1. *Given a cyclic 2n-gon with winding number W about the circumcircle's center, and n ordered pairs $(a_1,b_1)\dots(a_n,b_n)$ such that*

$\{a_i\} \cup \{b_i\} = \{1\dots 2n\}$, $b_i > a_i$ and $b_i - a_i$ is odd for each i

$$\sum_{i=1}^{n} (-1)^{b_i} \delta_{a_i b_i} = W\pi \tag{8}$$

Proof. Let $P_{i,j} = -2$ where $j = a_i$ or $j = b_i$ and 0 otherwise, let $Q_1 = 2\pi W$. For $i > 1$, let $Q_j = 2\delta_{a_i,b_i}$ if $j = b_i$ and 0 otherwise.

Analogous to equation (2) we have:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 1 & P_{1,1} & \cdots & P_{n,1} \\ 1 & 1 & \cdots & 0 & 0 & P_{1,2} & \cdots & P_{n,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 1 & P_{1,2n} & \cdots & P_{n,2n} \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{2n-1} \\ \theta_{2n} \\ \phi_1 \\ \vdots \\ \phi_n \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \\ \vdots \\ Q_{2n} \end{pmatrix} \tag{9}$$

Triangulation yields a matrix equation analogous to that of (4(

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & R_1 & \cdots & R_n \end{pmatrix} \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_{2n} \\ \phi_1 \\ \vdots \\ \phi_n \end{pmatrix} = \begin{pmatrix} \vdots \\ S \end{pmatrix} \tag{10}$$

where

$$R_i = \sum_{j=1}^{2n} (-1)^j P_{i,j} \tag{11}$$

and

$$S = \sum_{j=1}^{2n} (-1)^j Q_j \tag{12}$$

Hence

$$R_i = (-1)^{b_i} + (-1)^{a_i} \tag{13}$$

and

$$S = \sum_{i=1}^{n} (-1)^{b_i} 2\delta_{a_i b_i} - 2W\pi \tag{14}$$

As $b_i - a_i$ is odd, $R_i = 0$ for all i. Hence for consistency $S = 0$. Hence the result. □

3 Automated Discovery of Cyclic Polygon Theorems

We describe a mechanized process for automatic theorem generation for angles in cyclic polygons. Each theorem will establish a relationship between angles in a cyclic polygon. The process breaks down into three steps. First a set of side pairs is chosen which satisfy the criteria of Theorem 2.1. Secondly, a specific location is decided for the vertices of the polygon.and a geometry diagram created. Angles between the side pairs are given names and drawn on the diagram. Thirdly, (8) is expressed in terms of these angle names, yielding the conclusion of our theorem statement.

3.1 Choosing Side Pairs

In order to use theorem 2.1 to create cyclic polygon theorems, we need to first find a set of ordered pairs $(a_1, b_1) \ldots (a_n, b_n)$ such that $\{a_i\} \cup \{b_i\} = \{1, \ldots, 2n\}$ and $|b_i - a_i|$ is odd for each i

We then sort the elements so that $a_i < b_i$. Each pair (a_i, b_i) contains an odd and an even integer. Hence any set of pairs corresponds to a 1-1 mapping between $A = \{1, 3, \ldots, 2n-1\}$ and $B = \{2, 4, \ldots, 2n\}$. A random such mapping is selected. Applying theorem 2.1 yields an expression for a linear combination of the differences in direction $\delta_{a_i b_i}$.

The number of such mappings is $n!$. However, when considered as pairings between sides of a cyclic polygons, there are many which are simply rotated or reflected images. The number of non-isomorphic patterns for $n = 2$ to 7 are $1, 3, 5, 17, 53, 260$ [3]. Figure 3 depicts these patterns for n equal to 3 and 4.

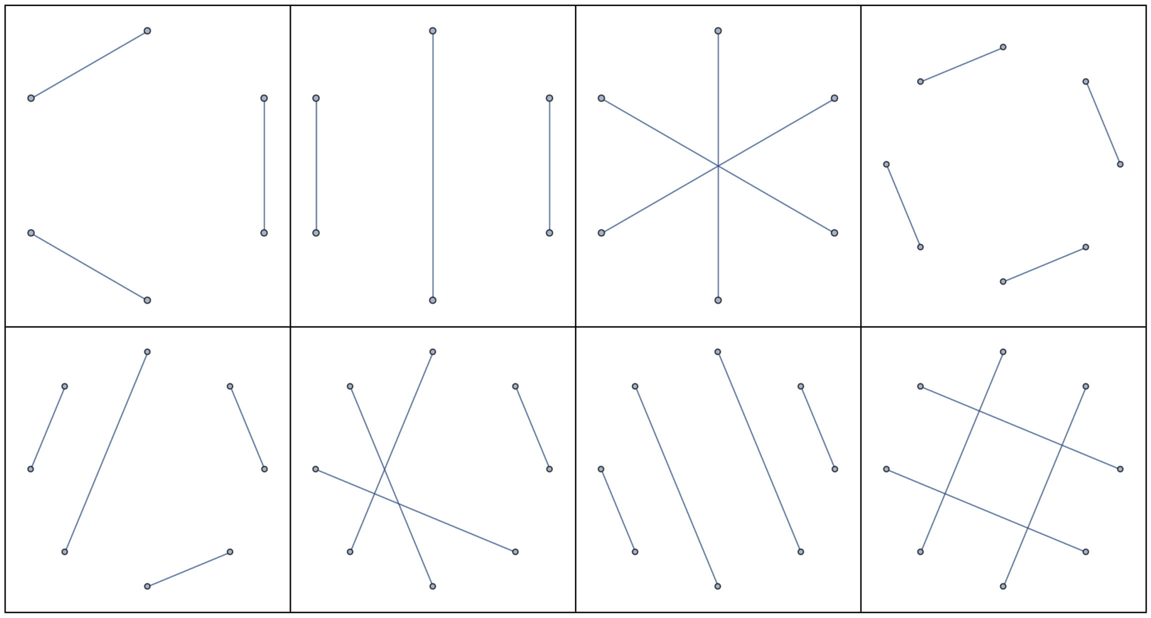


Figure 3: Graphs representing possible conforming sets of edge pairs for $n = 3,4$. Vertices on these graphs correspond to polygon sides. Edges indicate the side pairs whose angles will be specified

3.2 Diagram Creation

A diagram should be constructed containing the circle and the cyclic polygon. Points $q_{a_ib_i}$ should be computed and angles $\psi_{a_ib_i}$ should be marked on the diagram and named. We need to relate the named angles: $\psi_{a_ib_i}$ to the angles treated in our theorem: $\delta_{a_ib_i}$.

One straightforward approach is to create a polygon whose vertices are arranged clockwise around the circle, making a single circuit, but at non-regular intervals. Constraints may be set on the location of the polygon vertices to ensure that the angle between chosen side-pairs is not too small for their intersection to appear on the diagram. This intersection would be displayed, along with a named angle (Figure 5).

3.3 Geometric Angle Conclusion

The relation between δ_{ij} and ψ_{ij} depends on whether the circle center is on the same side of w_i and w_j and whether a rotation from w_i to w_j is clockwise or counter-clockwise.

When the polygon is convex and has a winding number of 1 around the circumcenter, the formula is as follows

Let $s = sgn(w_i \wedge w_j)$ and

$$\delta_{ij} = \pi - s \cdot (\psi_{ij}) \tag{15}$$

4 Example

As an example, we start with one of the sets of pairings for the decagon: $\{(1,2),(3,10),(4,7),(5,8),(6,9)\}$ (Figure 4).

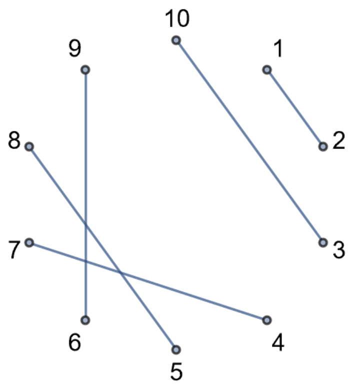


Figure 4: Graph for the set of side pairings $\{(1,2),(3,10),(4,7),(5,8),(6,9)\}$

Applying Theorem 2.1 to this set of side pairings, and assuming a winding number of 1 gives the following:

$$\delta_{1,2} + \delta_{3,10} - \delta_{4,7} + \delta_{5,8} - \delta_{6,9} = \pi \tag{16}$$

A cyclic decagon is drawn, where the vertices are placed at random subject to constraints that the polygon does not self-intersect, the sides are not too small, and the winding number is 1. Angles between the five side pairs are marked and named. In the case of non-contiguous sides, the sides are extended in the diagram to their intersection (Figure 5). As their angles are counter-clockwise, we have:

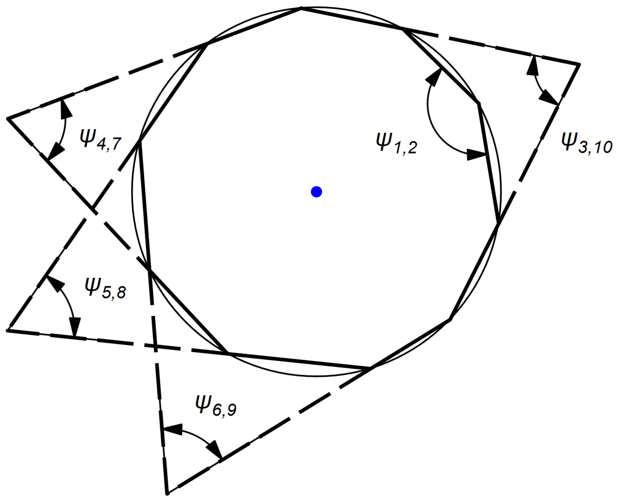


Figure 5: Cyclic decagon with angles marked for side pairings $\{(1,2),(3,10),(4,7),(5,8),(6,9)\}$

$$\begin{aligned}\delta_{1,2} &= \pi - \psi_{1,2}\\ \delta_{4,7} &= \pi - \psi_{4,7}\\ \delta_{5,8} &= \pi - \psi_{5,8}\\ \delta_{6,9} &= \pi - \psi_{6,9}\end{aligned} \tag{17}$$

