Supporting Proving and Discovering Geometric Inequalities in GeoGebra by using Tarski

Christopher W. Brown[0000-0001-8334-0980]

United States Naval Academy
Annapolis, MD, USA

`wcbrown@usna.edu`

Zoltán Kovács[0000-0003-2512-5793]

The Private University College of
Education of the Diocese of Linz
Linz, Austria

`zoltan@geogebra.org`

Róbert Vajda[0000-0002-2439-6949]

Bolyai Institute, University of Szeged
Szeged, Hungary

`vajda@math.u-szeged.hu`

We introduce a system of software tools that can automatically prove or discover geometric inequalities. The system, called *GeoGebra Discovery*, consisting of an extended version of *GeoGebra*, a controller web service *realgeom*, and the computational tool *Tarski* (with the extensive help of the *QEPCAD B* system) successfully solves several non-trivial problems in Euclidean planar geometry related to inequalities.

1 Introduction

For geometric *equalities* there are already integrated tools in some dynamic geometry systems (DGS). A well-known application is GeoGebra—it provides an automated reasoning toolset (ART) that supports symbolic check of equality of lengths of line segments in a geometric configuration, or, the equality of expressions. Both features are supported by the *Relation* tool or command, or the low-level commands *Prove* and *ProveDetails* directly [14]. GeoGebra's ART successfully proves hundreds of well-known theorems in elementary planar geometry (see `https://prover-test.geogebra.org/job/GeoGebr aDiscovery-provertest/66/artifact/fork/geogebra/test/scripts/benchmark/prover /html/all.html` for a recent benchmark of 291 test cases), however, all of these results are related to equalities.

In this extended abstract we describe our current work that focuses on *inequalities*. In our work we partly use the same algebro-geometric theory that plays a crucial role in GeoGebra's toolset, based on the revolutionary work of Wu [26], Chou [7], and improved later by Recio and Vélez [20] with elimination theory. This is the required algebraic basis for gaining a conjecture on *a possibly fixed ratio of two quantities*, if equality does not hold between them. For example, the centroid of a triangle divides the medians in ratio 2 : 1, so here there is no equality between the parts, but a fixed a ratio, namely 2. Once the elimination method is able to provide the exact ratio, the obtained conjecture becomes a mathematically proven proposition.

On the other hand, we partly use general purpose real quantifier elimination (RQE) methods to find the best possible geometric constants in the related inequalities if two expressions do not have a fixed ratio. Our implementation uses cylindrical algebraic decomposition (CAD) that promotes effective RQE. As a special case, we can also directly prove geometric inequalities. For example, theorems like the *triangle inequality* or the inequality $s \leq 3\sqrt{3}R$ (where s stands for the semiperimeter and R for the circumradius) can already be mechanically proven by our toolset in an intuitive way.

In this communication we do not go into the further detail on the hidden technical difficulties, but refer to two recent papers: [15] explains the mathematical background on obtaining ratios in regular polygons, while [22] focuses on the RQE related issues, and points to a large set of benchmarks based on our tool. Instead, we focus on the practical use: how our work can be fruitful for the student and the teacher in a classroom. These are the two possible ways how *GeoGebra Discovery* obtains proofs of geometric inequalities:

- In the case of exploration it tries to solve the problem with elimination first, and if this step is unsuccessful, then the given RQE problem setting will be outsourced to the external tool *realgeom*. We highlight that realgeom can work together with Wolfram's *Mathematica* (in this case the end user is expected to have a Mathematica subscription or to own a *Raspberry Pi* system that offers free access to Wolfram's tool, or use `wolframscript` that offers a light version of the Mathematica kernel), or the free systems *QEPCAD B* and *Tarski*.

- In the case of request for a direct proof of an inequality, at the moment, the Tarski system is used.

Our experimental system is already capable of solving a large set of open questions in planar Euclidean geometry. Even if the mathematical methods we use are mostly well-known, a full implementation of the various techniques in a single graphical application—being freely available for millions of potential users—is completely new.

As precursors of our contribution we refer the reader to some related work. These include the software packages SCRATCHPAD II [19] (the first uniform implementation of the geometry prover prototypes) and Java Geometry Explorer [30] (a program with graphical user interface that can give animated explanations of proofs of *equation based* geometry statements), and the BOTTEMA solver [29] (which is able to prove several inequalities from [1]); and the paper [13] on a survey of human readable proofs. Further precursors of our work can be found in [8, 21, 24, 27, 28].

