1 2 9 0

UNIVERSIDADE D
COIMBRA

Nuno Miguel dos Santos Baeta

Working Environment for Automated Deduction in Geometry

Tese no âmbito do Doutoramento em Álgebra Computacional orientada pelo Professor Doutor Pedro Henrique e Figueiredo Quaresma de Almeida e apresentada ao Departamento de Matemática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho de 2023

Working Environment for Automated Deduction in Geometry

Nuno Miguel dos Santos Baeta

UC|UAb Joint PhD Program in Computational Algebra

Doutoramento em Álgebra Computacional

PhD Thesis | Tese de Doutoramento

Julho de 2023

Acknowledgements

I wish to thank Professor Pedro Quaresma for his encouragement and guidance in carrying out this work.

I would also like to thank my wife Cristina, sons Francisco and José, and remaining family, who patiently supported me during this period.

Abstract

Given its formal, logical and spatial properties, geometry is well suited to be used in teaching environments that include dynamic geometry systems (DGS), geometry automated theorem provers (GATP) and repositories of geometric problems. With the help of the DGS, students can build geometric constructions and conjecture about their properties. With the GATP, students may check the soundness of the constructions (e.g., if two given lines are parallel) and also create formal proofs of conjectures. These tools, backed by repositories of geometric knowledge, provide teachers and students with a framework supported by a large set of geometric constructions and conjectures, for the development of experiments.

The ultimate goal is to contribute towards the development of a working environment, where the dynamic geometry component and the automated deduction component work together, providing proofs for plane geometry, in natural language and with a visually dynamic presentation. Quoting Gila Hanna "the best proof is one that also helps understand the meaning of the theorem being proved: to see not only that it is true, but also why it is true".

Building on the work already done in the open source project *Open Geometry Prover* (OGP), the initial goal was to complete the GATP based on the full-angle method, improving OGP's library, and allowing the automated deduction in geometry community to profit from a new tool. The OGP project was originally developed by PhD student Ivan Petrović, under the guidance of Predrag Janičić of the University of Belgrade, Serbia, but it was not completed. Having taken over the project, now designated as the *Open Geometry Prover Community Project* (OGPCP), it was opted for a totally new implementation of a GATP based on the deductive databases for geometry method, having concluded a first version of the automatic theorem prover, the *Geometry Deductive Database Prover*, fully integrated in the OGPCP.

Resumo

Dadas as suas propriedades formais, lógicas e espaciais, a geometria é adequada para ambientes de ensino que incluam sistemas de geometria dinâmica (DGS), demonstradores automáticos de teoremas para a geometria (GATP) e repositórios de problemas geométricos. Com a ajuda dos DGS, os alunos podem efectuar construções geométricas e conjecturar sobre as propriedades das mesmas. Com os GATP, podem verificar a correcção das construções (por exemplo, se duas linhas dadas são paralelas) e também demonstrar as conjecturas geométricas. Estas ferramentas, apoiadas em repositórios de conhecimento geométrico, fornecem aos professores e alunos um ambiente de trabalho suportado por um conjunto alargado de exemplos de construções geométricas e conjecturas, para o desenvolvimento de experiências.

Tem-se como objetivo final contribuir para o desenvolvimento de um ambiente de trabalho, onde as componentes de geometria dinâmica e de dedução automática trabalham em conjunto, providenciando demonstrações em geometria euclidiana, escritas em linguagem natural e com uma apresentação visualmente dinâmica. Citando Gila Hanna "a melhor prova é aquela que também ajuda a entender o significado do teorema a ser demonstrado: perceber que não é apenas verdadeiro, mas também por que é verdadeiro".

Tendo como ponto de partida o trabalho já desenvolvido no projecto de código aberto *Open Geometry Prover* (OGP), o objetivo inicial era completar o GATP do método de ângulo completo, melhorando a «biblioteca» do OGP, e permitindo à comunidade da dedução automática em geometria usufruir de uma nova ferramenta. O projecto OGP foi inicialmente desenvolvido pelo aluno de doutoramento Ivan Petrović, sob a orientação de Predrag Janičić da Universidade de Belgrado, Sérvia, não tendo sido, contudo, terminado. Tendo assumido o projeto, agora designado como *Open Geometry Prover Community Project* (OGPCP), optou-se por uma implementação totalmente nova de um GATP baseado no método das bases de dados dedutivas para a geometria, tendo concluído uma primeira versão do demonstrador automático de teoremas, o *Geometry Deductive Database Prover*, totalmente integrado ao OGPCP.

Contents

Chapter 1

Introduction

Given its formal, logical, and spatial properties, geometry is well suited to be taught using teaching environments that include dynamic geometry systems (DGS), geometry automated theorem provers (GATP) and repositories of geometric problems. With the help of the DGS, students can build geometric constructions and conjecture about their properties. With the GATP, students may check the soundness of the constructions (e.g. if two given lines are parallel) and also create formal proofs of the conjectures. Supported by repositories of geometric knowledge, these tools provide teachers and students with a framework backed by a large set of geometric constructions and conjectures, for the development of experiments. Following Gila Hanna's argument [23, p. 8] that "the best proof is one that also helps understand the meaning of the theorem being proved: to see not only that it is true, but also why it is true", the large number of articles on proof and proving in mathematics education from the ICMI Study 19 Conference [24, 36, 37] and books about the subject [25, 49], the focus of this thesis is on the implementation of an open source GATP capable of being used on the practice of verification, explanation, and discovery in the teaching and learning of geometry.

An initial goal of implementing a GATP based on the full-angle method [11, 14, 61] was pursued. The goal was to pick-up on the work already done by Ivan Petrović, under the supervision of Predrag Janičić, from the University of Belgrade, Serbia, that developed an incomplete implementation, in *Java*, of a GATP based on the full-angle method.[1] Unfortunately, given the lack of documentation, the unavailability of help from the initial developer and a complex and very intricate *Java* implementation, although a fair amount of work was done, this task was not successfully complete [1, 2]. The implementation of this GATP was to be open source and integrated in the *Open Geometry Prover* project led by Predrag Janičić, more about this later.

After having concluded that it would be better to build a GATP from scratch, abandoning the idea of finishing the initial, incomplete, implementation, the planning for a *C++* implementation begun. At that moment it was also decided that a GATP based on the deductive databases approach, following the method described in [15], would be more appropriated for the goal of a

[1] Ivan Petrović did not complete this project given that he decided to pursue other goals.

GATP capable of being used on the practice of verification, explanation, and discovery in the teaching and learning of geometry [44, 52, 55]. A first functional implementation of this GATP was done [5] (see Chapter 4).

The *Geometry Deductive Database Prover* is being developed as an open source GATP available from *GitHub*.[2] It is integrated in the *Open Geometry Prover Community Project* library of GATP, also available from *GitHub*.[3] The *Open Geometry Prover Community Project* is a long term effort from the automated deduction in geometry community.[4] It has been started by Predrag Janičić from the University of Belgrade, Serbia, and it is now led by Pedro Quaresma and Nuno Baeta from the Centre for Informatics and Systems of the University of Coimbra [4] (see Chapter 3).

Apart from this two contributions that are the backbone of this thesis, the contributions around the theme of automated deduction in geometry are numerous. The development of a common format for GATP, the I2GATP [43], based on the I2G common format for DGS [50]. An implementation of an exchange protocol [45] that allows the remote query of the Thousands of Geometric Problems for Geometric Theorem Provers (TGTP[5]) [40] repository of geometric knowledge from within other systems, e.g. the Web Geometry Laboratory (WGL[6]) project [46, 48]. Establishing a ranking among GATP [3] and, keeping that goal in mind, also to help improve efficiency among GATP, the development of a GATP competition, the *Geometry Automated Provers Systems Competition* (GASC) [6]. With regard to education applications, the modernisation of the *Geometrography* approach [35, 38] to DGS [51], alongside the development of a taxonomy of geometric problems [47].

This thesis is organised as follows. In chapter 2 the area of automated deduction in geometry is described. In chapter 3 the *Open Geometry Prover Community Project* is presented — included in this chapter it is the article, Nuno Baeta and Pedro Quaresma, *Open Geometry Prover Community Project*, published in volume 352 of the *Electronic Proceedings in Theoretical Computer Science*, in 2021. In chapter 4 the GATP, *Geometry Deductive Database Prover* is described — included in this chapter it is the article, Nuno Baeta and Pedro Quaresma, *Towards a Geometry Deductive Database Prover*, published in the journal Annals of Mathematics and Artificial Intelligence, Springer, in 2023 (available online). Finally, in chapter 5 conclusions are drawn and future research is foreseen.

[2] https://github.com/opengeometryprover/OpenGeometryProver/
[3] https://github.com/opengeometryprover/
[4] The 14th International Conference on Automated Deduction in Geometry will be held on Belgrade, Serbia, September 20-22, 2023, https://adg2023.matf.bg.ac.rs/
[5] http://hilbert.mat.uc.pt/TGTP/
[6] http://hilbert.mat.uc.pt/WebGeometryLab/

Chapter 2

Automated Deduction in Geometry

For the last four decades, automated deduction in geometry has become one of the most successful fields in automated reasoning. Various methods and techniques have been studied and developed for automatically proving and discovering geometric theorems [9, 26, 28, 53].

Adapting general-purpose reasoning approaches developed in the field of artificial intelligence, synthetic methods, such as the approaches by Gelernter [21], Nevins [39], Elcock [18], Greeno et al. [22], Coelho and Pereira [16], Chou, Gao, and Zhang [10], are dedicated to automating traditional proving processes. Making use of axiomatic systems close to the ones used in secondary schools, these systems try to provide readable (by students and teachers) proofs. Unfortunately, the combinatorial explosion caused by the repeated application of postulates, made it necessary to use suitable heuristics. In turn, these (heuristics) narrow the scope of the geometry automated theorem provers (GATP) and do not allow the development of general-purpose efficient GATP. A more recent approach is given by the rule based theorem provers, e.g. the deductive database method for geometry [15, 20, 60, 61] and the implementation describe in this thesis (see chapter 4) [5], and, in a certain way, a rule based theorem prover recently incorporated in a tutorial system by L. Leduc [34].

The algebraic methods, such as the quantifier elimination method of Tarski [17, 54], the characteristic set method, also known as Wu's method [8, 57, 59], the polynomial elimination method [56], the Gröbner basis method [30, 31], and the Clifford algebra approach [26], subtly reduce the complexity of logical inferences. This is achieved by computing relations between coordinates of geometric entities, thus successfully (efficiently) solving many complicated geometric problems and discover new theorems. The price to pay is the absence of geometric proofs, or, when proofs are provided, they are only (barely) readable by experts.

