

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>

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* utility³) 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 details⁶), 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/>

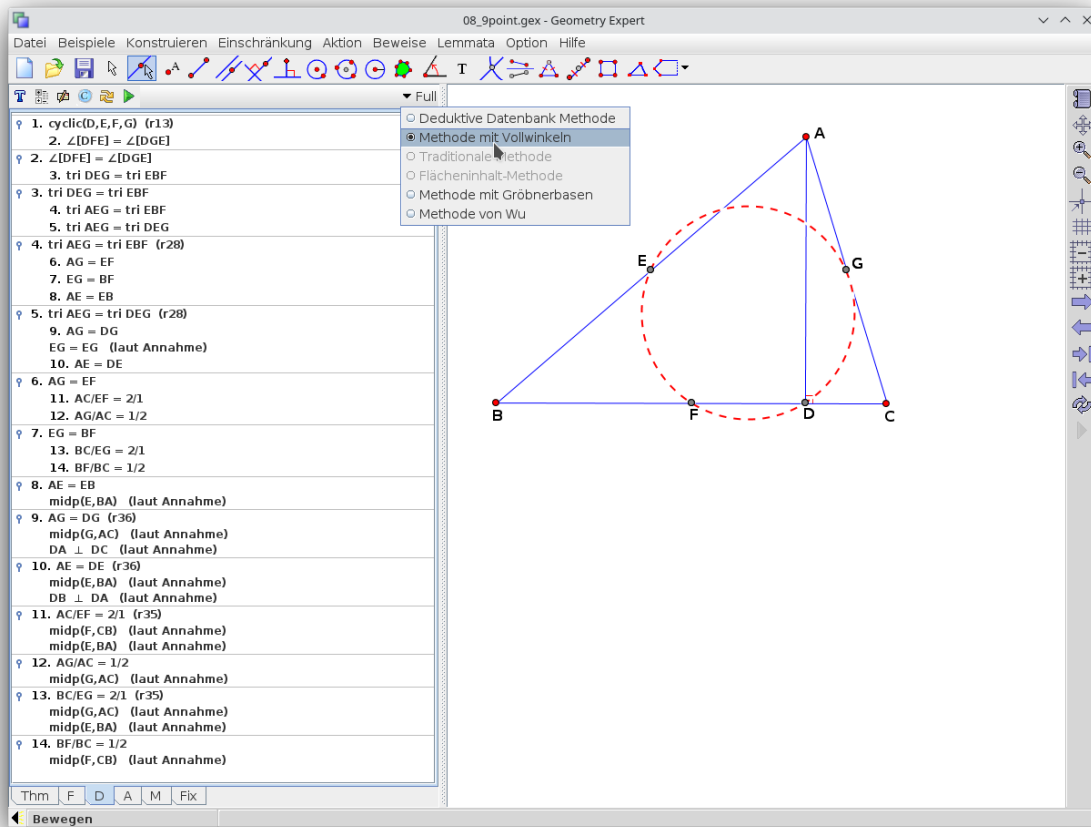


Figure 1: JGEX proves a part of the nine-point theorem by using the GDD method. Its user interface is set to German.

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.

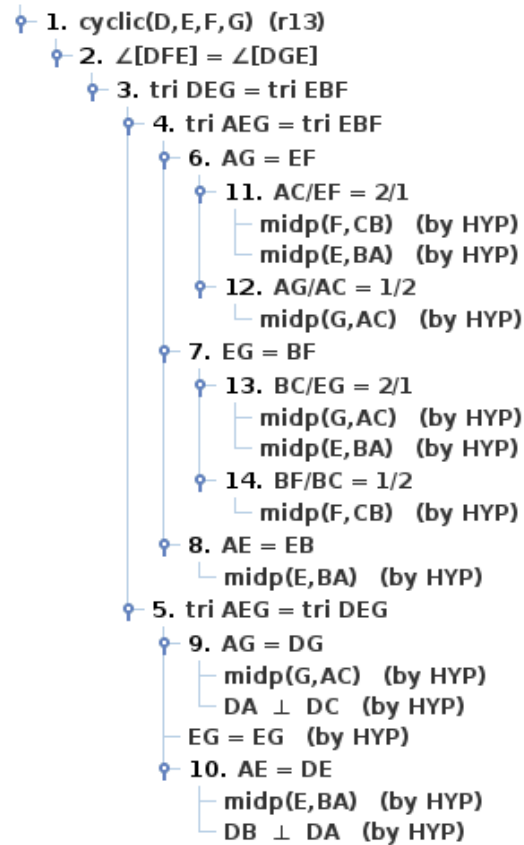


Figure 2: The output of the GDD method is visualized as a tree.