However, the angle for side pair (3,10) is clockwise, so

$$\delta_{3,10} = \pi + \psi_{3,10} \tag{18}$$

Substituting these values into (16) gives the following

$$\pi - \psi_{1,2} + \pi + \psi_{3,10} - (\pi - \psi_{4,7}) + \pi - \psi_{5,8} - (\pi - \psi_{6,9}) = \pi \tag{19}$$

which can be presented as this theorem conclusion

$$\psi_{3,10} + \psi_{4,7} + \psi_{6,9} = \psi_{1,2} + \psi_{5,8} \tag{20}$$

We note that it would be possible to draw a polygon such that the angle between lines 3 and 10 is clockwise. In this case (21) becomes

$$\delta_{3,10} = \pi - \psi_{3,10} \tag{21}$$

and the theorem conclusion (20) becomes

$$\psi_{4,7} + \psi_{6,9} = \psi_{1,2} + \psi_{5,8} + \psi_{3,10} \tag{22}$$

5 An Automated Problem Generator

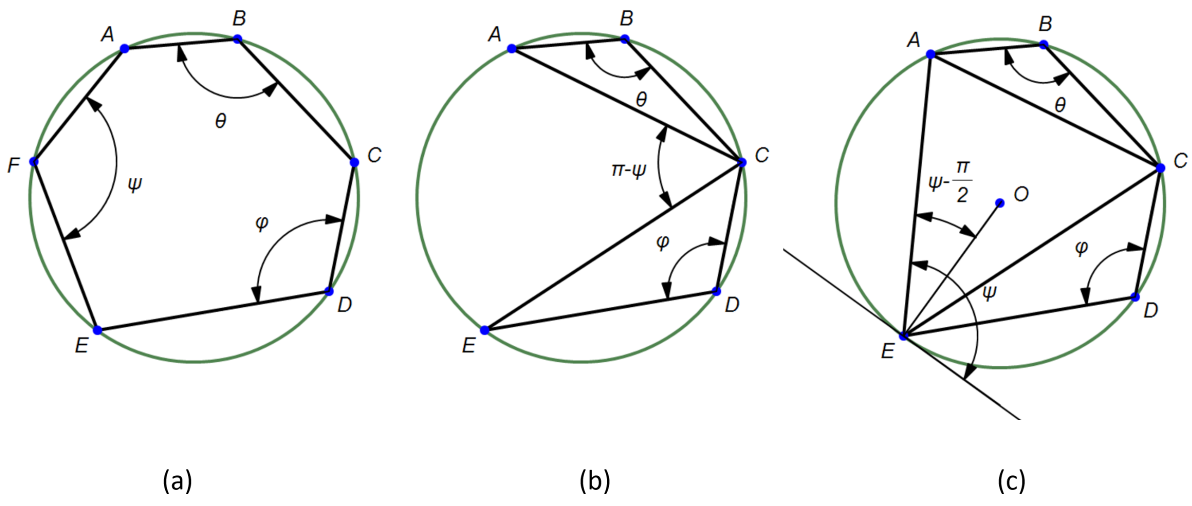


Figure 6: (a) Given convex cyclic hexagon $ABCDEF$, $\angle ABC + \angle CDE + \angle EFA = 2\pi$. (b) Given convex cyclic pentagon $ABCDE$, $\angle ABC + \angle CDE - \angle ECA = \pi$. (c) Given convex cyclic hexagon $ABCDE$ with center O, $\angle ABC + \angle CDE + \angle OEA = \frac{3\pi}{2}$.

An automated problem generator based on the above approach was created [5], and introduces further elaborations. First, rather than the vertices of the polygon lying in order around the circle, we allow any permutation of the vertices. With any but the identity permutation, the polygon will be self-intersecting and will, in fact, be made up of some subset of the sides and diagonals of a convex polygon.

A theorem for a polygon with an odd number of sides may be constructed from a polygon with one more side by merging a pair of points in the larger polygon. For example, figure 6 (a) shows a theorem linking three alternating angles of cyclic polygon $ABCDEF$. Figure 6(b) shows the result of merging points F and C of the hexagon. Edges EF and FA now become the diagonals EC and CA of a cyclic pentagon $ABCDE$, and the theorem relates two angles of the pentagon and an angle between two of its diagonals. If the points which are merged share an edge of the larger polygon, that edge will be replaced by a line joining the center of the circle to the merged point, and the theorem statement will be altered by $\frac{\pi}{2}$. Figure 6(c) shows the result of merging points F and E of the hexagon $ABCDEF$. Angle AFE becomes, in the limit, the angle between AE and the tangent to the circle at E, which differs from OE by $\frac{\pi}{2}$ (where O is the circle center).

In addition to presenting the theorems, our problem generator [5] can create a human readable proof by propagating a handful of simple angle generation rules. A single step of the proof generator proceeds by finding all the angles determined by the application of the following set of rules to the known angles:

1. If AD is between AB and AC then $\angle BAC = \angle DAB + \angle DAC$.
2. The angles of a triangle add to π.
3. Two angles which form a line add to π.
4. Two angles on the same side of the same chord of a circle are the same.
5. Opposite angles of a cyclic quadrilateral add to π.
6. The angle at the center of a circle is twice the angle on the same chord at the circumference.
7. If AB is a chord of circle centered O then $\angle OAB = \angle OBA = \frac{1}{2}(\pi - \angle AOB)$.

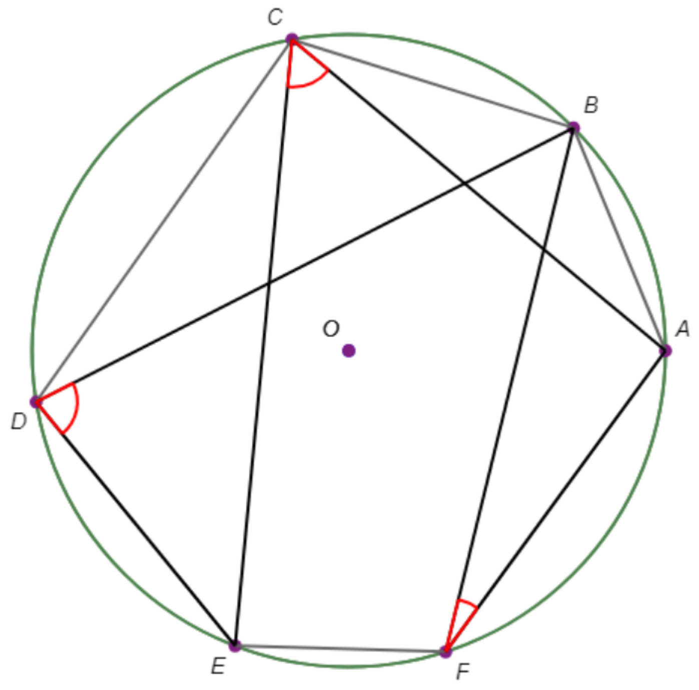


Figure 7: Automatically generated diagram with problem statement: Let $ABCDEF$ be a cyclic hexagon with center O. Prove that $BDE = ACE + AFB$

Our theorems can all be expressed as a linear combination of the angles and π. The result of each angle generation rule is a linear combination of its inputs and π. Hence a convenient way of storing the computed value of any angle is as a vector of coefficients of the given angles and π. The proof generator is started with the known angles and proceeds to compute values for all angles which can be computed from those by application of the above rules (breadth first). The angle value is stored, as a vector of coefficients of the known angles and π. Along with the value, a reference is stored to the rule which was fired, and the arguments it was applied to. If the process finds a new value for a given angle, this new value can be equated to the given angle yielding a linear expression. The sequence of rule applications used to reach the final expression may be turned into the words of a proof.

For example, in figure 7, we set $\angle BDE = (1,0,0,0)$, $\angle ACE = (0,1,0,0)$, $\angle AFB = (0,0,1,0)$.

Applying (4) to $\angle AFB$ gives $\angle ACB = (0,0,1,0)$.

Applying (4) to $\angle BDE$ gives $\angle BCE = (1,0,0,0)$.

Applying (1) to $\angle BCE$ and $\angle ACE$ gives $\angle ACB = (1,-1,0,0)$.

But we already have $\angle ACB = (0,0,1,0)$ so our linear relation can be expressed as $(1,-1,-1,0) = 0$.

This would result in the following proof.

Let $BDE = x$. Let $ACE = y$. $Let AFB = z$.

As AFB and ACB stand on the same chord, $ACB = AFB$, so $ACB = z$.

As BDE and BCE stand on the same chord, $BCE = BDE$, so $BCE = x$.

As $ACE = y$, $ACB = x - y$.

But $ACB = z$, so $x - y = z$, or $x = y + z$, or $BDE = ACE + AFB$.