Roughly speaking, these methods can address those geometry statements of equality type, for which, in their algebraic form, the hypotheses can be expressed by a set (conjunction) of

equations [9]:

$$\begin{aligned} h_1(y_1, \ldots, y_m) &= 0 \\ h_2(y_1, \ldots, y_m) &= 0 \\ &\vdots \\ h_r(y_1, \ldots, y_m) &= 0, \end{aligned}$$

and the conclusion is also an equation $c(y_1, \ldots, y_m) = 0$, where the h's and c are polynomials with coefficients in a base field K. Usually, it is assumed that $K = \mathbb{Q}$, the field of rational numbers. Thus the algebraic form of the geometry statement would be:

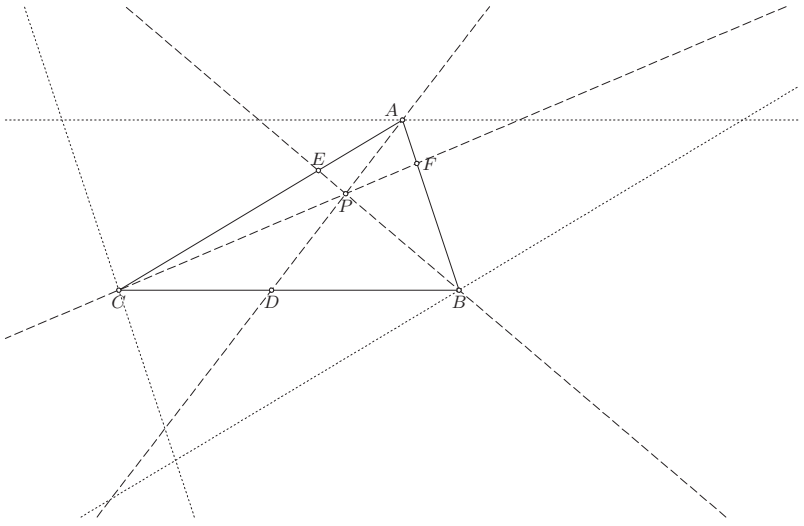
$$\forall_y (h_1 = 0 \wedge \ldots \wedge h_r) \Rightarrow c = 0.$$

In many situations, a set of non-degenerate (ndg) conditions must be added to the premises for the theorem to became valid.

The following simple example briefly illustrates some key features of these methods.

Theorem 2.1 (Ceva's Theorem) *Let $\triangle ABC$ be a triangle and P be an arbitrary point in the plane. Let D be the intersection of AP and BC, E be the intersection of BP and AC, and F the intersection of CP and AB. Then:*

$$\frac{\overline{AF}}{\overline{FB}} \frac{\overline{BD}}{\overline{DC}} \frac{\overline{CE}}{\overline{EA}} = 1$$



Using the syntax of the geometry automated theorem prover *GCLC* [27], this theorem can be formalised (see Listing 2.1).

Listing 2.1 Ceva's Theorem formalised in GCLC

```
% Free points (the coordinates are only for the drawing purposes)
point A 90 70
```

```
point B 110 10
point C 10 10
point P 75 40
% Lines BC, AC and AB
line a B C
line b A C
line c A B
% Lines PA, PB and PC
line pa P A
line pb P B
line pc P C
% Intersection points D, E and F
intersec D a pa
intersec E b pb
intersec F c pc
% The conjecture
prove {
 equal
  { mult  { mult { sratio A F F B } { sratio B D D C } } { sratio C E E A } }
  { 1 }
}
```

Using an algebraic method, e.g. the Wu's method, *GCLC* is able to prove this theorem in 0.002 seconds. It produces a seven pages long algebraic proof, "readable" by an expert (see Appendix A), but with no geometric interpretation whatsoever, all the steps are given by algebraic manipulations.

In order to combine the readability of synthetic methods and the efficiency of algebraic methods, some approaches, such as the area method [10, 12, 13, 28], the full angle method [14], and coherent-logic-based method [53], have been developed. Representing geometric knowledge in the form of expressions with respect to geometric invariants, the GATP implementing these methods are capable of producing formal and readable proofs, efficiently. However, even though the proofs produced are readable by secondary schools' students and teachers, given the fact that these methods do not employ the usual axioms systems used in secondary schools, a previous study of the axioms system used by the GATP is mandatory.

The area method is the best well-known of these methods, with several implementations available. The method is able to produce human-readable proofs for hard geometry theorems efficiently. Instead of coordinates, three basic geometric quantities, the ratio of parallel line segments, the signed area, and the Pythagorean difference, are used. The basic propositions, which formally describe the properties of these quantities, are the deductive basis of the area method. The method involves the elimination of the constructed points from the conclusion, using these basic geometry propositions [28].

Using, again, the Ceva's theorem as a test-case (see Theorem 2.1, Listing 2.1), it can be seen that the *GCLC* GATP is able to prove this theorem in 0.000 seconds (meaning that the proof took less the 1 milliseconds) and it is a two pages long proof. It is a synthetic proof (see Appendix B) easy to follow, with the only drawback that it uses the deductive rules of the area method, and not the usual ones used at secondary schools.

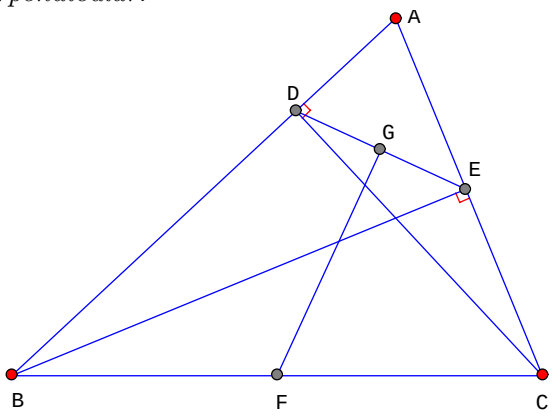
The synthetic methods, based in axiomatic theories for geometry, try to automate traditional proving processes. Making use of axiomatic systems close to the ones used in secondary schools, these systems try to provide readable (by students and teachers) proofs.

The geometry deductive database method is a synthetic method that uses forward chaining to prove non-trivial geometry theorems efficiently [15]. Given a geometric configuration, the conjecture, the prover finds its fix-point with respect to a predefined set of inference rules/axioms. In other words, it finds all the properties that can be deduced from that configuration, using those axioms. A conjecture is proved if it is among the deduced facts. Otherwise, the conjecture is not proved, and a different approach must be used to prove or disprove it.

The algorithm is a data-based search strategy, where a list of new facts is kept and for each new fact the system applies all possible inference rules, eventually generating other new facts. The facts that are being discovered along the process are kept in an old facts list, being usable later in other inferences. In the original implementation [15] the process could be "restarted" by the introduction of auxiliary points. The list of rules that are able to introduce new points can be applied with the restriction that no new point can be introduced using a previously introduced point, ensuring that only a finite number of new points can be created.

This time the Ceva's theorem cannot be used as a test-case. The current implementations of the method do not deal with numerical equalities involving lengths of line segments. Using, instead, the example `01.gex` belonging to the *JGEx*'s `GDD-Full` set of examples [62] (see Theorem 2.2).

Theorem 2.2 (JGEx Example GDD-Full 01) *Let $\triangle ABC$ be a triangle, D the foot of C in AB, E the foot of B in AC, F the midpoint of BC and G the midpoint of DE, then the lines FG and DE are perpendicular.*



The *JGEx* implementation of the geometry deductive database method is capable of finding the fix-point very fast (0.01 seconds) and from that to produce a nice synthetic proof (see Figure 2.1).

Our implementation, described in chapter 4, is also capable of reaching a fix-point and to prove it. In a development version, `0.6.1` it takes 62.9 seconds to reach that fix-point (only 1.1 seconds to find the proof), clearly it has, for now, efficiency issues that need to be addressed.

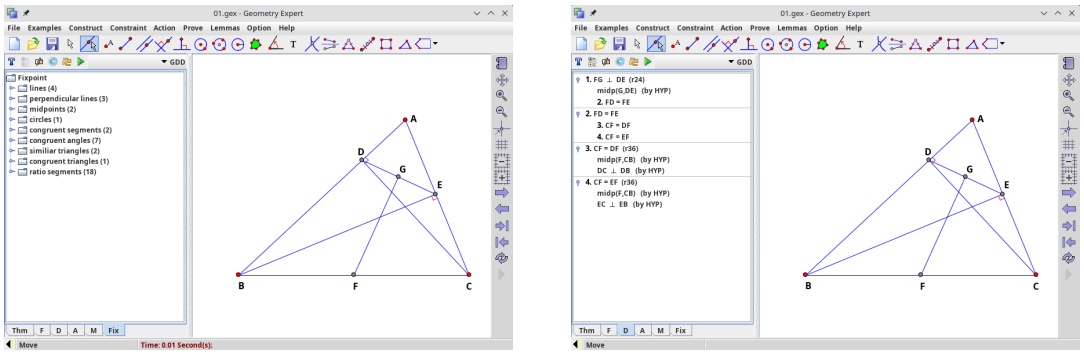


Figure 2.1 JGEx Example GDD-Full 01, Fix-point and Proof

Overall, the existing methods for automated theorem proving and discovering in geometry are very efficient and successful. Nevertheless, when educational uses are to be considered [7, 29, 41, 42, 58, 63], there is still a lot of room for further improvement, in terms of axioms systems used, speed, readability of the outputs and its integration in learning environments. The integration of theorem proving features in a DGS is not an ad-hoc task. It is a complex process yielding an evolving system, meeting users' needs and progress in theorem proving technology. Once more, from an educational perspective, DGS could then be used as an expert system in elementary geometry, which not only tell a yes/no answer, but are capable of showing a step-by-step explanation, if a machine generated proof is considered human readable. Systems like *Java Geometry Expert (JGEx)* [63], *Geometry Constructions to LaTeX Converter (GCLC)* [27] and *Geogebra* [32, 33], among others, incorporate several GATP, providing automated deductive methods for the mathematical practitioner.

Chapter 3

Open Geometry Prover Community Project

The *Open Geometry Prover Community Project* is a long term effort from the automated deduction in geometry community. It has been started by Predrag Janičić from the University of Belgrade, Serbia, as the *Open Geometry Prover* project, with the implementation, in *Java*, of the full-angle method by Ivan Petrović, under the supervision of Predrag Janičić. This first effort was, unfortunately, not concluded given that Ivan Petrović was "bewitched by the song of the sirens", and went working on a software house. Given the complexity of this first (incomplete) implementation, lack of documentation and lack of support from the original programmer, a first attempt to finish this GATP was unsuccessful [1, 2]. It was then decided to start over, beginning by laying down all the basis for a true community project where all different efforts by the community can found a common "house".

The *Open Geometry Prover Community Project* (OGPCP) aims at the integration of the different efforts made in the development of geometry automated theorem provers, under a common "umbrella". Therefore, one of OGPCP's goals is to become a library of GATP, that is, a set of GATP that can be used as stand-alone programs, but mainly as "books" in a "library", ready to be "read" by other programs, e.g. to be integrated in tutorial systems or in DGS, providing the automated deduction features needed by those systems.

Nuno Baeta is responsible for conducting this project, setting the source repository, the application programming interface, programming some of the filters already contained in the library and implementing a GATP base in the deductive databases method (see Chapter 4).

Following, the article [4] — Nuno Baeta and Pedro Quaresma, *Open Geometry Prover Community Project*, Electronic Proceedings in Theoretical Computer Science, volume 352, 30 December 2021, doi:10.4204/EPTCS.352.14 — describes the *Open Geometry Prover Community Project*, its goals, current status and future endeavours.

Open Geometry Prover Community Project *

Nuno Baeta[0000-0002-1629-7924]
CISUC
University of Coimbra, Portugal
nmsbaeta@gmail.com

Pedro Quaresma[0000-0001-7728-4935]
CISUC, Department of Mathematics
University of Coimbra, Portugal
pedro@mat.uc.pt

Mathematical proof is undoubtedly the cornerstone of mathematics. The emergence, in the last years, of computing and reasoning tools, in particular automated geometry theorem provers, has enriched our experience with mathematics immensely. To avoid disparate efforts, the *Open Geometry Prover Community Project* aims at the integration of the different efforts for the development of geometry automated theorem provers, under a common "umbrella". In this article the necessary steps to such integration are specified and the current implementation of some of those steps is described.