2 Multiple ways for users to enter input

School curriculum usually includes several relationships between expressions in a planar construction. Every student has to learn the Pythagorean theorem which is an equality of two expressions, namely $a^2 + b^2$ and c^2 where a, b and c correspond to the lengths of sides in a right triangle. In general, however, a simple question is not discussed: what happens if we omit the assumption that the triangle has a right angle? By using our tool it can be quickly shown mechanically (here "mechanically" means that a *machine* works in the background) that in general the equality $a^2 + b^2 > \dfrac{c^2}{2}$ holds.

GeoGebra Discovery offers several ways to obtain this result. All of them require the first step that a triangle *ABC* must be drawn with GeoGebra's toolset. GeoGebra automatically labels the sides a, b and c. After this, the user has the following options:

1. Directly proving the inequality by typing `Prove(`$a^2 + b^2 > c^2/2$`)`. This is simple to input, but of course the user would have to know or guess that this inequality is the right one to try.

    It is possible to arrive at the inequality by through multiple attempts, e.g. `Prove(`$a^2 + b^2 > c^2$`)` (which is false), or `Prove(`$a^2 + b^2 > c^2/3$`)` (which is true), and then manually finding the largest value μ such that $a^2 + b^2 > \mu \cdot c^2$ holds in general.

2. By entering the command `Compare(`$a^2 + b^2, c^2$`)` the computer is asked to search for such a μ value, or, more generally, to find sharp non-negative constants m and M such that

$$m \cdot (a^2 + b^2) \underset{(=)}{\leq} c^2 \underset{(=)}{\leq} M \cdot (a^2 + b^2) \tag{1}$$
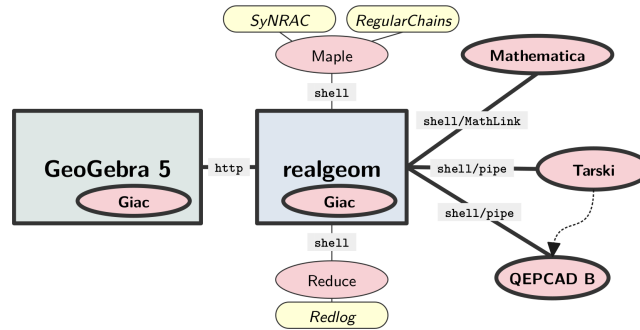
Figure 1: Computation hierarchy for GeoGebra 5

always holds. This command immediately prints the result in both the Algebra and Graphics Views of GeoGebra, in a raw textual form.

3. For most users there is a high-level method available. After typing `Relation`$(a^2 + b^2, c^2)$ a popup window is shown in the program by reporting that $a^2 + b^2$ and c^2 numerically differ in the particular case for the given triangle. After then, the user needs to click "More…" to obtain the exact symbolic result that considers all possible positions for points A, B and C.

   Alternatively, the user may define the sum $a^2 + b^2$ by typing it: GeoGebra will automatically denote it by d. Now, by using the Relation tool $a \overset{?}{=} b$ the user can directly compare c and d. The underlying system learns that c is a linear quantity (in terms of some appearing lengths in the figure), while d is a quadratic one (again, in terms of lengths), so the investigated inequality should be of the form

$$m \cdot d^1 \underset{(=)}{<} c^2 \underset{(=)}{<} M \cdot d^1. \tag{2}$$

These methods are supported in GeoGebra Discovery's latest versions [16], but in the mainstream version not yet.

3   Hierarchical structure of the system of tools

Fig. 1 gives an overview of the underlying technical hierarchy. Our system is modularized to be able to introduce other systems as optional parts at a later point. Here we emphasize that it seems very difficult to start writing a completely new computation system from scratch that provides all the mathematical background needed to achieve our aims in a feasible timelimit for our input problems. That is, re-using existing systems seems to be a much more reasonable way.

The boldface items are used in reality to obtain the sharp constants m and M, described in the formulas above. That is, the current technical hierarchy allows using *Maple/SyNRAC* [12], Maple/*RegularChains* [6], and *Reduce/Redlog* [11] in the near future, but for the moment only Mathematica's [25] RQE subsystem, *QEPCAD B* [4,9] and Tarski [23] are available for the end user. GeoGebra internally uses the Giac computer algebra system to perform symbolic computations [17].