If the proof generator does not run to completion, additional geometry is added to the diagram in two phases. First, missing line segments are added, which allow additional angles to be derived (angles are only computed when the line segments joining their vertices are present). A second step is for an additional angle to be specified as known, and given a name. As long as the new angle is not in the span of the known angles, it cannot show up in the eventual linear relation, and hence is guaranteed to simplify out in the course of the proof.

For example the following is the proof generated for the theorem stated in figure 8.

Theorem 5.1. *Let ABCDE be a cyclic pentagon with center O. Let F be the intersection of AE and DC. Let G be the intersection of OE and CB. Let H be the intersection of ED and BA.* $BGE + AHE = DFE + 90$.

Proof. Draw line BE.

Let $\angle DFE = x$. Let $\angle BGE = y$. Let $\angle AHE = z$.

Let $\angle AEH = w$.

As$\angle AHE = z$, $\angle EAH = 180 - z - w$.

As $\angle EAH = 180 - z - w$, $\angle EAB = z + w$.

As $\angle AEH = w$, $\angle HEF = 180 - w$.

As $\angle FEH = 180 - w$, $\angle FED = w$.

As $\angle DEF = w$, $\angle EDF = 180 - x - w$.

As $\angle EDF = 180 - x - w$, $\angle EDC = x + w$.

As $CDEB$ is a cyclic quadrilateral, $\angle CBE = 180 - \angle CDE$, so $\angle CBE = 180 - x - w$.

As $\angle EBG = 180 - x - w$, $\angle BEG = x + w - y$.

As triangle BEO is isosceles, $\angle BOE = 2y - 2x - 2w + 180$.

As $\angle BOE$ is at the center of a circle on the same chord, but in the opposite direction to $\angle BAE$, $\angle BOE = 360 - 2BAE$, so $\angle BAE = x + w - y + 90$.

But $\angle BAE = z + w$, so $x + w - y + 90 = z + w$, or $x + 90 = y + z$, or $\angle DFE + 90 = \angle BGE + \angle AHE$. □

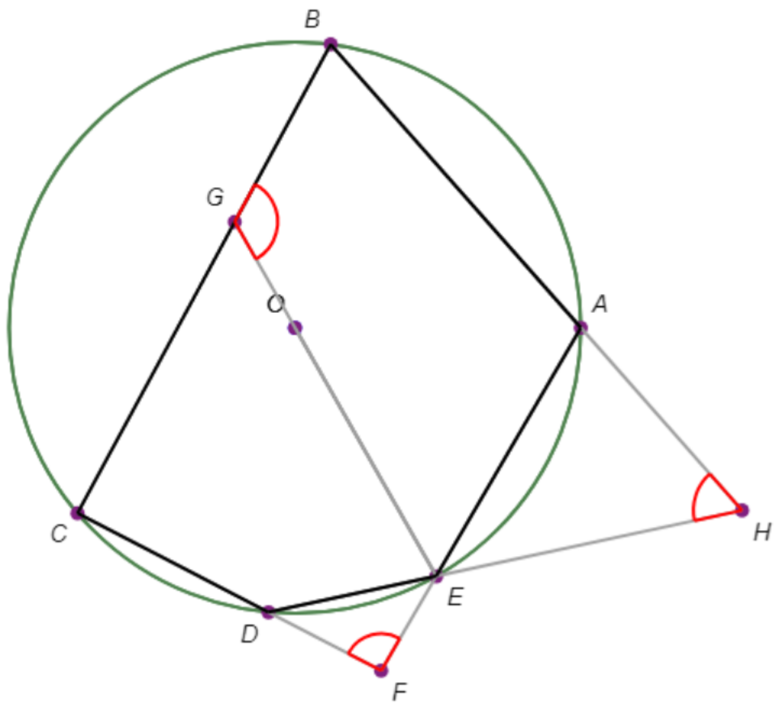


Figure 8: Let $ABCDE$ be a cyclic pentagon with center O. Let F be the intersection of AE and DC. Let G be the intersection of OE and CB. Let H be the intersection of ED and BA. Prove that $BGE + AHE = DFE + 90$.

Automated proofs can be simply adapted to provide step-by-step solutions to problems framed in terms of determining an unknown angle, either numerically or algebraically.

Our generator can be set to create problem collections. It is important for such collections to eliminate problems which are simply rotated or reflected duplicates. For a hexagon, for example, we see from figure 3 that there are 3 different pair patterns. When we combine this with the different permutations of 6 points, and remove rotated and reflected duplicates, we end up with 49 distinct diagrams.

For the pentagon, we can choose any pair of points to merge, but again need to avoid duplication by keeping track not only of the diagrams which have been produced, but also of their rotated and reflected images. The pentagon yields 54 distinct diagrams. [4] shows the complete collection of pentagon and hexagon theorems generated in this way.

A heptagon theorem may be generated from an octagon theorem by merging a pair of points. A hexagon theorem may be formed from the heptagon theorem by merging a further pair of points. Admitting such "four angle" hexagon theorems expands the number of possibilities to hundreds. Admitting octagons and heptagons expands the possible theorem count to thousands. [6] is a collection of 200 random theorems and their proofs generated by [5] .

6 Conclusion

The angles considered in this paper are not the full angles which have their own place in automated deduction in geometry [1], and which are largely impervious to changes in the diagram caused by simply positioning the points in different locations. The angles we use are well defined: an angle ABC means the non-reflex undirected angle defined by the line segments AB and BC. While the angle is well defined, its relationship to the directed angle, which is the basis of, for example, (16) is highly dependent on the relative location of the points on the diagram. When an angle passes through π, the relation between the ordered angle and the unordered non-reflex angle changes and, in our setting, the theorem statement. This is both a blessing and a curse. A curse because it is difficult to give a succinct general theorem statement without a complicated set of side conditions. In the context of automated problem generation, however, the curse becomes a blessing, as it proliferates the number of distinct examples which may be generated.

From a pedagogical standpoint, the problems generated have several advantages. They can all be solved by applying a small set of geometric facts, however some ingenuity in applying them is called for. Algebraically, the problems require only the manipulation of linear equations. Problems can be phrased as the determination of unknown quantities, either numerically or symbolically, or as proof problems. Finally the availability of machine generated human-readable proofs gives a convenient means of scaffolding.

References

[1] Shang-Ching Chou, Xiao-Shan Gao & Jing-Zhong Zhang (1996): *Automated generation of readable proofs with geometric invariants. II. Theorem proving with full-angles*. Journal of Automated Reasoning 17(3), pp. 349–370, doi:10.1007/BF00283134.

[2] B. D. Saunders (2015): *Matrices with two nonzero entries per row*. In: Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation, pp. 323–330, doi:10.1109/TIT.1986.1057137.

[3] N. J. A. Sloane: *Sequences A357442 in "The On-Line Encyclopedia of Integer Sequences."*. Available at `https://oeis.org/A357442`.

[4] P. H. Todd (2023): *The Complete Set of Automatically Generated Angle Problems for Cyclic Hexagon and Pentagon. Saltire Technical Report 23-1*. Available at `https://saltire.com/download/TR2023-1.pdf`.

[5] P. H. Todd (2023): *Cyclic Polygon Angle Problem Generator*. Available at `https://www.saltire.com/ProblemGenerator`.

[6] P. H. Todd (2023): *A Set of 200 Cyclic Polygon Angle Problems. Saltire Technical Report 23-2*. Available at `https://saltire.com/download/TR2023-2.pdf`.

[7] P. H. Todd & D. Aley (2023): *A program to create new geometry proof problems*. Annals of Mathematics and Artificial Intelligence, pp. 779–795, doi:10.1007/s10472-023-09854-1.

[8] Philip Todd (2023): *Automated Discovery of Angle Theorems*. Annals of Mathematics and Artificial Intelligence, doi:10.1007/s11786-020-00490-0.

Improving Angular Speed Uniformity by Piecewise Radical Reparameterization

Hoon Hong
Department of Mathematics
North Carolina State University
Box 8205, Raleigh, NC 27695, USA
hong@ncsu.edu

Dongming Wang
LMIB – IAI – School of Mathematical Sciences
Beihang University
Beijing 100191, China
Dongming.Wang@cnrs.fr

Jing Yang*
SMS – HCIC – School of Mathematics and Physics
Center for Applied Mathematics of Guangxi
Guangxi Minzu University
Nanning 530006, China
yangjing0930@gmail.com

For a rational parameterization of a curve, it is desirable that its angular speed is as uniform as possible. Hence, given a rational parameterization, one wants to find *re*-parameterization with better uniformity. One natural way is to use *piecewise* rational reparameterization. However, it turns out that the piecewise rational reparameterization does *not* help when the angular speed of the given rational parameterization is zero at some points on the curve. In this paper, we show how to overcome the challenge by using piecewise *radical* reparameterization.

1 Introduction

Parametric curves and surfaces are fundamental objects that are most frequently used in computer aided geometric design. A given curve or surface may have many different parameterizations, of which some may possess better properties and thus are more suitable for certain applications than the others. Thus, one often needs to convert one parameterization into another, i.e., to *re*-parameterize the given parameterization (see, e.g., [1, 2, 3, 5, 6, 7, 8, 9, 10]). In this paper, we focus our investigation on an important class of parameterizations, called uniform (angular-speed) parameterizations, where the distribution of points are determined by the local curvature and show how to construct such reparameterizations for a specific class of curves.