1 Introduction

Mathematical proof is undoubtedly the cornerstone of mathematics. All mathematics practitioner know its centrality and the difficulty in mastering it [8]. The emergence, in the last years, of computing and reasoning tools, in particular automated geometry theorem provers, has enriched our experience with mathematics immensely. Building such tools and exploring their applicability require a coherent, well-organized community of researchers working in a collaborative way, to avoid disparate efforts, as recalled by T. Han et al. [7]. Reuse of previous knowledge is vital for human beings in all kinds of learning activities, and so much more in mathematics. The reuse of practical implementations of an abstract idea is usually much harder than the reuse of the abstract idea itself. The same algorithm may be implemented several times using different programming languages and data formats due to engineering mismatches.

The *Open Geometry Prover Community Project* (*OGPCP*) aims at the integration of the different efforts for the development of geometry automated theorem provers, under a common "umbrella". As such, a contribution to the larger goal of establishing a network of researchers working in the area of formal reasoning, knowledge-based intelligent software and geometric knowledge management, to explore efficient methodologies for the creation and reuse of electronic tools in geometry.

To bring up such a framework a series of tools and protocols must be implemented/established. The *Open Geometry Prover Community Project* framework, goals are:

- to provide a common open access repository for the development of Geometry Automated Theorem Provers (GATP);
- to provide an API to the different GATP in such a way that they can be easily used by users, stand-alone or integrated in other tools;
- to develop portfolio strategies to allow choosing the best GATP for any given geometric conjecture;
- to interface with repositories of geometric knowledge [27] (e.g. *TGTP*[1] [24], *TPTP*[2] [30]);
- to develop a GATP System Competition to be able to rate GATPs [2, 25].

*This work is funded by national funds through the 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.

[1]Thousand of Geometric problems for geometric Theorem Provers, http://hilbert.mat.uc.pt/TGTP/

[2]Thousands of Problems for Theorem Provers, http://www.tptp.org/

Janičić, Kovács (Eds.): Automated Deduction in Geometry (ADG 2021)
EPTCS 352, 2021, pp. 129–138, doi:10.4204/EPTCS.352.14

Overview of the paper. The paper is organised as follows: first, in §2, the current status of the framework implementation is described. In §3 a short description of the GATP currently incorporated in the *OGPCP* is given. Finally, in §4, conclusions are drawn and future work is discussed.

2 *OGPCP* Implementation Status

The *OGPCP* framework is a never-ending project in the sense that new GATP can be proposed and incorporated in the project at any given moment. Nevertheless many of the steps necessary for its current use and for an easy integration of new future projects are already done.

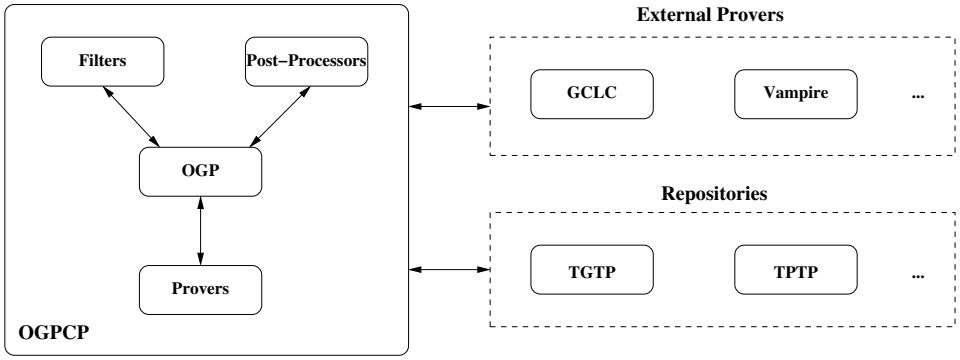


Figure 1: OGPCP Framework

2.1 *OGPCP* Source Repository

The *OGPCP* is hosted at GitHub.[3] The code is made available under the GNU General Public License,[4] version 3 or later, and the documentation under the GNU Free Documentation License,[5] version 1 or later.

OGPCP is available only as source-code and its installation in Unix systems is a straightforward process, provided that GNU Make, Apache Ant and OpenJDK are installed. After downloading the code from the GitHub repository, in the command line just type

```
$ make
$ make install
```

2.2 *OGPCP* Application Programming Interface

OGPCP API is a combination of several command-line tools, e.g., native (i.e. done by the *OGPCP* team and sharing a common base code) and external provers, filters, post-processors, prepared to seamlessly work together or with independent tools.

Native *OGPCP* provers must:

[3] `https://github.com/opengeometryprover/OpenGeometryProver`
[4] `https://www.gnu.org/licenses/gpl.html`
[5] `https://www.gnu.org/licenses/fdl.html`

- use *TPTP*'s first-order format (FOF) syntax as their default conjecture format;
- accept the same command-line arguments;
- provide the same output;

All this is explained in Open Geometry Prover Community Project *Programmer Manual*, available at the *OGPCP*'s GitHub repository.

External provers will be developed by other teams, with different base codes, even, eventually, using different programming languages (no enforcement is done on those matters). External provers conjecture's format may not adhere to TPTP's FOF syntax (see Section 4). In such cases, for *OGPCP* to take advantage of those provers, and vice-versa, filters to/from the FOF syntax must be written, as well as post-processors to interpret the output of those provers.

Example of usage and already implemented features:

Using conjectures *in situ*, contained in local files

`$ ogp ceva.gcl`	use of GCL prover, native language, area method
`$ ogp ceva.gcl -w`	use of GCL prover, native language, Wu's method
`$ ogp ceva.coqam`	use of CoqAM, native language, area method
`$ ogp ceva.fof`	use of inbuilt portfolio mechanism
`$ ogp -t 30 ceva.fof`	use of inbuilt portfolio mechanism with a time limit (30 seconds)

Using conjectures in a remote repository

`$ ogp --tgtp=GEO0001 gclc`	connection to the *TGTP* repository (see § 2.5).

The command line *OGPCP* meta-syntax is the following:

`ogp [<option>] [<conjecture> [<prover> [<prover-options>]]]`

The available options are:

`-h`, `--help`
: prints a help message and exits — to be used alone;

`-p`, `--provers`
: lists the available (to *OGPCP*) GATP and exits — to be used alone;

`-V`, `--version`
: prints *OGPCP* version and exits — to be used alone;

`-t <time>`, `--timeout=<time>`
: redefines the default time limit, in seconds, when proving a conjecture.

The conjecture is provided to the prover in a local file or, when using the remote repository *TGTP*, by its unique id in the repository, using the syntax

`--tgtp=<conjecture_id>`.

When attempting to prove a conjecture, the choice of the prover, if none is indicated, proceeds according to the following rules:

1. if the conjecture is provided in a local file, then
   (a) if file name extension is `fof`, then use *OGPCP* portfolio prover;
   (b) otherwise, use the default prover associated with that extension
2. otherwise, use *OGPCP* portfolio prover.

When the prover is given, a check is made to certify if the conjecture's format is one accepted by the prover. If that is not the case, whenever possible filters are used to convert to a format accepted by the prover. If all this fails, an error occurs and the process ended.

2.3 *OGPCP* Filters & Post-processors

A set of filters are already ready to be used.

`filterGCLtoFOF`	GCL language to FOF
`filterGEOGEBRAtoFOF`	GeoGebra[6] to FOF
`filterJGEXtoFOF`	JGEX to FOF

for the moment all these filters, `filter*toFOF`, assume the inclusion of the axioms of the deductive database full-angle method [6], given that these are already converted to FOF syntax.[7] That is, a plain conversion is made and an `include` instruction is added at the begin with the above mentioned axiom set.

Post-processors are to be used in conjunction with independent provers. They are used to obtain information about the proof's result, e.g., if the proof was successful or not, time, file with the proof steps, if any, etc., as the output of an independent prover must not adhere to that of a native *OGPCP* prover.

As of this writing, there is only one post-processor — for the Vampire ATP, to get the time of a proof.

2.4 *OGPCP* Portfolio

Portfolio problem solving is an approach in which for an individual instance of a specific problem, one particular, hopefully the most appropriate, solving technique is automatically selected among several available ones and used. Weindenbach [32] makes the distinction between syntactic and semantic approaches. With a *Simple-Syntactic portfolio solver* the selection of the core solvers is done by purely syntactic problem properties and there is no exchange of results between different core solvers. In a *Sophisticated-Semantic portfolio solver* the selection of the core solvers is done by semantic or structural problem properties and the solvers exchange results [32].

Already some work in the area of geometric automated theorem proving has been done, namely in the prover mechanism implemented in GeoGebra [16, 17, 22]. It is expected that this research can be incorporated into the *OGPCP*.

2.5 *OGPCP* Interface with Repositories

A server/client architecture to connect *OGPCP* and *TGTP* is already available. On the side of the *TGTP* repository a query-server is already implemented, always listening to client requests.

The code for the clients is open-source and available as part of the *OGPCP* project. The clients are build in such a way that a SQL query can be send to the *TGTP* database, receiving in return the code of the desired conjecture. The exchange of information between the server and the client is done using the *JSON* format.[8] The implementation of new clients to other GATP it is easy and opens the use of the information contained in *TGTP* from any GATP. This server/client architecture is currently being used to establish a connection between the e-learning environment *Web Geometry Laboratory* and the *TGTP* repository [27, 28].

For example, using the *OGPCP* API we could write: `ogp --tgtp=GEO0001 gclc`. This call will trigger the `tgtpToOgpcp` client, sending a query about the *gclc* code for problem *GEO0001* in *TGTP*. If

[6] `https://www.geogebra.org/`

[7] GEO012+0.ax, `http://www.tptp.org/cgi-bin/SeeTPTP?Category=Axioms`

[8] `https://www.json.org/`

the problem do not exist an error code will be returned, if the *gclc* for such a problem do not exist, the *FOF* code for that problem will be returned. After receiving an error free answer, the `ogp` command will pursue as usual.

2.6 Geometry Automated Theorem Provers Systems Competition

To be able to compare the different methods and implementations, a competition will have the virtue of pushing towards the standardization of the input language, the standardization of test sets, the direct comparability and the easier exchange of ideas and algorithmic techniques. The results of such a competition will also constitute a showcase, where potential users will look for the best GATP for their goals [2, 25].

A first trial-run of the *Geometry Automated Theorem Provers Systems Competition*, GASC 0.2, was already run, at ThEdu'19, the *8th International Workshop on Theorem proving components for Educational software*, August 2019, Natal, Brazil [25, 26] and a second trial-run is being prepared.

Not being directly related to the *OGPCP* the GASC will be used to test the different GATP in the project, pushing towards the development of new and better implementations.

3 External Geometry Automated Theorem Provers

A set of external GATP are already part of the OGPCP. These are autonomous open source projects that recognise the *OGPCP* and from which filters to/from the native syntax and FOF are already implemented, or will be implemented in a near future.

Those GATP must be downloaded and installed in a separate way, simple instructions on how to do it will be part of the *OGPCP* documentation.

GeoGebra Automated Reasoning Tools. The standard version of *GeoGebra*[9] includes several Automated Reasoning Tools (ART):

- for conjecturing a geometric property (e.g. such three points visually "seem" to be aligned), the `Relation` command;
- for rigorously denying or confirming a given conjecture (e.g. providing an affirmative answer to the conjecture after internally verifying, using Computer Algebra tools, that some determinant involving the coordinates of the three selected points is zero), the `Prove` and `ProveDetails` commands;
- for presenting some complementary hypotheses for the truth of a given (actually false) statement (e.g. remarking that the truth of the proposed statement needs some further steps in the geometric construction describing the statement), the `LocusEquation` command.

See [3] for a detailed explanation about the project and [15] for a tutorial-like paper about the different commands, as well as [19] for a quite updated version.