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-

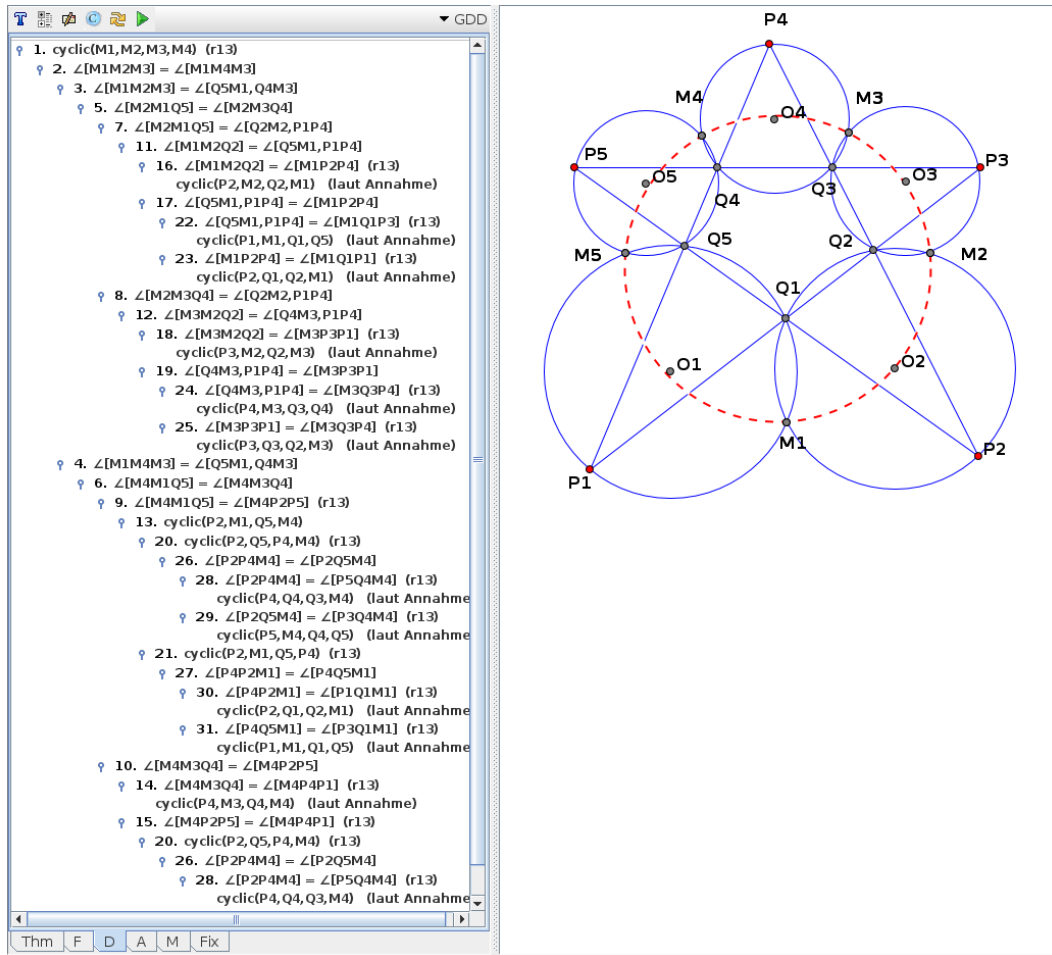


Figure 3: The GDD method finds a proof that is not a tree. The interface is set to German and the connecting edges of the graph are not shown.

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.

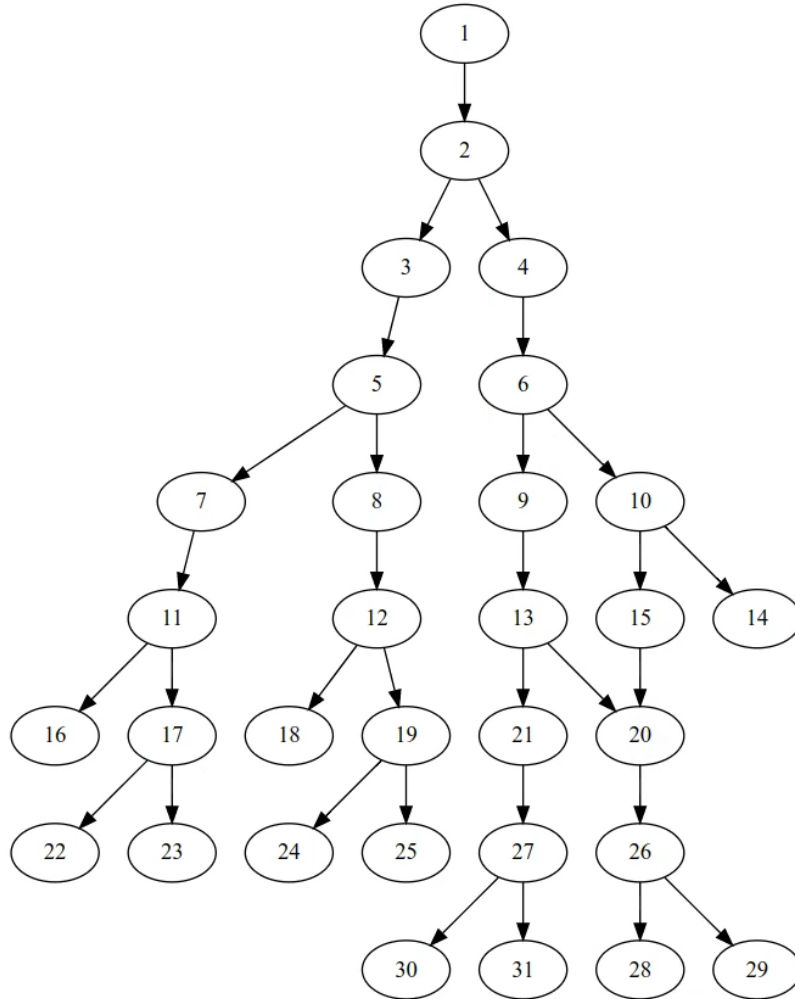


Figure 4: A better visualization of the GDD proof by using GraphViz.

- [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.