Uniform parameterization has been studied in a series of papers (see [4, 6, 8, 11, 14, 13, 12] and references therein). The authors have defined a function of angular speed uniformity to measure the quality of any given parameterization of a plane curve and proposed a method to compute its uniform reparameterization. However, the computed reparameterization is irrational in most cases (with straight lines as exceptions). For the sake of efficiency, a framework has been proposed for the computation of rational approximations of uniform parameterizations [4]. Four different methods of reparameterization (i.e., optimal reparameterization with fixed degree, C^0 and C^1 optimal piecewise reparameterization, and nearly optimal C^1 piecewise reparameterization) have been integrated into this framework. They have also been generalized to compute uniform quasi-speed reparameterizations of parametric curves in n-dimensional space.

*Corresponding author.

However, there is still a major challenge: all the above-mentioned methods do *not* work well when the angular speed of the given rational parameterization is zero at some points on the curve. This is due to an intrinsic property of the angular speed function [13]:

Let ω_p be an angular speed function of a curve p and r be a proper transformation. Then

$$\omega_{p\circ r} = (\omega_p \circ r)\cdot r'. \tag{1}$$

Uniformizing the angular speed can be seen as modifying the angular speed value at each point under the constraint (1) iteratively until all the values are equal to the average. However, the constraint indicates that $\omega_{p\circ r}$ will never reach the average value for any rational r' when $\omega_p(t) = 0$ for some t.

In this paper, we propose to overcome the challenge by using *radical* transformations instead of rational ones. We show that radical transformations allow one to increase the angular speed toward the average value at the points where the angular speed is zero. Then we adapt the idea of piecewise Möbius transformation from [11] and the strategies in [12] to optimally improve the uniformity of angular speed.

Experiments show that the proposed approach can improve the angular speed uniformity significantly when the angular speed of the given parameterization vanishes at some point on the curve.

The rest of the paper is structured as follows. In Section 2, we formulate the problem precisely. For this, we also introduce all the needed notations and notions. In Section 3, we develop mathematical theory to tackle the problem. In particular, we show how to use piecewise radical transformation to transform an angular speed function with zeros into one without zero in such a way that the parameters involved are also optimized. In Section 4, we summarize the theoretical results into an algorithm and illustrate its performance on an example. In Section 5, we briefly discuss implementational issues/suggestions when floating point arithmetic is used.

2 Problem

Consider a regular parametric curve

$$p = (x_1(t),\ldots,x_n(t)) : \mathbb{R} \mapsto \mathbb{R}^n.$$

Its *angular speed* ω_p is given by the following expression (see [4]):[1]

$$\omega_p = \frac{\sqrt{\sum\limits_{1\le i<j\le n} \begin{vmatrix} x_i'' & x_j'' \\ x_i' & x_j' \end{vmatrix}^2}}{\sum\limits_{i=1}^{n} x_i'^2}. \tag{2}$$

Recall that the mean μ_p and the variation σ_p^2 of ω_p are given by

$$\mu_p = \int_0^1 \omega_p(t)\,dt, \quad \text{and} \quad \sigma_p^2 = \int_0^1 (\omega_p(t) - \mu_p)^2\,dt.$$

Definition 1 *The* angular speed uniformity u_p *of a parameterization p is defined as*

$$u_p = \begin{cases} \dfrac{1}{1+\sigma_p^2/\mu_p^2} & \textit{if } \mu_p \neq 0, \\ 1 & \textit{otherwise.} \end{cases} \tag{3}$$

[1]The concept of angular speed is defined in the same manner as the one in physics but for $p'(t)$. The reason is to make the angular speed independent of the origin.

Example 2 (Running) *Consider the parametric curve $p = (t, t^3)$. Then*

$$\omega_p = \frac{6t}{9t^4+1},$$

$\mu_p \doteq 1.249$ and $u_p \doteq 0.846$. The goal is to find a proper parameter transformation r over $[0,1]$ in order to increase the uniformity.

Recall the following results from [4]. For any proper parameter transformation r over $[0,1]$, we have

$$\omega_{p\circ r}(s) = (\omega_p \circ r)(s) \cdot r'(s) \tag{4}$$

and

$$u_{p\circ r} = \frac{\mu_p^2}{\eta_{p,r}}, \quad \text{where} \quad \eta_{p,r} = \int_0^1 \frac{\omega_p^2}{(r^{-1})'}(t)\,dt. \tag{5}$$

By [13, Theorem 2], one can construct a uniform reparameterization from p, but such a reparameterization is irrational in most cases. Therefore, we proposed several methods in [4] to improve the angular speed uniformity by computing piecewise rational reparameterizations. However, those methods are not applicable to curves whose angular speed may vanish over $[0,1]$. Intuitively speaking, uniformizing the angular speed over $[0,1]$ can be viewed as getting all the values of $\omega_p(t)$ (for all $t \in [0,1]$) as close to μ_p as possible.

If r is a continuous rational function over $[0,1]$, then r' is bounded. Suppose that $\omega_p(t_0) = 0$ for some $t_0 \in [0,1]$ and $\mu_p \neq 0$. Then by (4), there must exist some $s_0 \in [0,1]$ such that $\omega_{p\circ r}(s_0) = 0$, which is not close to μ_p at all. This makes rational proper parameter transformations invalid. In what follows, we resort to radical transformations and develop a new approach to uniformize the angular speed of parametric curves which has zeros over $[0,1]$.

Let p be a parametric curve. Without loss of generality, we assume that

$$p'(t) = (x_1'(t), \ldots, x_n'(t)) = \Big(\frac{X_1(t)}{W(t)}, \cdots, \frac{X_n(t)}{W(t)}\Big),$$

where $X_i(t), W(t) \in \mathbb{R}[t]$ and $\gcd(X_1(t), \ldots, X_n(t), W(t)) = 1$. One can verify that

$$\omega_p = \frac{\sqrt{F}}{\sum_i X_i^2}, \quad \text{where} \quad F = \sum_{i \neq j} \begin{vmatrix} X_i' & X_j' \\ X_i & X_j \end{vmatrix}^2.$$

Let F be written as $F = \big(\prod_{i=0}^k (t-\tilde{t}_i)^{2\mu_i}\big)\,\zeta(t)$ for positive k. Note that $\{\tilde{t}_i : \tilde{t}_i < \tilde{t}_{i+1} \text{ for } 0 \le i < k\}$ contains all the zeros of F over $[0,1]$. It is allowed that some t_i's are not the roots of F. The positive integer $\mu_i \in \mathbb{N}$ is called the *multiplicity* of $\tilde{t}_i$ in ω_p and denoted by $\mathtt{mult}(\omega_p, \tilde{t}_i)$. If $\omega_p(\tilde{t}_i) \neq 0$, then $\mathtt{mult}(\omega_p, \tilde{t}_i) = 0$.

Let

$$T = (t_0, \ldots, t_N), \quad S = (s_0, \ldots, s_N), \quad Z = (z_0, \ldots, z_N), \quad \alpha = (\alpha_0, \ldots, \alpha_{N-1})$$

be sequences such that

- $0 = t_0 < \cdots < t_N = 1$, $0 = z_0 < \cdots < z_N = 1$, $0 = s_0 < \cdots < s_N = 1$, $0 < \alpha_i < 1$;
- at most one of $\omega_p(t_i) = 0$ and $\omega_p(t_{i+1}) = 0$ holds for $0 \le i < N$, that is, the successive appearance of two zeros of ω_p are not allowed;
- the multiplicity of t_i in ω_p is μ_i;

- $\omega_p(t) \neq 0$ for all $t \in (t_i, t_{i+1})$.

Definition 3 (Elementary Piecewise Radical Transformation) *Let p be a parametric curve with T, S defined above. Then φ is called an* elementary piecewise radical transformation *associated to p if φ has the following form:*

$$\varphi(s) = \begin{cases} \vdots & \\ \varphi_i(s) & \text{if} \quad s \in [s_i, s_{i+1}], \\ \vdots & \end{cases}$$

where

$$\varphi_i(s) = \begin{cases} t_i + \Delta t_i \sqrt[\mu_i+1]{\tilde{s}} & \text{if } \omega_p(t_i) = 0; \\ t_i + \Delta t_i (1 - \sqrt[\mu_{i+1}+1]{1-\tilde{s}}) & \text{if } \omega_p(t_{i+1}) = 0; \\ t_i + \Delta t_i \cdot \tilde{s} & \text{otherwise,} \end{cases} \tag{6}$$

and $\Delta t_i = t_{i+1} - t_i$, $\Delta s_i = s_{i+1} - s_i$, $\tilde{s} = (s - s_i)/\Delta s_i$.

Remark 4

1. *It can be verified that $\varphi(s_i) = t_i$ and $\varphi(s_{i+1}) = t_{i+1}$, which implies that φ is with C^0 continuity.*
2. *It is allowed that more than one intermediate point lie between two zeros of ω_p because it can reduce the number of radical pieces and thus enhance the efficiency of generating points with the new parameterization.*

It can be shown that $\omega_{p\circ\varphi}(s) \neq 0$ (see Theorem 7). Next let $q = p \circ \varphi$ and thus q has no inflation point. We adapt the reparameterization methods from [12] to increase the uniformity of ω_q to any value close to 1. For this purpose, we recall the following piecewise Möbius transformation.