Here we highlight that an important amount of work was performed to make our system work in educational applications. First of all, each discussed component is free software. Second, we ported the QEPCAD B and Tarski systems to Windows and made them available for the most recent macOS versions to reach a large amount of users including teachers and students. The system is available at [16] for all three majors platforms (for Linux a 64 bit version and a Raspbian variant are published).
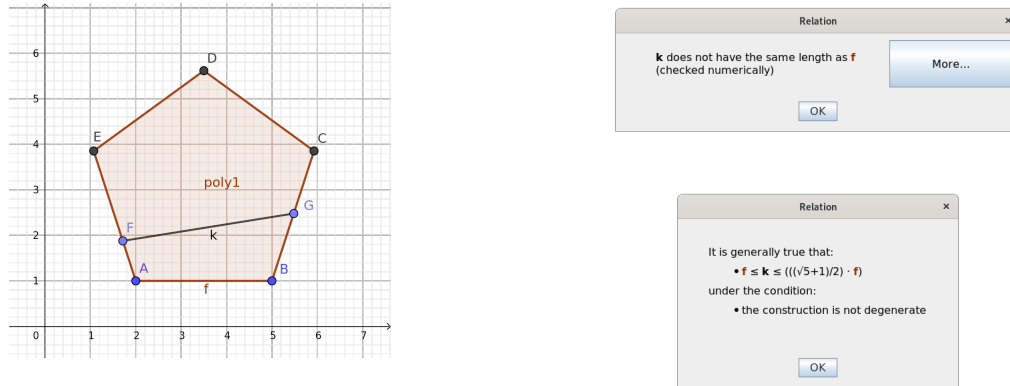
Figure 2: An example with a regular pentagon

4 A detailed example

Let us consider a regular pentagon. We are interested in how far the points of two sides are from each other.

To formulate this precisely, we consider Fig. 2 (left). A regular pentagon $ABCDE$ is given, and on side AE and BC, respectively, two points: F and G. We connect them by a segment k. We denote the segment AB by f. Question: Is there a general relationship between f and k?

When focusing on inequalities, this question can be split into two questions, namely:

1. What is m such that $m \leq k/f$, and

2. what is M such that $k/f \leq M$?

The first question seems to be simple to answer: $m = 1$, and this case can be reached when $F = A$ and $B = G$. To find M in the second question, however, some deeper considerations may be needed.

Here we emphasize that points A and B are freely chosen, and F and G have a one-dimensional freedom. For the user, the easiest way to get a symbolic answer is the method we described in Section 2, item 3: by clicking on the Relation tool and then on each segment of f and k we run into a numerical comparison first (Fig. 2, right, top). Then, by clicking on "More…" we get, after a couple of seconds, the result shown in Fig. 2 (right, bottom).

The result is as expected, but we highlight here that the translation of the geometric setup into a semi-algebraic system is not completely straightforward. Fig. 3 shows how the geometric figure is translated into a set of equations and inequalities in an automated way.

The translation is as follows. First free points A and B are designated by coordinates (v_1, v_2) and (v_3, v_4), respectively. Coordinates of points $C(v_7, v_8)$, $D(v_9, v_{10})$ and $E(v_{11}, v_{12})$ are uniquely determined by the equations and inequalities shown inside the pentagon, presented in red, but repeated here in the text as well. That is,

- $-v_7 v_4 + v_8 v_3 + v_7 v_2 - v_3 v_2 - v_8 v_1 + v_4 v_1 > 0$ designates C to be in the "upper" half-plane of line AB,

- $v_8 v_4 - v_4^2 + v_7 v_3 - v_3^2 - v_8 v_2 + v_4 v_2 - v_7 v_1 + v_3 v_1 > 0$ designates C to be in the "right" half-plane of a line going through B and perpendicular to AB,