Moreover there are two other reasoning toolsets, already implemented but in (yet) non-standard versions of *GeoGebra* [4, 20]. The first one contains the `Discover` tool and command, and the `Compare` command, can be used in the *GeoGebra Discovery* fork,[10] available in two different options: one, operating over *GeoGebra Classic 5*, for *MS-Windows*, *Mac* and *Linux* systems; and, the other, working over *GeoGebra Classic 6*, that requires starting it in a browser, for tablets and smartphones.

[9] `https://www.geogebra.org/download`

[10] `https://github.com/kovzol/geogebra-discovery`, `http://autgeo.online/geogebra-discovery/`

The `Discover` command automatically finds all theorems (of a certain kind: parallelism, congruence, perpendicularity, etc.) holding over a given element of a construction (e.g. involving a point), by considering some combinatorial heuristics to formulate different *Relation* tests involving always the selected element, plus some other one, and presenting as output the collection of obtained properties. The `Compare` command is used to find a general relationship between two quantities (for example, by comparing the sum of the lengths of the catheti a and b and the length of the hypotenuse c in a right triangle—clearly, here the relationship is an inequality, namely, $c < a+b < \sqrt{2}c$). This low-level command is usually called from an improved version of the `Relation` command [31].

The second currently on-going improvement deals with the development of an *AG=Automated Geometer*,[11] a "geometer" that does not require human intervention, except that of launching the computation process over a figure. It is a web-based module that allows GeoGebra to automatically produce different conjectures over the given geometric construction, and to internally confirm or deny them using tools similar to those in GeoGebra Discovery, but here not limited to exploring relations involving a single, specific element.

The algorithms behind all these tools deal with the algebraic translation of the geometric statements and the symbolic manipulation—via the embedded computer algebra system GIAC[12] [13]—of the corresponding complex algebraic geometry varieties. See [14, 18, 21, 29], for a detailed description of the involved theoretical approach.

It must be remarked that the chosen method is quite effective and is able to deal, in milliseconds and over a variety of popular electronic devices (laptops, smartphones, etc.), with very complicated statements but, on the other hand, it does not provide any human-understandable arguments for the declared truth/falsity of the involved statements.

Finally, we summarize how GeoGebra's features can be directly exploited by OGPCP. GeoGebra offers two application programming interfaces for external programs:

- A JavaScript Application Programming Interface (API), available for web applications. The suggested method is to set up the construction via JavaScript calls and then execute the command `ProveDetails` to obtain the result.
- The desktop application can be directly called with an input GeoGebra file. The file structure is given in XML. By creating the XML data as input, and calling GeoGebra via command line, the debug information can be directly processed to get the result.

GCLC Automated Reasoning Tools. Within the mathematical software GCLC, there is an implementation of the area method [12] by Janičić and Quaresma, and implementations of (simple) Wu's method and Gröbner based method, by Predović and Janičić [11]. Apart a graphical user interface all the GATP can be used in stand-alone mode, begin usable in the overall *Open Geometry Prover Community Project* interface.

CoqAM. The formalization within the *Coq* proof assistant of the area method, a decision procedure for affine plane geometry [12, 23].[13] It can be used in stand-alone mode (*Coq* must be installed), being possible its use within overall *Open Geometry Prover Community Project* interface.

[11] `http://autgeo.online/ag/`, `https://github.com/kovzol/ag`
[12] Giac/Xcas, `https://www-fourier.ujf-grenoble.fr/~parisse/giac.html`
[13] `https://dpt-info.u-strasbg.fr/~narboux/area_method.html`

JGEX. Java Geometry Expert, *JGEX*, is a software which combines dynamic geometry software (DGS), automated geometry theorem prover (GATP) and an approach for visually dynamic presentation of proofs. As a dynamic geometry software, *JGEX* can be used to build dynamic visual models to assist teaching and learning of various mathematical concepts.[14]

Apart the use of the GATP systems inside the overall graphical DGS interface, they can be used standalone, being possible its use within the overall *Open Geometry Prover Community Project* interface.

Generic ATP Apart from these ATP, specific to geometry (GATP), the generic ATP can also be used. It is a question of including an axiomatic theory specific to geometry, e.g. those in the *Geo* domain in *TPTP* (Hilbert geometry; Tarski geometry, Rules of construction (von Plato), deductive database method, among others). Not being in the core of the project its use is, nevertheless, being taken in consideration and the *Open Geometry Prover Community Project* command line tool will process an input related to such tools.

4 Conclusions and Future Work

The problems related to the integration between different geometry provers can be much more harder than the presented above. Different algorithms/provers do not assume all the same mathematical setting. Different axiomatizations exist, e.g. Tarski's, Hilbert's, von Plato's; Area method. Different kinds of geometry, e.g. euclidean 2D or 3D, non-euclidean. Different types of approaches, geometric, e.g. area method, algebraic, e.g. Wu's method. More than a, maybe unrealistic, full integration, the *OGPCP* should aim to: give a simple, documented, open source, API to allow the use of GATP by experts and non-experts and to constitute itself as a forum, a space of discussion, about the deductive tools for geometry.

Apart many improvements in the existing framework, e.g. improve the API, linking with external provers, filters and post-processing, new native provers are planned: a new implementation of the full-angle method [5], the deductive database method [6] (using the axioms of the full-angle method), and a novel approach, the deductive graphs method, based on the deductive database method but using deductive graphs. Some initial work has already been done in those methods [1, 9, 10].

As said at the beginning the *OGPCP* is meant to be a never ending project in the sense that new improvements in the area of automated deduction will be made and incorporated in it. New methods, new implementations, improvements in the existing approaches, etc. To enlarge the usefulness and conquer new "audiences" (e.g. teachers in primary and secondary tools) the GATP need to be more modular, being able to be incorporated into "friendly" tools, that can cover the "difficult nature" of many GATP. The *OGPCP* should help on the goal of "bring the automated deduction to all geometers".

Acknowledgements

Zoltán Kovács, Tomás Recio, Francisco Botana (*GeoGebra*), Predrag Janičić (*GCLC*) and Julien Narboux (*COQAm*), contributed to this work by helping writing section 3.

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

References

[1] Nuno Baeta & Pedro Quaresma (2013): *The full angle method on the OpenGeoProver*. In C. Lange, D. Aspinall, J. Carette, J. Davenport, A. Kohlhase, M. Kohlhase, P. Libbrecht, P. Quaresma, F. Rabe, P. Sojka, I. Whiteside & W. Windsteiger, editors: *MathUI, OpenMath, PLMMS and ThEdu Workshops and Work in Progress at the Conference on Intelligent Computer Mathematics*, *CEUR Workshop Proceedings* 1010, Aachen. Available at `http://ceur-ws.org/Vol-1010/paper-08.pdf`.

[2] Nuno Baeta, Pedro Quaresma & Zoltán Kovács (2020): *Towards a Geometry Automated Provers Competition*. In: *Proceedings 8th International Workshop on Theorem proving components for Educational software*, *Electronic Proceedings in Theoretical Computer Science* 313, pp. 93–100, doi:10.4204/EPTCS.313.6. (ThEdu'19), Natal, Brazil, 25th August 2019,.

[3] 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.

[4] Francisco Botana, Zoltán Kovács & Tomás Recio (2020): *A Mechanical Geometer*. *Mathematics in Computer Science*, doi:10.1007/s11786-020-00497-7.

[5] 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(13), pp. 349–370, doi:10.1007/BF00283134.

[6] 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, p. 219–246, doi:10.1023/A:1006171315513.

[7] The Anh Han, Lúis Moniz Pereira & Tom Lenaerts (2019): *Modelling and Influencing the AI Bidding War: A Research Agenda*. In: *AAAI/ACM conference on AI, Ethics and Society 2019*, doi:10.1145/3306618.3314265. Available at `http://www.aies-conference.com/wp-content/papers/main/AIES-19_paper_28.pdf`.

[8] 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.

[9] Yannis Haralambous & Pedro Quaresma (2014): *Querying Geometric Figures Using a Controlled Language, Ontological Graphs and Dependency Lattices*. In S. Watt et al., editor: *CICM 2014*, *LNAI* 8543, Springer, pp. 298–311, doi:10.1007/978-3-319-08434-3_22.

[10] Yannis Haralambous & Pedro Quaresma (2018): *Geometric Search in TGTP*. In Hongbo Li, editor: *Proceedings of the 12th International Conference on Automated Deduction in Geometry*, SMS International. Available at `http://adg2018.cc4cm.org/ADG2018Proceedings`.

[11] Predrag Janičić (2006): *GCLC — A Tool for Constructive Euclidean Geometry and More Than That*. In Andrés Iglesias & Nobuki Takayama, editors: *Mathematical Software - ICMS 2006*, *Lecture Notes in Computer Science* 4151, Springer, pp. 58–73, doi:10.1007/11832225_6.

[12] 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.

[13] Z. Kovács & B. Parisse (2015): *Computer algebra and polynomials*, chapter Giac and GeoGebra – improved Gröbner basis computations, pp. 126–138. *LNCS* 8942, Springer Cham, doi:10.1007/978-3-319-15081-9_7.

[14] Z. Kovács, T. Recio & C. Sólyom-Gecse (2019): *Rewriting input expressions in complex algebraic geometry provers*. *Annals of Mathematics and Artificial Intelligence* 85(2-4), pp. 73–87, doi:10.1007/s10472-018-9590-1.

[15] Zltan Kovács, Tomas Recio & Maria Pilar. Vélez (2018): *Using Automated Reasoning Tools in GeoGebra in the Teaching and Learning of Proving in Geometry*. *International Journal for Technology in Mathematics Education* 25(2), pp. 33–50, doi:10.1564/tme_v25.2.03.

[16] Zoltán Kovács (2014): *The portfolio prover in GeoGebra 5*. In: *Proceedings of the 10th International Workshop on Automated Deduction in Geometry (ADG 2014)*.

[17] Zoltán Kovács (2015): *The Relation Tool in GeoGebra 5*. In Francisco Botana & Pedro Quaresma, editors: *Automated Deduction in Geometry*, *Lecture Notes in Computer Science* 9201, Springer International Publishing, pp. 53–71, doi:10.1007/978-3-319-21362-0_4.

[18] Zoltán Kovács, Tomás Recio & M. Pilar Vélez (2019): *Detecting truth, just on parts*. *Revista Matemática Complutense* 32(2), pp. 451–474, doi:10.1007/s13163-018-0286-1.

[19] Zoltan Kovács, Tomas Recio & Maria Pilar Vélez (2022): *Mathematics Education in the Age of Artificial Intelligence*, chapter Automated Reasoning Tools with GeoGebra: What are they? What are they good for? Springer Nature. (to appear).

[20] Zoltán Kovács & Tomas Recio (2020): *GeoGebra Reasoning Tools for Humans and for Automatons*. In: *Electronic Proceedings of the 25th Asian Technology Conference in Mathematics*, Radford University, Radford, Virginia, USA, and Suan Sunandha Rajabhat University, Thailand, Mathematics and Technology, LLC, pp. 16–30, doi:10.13140/RG.2.2.26851.58407. Available at `http://atcm.mathandtech.org/EP2020/invited/21786.pdf`.

[21] Manuel Ladra, Pilar Páez-Guillán & Tomás Recio (2020): *Dealing with negative conditions in automated proving: tools and challenges. The unexpected consequences of Rabinowitsch's trick*. *Revista de la Real Academia de Ciencias Exactas, Físicas y Naturales. Serie A. Matemáticas* 114(4), doi:10.1007/s13398-020-00874-8.

[22] Zoltán Kovács & Predrag Janičić Mladen Nikolić, Vesna Marinković (2019): *Portfolio theorem proving and prover runtime prediction for geometry*. *Annals of Mathematics and Artificial Intelligence* 85(2-4), pp. 119–146, doi:10.1007/s10472-018-9598-6.