Definition 5 (Piecewise Möbius Transformation) *Let p be a parametric curve with S, Z, α defined above. Then m is called a* piecewise Möbius transformation *associated to p if m has the following form:*

$$m(z) = \begin{cases} \vdots & \\ m_i(z) & \text{if} \quad z \in [z_i, z_{i+1}], \\ \vdots & \end{cases}$$

where

$$m_i(z) = s_i + \Delta s_i \cdot \frac{(1-\alpha_i)\tilde{z}}{(1-\alpha_i)\tilde{z} + \alpha_i(1-\tilde{z})} \tag{7}$$

and $\Delta z_i = z_{i+1} - z_i$, $\Delta s_i = s_{i+1} - s_i$, $\tilde{z} = (z - z_i)/\Delta z_i$.

The problem addressed in this paper may be formulated as follows.

Problem 6 *Given a parametric curve p with $\omega_p(t) = 0$ for some $t \in [0,1]$, find a radical piecewise transformation φ and an optimal piecewise Möbius transformation m over $[0,1]$ such that*

- $u_{p\circ\varphi\circ m} \doteq 1$;
- $\forall s \in [0,1]$, $\omega_{p\circ\varphi\circ m}(s) \neq 0$.

3 Theory

3.1 Property of φ

Theorem 7 *For any $s \in [0,1]$, $\omega_{p\circ\varphi}(s) \neq 0$.*

Proof: Taking derivative of φ_i, we have

$$\varphi_i'(s) = \begin{cases} \dfrac{\Delta t_i}{\mu_i+1} \cdot \dfrac{1}{\sqrt[\mu_i+1]{\tilde{s}^{\mu_i}}} \cdot \dfrac{1}{\Delta s_i} & \text{if } \omega_p(t_i) = 0; \\ \dfrac{\Delta t_i}{\mu_{i+1}+1} \cdot \dfrac{1}{\sqrt[\mu_{i+1}+1]{(1-\tilde{s})^{\mu_{i+1}}}} \cdot \dfrac{1}{\Delta s_i} & \text{if } \omega_p(t_{i+1}) = 0; \\ \dfrac{\Delta t_i}{\Delta s_i} & \text{otherwise.} \end{cases}$$

Next we show that in the above three cases, $\omega_{p\circ\varphi}(s) \neq 0$.

Case 1: $\omega_p(t_i) = 0$.

Assume that $\mu_i = \texttt{mult}(\omega_p, t_i)$. Then ω_p can be written as

$$\omega_p = |t - t_i|^{\mu_i} \cdot \tilde{\zeta}(t),$$

where $\tilde{\zeta}(t) > 0$ for $t \in [t_i, t_{i+1}]$. Therefore,

$$\begin{aligned} \omega_{p\circ\varphi}(s) = {} & |\varphi_i(s) - t_i|^{\mu_i} \cdot (\tilde{\zeta} \circ \varphi_i)(s) \cdot \varphi_i'(s) \\ = {} & (\Delta t_i \tilde{s}^{\frac{1}{\mu_i+1}})^{\mu_i} \cdot (\tilde{\zeta} \circ \varphi_i)(s) \cdot \left[\frac{\Delta t_i}{\mu_i+1} \cdot \frac{1}{\sqrt[\mu_i+1]{\tilde{s}^{\mu_i}}} \cdot \frac{1}{\Delta s_i} \right] \\ = {} & \frac{\Delta t_i^{\mu_i+1}}{\mu_i+1} \cdot (\tilde{\zeta} \circ \varphi_i)(s) \cdot \frac{1}{\Delta s_i} \\ = {} & \frac{\Delta t_i^{\mu_i+1}}{\mu_i+1} \cdot \frac{1}{\Delta s_i} \cdot \tilde{\zeta}(t) \neq 0 \end{aligned}$$

for $s \in [s_i, s_{i+1}]$.

Case 2: $\omega_p(t_{i+1}) = 0$.

Assume that $\mu_{i+1} = \texttt{mult}(\omega_p, t_{i+1})$. Then ω_p can be written as

$$\omega_p = |t_{i+1} - t|^{\mu_{i+1}} \cdot \tilde{\zeta}(t),$$

where $\tilde{\zeta}(t) > 0$ for $t \in [t_i, t_{i+1}]$. Therefore,

$$\begin{aligned} \omega_{p\circ\varphi}(s) = {} & |t_{i+1} - \varphi_i(s)|^{\mu_{i+1}} \cdot (\tilde{\zeta} \circ \varphi_i)(s) \cdot \varphi_i'(s) \\ = {} & \frac{\Delta t_i}{\mu_{i+1}+1} \cdot \frac{1}{\sqrt[\mu_{i+1}+1]{(1-\tilde{s})^{\mu_{i+1}}}} \cdot \frac{1}{\Delta s_i} \\ = {} & [\Delta t_i (1-\tilde{s})^{\frac{1}{\mu_{i+1}+1}}]^{\mu_{i+1}} \\ & \cdot (\tilde{\zeta} \circ \varphi_i)(s) \cdot \frac{\Delta t_i}{\mu_{i+1}+1} \cdot \frac{1}{\sqrt[\mu_{i+1}+1]{(1-\tilde{s})^{\mu_{i+1}}}} \cdot \frac{1}{\Delta s_i} \\ = {} & \frac{\Delta t_i^{\mu_{i+1}+1}}{\mu_{i+1}+1} \cdot (\tilde{\zeta} \circ \varphi_i)(s) \cdot \frac{1}{\Delta s_i} \\ = {} & \frac{\Delta t_i^{\mu_{i+1}+1}}{\mu_{i+1}+1} \cdot \tilde{\zeta}(t) \cdot \frac{1}{\Delta s_i} \neq 0. \end{aligned}$$

Case 3: $\omega_p(t_i)\omega(t_{i+1}) \neq 0$.

Combining $\omega_p(t) \neq 0$ for $t \in [t_i, t_{i+1}]$, $\Delta t_i > 0$ and $\Delta s_i > 0$, we have

$$\omega_{p\circ\varphi}(s) = (\omega_p \circ \varphi)(s) \cdot \varphi'(s) = \omega_p(t) \cdot \frac{\Delta t_i}{\Delta s_i} \neq 0.$$

To sum up, we have $\omega_{p\circ\varphi}(s) \neq 0$ when $s \in [s_i, s_{i+1}]$. □

Example 8 (Continued from Example 2) *For the cubic curve* $p = (t, t^3)$ *whose angular speed is* $\omega_p = \dfrac{6t}{9t^4+1}$, *it is easy to see that* $t = 0$ *is a zero of* ω_p *with multiplicity* 1. *Let* $T = (0,1)$ *and* $S = (0,1)$. *Then the constructed* φ *is* $\varphi(s) = \sqrt{s}$. *It follows that*

$$\omega_{p\circ\varphi}(s) = (\omega_p \circ \varphi)(s) \cdot \varphi'(s) = \frac{6\sqrt{s}}{9s^2+1} \cdot \frac{1}{2\sqrt{s}} = \frac{3}{9s^2+1}$$

which is nonzero over $[0,1]$.

Remark 9 *It may be further deduced that* $\omega_{p\circ\varphi}(s)$ *is discontinuous at* $s = s_i$.

3.2 Choice of T

By Definition 3, T should contain all the zeros of ω_p over $[0,1]$ and some intermediate points in the subintervals separated by the zeros of ω_p. One question is how to choose intermediate points to make the uniformity improvement as significant as possible. In this subsection, we present a strategy similar to the one introduced in [12] for determining such points.

Recall [13, Theorem 2] which states that the uniformizing parameter transformation r_p of p satisfies

$$(r_p)^{-1} = \int_0^t \omega_p(\gamma)d\gamma/\mu_p.$$

Let φ be a piecewise radical transformation associated to p. If $(r_p)^{-1}$ and φ^{-1} share some common properties, we say informally that r_p and φ are similar to each other.

First of all, the following can be derived:

$$\varphi^{-1}(t) = \begin{cases} s_i + \Delta s_i \cdot \tilde{t}^{\mu_i+1} & \text{if } \omega_p(t_i) = 0; \\ s_i + \Delta s_i \cdot [1 - (1-\tilde{t})^{\mu_{i+1}+1}] & \text{if } \omega_p(t_{i+1}) = 0; \\ s_i + \Delta s_i \cdot \tilde{t} & \text{otherwise,} \end{cases} \tag{8}$$

where $\tilde{t} = (t - t_i)/\Delta t_i$. Furthermore,

$$[\varphi^{-1}]' = \begin{cases} \dfrac{\Delta s_i}{\Delta t_i} \cdot (\mu_i + 1) \cdot \tilde{t}^{\mu_i} & \text{if } \omega_p(t_i) = 0; \\ \dfrac{\Delta s_i}{\Delta t_i} \cdot (\mu_{i+1} + 1) \cdot (1-\tilde{t})^{\mu_{i+1}} & \text{if } \omega_p(t_{i+1}) = 0; \\ \dfrac{\Delta s_i}{\Delta t_i} & \text{otherwise;} \end{cases}$$

$$[\varphi^{-1}]'' = \begin{cases} \dfrac{\Delta s_i}{\Delta t_i^2} \cdot (\mu_i + 1)\mu_i \cdot \tilde{t}^{\mu_i - 1} & \text{if } \omega_p(t_i) = 0; \\ -\dfrac{\Delta s_i}{\Delta t_i^2} \cdot (\mu_{i+1} + 1)\mu_{i+1} \cdot (1-\tilde{t})^{\mu_{i+1}-1} & \text{if } \omega_p(t_{i+1}) = 0; \\ 0 & \text{otherwise.} \end{cases}$$

Note that φ has the properties listed below.