- $-1 + 2v_5 + 4v_5^2 = 0$ defines the minimal polynomial of $\cos(2\pi/5)$ in the variable v_5, that is, one of the roots of this polynomial is $\cos(2\pi/5)$,
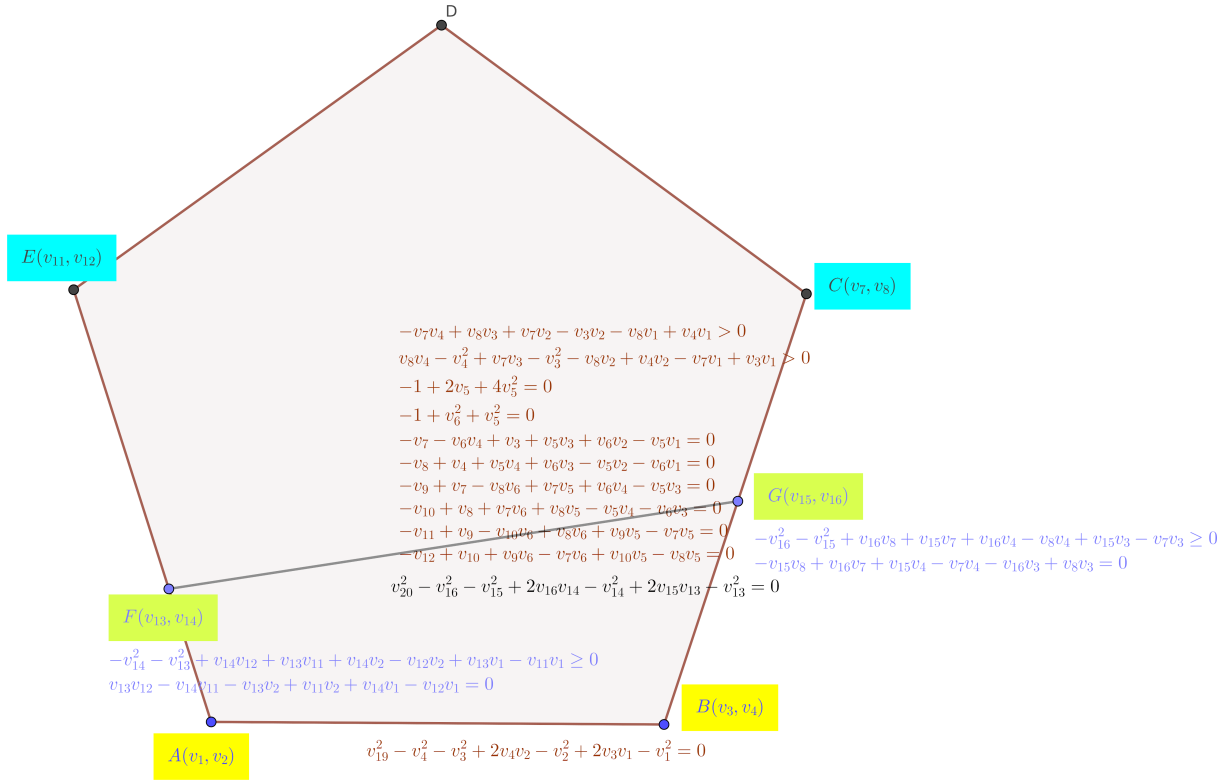
Figure 3: Translation of the figure into a semi-algebraic system

- $-1 + v_6^2 + v_5^2 = 0$ defines v_6 to express $\sin(2\pi/5)$.

- The equations $-v_7 - v_6 v_4 + v_3 + v_5 v_3 + v_6 v_2 - v_5 v_1 = 0$, $-v_8 + v_4 + v_5 v_4 + v_6 v_3 - v_5 v_2 - v_6 v_1 = 0$, $-v_9 + v_7 - v_8 v_6 + v_7 v_5 + v_6 v_4 - v_5 v_3 = 0$, $-v_{10} + v_8 + v_7 v_6 + v_8 v_5 - v_5 v_4 - v_6 v_3 = 0$, $-v_{11} + v_9 - v_{10} v_6 + v_8 v_6 + v_9 v_5 - v_7 v_5 = 0$ and $-v_{12} + v_{10} + v_9 v_6 - v_7 v_6 + v_{10} v_5 - v_8 v_5 = 0$ designate the coordinates of the remaining points of the regular pentagon. See [15] for more details on this technique.

To put point $F(v_{13}, v_{14})$ on line AE, we set the collinearity condition $v_{13} v_{12} - v_{14} v_{11} - v_{13} v_2 + v_{11} v_2 + v_{14} v_1 - v_{12} v_1 = 0$, and also the inequality $-v_{14}^2 - v_{13}^2 + v_{14} v_{12} + v_{13} v_{11} + v_{14} v_2 - v_{12} v_2 + v_{13} v_1 - v_{11} v_1 \geq 0$ is set to ensure that F lies between A and E. An analogue equation and inequality is set up for point $G(v_{15}, v_{16})$.