[23] Julien Narboux (2004): *A Decision Procedure for Geometry in Coq*. *Lecture Notes in Computer Science* 3223, pp. 225–240, doi:10.1007/b100400. Available at `http://portal.acm.org/citation.cfm?id=1784950.1784959`.

[24] Pedro Quaresma (2011): *Thousands of Geometric Problems for Geometric Theorem Provers (TGTP)*. In Pascal Schreck, Julien Narboux & Jürgen Richter-Gebert, editors: *Automated Deduction in Geometry*, *Lecture Notes in Computer Science* 6877, Springer, pp. 169–181, doi:10.1007/978-3-642-25070-5_10.

[25] Pedro Quaresma & Nuno Baeta (2019): *Geometry Automated Theorem Provers Systems Competition 0.2 Report*. techreport 1, CISUC. Available at `https://www.cisuc.uc.pt/ckfinder/userfiles/files/TR2019-01.pdf`.

[26] Pedro Quaresma, Walther Neuper & João Marcos, editors (2020): *Proceedings 8th International Workshop on Theorem Proving Components for Educational Software*. 313, Open Publishing Association, doi:10.4204/EPTCS.313.

[27] Pedro Quaresma, Vanda Santos & Nuno Baeta (2018): *Exchange of Geometric Information Between Applications*. *Electronic Proceedings in Theoretical Computer Science* 267, pp. 108–119, doi:10.4204/eptcs.267.7.

[28] Pedro Quaresma, Vanda Santos & Milena Marić (2018): *WGL, a web laboratory for geometry*. *Education and Information Technologies* 23(1), pp. 237–252, doi:10.1007/s10639-017-9597-y.

[29] 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`.

[30] G. Sutcliffe (2017): *The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0*. *Journal of Automated Reasoning* 59(4), pp. 483–502, doi:10.1007/s10817-017-9407-7.

[31] Róbert Vajda & Zoltán Kovács (2020): *GeoGebra and the realgeom Reasoning Tool*. In P. Fontaine, K. Korovin, I. S. Kotsireas, P. Rümmer & S. Tourret, editors: *PAAR+SC-Square 2020. Workshop on Practical Aspects of Automated Reasoning and Satisfiability Checking and Symbolic Computation Work shop 2020*, pp. 204–219. arXiv:http://ceur-ws.org/Vol-2752/paper15.pdf.

[32] Christoph Weidenbach (2017): *Do Portfolio Solvers Harm?* In Giles Reger & Dmitriy Traytel, editors: *ARCADE 2017. 1st International Workshop on Automated Reasoning: Challenges, Applications, Directions, Exemplary Achievements*, *EPiC Series in Computing* 51, EasyChair, pp. 76–81, doi:10.29007/vpxm.

Chapter 4

Geometry Deductive Database Prover

A first attempt to develop a geometry automatic theorem prover based in a deductive database approach was reported by Chou et al. in [15]. The basic idea is to use a deductive database [19] to be able to prove or discover nontrivial geometry theorems, by exploring the full strength of forward chaining. For a given geometric configuration, a GATP based in this approach can find its fix-point with respect to a fixed set of geometric rules or axioms; in other words, it can find all the properties of the configuration that can be deduced using these axioms and if the conjecture is among those facts, it is a theorem. Being based in a set of geometric inference rules and in forward chaining reasoning, this method has the attractiveness of being able to produce a geometric proof. A first implementation is included in the software *JGEx* [62].

In approaching a new implementation of this method two main goals were established, both new when taking in consideration the known implementation. An implementation as a library, a "book" to be integrated in *Open Geometry Prover Community Project*. A generic prover, being able to accept different sets of rules and develop proofs in those sets. To fulfil these specifications it, was decided to use the *C++* programming language and a *SQL* database implemented as a library and that allowed to use in-memory databases.[1] After having taken that decision, the overall layout of the prover was established and a first prototype was implemented. Nuno Baeta is responsible for the implementation of the GATP, integrating all the rules proposed in [15], correcting the founded inconsistencies (see Appendix C) and incorporating it in the *Open Geometry Prover Community Project* library.

In the current version `0.6.1`[2] the *OGPGDDM* prover is already capable of proving some simple geometric conjectures, for example the theorem that states that (see Listing 4.1), *if a line intersects two sides of a triangle and splits those sides in equal proportions, then that line must be parallel to the third side of the triangle*, can be proved in 0.016179 seconds and it reaches

[1] The library, *SQLite*, was used, https://www.sqlite.org/

[2] Available at GitHub: https://github.com/opengeometryprover/OpenGeometryProver/tree/master/provers/ogpgddm (open access)

the fix-point in 88.4349 seconds (see Listing 4.2[3]), producing a file with all the geometric facts derived from the original construction (see Appendix D).

Listing 4.1 Parallel Lines in a Triangle Theorem

```
%----Include Geometry Deductive Database Method axioms
include('geometryDeductiveDatabaseMethod.ax').

fof (geo0007,conjecture,! [A,B,C,D,E] :
     ( ( midp(D,C,A) & midp(E,A,B) ) =>  para(B,C,E,D) ) ) .
```

Listing 4.2 Parallel Lines in a Triangle Theorem, OGPGDDM output

```
$ ./ogpgddm geo0007.p
OGP GDDM 0.6.1
Copyright (C) 2022 Nuno Baeta, Pedro Quaresma
Distributed under GNU GPL 3.0 or later
Conjecture is PROVED, in: 0.016179s
Fix-point found, in: 88.4349s
Fix-point saved to file 'geo0007.fp'.
```

A first presentation of this GATP — Nuno Baeta, Pedro Teixeira and Pedro Quaresma, *Rule Based Geometry Automated Theorem Provers* (extended abstract) — was done in the *11th International Workshop on Theorem proving components for Educational software*, 11 August, workshop at *FLoC 2022*, July 31 – August 12, 2022, Haifa, Israel.[4] In the following, the article [5] — Nuno Baeta and Pedro Quaresma, *Towards a Geometry Deductive Database Prover*, Annals of Mathematics and Artificial Intelligence, Springer, 2023, accessible online, doi:10.1007/s10472-023-09839-0 — is presented.

[3] In an Intel(R) Core(TM) i5-4430 CPU @ 3.00GHz, 15GiB of Memory computer, running Debian GNU/Linux 12

[4] https://www.uc.pt/en/congressos/thedu/ThEdu22/programme.

Annals of Mathematics and Artificial Intelligence
https://doi.org/10.1007/s10472-023-09839-0

Towards a geometry deductive database prover

Nuno Baeta[1] · **Pedro Quaresma**[2]

Accepted: 8 February 2023

Abstract
The Geometry Automated-Theorem-Provers (GATP) based on the deductive database method use a data-based search strategy to improve the efficiency of forward chaining. An implementation of such a method is expected to be able to efficiently prove a large set of geometric conjectures, producing readable proofs. The number of conjectures a given implementation can prove will depend on the set of inference rules chosen, the deductive database method is not a decision procedure. Using an approach based in an *SQL* database library and using an in-memory database, the implementation described in this paper tries to achieve the following goals. Efficiency in the management of the inference rules, the set of already known facts and the new facts discovered, by the use of the efficient data manipulation techniques of the *SQL* library. Flexibility, by transforming the inference rules in *SQL* data manipulation language queries, will open the possibility of meta-development of GATP based on a provided set of rules. Natural language and visual renderings, possible by the use of a synthetic forward chaining method. Implemented as an open source library, that will open its use by third-party programs, e.g. the dynamic geometry systems.

Keywords Automated deduction in geometry · Automated geometry theorem proving · Deductive databases

Mathematics Subject Classification (2010) 51-04 · 68T15

1 Introduction

Geometry automated-theorem-proving began, in the late 1950, by adapting the traditional geometric proof methods to the general-purpose reasoning approaches developed in artificial intelligence [8]. Subsequent work, e.g., [5, 11], followed, mostly, the same *synthetic* reasoning of automating the traditional proof methods. Despite their initial success, and

✉ Pedro Quaresma
pedro@mat.uc.pt

Nuno Baeta
nmsbaeta@gmail.com

[1] CISUC, University of Coimbra, Coimbra, Portugal

[2] CISUC, Department of Mathematics, University of Coimbra, Coimbra, Portugal

Published online: 24 May 2023

even though being able to produce readable proofs, these *synthetic methods* did not make much progress as they revealed to be narrow-scoped and inefficient.

In recent years, *synthetic methods* have seen a resurgence, with results like the *ArgoCLP* theorem prover [17], based on coherent logic, or the deductive database approach [4], with mixed results in different classes of geometric problems.

Since the early implementations of automated theorem provers for geometry, synthetic provers based on inference rules and using forward chaining reasoning are considered to be more suited for education proposes. Unlike the *algebraic methods*, and the *semi-synthetic methods* [3], they can produce readable synthetic proofs and, depending on the chosen set of rules, more adapted to a given school audience, e.g. secondary school students [19].

The authors will now present their recent efforts in writing a prover using the geometry deductive database method. The goal of such endeavour are to produce a GATP that is: efficient, flexible, with natural language and visual renderings and implemented as an open source library.

Overview of the paper The paper is organised as follows: first, in Section 2, a brief description of the geometry deductive database method is presented. In Section 3 the prover is discussed. Final conclusions are drawn, and future work is foreseen in Section 4.

2 The geometry deductive database method

We now present a brief description of the geometry deductive database method (*GDDM*), and make some remarks regarding the inference rules and the deductive database. A thorough explanation of this method is provided in [4].

2.1 Brief description

The geometry deductive database method is a synthetic method that uses forward chaining to prove non-trivial geometry theorems efficiently.

Given a geometric configuration, the conjecture, the prover finds its fix-point with respect to a predefined set of inference rules/axioms. In other words, it finds all the properties that can be deduced from that configuration, using those axioms. A conjecture is proved if it is among the deduced facts. Otherwise, the conjecture is not proved, and a different approach must be used to prove or disprove it.

The algorithm is a data-based search strategy, where a list of *new facts* is kept and for each *new fact* the system applies all possible inference rules, eventually generating other *new facts* (see Fig. 1). The facts that are being discovered along the process are kept in an *old facts* list, being usable later in other inferences. In the original implementation [4] the process could be "restarted" by the introduction of auxiliary points. The list of rules that are able to introduce new points can be applied with the restriction that no new point can be introduced using a previously introduced point, ensuring that only a finite number of new points can be created.