- $\varphi^{-1}(0) = 0$, $\varphi^{-1}(1) = 1$.
- φ_i^{-1} is monotonic over (t_i, t_{i+1}) because $(\varphi_i^{-1})'(t) \geq 0$ for all $t \in (t_i, t_{i+1})$; since φ^{-1} is continuous over $[0,1]$, φ^{-1} is monotonic over $[0,1]$.
- $[\varphi_i^{-1}]'$ is monotonic over (t_i, t_{i+1}) because $[\varphi_i^{-1}]''$ has a constant sign over (t_i, t_{i+1}).

The above properties indicate that φ is composed of some monotonically increasing convex or concave pieces. Moreover, it can be verified that

- $r_p^{-1}(0) = \int_0^0 \omega_p(\gamma)\,d\gamma/\mu_p = 0$, $r_p^{-1}(1) = \int_0^1 \omega_p(\gamma)\,d\gamma/\mu_p = 1$;
- r_p^{-1} is monotonic over $[0,1]$ because $(r_p^{-1})'(t) = \omega_p(t)/\mu_p \geq 0$.

One may observe that φ shares the first two properties with r_p. If r_p possesses the third property of φ, then r_p and φ are expected to be similar. This inspires us to divide $[0,1]$ into some monotonic intervals of $(r_p^{-1})'(t)$ (i.e., ω_p). Thus we may try to choose the intermediate t_i in T by solving

$$\omega_p(t_i)\omega_p'(t_i) = 0.$$

Note that $\omega_p(t)$ is nonnegative. Thus 0 is the local minimum value of ω_p. In this sense, T consists of all the local extreme points of ω_p and the two boundary points of the unit interval.

With the above operation, r_p is divided into some monotonically increasing/decreasing convex or concave pieces with each piece having a corresponding one in φ. Therefore, T can be obtained by collecting and inserting the zeros of ω_p and ω_p' into $[0,1]$ in order.

Example 10 (Continued from Example 8) *One may compute that*

$$\omega_p'(t) = -\frac{6\,(27t^4 - 1)}{(9t^4 + 1)^2}.$$

Then the solution of $\omega_p(t)\omega_p'(t) = 0$ *over* $[0,1]$ *gives us a partition of* $[0,1]$, *i.e.,*

$$T \doteq (0, 0.439, 1).$$

Furthermore, one may check that the multiplicities of t_0, t_1, t_2 *as roots of* ω_p *are* $1, 0$ *and* 0*, respectively.*

3.3 Determination of S

Once a partition T of $[0,1]$ is obtained, one can compute the sequence S in various ways. In this subsection, we present an optimization strategy for the computation of S.

When T is fixed, $u_{p\circ\varphi}$ becomes a function of s_i $(i = 1, \ldots, N-1)$. The following theorem provides a formula for computing the optimal values for s_i's.

Theorem 11 *The uniformity $u_{p\circ\varphi}$ reaches the maximum when*

$$s_i = s_i^* = \frac{\sum_{k=0}^{i-1}\sqrt{L_k}}{\sum_{k=0}^{N-1}\sqrt{L_k}}, \tag{9}$$

where

$$L_k = \begin{cases} \Delta t_k \int_{t_k}^{t_{k+1}} \dfrac{\omega_p^2(t)}{(\mu_k+1)\tilde{t}^{\mu_k}}\,dt & \text{if} \quad \omega_p(t_k)=0; \\ \Delta t_k \int_{t_k}^{t_{k+1}} \dfrac{\omega_p^2(t)}{(\mu_{k+1}+1)(1-\tilde{t})^{\mu_{k+1}}}\,dt & \text{if} \quad \omega_p(t_{k+1})=0; \\ \Delta t_k \int_{t_k}^{t_{k+1}} \omega_p^2(t)\,dt & \text{otherwise.} \end{cases} \tag{10}$$

The maximum value of $u_{p\circ\varphi}$ is

$$u_{p\circ\varphi}^* = \mu_p^2/\eta_{p,\varphi}^*, \quad \text{where} \quad \eta_{p,\varphi}^* = \left(\sum_{i=0}^{N-1}\sqrt{L_i}\right)^2.$$

Proof: Recall (5). Since μ_p is a constant for any given p, the problem of maximizing $u_{p\circ\varphi}$ can be reduced to that of minimizing

$$\eta_{p,\varphi} = \int_0^1 \frac{\omega_p^2}{(\varphi^{-1})'}(t)\,dt = \sum_{i=0}^{N-1}\int_{t_i}^{t_{i+1}} \frac{\omega_p^2}{(\varphi_i^{-1})'}(t)\,dt.$$

We first simplify each component in the above equation. Denote $\int_{t_i}^{t_{i+1}} \frac{\omega_p^2}{(\varphi_i^{-1})'}(t)\,dt$ by I_i. Note that

$$\frac{1}{(\varphi_i^{-1})'(t)} = \begin{cases} \dfrac{\Delta t_i}{\Delta s_i}\cdot\dfrac{1}{\mu_i+1}\cdot\dfrac{1}{\tilde{t}^{\mu_i}} & \text{if} \quad \omega_p(t_i)=0; \\ \dfrac{\Delta t_i}{\Delta s_i}\cdot\dfrac{1}{\mu_{i+1}+1}\cdot\dfrac{1}{(1-\tilde{t})^{\mu_{i+1}}} & \text{if} \quad \omega_p(t_{i+1})=0; \\ \dfrac{\Delta t_i}{\Delta s_i} & \text{otherwise.} \end{cases}$$

When $\omega_p(t_i)=0$,

$$I_i = \frac{\Delta t_i}{\Delta s_i}\cdot\frac{1}{\mu_i+1}\int_{t_i}^{t_{i+1}}\frac{\omega_p^2}{\tilde{t}^{\mu_i}}\,dt = L_i/\Delta s_i.$$

Similarly, when $\omega_p(t_{i+1})=0$,

$$I_i = \frac{\Delta t_i}{\Delta s_i}\cdot\frac{1}{\mu_{i+1}+1}\int_{t_i}^{t_{i+1}}\frac{\omega_p^2}{(1-\tilde{t})^{\mu_{i+1}}}\,dt = L_i/\Delta s_i.$$

When $\omega_p(t_i)\cdot\omega_p(t_{i+1}) \neq 0$,

$$I_i = \frac{\Delta t_i}{\Delta s_i}\cdot\int_i^{t_{i+1}}\omega_p^2\,dt = L_i/\Delta s_i.$$

It is obvious that $\eta_{p,\varphi} = \sum_{i=0}^{N-1} I_i > 0$; it increases to $+\infty$ when s_i approaches the boundary of the feasible set of parameters. Now we compute the extrema of $\eta_{p,\varphi}$. Let

$$\frac{\partial \eta_{p,\varphi}}{\partial s_i} = 0,$$

i.e.,

$$\frac{L_i}{\Delta s_i^2} - \frac{L_{i-1}}{\Delta s_{i-1}^2} = 0,$$

where L_i is as in (10). Solving the above equation, we obtain

$$\Delta s_i = \Delta s_i^* = \Delta s_0^* \sqrt{L_i/L_0}.$$

Note that $\sum_{i=0}^{N-1} \Delta s_i^* = 1$. Thus

$$\Delta s_0^* = \left(\sum_{k=0}^{N-1} \sqrt{L_k/L_0} \right)^{-1}, \quad s_i^* = \sum_{k=0}^{i-1} \Delta s_k^* = \frac{\sum_{k=0}^{i-1} \sqrt{L_k/L_0}}{\sum_{k=0}^{N-1} \sqrt{L_k/L_0}} = \frac{\sum_{k=0}^{i-1} \sqrt{L_k}}{\sum_{k=0}^{N-1} \sqrt{L_k}}.$$

Therefore,

$$\Delta s_i^* = \frac{\sqrt{L_i}}{\sum_{k=0}^{N-1} \sqrt{L_k}}.$$

Moreover, the optimal value of $\eta_{p,\varphi}$ is

$$\eta_{p,\varphi} = \eta_{p,\varphi}^* = \sum_{i=0}^{N-1} \frac{L_i}{\Delta s_i^*} = \sum_{k=0}^{N-1} \frac{L_i}{\frac{\sqrt{L_i}}{\sum_{i=0}^{N-1} \sqrt{L_k}}} = \left(\sum_{k=0}^{N-1} \sqrt{L_k} \right)^2,$$

from which it follows that the optimal value of $u_{p\circ\varphi}$ is

$$u_{p\circ\varphi} = u_{p\circ\varphi}^* = \frac{\mu_p^2}{\eta_{p,\varphi}^*} = \frac{\mu_p^2}{\left(\sum_{k=0}^{N-1} \sqrt{L_k}\right)^2}.$$

The proof is completed. □

Example 12 (Continued from Example 10) *By using* (10)*, we compute the values of* L_0 *and* L_1 *and obtain*

$$L_0 \doteq 0.439 \int_0^{0.439} \frac{\left(\frac{6t}{9t^4+1}\right)^2}{2 \cdot \frac{t-0}{0.439}} dt \doteq 0.276,$$

$$L_1 \doteq (1-0.439) \int_{0.439}^{1} \left(\frac{6t}{9t^4+1}\right)^2 dt \doteq 0.590.$$

By (9)*, we have* $s_1 = 0.406$*. Thus* $S \doteq (0, 0.406, 1)$*. Furthermore, one may calculate the optimal value of* $u_{p\circ\varphi}$ *and obtain* $u_{p\circ\varphi}^* \doteq 0.932$*.*

3.4 Determination of Z and α

Once a partition S of $[0,1]$ is obtained, one can compute the sequence Z. In this subsection, we give explicit formulae for the optimal values of Z and α which are directly computed from the sequence T. For this purpose, we first recall the the following result from [11].