In this example we skip variables v_{17} and v_{18} (they were used for an auxiliary computation), but mention v_{19} and v_{20} that correspond to the length of AB and FG, respectively. They are defined by the polynomials $v_{19}^2 - v_4^2 - v_3^2 + 2v_4 v_2 - v_2^2 + 2v_3 v_1 - v_1^2 = 0$ and $v_{20}^2 - v_{16}^2 - v_{15}^2 + 2v_{16} v_{14} - v_{14}^2 + 2v_{15} v_{13} - v_{13}^2 = 0$, respectively.

Let us emphasize that we will focus on determining the possible values of m (and M) by setting up an equality among v_{19}, v_{20} and m. This will be performed in step 2 (see below).

Having successfully translated the geometric figure in a set of semi-algebraic hypotheses $H = H_e \cup H_i$ (where H_e stands for the set of equalities and H_i for the inequalities), we consider the following steps:

1. We put A and B to $(0,0)$ and $(1,0)$, respectively, by learning that this can be done without loss of generality.

2. It is possible that there are a finite number of candidate solutions that we can compute without considering H_i, so, roughly speaking, we construct the elimination ideal $(H_e \cup \{m \cdot v_{19} - v_{20}\}) \cap \mathbb{Q}[m]$, and by using Gröbner basis computations, we hope for a zero-dimensional result. (Technically, this step is done in GeoGebra, by using Giac.) Here this is not the case, so we go further.

3. To prepare the data for further processing, we *delinearize H_e*. Roughly speaking, this means that we try to eliminate as many variables as possible to prepare for the next step. In our case we obtain the new equalities H'_e:

$$-v_{13}^2 + 2v_{13}v_{15} - v_{15}^2 - v_{14}^2 + 2v_{14}v_{16} - v_{16}^2 + v_{20}^2 = 0, \tag{3}$$
$$v_{10}v_7 - 2v_7v_8 - v_{12} + v_8v_9 + v_8 = 0, \tag{4}$$
$$-v_{10}v_8 + v_8^2 - v_{11} - v_7^2 + v_7v_9 + v_7 = 0, \tag{5}$$
$$v_7^2 - 2v_7 + v_8^2 = 0, \tag{6}$$
$$4v_7^2 - 6v_7 + 1 = 0, \tag{7}$$
$$-v_{15}v_8 + v_8 + v_{16}v_7 - v_{16} = 0, \tag{8}$$
$$-v_{10} + 2v_7v_8 - v_8 = 0, \tag{9}$$
$$v_7^2 - v_7 - v_8^2 - v_9 + 1 = 0, \tag{10}$$
$$-v_{11}v_{14} + v_{12}v_{13} = 0, \tag{11}$$
$$-m + v_{20} = 0, \tag{12}$$
$$-v_{19} + 1 = 0. \tag{13}$$

This step is performed in the realgeom tool.

4. By creating a quantified formula, here

$$\exists_{v_7, v_8, \ldots, v_{20}} (H'_e \wedge H_i \wedge v_{19} > 0 \wedge v_{20} > 0), \tag{14}$$

we compute an equivalent, quantifier-free formula, which is

$$m > 0 \wedge m - 1 \geq 0 \wedge m^2 - m - 1 \leq 0. \tag{15}$$

This step is computed with the help of the Tarski system. It reformulates the input into a form that is hopefully easy to handle by QEPCAD B.

The quantifier elimination problems generated by this process are all existentially closed except for the free variable m, and generally have a number of equations in which some variables occur linearly. These problems are attacked via a script in Tarski's input language that does the following:

   (a) applies quick simplification (command `bbwb`);
   (b) performs linear substitutions with case splitting to handle vanishing denominators and further quick simplification to eliminate as many variables as possible (command `qfr`)—note that the resulting formula is a disjunction of existentially quantified conjunctions—and
   (c) formulates each quantified conjunction for QEPCAD B and sends it to be solved independently (command `qepcad-qe`). The final solution is then the disjunction of each of the sub-solutions which, recall, are formulas in the variable m alone.

5. We turn the quantifier-free formula into an interval that is simpler to interpret:

$$m \geq 1 \wedge m \leq \frac{\sqrt{5}+1}{2}. \tag{16}$$

This step is done in the realgeom tool by utilizing Giac.

6. Finally, GeoGebra reformulates this result as seen in Fig. 2 (right, bottom).

Here is how much time is spent in each step: The algebraic translation requires 50 ms, the substitution and elimination steps (1–2) need 32 ms. For delinearization (3) 27 ms is needed. Computation of the quantifier-free formula (4) requires 2216 ms, its rewrite to an interval (5) needs 1 ms, and the final communication (6) requires another 1 ms. As a total, 2327 ms were needed on a recent workstation (Ubuntu Linux 20.04, 8×Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 32 GB RAM).