2.2 Inference rules

In [4] a set of inference rules is proposed, along with some guidelines on how to select a "better" set of geometric (inference) rules. However, due to syntactic inconsistencies found

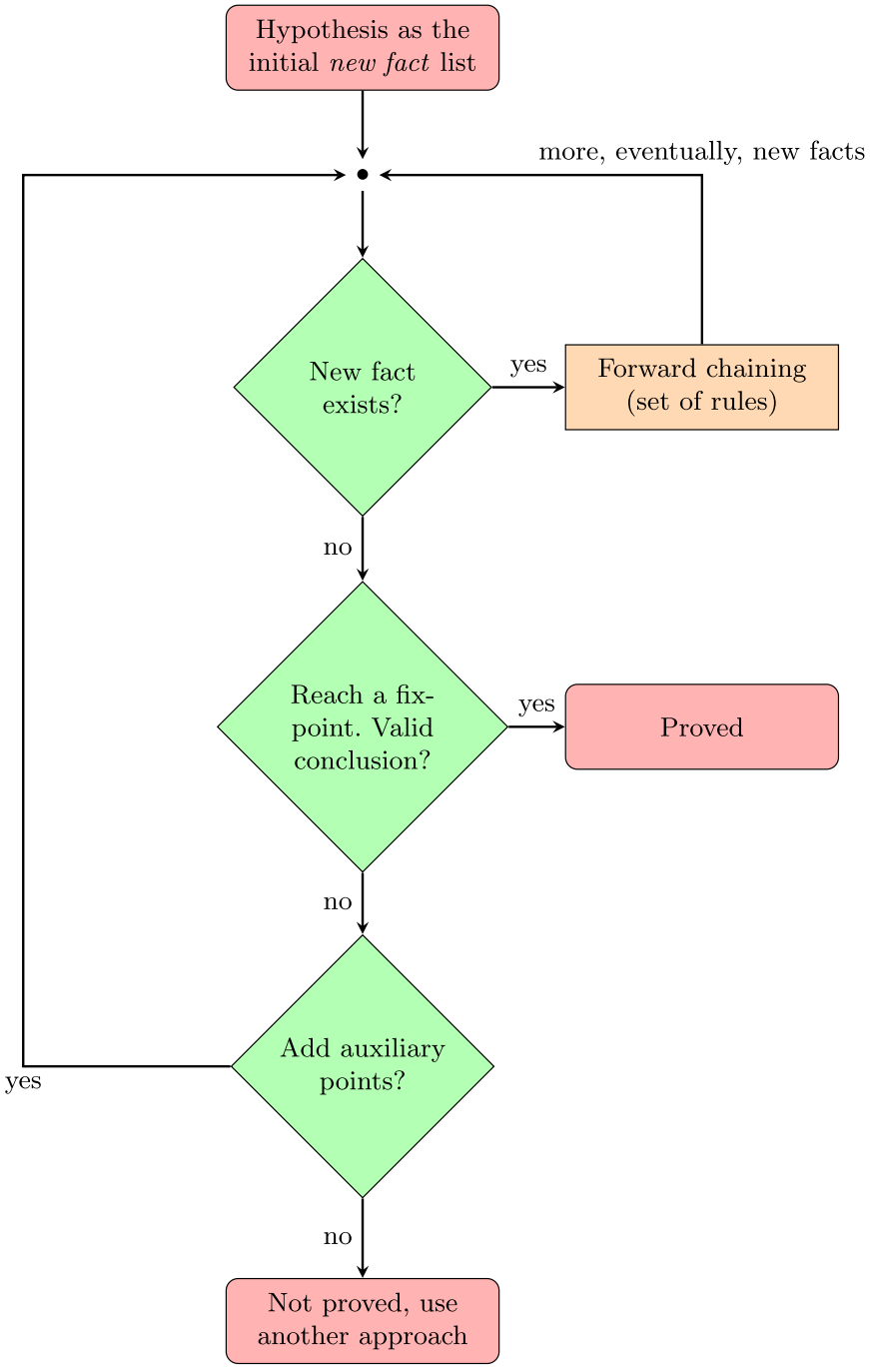


Fig. 1 GDDM algorithm

in that list, some adaptations/correction were made. We now present two examples were correction were necessary:

– Rule D42 states in its assumptions that points `P`, `Q`, `A` and `B` are collinear, i.e., `coll(P, Q, A, B)`, but `coll` (collinear) is a 3-ary predicate. The solution is simple, assume `coll(P, Q, A) & coll(P, Q, B)`.

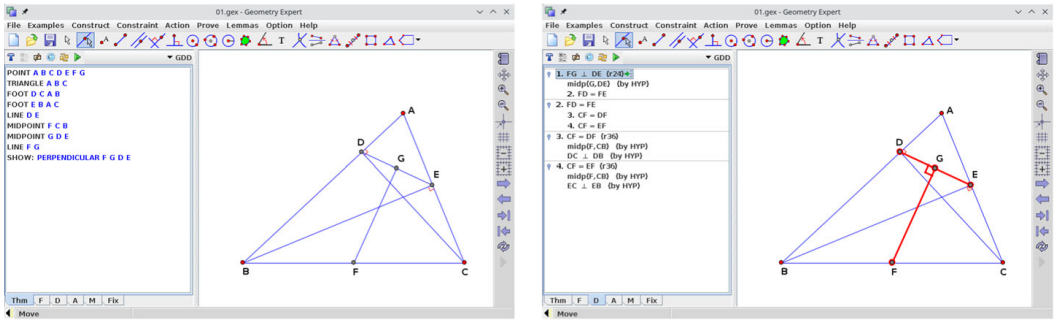


Fig. 2 Problem 01, *JGEx*, GDD method

– Rule 61 uses the expression `AB = PQ`, and nowhere is defined the equality of line segments. However, the predicate congruent segments is defined. Therefore, we used `cong(A, B, P, Q)` instead.

The set of rules presented in [4] is consistent[1] but it is far from adequate. *The Java Geometry Expert* (*JGEx*) [20], written by the authors of [4], and the only, as far as the authors are aware, GATP that implements the *GDDM*, uses a different set of rules than the ones in [4]. Indeed, there are some *JGEx* examples that cannot be proven by the set of rules presented in [4].

For example, for the problem `01.gex` of the set of problems in *JGEx* related to the the *GDD* prover (see Fig. 2), the following rules are used: steps 3, 4 (`r36`) and 2 (rule without an explicit identifier) are in correspondence with rules `D52` and `D25` of the rule-set presented in [4], but the final step, rule `r24`, have no counterpart in that rule-set. A new rule[2] must be added in order to be able to prove this example.

```
B1. perp(F,G,D,E) :- cong(F,D,F,E), midp(G,D,E), ¬coll(F,G,D)
```

where the condition, `¬col(F,G,D)`, is a non-degenerate (ndg) condition.

In example `02.gex`, the very first step of the proof needs, again, this rule. To build an adequate set of rules, based on this set of rules, is among our ongoing projects.[3]

3 The *OGP-GDDM* prover

The goals of the *OGP-GDDM* Prover are: to produce a GATP that is efficient, flexible, with natural language and visual renderings and implemented as an open source library.

Efficient by the use of an in-memory *SQL* database library, a data-based search strategy can be efficiently implemented. This is a still unproved claim. We expected to get a more definitive conclusion as soon as we improve our current implementation.

Flexible by transforming the inference rules in *SQL* data manipulation language (DML) queries, we claim that a generic prover can be built. The prover would begin by including

[1] The generic ATP *Vampire* was used to prove this fact.

[2] Written as a *definite Horn clause*.

[3] In collaboration with Predrag Janičić, from the University of Belgrade, Serbia.

the rules of inference, transformed in DML queries, and then the attempt to prove the conjecture would be made, based of that "set or rules".

Rendering being a synthetic method that uses a forward chaining reasoning, the natural and visual rendering of the proofs will be possible (e.g. see the implementation of the method in *JGEx*)[4] [20].

Open Source the implementation as a library will allow the use of the GATP by third-party programs (e.g. *GeoGebra* [10]). Its integration in the *Open Geometry Prover Community Project* (*OGPCP*)[5] [1] and the use of the *First Order Format* (FOF), as specified in [18], will allow an easy implementation of filters from/to the prover to/from other programs. The geometric predicates used are those described in [4].

3.1 Efficiency considerations

The geometry deductive databases method, as described in [4], uses the ideas from deductive database theory, e.g. fix.points and the treatment of negative clauses, proposing a structured deductive database and a data-based search strategy to improve search efficiency. Our idea is to avoid a painful and difficult task of implement such structured deductive database by using an off-the-shelf *SQL* database engine. The easy-of-use and power gain by the use of the *Data Definition Language* (DDL) and *Data Manipulation Language* (DML) will give an efficient (still to be proved) and flexible (see Section 3.2) solution.

Efficiency consideration strongly implied that a library, in-memory, database must be used. The library implementation to allow an easy integration in our prover and, given the expected dimension of the database, an in-memory solution is appropriated. The choice fell on *SQLite*.[6] It provides an efficient and reliable management and query system. It is a library easily integrated in *C/C++* programs, and with the needed option of an in-memory database.

3.2 Flexibility considerations

A rule based theorem prover, like the deductive database method, will reach its maximum usefulness (in terms of wider users base) if it can accept different rules sets. By transforming the inference rules in *SQL* data manipulation language queries, we aim at getting that degree of flexibility.

For now, the implementation has the rules in [4] (see Section 2.2) hard-coded. A modular approach was used with a clear separation between the parsing of the FOF input text, the application of the different inference rules, the overall prover mechanism, etc.

As an example, let's consider rule D1 about collinear points:

```
D1.   coll(A,C,B) :- coll(A,B,C)
```

translated (by hand) to FOF:

```
fof(ruleD1,axiom,
    ( ! [A,B,C] : (coll(A,B,C) => coll(A,C,B)) )).
```

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

[5] https://github.com/opengeometryprover/

[6] https://www.sqlite.org/index.html

The steps to add this rule to the prover are the following:

1. If needed, add a rule to the scanner (`scanner.ll`)

```
"coll"      return yy::parser::make_COLL(loc);
```

and the correspondent lines of code to the parser (`parser.yy`)

```
%token    COLL      "coll"
(...)
%type <int> coll;
(...)
coll:
   "coll" "(" "identifier" "," "identifier" ","
   "identifier" ")"
    {
        drv.typeGeoCmd[drv.numGeoCmd] = "coll";
        drv.point1[drv.numGeoCmd] = $3;
        drv.point2[drv.numGeoCmd] = $5;
        drv.point3[drv.numGeoCmd] = $7;
    };
```

2. If needed, in the file `dbRAM.cpp`, create a new table in the database for the collinear predicate—for each predicate there is one table.
3. If needed, in the file `foftodb.cpp`, modify the function `readFileLoadDB()` so that collinear facts may be added to the database, in this case the fact `coll(A, B, C)`.
4. Add a rule to D1 as a function in `prover.cpp` (see Listing 1).
5. In the file `prover.cpp`, modify the function `fixedPoint()`, where the fix-point is calculated, to used the rule D1.
6. If needed, in the file `prover.cpp`, modify the function `proved()`, where the fix-point is searched to check if the method proved a conjecture, in this case to search for `coll(A, C, B)`.
7. If needed, in the file `prover.cpp`, modify the function `showFixedPoint()`, where the calculated fix-point is displayed, so that collinear facts are shown.

Our ultimate goal is to write a generic prover, capable of accept different set of rules and use then as an inference base.

The planned work goes in the direction of building a meta-prover-builder, that is, a program that, given a set of inference rules, previously checked for consistency, synthesise another program, a gddm-prover based on the rules just provided. If successful we will try to work on the next step a generic rule-based geometric automated theorem prover.