Theorem 13 *Let q be a rational parameterization such that $\omega_q(s) \neq 0$ over $[0,1]$ and m be a piecewise Möbius transformation determined by S, Z and α. For a given sequence S, the uniformity $u_{q\circ m}$ reaches the maximum when*

$$\alpha_i = \alpha_i^* = \frac{1}{1+\sqrt{C_i/A_i}}, \quad z_i = z_i^* = \frac{\sum_{k=0}^{i-1}\sqrt{M_k}}{\sum_{k=0}^{N-1}\sqrt{M_k}}, \tag{11}$$

where

$$A_i = \int_{s_i}^{s_{i+1}} \omega_q^2 \cdot (1-\tilde{s})^2 ds, \qquad B_i = \int_{s_i}^{s_{i+1}} \omega_q^2 \cdot 2\tilde{s}(1-\tilde{s}) ds,$$

$$C_i = \int_{s_i}^{s_{i+1}} \omega_q^2 \cdot \tilde{s}^2 ds, \qquad M_k = \Delta s_k \left(2\sqrt{A_k C_k} + B_k\right).$$

Let m^* be the piecewise Möbius transformation determined by S, Z^* and α^*. Then the maximum value of $u_{q\circ m}$ is $u_{q\circ m^*} = \mu_q^2/\eta_{q,m^*}$ where

$$\eta_{q,m^*} = \left(\sum_{i=0}^{N-1}\sqrt{M_k}\right)^2.$$

Remark 14 *Let φ be an elementary radical transformation as in Definition 3 and $q = p\circ\varphi$. Note that*

$$\mu_q = \int_0^1 \omega_q ds = \int_0^1 \omega_{p\circ\varphi} ds = \int_0^1 (\omega_p\circ\varphi)(s)\cdot\varphi'(s) ds = \int_0^1 \omega_p dt = \mu_p.$$

Thus

$$u_{p\circ\varphi\circ m^*} = u_{q\circ m^*} = \mu_q^2/\eta_{q,m^*} = \mu_p^2 \Big/ \left(\sum_{i=0}^{N-1}\sqrt{M_k}\right)^2.$$

Let φ and q be defined as before. By Theorem 7, $\omega_q \neq 0$ over $[0,1]$. One may compute the optimal values of S, α and Z by Theorems 11 and 13. However, $p\circ\varphi$ is a composition of radical function and rational function and the composition will cause an increase of complexity because ω_q is radical. In what follows, we simplify the formulae for A_i, B_i and C_i with the goal of computing the values of A_i, B_i and C_i directly from p.

The formula of A_i $(0 \le i \le N-1)$ is derived via the following steps:

$$\begin{aligned}
A_i &= \int_{s_i}^{s_{i+1}} \omega_q^2 \cdot (1-\tilde{s})^2 ds \\
&= \int_{s_i}^{s_{i+1}} [(\omega_p\circ\varphi)(s)]^2 \cdot [\varphi'(s)]^2 \cdot (1-\tilde{s})^2 ds \\
&= \int_{s_i}^{s_{i+1}} [(\omega_p\circ\varphi)(s)]^2 \cdot [\varphi'(s)] \cdot (1-\tilde{s})^2 \left[\varphi'(s) ds\right] \\
&= \int_{t_i}^{t_{i+1}} \frac{\omega_p^2}{(\varphi^{-1})'}(t) \cdot \left(1 - \frac{\varphi^{-1}-s_i}{\Delta s_i}\right)^2 dt \qquad \text{by (8)} \\
&= \begin{cases}
\int_{t_i}^{t_{i+1}} \frac{\omega_p^2}{(\varphi^{-1})'}(t)\cdot(1-\tilde{t}^{\mu_i+1})^2 dt & \text{if } \omega(t_i)=0; \\
\int_{t_i}^{t_{i+1}} \frac{\omega_p^2}{(\varphi^{-1})'}(t)\cdot(1-\tilde{t})^{2(\mu_{i+1}+1)} dt & \text{if } \omega(t_{i+1})=0; \\
\int_{t_i}^{t_{i+1}} \frac{\omega_p^2}{(\varphi^{-1})'}(t)\cdot(1-\tilde{t})^2 dt & \text{otherwise;}
\end{cases}
\end{aligned}$$

$$= \begin{cases} \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \dfrac{\omega_p^2(t)}{(\mu_i+1)\tilde{t}^{\mu_i}} \cdot (1-\tilde{t}^{\mu_i+1})^2 dt & \text{if } \omega(t_i)=0; \\ \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \dfrac{\omega_p^2(t)}{\mu_{i+1}+1} \cdot (1-\tilde{t})^{\mu_{i+1}+2} dt & \text{if } \omega(t_{i+1})=0; \\ \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \omega_p^2(t) \cdot (1-\tilde{t})^2 dt & \text{otherwise.} \end{cases}$$

Similarly, we have

$$B_i = \begin{cases} \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \dfrac{\omega_p^2(t)}{\mu_i+1} \cdot 2\tilde{t}(1-\tilde{t}^{\mu_i+1}) dt & \text{if } \omega(t_i)=0; \\ \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \dfrac{\omega_p^2(t)}{\mu_{i+1}+1} \cdot 2[1-(1-\tilde{t})^{\mu_{i+1}+1}](1-\tilde{t}) dt & \text{if } \omega(t_{i+1})=0; \\ \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \omega_p^2(t) \cdot (1-\tilde{t})^2 dt & \text{otherwise;} \end{cases}$$

$$C_i = \begin{cases} \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \dfrac{\omega_p^2(t)}{\mu_i+1} \cdot \tilde{t}^{\mu_i+2} dt & \text{if } \omega(t_i)=0; \\ \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \dfrac{\omega_p^2(t)}{(\mu_{i+1}+1)(1-\tilde{t})^{\mu_{i+1}}} \cdot [1-(1-\tilde{t})^{\mu_{i+1}+1}]^2 dt & \text{if } \omega(t_{i+1})=0; \\ \dfrac{\Delta t_i}{\Delta s_i} \int_{t_i}^{t_{i+1}} \omega_p^2(t) \cdot (1-\tilde{t})^2 dt & \text{otherwise.} \end{cases}$$

Example 15 (Continued from Example 12) *With the above formulae and T, S as in Examples 10 and 12, one may obtain the following:*

$$\begin{aligned} &A_0 \doteq 0.258, \quad && B_0 \doteq 0.229, \quad && C_0 \doteq 0.193, \\ &A_1 \doteq 0.518, && B_1 \doteq 0.317, && C_1 \doteq 0.159. \end{aligned}$$

Thus

$$\begin{aligned} M_0 &\doteq 0.406(2\sqrt{0.258 \cdot 0.193} + 0.229) \doteq 0.274, \\ M_1 &\doteq (1-0.406)(2\sqrt{0.518 \cdot 0.159} + 0.317) \doteq 0.529. \end{aligned}$$

By Theorem 13, we obtain

$$\begin{aligned} \alpha_0 &\doteq 1/(1+\sqrt{0.193/0.258}) \doteq 0.536, \\ \alpha_1 &\doteq 1/(1+\sqrt{0.159/0.518}) \doteq 0.643, \end{aligned}$$

and $z_1^* \doteq 0.419$. *Thus* $\alpha \doteq (0.536, 0.643)$ *and* $Z \doteq (0, .419, 1)$. *One may further calculate*

$$u_{q\circ m^*} \doteq \frac{1.249^2}{(\sqrt{0.274}+\sqrt{0.529})^2} \doteq 0.997.$$

4 Algorithm

In this section, we summarize the above ideas and results as Algorithm 1 and illustrate how the algorithm works for the cubic curve in Example 2.

```
Algorithm 1: Optimal_Radical_Transformation

Input:   $p$, a rational parameterization of a plane curve.
Output:  $r$, the optimal piecewise radical transformation of $p$ such that $u_{p\circ r} > u_p$.
1.  Compute $\omega_p$ and $\mu_p$ using (2), $u_p$ using (3) and $\omega'_p$.
2.  Solve $\omega_p\omega'_p = 0$ and get $T$.
3.  Compute $S$, $Z$, $\alpha$ and $u$ using (9) and (11).
4.  Construct $\varphi$ with $T$, $S$ and $m$ with $S$, $Z$, $\alpha$ using (6) and (7).
5.  $r \leftarrow \varphi \circ m$.
6.  Return $r$.
```

Example 16 (Continued from Example 15) *Given $p = (t, t^3)$, after the above calculation, one may obtain*

$$T \doteq (0, 0.439, 1),\;\; S \doteq (0, 0.406, 1),\;\; Z \doteq (0, .419, 1),\;\; \alpha \doteq (0.536, 0.643).$$

Then one may construct φ with T and S, and m with S, Z and α, and obtain

$$\varphi \doteq \begin{cases} 0.688\sqrt{s} & \text{if } \; 0.000 \le s \le 0.406; \\ 0.055 + 0.945s & \text{if } \; 0.406 \le s \le 1.000; \end{cases}$$

$$m \doteq \begin{cases} \dfrac{-0.450z}{0.172z - 0.536} & \text{if } \; 0.000 \le z \le 0.049; \\ -\dfrac{0.165z + 0.192z}{0.492z - 0.849} & \text{if } \; 0.419 \le z \le 1.000. \end{cases}$$