This is one example among about two-hundred working test cases that vary between simple constructions and some more complicated ones. We provide a benchmark of 131 examples, available at `https://prover-test.geogebra.org/job/GeoGebra`$_D$`iscovery-comparetest/113/artifa` `ct/fork/geogebra/test/scripts/benchmark/compare/html/all.html` that include several entries from the well-known Bottema database [1]. Also, direct proofs for 46 problems are benchmarked in the database `https://prover-test.geogebra.org/job/GeoGebra`$_D$`iscovery-proverrgtest` `/33/artifact/fork/geogebra/test/scripts/benchmark/prover-rg/html/all.html`. Both lists contain several examples that can be directly used in a classroom situation, either to get a conjecture (with proof), or a confirmation/disproof, if the students already made a conjecture.

5   Conclusion

According to our benchmark, the Tarski system manages to solve 117 and 35 input problems within 30 secs timeout, respectively, outperforming Mathematica with the number of successful tests (117 and 33 successful results, respectively).

By comparing the performance of Mathematica and Tarski we see that both systems are very fast. In our benchmark they usually deliver the solution far below 1 second. This supports the idea that Tarski can be used to obtain non-trivial results, by allowing a large scale of users to study inequalities in a planar Euclidean geometry construction. In classroom situations speed is very important: young learners are not expected to wait minutes, hours or even days to obtain a result. Fast results are however not expected in a scientific environment.

We provide a figure that shows the density estimate of time output of the first benchmark, visualized with statistics software R's sm package [2] on those 102 tests that uniformly work on both systems (see Fig. 4). The *x*-axis shows the overall time in milliseconds. According to this diagram, Mathematica's advantage is clear because of its smaller mean (1366 secs) and deviation (3485 secs), however, Tarski's performance is also remarkable (2365 and 3940 secs, respectively).

In fact, more complicated inputs may result in getting no answer in feasible time in both systems. It is well known that CAD has doubly exponential complexity in the number of variables (see [3, 10]), so in general there is no hope to cover all problems in feasible time. But the questions we raise are quite specific, so there seems to be much room for remarkable speedups.

We can consider the "classical way" of translating geometry into algebra as founded by Wu and Chou in the 1980s, and its "extension" by adding non-equational constraints. Surprisingly, just a couple of such constraints are required to faithfully describe a planar Euclidean figure, if we consider the usual construction steps that are used at secondary school level. These include

- the assumption of positive sign for the lengths (e.g. those that belong to the sides of a triangle),

- the *betweenness* property if a point is attached to a segment (cf. $F \in AE$ in Fig. 3),

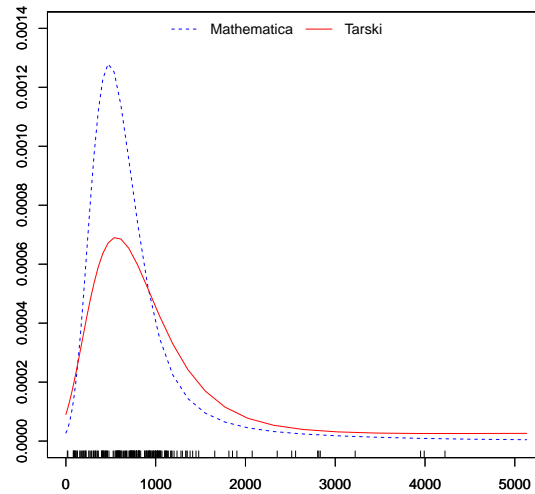- the condition to consider only the internal angle bisector (that is, to ignore the external ones),

Figure 4: Comparison of Mathematica (- - -) and Tarski (—)

- as a special case of the previous one, the restriction that the center of the incircle of a triangle is a unique point (unlike 4 points that include the centers of the excircles as well),

- the constraint that a point is an inner point of a triangle, and

- the uniqueness of a regular *n*-gon in the cases $n > 2$, $n \neq 4$ (unlike considering some regular star-polygons at the same time, because they are indistinguishable in the "classical" *complex algebraic geometry setup*),

for the given hypotheses. These former inconveniencies can be handled properly in the "extended" *real algebraic geometry setup*, and all of them are implemented in the most recent version of GeoGebra Discovery. In addition, we can avoid some difficulties that appeared when the thesis included a polynomial equality. Formerly the thesis usually had to be rewritten to higher degree polynomials (e.g. instead of $a + b = c$, that is equivalent to $a + b - c = 0$, the following equation had to be used: $(a + b - c) \cdot (b + c - a) \cdot (c + a - b) \cdot (a + b + c) = 0$, see [18]). Real algebraic geometry successfully brings the possibility to forget these difficulties.