3.3 Rendering considerations

In a parallel project devoted to the use of automated deduction tools and methods in secondary schools, where the authors also participate, it is claimed that, one of the bottlenecks that the introduction of automated deduction systems in secondary schools face is the difficulty of tackling the task by automatic means [14]. There is the need of efficient provers (e.g. Wu's (algebraic) method), but there is also the need of readable proof scripts (e.g. Area Method, semi-synthetic) and this two objectives are difficult to attain together [12, 13]. There is also the need of an automated deduction method that is adapted to the secondary

```
// Rule D1: coll(A, B, C) => coll(A, C, B)
DBinMemory Prover::ruleD1(DBinMemory dbim, std::string point1,
                std::string point2, std::string point3) {
  bool correctTransaction;
  std::string insertionColl,insertNewFact;
  std::string lastInsertedRowId,lstInsRwId;

  insertNewFact = "INSERT INTO NewFact(typeGeoCmd) VALUES ('coll')";
  lastInsertedRowId = "SELECT last_insert_rowid()";

  sqlite3_exec(dbim.db, "begin;", 0, 0, &(dbim.zErrMsg));
  correctTransaction = true;
  dbim.rc = sqlite3_prepare_v2(dbim.db, insertNewFact.c_str() ,
                    insertNewFact.size(), &(dbim.stmt), NULL);
  if (sqlite3_step(dbim.stmt) != SQLITE_DONE) {
  correctTransaction = false;
  }
  dbim.rc = sqlite3_prepare_v2(dbim.db,lastInsertedRowId.c_str(),
       lastInsertedRowId.size(), &(dbim.stmt), NULL);
  sqlite3_step(dbim.stmt);
  lstInsRwId = (char*)sqlite3_column_text(dbim.stmt, 0);

  insertionColl = "INSERT
      INTO Collinear(typeGeoCmd,point1,point2,point3,newFact)
      VALUES ('coll','"+point1+"','"+point3+"','"+point2+"','"
      +lstInsRwId+"')";

  dbim.rc = sqlite3_prepare_v2(dbim.db, insertionColl.c_str(),
       insertionColl.size(), &(dbim.stmt), NULL);
  if (sqlite3_step(dbim.stmt) != SQLITE_DONE) {
  correctTransaction = false;
  }
  if (correctTransaction) {
  sqlite3_exec(dbim.db, "commit;", 0, 0, 0);
  } else {
  sqlite3_exec(dbim.db, "rollback;", 0, 0, 0);
  }
  return dbim;
}
```

Listing 1 Rule D1

schools learning, i.e. a method that uses the usual set of rules used on those schools and a method capable of rendering the proofs produced in the usual (in)formalism of secondary schools. As defended in [19], synthetic provers based on inference rules and using forward chaining reasoning are more suited for education proposes.

In [19] two problems, suited to a 7th year class (≈12-year-old students) are presented. An appropriated set of rules based on the set of rules [6, 7] is presented and the proofs (that could be) produced (by an appropriated implementation) of the GDDM method are shown.

The GDDM method is not a decision procedure: this is not an important question when secondary schools education is concerned. The GDDM method is not the most efficient GATP (unproven statement, but probably valid), but that it is not the most important question when secondary schools education is concerned. The important question is the possibility to have the synthetic proof and in a language that students and teachers can understand [9].

Looking again to Fig. 2, where *JGEx*'s GDD prover was used to prove a given geometric conjecture, we can see the construction and the conjecture (left window) and the proof

developed by the *GDD* prover (right window). Looking now only to the right window, it can be seen that in the left section a synthetic proof is shown and in the right section the construction is shown. The conjecture (on the left) and its visual rendering (on the right) are both highlighted and this connection between the proof and the construction can be explored for each step of the proof. What is missing? A set of rules close to the needs of the secondary schools students and teachers, and a tool that can be easily used by them [20].[7]

The development of the *OGP-GDDM* is an ongoing project, one of its goals is: to be able to render the synthetic proof produce by the GDDM in a (in)formal proof, hiding all the necessary details, translating it to a natural language form, etc.; to establish the connection with the construction, linking the proof and its visualisation. The first step is easy, it is a natural "collateral effect" of the GDDM prover, the others are more challenging. The idea is to use the environment *Web Geometry Laboratory* where an integrated DGS (*GeoGebra*) will allow to develop the construction, then the conjecture could be written by the students, and the GDDM prover call in the background and, after the proof was completed, the proof scrip and its visualisation on the construction (in *GeoGebra*), could be explored [15, 16].

3.4 Open source library

If the target audience is not only the ATP community, but the "public" in general (e.g. students and teachers in Secondary Schools) a close-box prover with esoteric input and output languages will be close to completely useless. To reach a larger audience the GATP should be a "book" in a "library", ready to be "read" by other programs, i.e. they should be developed as a library, with a clear documented API,[8] ready to be incorporated in third-party programs (e.g. *GeoGebra*).

The GDDM prover is being developed as an open source library, available at GitHub[9] [1]. It uses, as input language, the *TPTP*,[10] *First Order Format* (FOF), as specified in [18], to allow an easy implementation of filters from/to the prover to/from other programs. It is part of the *Open Geometry Prover Community Project* (*OGPCP*)[11] [1]. The *OGPCP*, aims to integrate different efforts in the development of GATP, namely: to provide a common open access repository for the development of GATPs; to provide an API to the different GATP in such a way that they can be easily used by users; to develop a portfolio strategies to allow choosing the best GATP for any given geometric conjecture; to interface with repositories of geometric knowledge, e.g. *TGTP*,[12] *TPTP*; to develop a *GATP System Competition* (GASC) to allow rating GATPs [2].

Source repository The *OGP-GDDM* is hosted at GitHub.[9] It is code is be made available under the GNU General Public Licence,[13] version 3 or later, and the documentation under the GNU Free Documentation Licence,[14] version 1 or later. It is available only as source-code. Provided that you use a *Unix*-like environment and have the usual tools *make*, *flex*,

[7] *JGEx* is not currently being developed and/or supported, it is available in open source form at: https://github.com/yezheng1981/Java-Geometry-Expert

[8] Application Programming Interface

[9] https://github.com/opengeometryprover/OpenGeometryProver/tree/master/provers/ogpgddm

[10] https://tptp.org/

[11] https://github.com/opengeometryprover/

[12] http://hilbert.mat.uc.pt/TGTP/

[13] https://www.gnu.org/licenses/gpl.html

[14] https://www.gnu.org/licenses/fdl.html

bison, a *C++* compiler and the *SQLite* library installed, it is a straightforward process to compile and install. Just change to the source code directory, `provers/ogpgddm`, and type the following commands:

```
$ make
$ sudo make install
```

Again, this is an ongoing project. The goals are: to build a set of consistent set of rules that can mimic those used in the secondary schools; to implement those rules in an (reasonably) efficient GDDM prover (see Sections 3.1 and 3.2); to be capable of producing natural language and visual renderings (see Section 3.3; to be able to integrate all this in a tool that can be easily used in secondary schools (see Section 3.4);

3.5 Usage and examples

Considering that *OGP-GDDM* is part of *OGPCP* it must adhere to its API, that is, using the FOF format and, in the stand-alone version, having a behaviour consistent with the other *OGPCP* provers.

Usage as a library The static library, `libogpgddm.a`, was created, it contains the necessary "books" to build a prover based in this library. It is enough to include the different "books" (e.g. `#include "prover.hpp"`) and then, at compilation time, the option, `-logpgddm`, must be included. All the files, including the documentation, `ogppm.pdf`, are available in the GitHub repository.

Usage as a stand-alone program A stand-alone version of the prover was already built, using the library. Its command line syntax is:

```
ogpgddm <option> | <conjecture>
```

where, `option`, is one of: "`-h`" or "`--help`", prints a help message and exits; "`-V`" or "`--version`", prints *OGP-GDDM* version and exits. Considerations about the conjecture will be made below. The result of a proof is sent to the standard output, errors included. In its current stage, no file is created, but that will change in a future version—a file with the proof will be created.

Conjecture format Once more, because *OGP-GDDM* is an *OGPCP* GATP, the conjectures are written in FOF. In its current version the inference rules are hard-coded, as such the the inclusion of axioms is ignored.

As an example, the problem `TPTP Problem File: GEO547+1.p`[15] (see Listing 2), will parsed seamlessly, with comments and the line, `include(...)`, being ignored.

In its current version, `0.6.0`, *OGP-GDDM* is already capable to parse any problem in the FOF format. Even with the problems in the set of inference rules (see Section 2.2) *OGP-GDDM* is already capable to prove some problems. For example, the *Midpoint Theorem*, problem `GEO0007` of the *TGTP* repository of geometric problems.

Theorem 1 (Midpoint Theorem) *Let ABC be a triangle, and let D and E be the midpoints of AC and BC respectively. Then the line DE is parallel to the base AB.*

[15] https://www.tptp.org/cgi-bin/SeeTPTP?Category=Problems&Domain=GEO&File=GEO547+1.p

```
%------------------------------------------------------------------------------
% File     : GEO547+1 : TPTP v8.0.0. Released v7.5.0.
% Domain   : Geometry
% Problem  : JGEX problem 07
% Version  : [CGZ00] axioms.
% English  :
% Refs     : [CGZ00] Chou et al. (2000), A Deductive Database Approach
%          : [YCG08] Ye et al. (2008), An Introduction to Java Geometry
%          : [Qua20] Quaresma (2020), Email to Geoff Sutcliffe
% Source   : [Qua20]
% Names    : 07.p [Qua20]
% Status   : Theorem
% Rating   : 0.25 v7.5.0
% Syntax   : Number of formulae    :  95 (   0 unt;   0 def)
%            Number of atoms       : 292 (   1 equ)
%            Maximal formula atoms :   9 (   3 avg)
%            Number of connectives : 204 (   7   ~;   0   |; 102   &)
%                                        (   0 <=>;  95  =>;   0  <=;
%                                         0 <~>)
%            Maximal formula depth :  18 (   9 avg)
%            Maximal term depth    :   1 (   1 avg)
%            Number of predicates  :  12 (  11 usr;   0 prp; 2-8 aty)
%            Number of functors    :   0 (   0 usr;   0 con; ---- aty)
%            Number of variables   : 531 ( 511   !;  20   ?)
% SPC      : FOF_THM_RFO_SEQ
% Comments: Taken from JGEX [YCG08], converted by Pedro Quaresma.
%------------------------------------------------------------------------------
include('Axioms/GEO012+0.ax').
%------------------------------------------------------------------------------
fof(exemplo6GDDFULL012007,conjecture,
    ! [A,B,C,O,D,E,F,G,NWPNT1] :
      ( ( circle(O,A,B,C) & circle(O,A,D,NWPNT1)
        & perp(E,D,B,C) & coll(E,B,C)
        & perp(F,D,A,C) & coll(F,A,C)
        & perp(G,D,A,B) & coll(G,A,B) )
     => coll(E,F,G) ) ).
%------------------------------------------------------------------------------
```

Listing 2 TPTP Problem File: GEO547+1.p

```
fof(geo0007.p,conjecture,! [ A,B,C,D,E ] :
            ((midp(D,C,A) & midp(E,A,B)) =>  para(B,C,E,D))).
```

Listing 3 TGTP Problem GEO0007

```
$ ogpgddm geo0007.p

OGP GDDM 0.6.0
Copyright (C) 2022 Nuno Baeta, Pedro Quaresma
Distributed under GNU GPL 3.0 or later
Conjecture is PROVED, in: 0.009493s
Fix-point found, in: 56.7969s
Fixed point saved to file "geo0007.fp".
```

Listing 4 *OGP-GDDM*, GEO0007

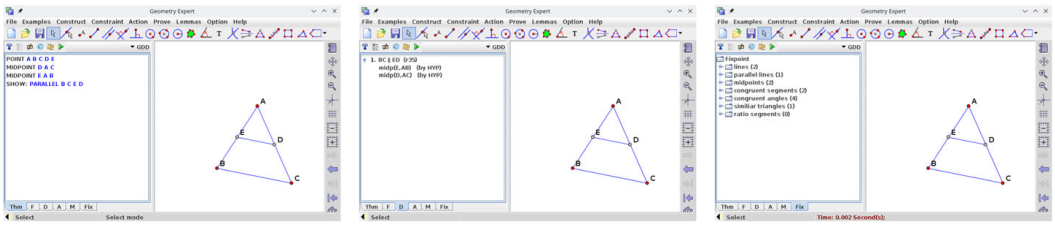


Fig. 3 *JGEx — TGTP*, GEO0007 problem

Given that the *JGEx* implementation (using a modified set of rules) have found the fixed-point in 0.002s (see the lower bar of the right window of Fig. 3), it is clear that the current implementation of GDDM still needs to be improved, in which efficiency is concerned. This is not a surprise, the current implementation of *OGP-GDDM* does not implement any efficiency oriented procedure.