Then the optimal transformation r is constructed below.

$$r = \varphi \circ m \doteq \begin{cases} 0.462\sqrt{-\dfrac{z}{0.172z - 0.536}} & \text{if } \; 0.000 \le z \le 0.419; \\ \dfrac{-0.129z - 0.228}{0.492z - 0.849} & \text{if } \; 0.419 \le z \le 1.000. \end{cases}$$

With the optimal radical transformation r, one may construct $p \circ r$ and obtain

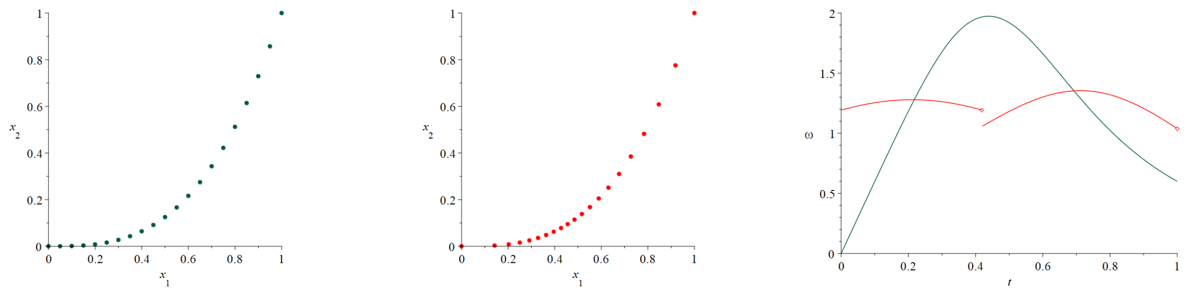
$$p \circ r \doteq \begin{cases} \left(\dfrac{-0.079\sqrt{z}\,(z - 3.116)}{(-0.172z + 0.5359)^{3/2}}, \dfrac{0.098\,z^{3/2}}{(-0.172z + 0.536)^{3/2}} \right) & \text{if } \; 0.000 \le z \le 0.419; \\ \left(\dfrac{-0.129z - 0.228}{0.492z - 0.849}, -\dfrac{0.002\,(z + 1.771)^3}{(0.492z - 0.849)^3} \right) & \text{if } \; 0.419 \le z \le 1.000. \end{cases}$$

The angular speed function of $p \circ r$ is

$$w_{p\circ r} \doteq \begin{cases} \dfrac{0.781}{z^2 - 0.420z + 0.655} & \text{if } \; 0.000 \le z \le 0.4187; \\ \dfrac{-1.379\,(z + 1.771)(z - 1.725)}{(z^2 - 1.456z + 0.899)(z^2 - 4.876z + 9.888)} & \text{if } \; 0.4187 \le z \le 1.000. \end{cases}$$

Furthermore, one may calculate its uniformity $u_{p\circ r} \doteq 0.997$.

The plots of p and $p\circ r$ as well as the behavior of their angular speed functions are shown below:



where

- *the left plot shows the equi-sampling of the original parameterization p (green);*
- *the middle plot shows the equi-sampling of the optimal piecewise radical reparameterization $p\circ r$ (red);*
- *the right plot shows the angular speed functions of p and $p\circ r$.*

It is seen that the angular speed uniformity is greatly improved by the piecewise radical reparameterization.

5 Implementational Issues/Suggestions

If one chooses to implement the proposed algorithm using floating-point arithmetic, then, as usual, one should be careful to avoid numerical instability.

For instance, if L_i is computed by (10) naively, then it leads to instability. For example, $t_i = 1/\sqrt{2}$ is a zero of ω_p with multiplicity 1, so

$$\omega_p(t) = |t - 1/\sqrt{2}| \cdot m(t),$$

where $m(1/\sqrt{2}) \neq 0$. The numeric solution over $[0,1]$ is $t_i = 0.707$. Thus

$$L_i = \Delta t_i \int_{t_i}^{t_{i+1}} \frac{\omega_p^2(t)}{(\mu_i+1)\tilde{t}^{\mu_i}}\,dt = \Delta t_i^2 \int_{t_i}^{t_{i+1}} \frac{(t-1/\sqrt{2})^2 \cdot m^2(t)}{2\,(t-0.707)}\,dt.$$

During integration, it is necessary to evaluate the integral at $t = 0.707$. When t approaches 0.707, the integral quickly increases to $+\infty$, causing numerical instability.

To avoid such cases, one could adopt a technique from symbolic computation to represent algebraic numbers. Suppose that $t = \gamma$ is a zero of $\omega_p(t)$ with multiplicity μ_i and t_i is its numerical approximation. By (2), $\omega_p^2(t)$ is a rational function. Let G and H be its numerator and denominator. Then $t = \gamma$ is a zero of G with multiplicity $2\,\mu_i$. Carrying out the Euclidean division with G as the dividend and $(t-\gamma)^{2\,\mu_i}$ as the divisor, we obtain

$$G(t) = (t-\gamma)^{2\,\mu_i} Q(t,\gamma) + R(t,\gamma).$$

Since $t = \gamma$ is a zero of G with multiplicity $2\,\mu_i$, it is also a zero of $R(t,\gamma)$ with multiplicity at least $2\,\mu_i$. Given that $\deg(R,t) < 2\,\mu_i$, $R(t,\gamma)$ must be zero, which leads to the following conclusion:

$$L_i = \Delta t_i \int_{t_i}^{t_{i+1}} \frac{\omega_p^2(t)}{(\mu_i+1)\,\tilde{t}^{\mu_i}}\,dt \doteq \frac{\Delta t_i^{\mu_i+1}}{\mu_i+1} \cdot \int_{t_i}^{t_{i+1}} \frac{Q(t,t_i)(t-t_i)^{\mu_i}}{H(t)}\,dt.$$

Acknowledgements Hoon Hong's work was supported by National Science Foundations of USA under Grant No. 1813340 and Jing Yang's work was supported by National Natural Science Foundation of China under Grant Nos. 11526060 and 12261010.

References

[1] P. Costantini, R. T. Farouki, C. Manni & Sestini A (2001): *Computation of optimal composite re-parameterizations*. Computer Aided Geometric Design 18(9), pp. 875–897, doi:10.1016/S0167-8396(01)00071-1.

[2] G. Farin (2006): *Rational quadratic circles are parametrized by chord length*. Computer Aided Geometric Design 23(9), pp. 722–724, doi:10.1016/j.cagd.2006.08.002.

[3] R. T. Farouki (1997): *Optimal parameterizations*. Computer Aided Geometric Design 14(2), pp. 153–168, doi:10.1016/S0167-8396(96)00026-X.

[4] H. Hong, D. Wang & J. Yang (2013): *A framework for improving uniformity of parameterizations of curves*. Science China Information Sciences 56(10), pp. 1–22, doi:10.1007/s11432-013-4924-4.

[5] B. Jüttler (1997): *A vegetarian approach to optimal parameterizations*. Computer Aided Geometric Design 14(9), pp. 887–890, doi:10.1016/S0167-8396(97)00044-7.

[6] M. Kosters (1991): *Curvature-dependent parameterization of curves and surfaces*. Computer-Aided Design 23(8), pp. 569–578, doi:10.1016/0010-4485(91)90058-5.

[7] X. Liang, C. Zhang, L. Zhong & Y. Liu (2005): *C^1 continuous rational re-parameterization using monotonic parametric speed partition*. In: Ninth International Conference on Computer Aided Design and Computer Graphics, IEEE Computer Society, Los Alamitos, CA, USA, pp. 16–21, doi:10.1109/CAD-CG.2005.23.

[8] R. Patterson & C. Bajaj (1989): *Curvature adjusted parameterization of curves*. Technical Report CSD-TR-907, Department of Computer Science, Purdue University, US. Available at `https://docs.lib.purdue.edu/cstech/773/`.

[9] J. R. Sendra & C. Villarino (2001): *Optimal reparameterization of polynomial algebraic curves*. International Journal of Computational Geometry & Applications 11(04), pp. 439–453, doi:10.1142/S0218195901000572.

[10] J. R. Sendra, F. Winkler & S. Pérez-Díaz (2008): *Rational algebraic curves. A computer algebra approach*. Algorithms and Computation in Mathematics 22, Springer Verlag, doi:10.1007/978-3-540-73725-4.

[11] J. Yang, D. Wang & H. Hong (2012): *Improving angular speed uniformity by optimal C^0 piecewise reparameterization*. In V. P. Gerdt, W. Koepf, E. W. Mayr & E. V. Vorozhtsov, editors: Computer Algebra in Scientific Computing, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 349–360, doi:10.1007/978-3-642-32973-9_29.

[12] J. Yang, D. Wang & H. Hong (2013): *Improving angular speed uniformity by C^1 piecewise reparameterization*. In T. Ida & J. Fleuriot, editors: Automated Deduction in Geometry, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 33–47, doi:10.1007/978-3-642-40672-0_3.

[13] J. Yang, D. Wang & H. Hong (2013): *Improving angular speed uniformity by reparameterization*. Computer Aided Geometric Design 30(7), pp. 636–652, doi:10.1016/j.cagd.2013.04.001.

[14] J. Yang, D. Wang & H. Hong (2014): *ImUp: a Maple package for uniformity-improved reparameterization of plane curves*. In R. Feng, W. Lee & Y. Sato, editors: Computer Mathematics, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 437–451, doi:10.1007/978-3-662-43799-5_29.