These extensions, however, have their price: higher computational complexity. To find an optimal balance between "imperfect but feasible" (via complex algebraic geometry) and "perfect but infeasible" (via real algebraic geometry) seems to be a challenging research for the next steps in our work.

6 Future work

Our final aim is to embed our experimental network of the four systems into one tool, namely, into the *mainstream version* of GeoGebra, developed at Johannes Kepler University of Linz, Austria. It runs on several platforms, including desktop computers with various operating systems (Windows, Mac and Linux), and the web platform, and also smartphones. This high variety of technological background re-

quires heavy simplification of the underlying technologies and it calls for minimizing the applied external tools.

That is, for the long term our aim is to embed all external parts of our machinery into GeoGebra. For the realgeom part, this seems to be achievable since it is a single Java application with no graphical user interface: only some sophisticated algorithms are included that can be copied to GeoGebra.

On the other hand, copying Tarski and QEPCAD B into GeoGebra seems to be somewhat more difficult. QEPCAD is based on Saclib [5], a C library that offers CAS computations. QEPCAD and Tarski are written in C++. Thus, it seems a viable way to compile all systems as a Java Native Interface (JNI) and include them as a dynamically loadable object on all desktop platforms. This project is an on-going work.

Besides the native platform, a web application and a smartphone version should also be supported. Compilation of C/C++ applications for the web is already well supported by various tools. Here we highlight Emscripten [31] that provides multiple targets including JavaScript or WebAssembly. Porting Tarski and QEPCAD B to the web is also an on-going work.

7   Acknowledgements

References

[1] O. Bottema, R.Z. Djordjevic, R.R. Janic, D.S. Mitrinovic & P.M. Vasic (1969): *Geometric Inequalities*. Wolters-Noordhoff Publishing, Groningen.

[2] A. W. Bowman & A. Azzalini (2013): *Package sm: non-parametric smoothing methods (version 2.2-5)*. Available at `http://www.stats.gla.ac.uk/~adrian/sm`.

[3] C. Brown & J.H. Davenport (2007): *The complexity of quantifier elimination and cylindrical algebraic decomposition*. In: *Proceedings of ISSAC '07*, ACM, pp. 54–60, doi:10.1145/1277548.1277557.

[4] C. W. Brown (2003): *An Overview of QEPCAD B: a Tool for Real Quantifier Elimination and Formula Simplification*. Journal of Japan Society for Symbolic and Algebraic Computation 10(1), pp. 13–22.

[5] Christopher W. Brown (2021): *Saclib*. A GitHub project. `https://github.com/chriswestbrown/saclib`.

[6] C. Chen & M. M. Maza (2016): *Quantifier Elimination by Cylindrical Algebraic Decomposition based on Regular Chains*. Journal of Symbolic Computation 75, pp. 74–93, doi:10.1016/j.jsc.2015.11.008.

[7] S. C. Chou (1988): *Mechanical Geometry Theorem Proving*. D. Reidel Publishing Company, Dordrecht, Netherlands.

[8] 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(3), pp. 219–246, doi:10.1023/A:1006171315513.

[9] G. E. Collins & H. Hong (1991): *Partial Cylindrical Algebraic Decomposition for Quantifier Elimination*. Journal of Symbolic Computation 12, pp. 299–328, doi:10.1016/S0747-7171(08)80152-6.

[10] J.H. Davenport & J. Heintz (1988): *Real quantifier elimination is doubly exponential*. Journal of Symbolic Computation 5(1-2), pp. 29–35, doi:10.1016/S0747-7171(88)80004-X.

[11] A. Dolzmann & T. Sturm (1997): *Redlog: Computer Algebra Meets Computer Logic*. *ACM SIGSAM Bulletin* 31(2), pp. 2–9, doi:10.1145/261320.261324.

[12] H. Iwane, H. Yanami & H Anai (2014): *SyNRAC: a toolbox for solving real algebraic constraints*. In: *Proceedings of ICMS-2014. LNCS, vol. 8592*, Springer, Heidelberg, pp. 518–522.

[13] Jianguo Jiang & Jing Zhong Zhang (2012): *A review and prospect of readable machine proofs for geometry theorems*. Journal of Systems Science and Complexity 25(4), doi:10.1007/s11424-012-2048-3.

[14] Zoltán Kovács (2015): *The Relation Tool in GeoGebra 5*. In Francisco Botana & Pedro Quaresma, editors: *Automated Deduction in Geometry: 10th International Workshop, ADG 2014, Coimbra, Portugal, July 9-11, 2014, Revised Selected Papers*, Springer International Publishing, Cham, pp. 53–71, doi:10.1007/978-3-319-21362-0_4.

[15] Zoltán Kovács (2020): *Automated Detection of Interesting Properties in Regular Polygons*. Mathematics in Computer Science 14, pp. 727–755, doi:10.1007/s11786-020-00491-z.

[16] Zoltán Kovács (2020): *GeoGebra Discovery*. A GitHub project. `https://github.com/kovzol/geogebra-discovery`.

[17] Zoltán Kovács & Bernard Parisse (2015): *Giac and GeoGebra – Improved Gröbner Basis Computations*. In Jaime Gutierrez, Josef Schicho & Martin Weimann, editors: *Computer Algebra and Polynomials*, Lecture Notes in Computer Science, Springer, pp. 126–138, doi:10.1007/978-3-319-15081-9_7.

[18] Zoltán Kovács, Tomás Recio & Csilla Sólyom-Gecse (2019): *Rewriting input expressions in complex algebraic geometry provers*. Annals of Mathematics and Artificial Intelligence 85, pp. 73–87, doi:10.1007/s10472-018-9590-1.

[19] K. Kusche, B. Kutzler & H. Mayr (1989): *Implementation of a geometry theorem proving package in SCRATCHPAD II*. In James H. Davenport, editor: *Eurocal '87*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 246–257, doi:10.1007/3-540-51517-8_123.

[20] Tomás Recio & M. Pilar Vélez (1999): *Automatic discovery of theorems in elementary geometry*. Journal of Automated Reasoning 23, pp. 63–82, doi:10.1023/A:1006135322108.

[21] Mohab Safey El Din (2007): *RAGlib: A library for real solving polynomial systems of equations and inequalities*. `https://www-polsys.lip6.fr/~safey/RAGLib/`.

[22] Róbert Vajda & Zoltán Kovács (2020): *GeoGebra and the realgeom Reasoning Tool*. CEUR Workshop Proceedings, pp. 204–219. Available at `http://ceur-ws.org/Vol-2752/paper15.pdf`.

[23] F. Vale-Enriquez & C.W. Brown (2018): *Polynomial Constraints and Unsat Cores in* TARSKI. In: *Mathematical Software – ICMS 2018. LNCS, vol. 10931*, Springer, Cham, pp. 466–474, doi:10.1007/978-3-319-96418-8_55.

[24] D. Wang (2002): *Geother 1.1: Handling and proving geometric theorems automatically*. In: *ADG 2002, Lecture Notes in Computer Science* 2930, Springer, pp. 194–215, doi:10.1007/978-3-540-24616-9_12.

[25] Wolfram Research, Inc. (2020): *Mathematica, Version 12.1*. Champaign, IL.

[26] W. T. Wu (1978): *On the decision problem and the mechanization of theorem proving in elementary geometry*. Scientia Sinica 21, pp. 157–179.

[27] B. Xia (2007): *DISCOVERER: A tool for solving semi-algebraic systems*. ACM SIGSAM Bulletin 41(3), pp. 102–103.

[28] L. Yang, X. Hou & B. Xia (1999): *Automated Discovering and Proving for Geometric Inequalities*. In: *ADG'1998, LNAI 1669*, Springer-Verlag, pp. 30–46.

[29] L. Yang & J. Zhang (2001): *A practical program of automated proving for a class of geometric inequalities*. In: *Automated Deduction in Geometry, Lecture Notes in Artificial Intelligence* 2061, Springer-Verlag, Berlin, Heidelberg, pp. 41–57, doi:10.5555/647693.731185.

[30] Zheng Ye, Shang Ching Chou & Xiao Shan Gao (2011): *An Introduction to Java Geometry Expert*. In: *Automated Deduction in Geometry*, Springer Science + Business Media, pp. 189–195, doi:10.1007/978-3-642-21046-4_10.

[31] Alon Zakai (2013): *Emscripten: An LLVM-to-JavaScript Compiler.* `https://github.com/kripken/emscripten/blob/master/docs/paper.pdf?raw=true`.