In a, still in development, new version, `0.7.0`, the rule B1 (see Section 2.2) was added. With that extra rule, the problem `01` of *JGEx* was already proved by *OGP-GDDM*. This clearly indicates that the current set of rules must be revised.

4 Conclusions and future work

In its current state, `0.6.0`, our implementation of the GDD method is already able to prove some simple geometric conjectures, but still far from our overall goals:

Efficiency Using [4] as a reference, some procedures should be optimised, also some changes in the structure of the database. After those improvements have being done, it will be necessary to validate the goodness of the approach used, i.e., the use of a *SQLite* in-memory database.

Flexibility In its current state the set of rules is hard-coded in the provers overall code. The implementation is very modular, so it is easy to add, delete or modify rules, but a more generic approach is thought. Two lines of research: the possibility to have a "prover-generator", i.e. a meta-program that, given a set of rules (transformed in DDL/DML queries) can synthesise another program, the prover for that set of rules. Another possibility is to build a generic GATP that can accept a set of rules, in a given format, and proceed with that set of rules. In the immediate future we will work on the first of this two approaches. Again, it will be necessary to validate the goodness of the used approach.

```
$ ogpgddm 01.p
OGP GDDM 0.7.0
Copyright (C) 2022 Nuno Baeta, Pedro Quaresma
Distributed under GNU GPL 3.0 or later
Conjecture is PROVED, in: 1.1283s
Fix-point found, in: 62.9388s
Fixed point saved to file "01.fp".
```

Listing 5 *OGP-GDDM*, *JGEx* problem 01

Natural language and visual renderings The development of a crude proof script will be easy to implement and it will be included in a next version of the *OGP-GDDM* prover. The natural language rendering, the next "natural" step will be, we anticipate, also not difficult. The linking of the natural language and the construction, with visual highlights, will require a third-party DGS, as already said we are planning to use the *Web Geometry Laboratory* and its embedded *GeoGebra*.

Open source library The *OGP-GDDM* is being developed under the *OGPCP*, as an open source GATP that can be used as a stand-alone, program or as a library, included as in *C++* program. We hope that "with a little help from our friends", *OGP-GDDM* can became an useful tool to many practitioners of the automated deduction area.

A parallel line of research, but with an important impact in the final outcome of the prover, is related to the set or rules used by the prover. This is an ongoing project, being done with the collaboration of other researchers, whose goal is to find a good set of rules to be used by rule based GATP.

The usefulness (or not) of the approach used, i.e., the use of a *SQLite* in-memory database, must be evaluated under two, somehow, conflicting, criteria, the efficiency and the flexibility. This is something that should be evaluated alongside the use of the GATP by different type of users, e.g. researchers, developers of programs incorporating automated deduction tools, students and teachers. We will try to follow those users and give support whenever necessary.

Funding Open access funding provided by FCT—FCCN (b-on). The authors were 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.

Data Availability The GDDM prover is being developed as an open source library, available at GitHub: https://github.com/opengeometryprover/OpenGeometryProver/tree/master/provers/ogpgddm

Declarations

Conflict of Interests All authors declare that they have no conflicts of interest.

References

1. Baeta, N., Quaresma, P.: Open geometry prover community project. Electron. Proc. Theor. Comput. Sci. **352**, 129–138 (2021). https://doi.org/10.4204/EPTCS.352.14
2. Baeta, N., Quaresma, P., Kovács, Z.: Towards a geometry automated provers competition. In: Proceedings 8th International Workshop on Theorem proving components for Educational software. Electronic Proceedings in Theoretical Computer Science, vol. 313, pp. 93–100 (2020). https://doi.org/10.4204/EPTCS.313.6, (ThEdu'19), Natal, Brazil, 25th August 2019

3. Chou, S.C., Gao, X.S.: Automated reasoning in geometry. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, pp. 707–749. Elsevier Science Publishers B.V (2001). https://doi.org/10.1016/B978-044450813-3/50013-8
4. Chou, S.C., Gao, X.S., Zhang, J.Z.: A deductive database approach to automated geometry theorem proving and discovering. J. Autom. Reason. **25**(3), 219–246 (2000). https://doi.org/10.1023/A:1006171315513
5. Coelho, H., Pereira, L.M.: Automated reasoning in geometry theorem proving with Prolog. J. Autom. Reason. **2**(4), 329–390 (1986). https://doi.org/10.1007/BF00248249
6. Font, L.: Génération automatique de preuves pour un logiciel tuteur en géométrie. phdthesis Polytechnique Montréal. https://publications.polymtl.ca/9090/ (2021)
7. Font, L., Richard, P.R., Gagnon, M.: Improving qed-tutrix by automating the generation of proofs. In: Quaresma, P., Neuper, W. (eds.) Proceedings 6th International Workshop on Theorem proving components for Educational software, Gothenburg, Sweden, 6 Aug 2017. Electronic Proceedings in Theoretical Computer Science, vol. 267, pp. 38–58. Open Publishing Association (2018). https://doi.org/10.4204/EPTCS.267.3
8. Gelernter, H., Hansen, J.R., Loveland, D.W.: Empirical explorations of the geometry theorem machine. In: Papers presented at the May 3-5, 1960, Western Joint IRE-AIEE-ACM computer conference. IRE-AIEE-ACM '60 (Western), pp. 143–149. ACM, New York (1960). https://doi.org/10.1145/1460361.1460381
9. Hanna, G., Reid, D., de Villiers, M. (eds.): Proof Technology in Mathematics Research and Teaching. Springer. https://doi.org/10.1007/978-3-030-28483-1 (2019)
10. Hohenwarter, M.: GeoGebra - a software system for dynamic geometry and algebra in the plane. Master's thesis, University of Salzburg, Austria (2002)
11. Nevins, A.: Plane geometry theorem proving using forward chaining. Artif. Intell. **6**(1), 1–23 (1975). http://hdl.handle.net/1721.1/6218
12. Quaresma, P.: Evolution of Automated Deduction and Dynamic Constructions in Geometry, Mathematics Education in the Digital Era, chap. 1, vol. 17, pp. 3–22. Springer, New York (2022). https://doi.org/10.1007/978-3-030-86909-0
13. Quaresma, P., Santos, V.: Proof Technology in Mathematics Research and Teaching, chap. Computer-generated geometry proofs in a learning context, pp. 237–253. Springer, New York (2019). https://doi.org/10.1007/978-3-030-28483-1_11
14. Quaresma, P., Santos, V.: Four geometry problems to introduce automated deduction in secondary schools. In: Proceedings 10th International Workshop on Theorem Proving Components for Educational Software. Electronic Proceedings in Theoretical Computer Science, vol. 354, pp. 27–42. Open Publishing Association (2022). https://doi.org/10.4204/eptcs.354.3
15. Quaresma, P., Santos, V., Marić, M.: WGL, a web laboratory for geometry. Educ. Inf. Technol. **23**(1), 237–252 (2018). https://doi.org/10.1007/s10639-017-9597-y
16. Santos, V., Quaresma, P., Marić, M., Campos, H.: Web geometry laboratory: case studies in Portugal and Serbia. Interact. Learn. Environ. **26**(1), 3–21 (2018). https://doi.org/10.1080/10494820.2016.1258715
17. Stojanović, S., Pavlović, V., Janičić, P.: A coherent logic based geometry theorem prover capable of producing formal and readable proofs. In: Schreck, P., Narboux, J., Richter-Gebert, J. (eds.) Automated Deduction in Geometry, Lecture Notes in Computer Science, vol. 6877, pp. 201–220. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-25070-5_12
18. Sutcliffe, G.: The TPTP problem library and associated infrastructure. J. Autom. Reason. **59**(4), 483–502 (2017). https://doi.org/10.1007/s10817-017-9407-7
19. Teles, J., Santos, V., Quaresma, P.: A rule based theorem prover: an introduction to proofs in secondary schools. Electronic Proceedings in Theoretical Computer Science, accepted for publication (2023)
20. Ye, Z., Chou, S.C., Gao, X.S.: An introduction to Java geometry expert. In: Sturm, T., Zengler, C. (eds.) Automated Deduction in Geometry, Lecture Notes in Computer Science, vol. 6301, pp. 189–195. Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-21046-4_10

Chapter 5

Conclusions and Future Work

As said in the AMAI article (see Chapter 4), the *Geometry Deductive Database Prover* in its current version, `0.6.1`, is already able to prove some simple geometric conjectures. However, it is still far from the overall goals of efficiency, flexibility, natural language and visual renderings.

Apart the short-term goal of improving its efficiency, using [15] as source of ideas, the set or rules used by the prover must be reviewed (the current set of rules follows [15] closely and can be seen in the appendix C). This is an ongoing project, being done in collaboration with other researchers, whose goal is to find a *good set* of rules to be used by a rule-based GATP.

A long-term goal is the development of a generic prover, that is, a prover that, alongside ATP like *Vampire*,[1] can accept axiomatic theories coded as a set of inference rules and a given conjecture, and develop the proof based on that specific set of rules. It is also planned to work on the development of a natural language rendering of the proof, linking the proof steps with the geometric construction, with visual effects to highlight such connections (in line with the visually dynamic presentation of proofs in plane geometry, as presented in [63]).

The *Geometry Deductive Database Prover* is being developed as a stand-alone program (to test the implementation), not losing from sight its use as a "book" in the library of GATP of the *Open Geometry Prover Community Project*. The success of this project will be measured by its usefulness to the Automated Deduction in Geometry (ADG) community. The hopes are high.

[1] https://vprover.github.io/

Bibliography

[1] Nuno Baeta. O método do ângulo-completo no sistema OpenGeoProver. master thesis, Faculdade de Ciências e Tecnologia da Universidade de Coimbra, Julho 2013. URL: http://hdl.handle.net/10316/47639.

[2] Nuno Baeta and Pedro Quaresma. The full-angle method on the OpenGeoProver. In Christoph Lange, David Aspinall, Jacques Carette, James Davenport, Andrea Kohlhase, Michael Kohlhase, Paul Libbrecht, Pedro Quaresma, Florian Rabe, Petr Sojka, Iain Whiteside, and Wolfgang Windsteiger, editors, *MathUI, OpenMath, PLMMS and ThEdu Workshops and Work in Progress at the Conference on Intelligent Computer Mathematics*, number 1010 in CEUR Workshop Proceedings, Aachen, 2013. URL: http://ceur-ws.org/Vol-1010/paper-08.pdf.

[3] Nuno Baeta and Pedro Quaresma. Towards ranking geometric automated theorem provers. In Pedro Quaresma and Walther Neuper, editors, Proceedings 7th International Workshop on *Theorem proving components for Educational software,* Oxford, United Kingdom, 18 july 2018, volume 290 of *Electronic Proceedings in Theoretical Computer Science*, pages 30–37. Open Publishing Association, 2019. doi:10.4204/EPTCS.290.3.

[4] Nuno Baeta and Pedro Quaresma. Open geometry prover community project. *Electronic Proceedings in Theoretical Computer Science*, 352:129–138, dec 2021. doi:10.4204/EPTCS.352.14.

[5] Nuno Baeta and Pedro Quaresma. Towards a geometry deductive database prover. *Annals of Mathematics and Artificial Intelligence*, 2023. available online. doi:10.1007/s10472-023-09839-0.

[6] Nuno Baeta, Pedro Quaresma, and Zoltán Kovács. Towards a geometry automated provers competition. In *Proceedings 8th International Workshop on Theorem proving components for Educational software*, volume 313 of *Electronic Proceedings in Theoretical Computer Science*, pages 93–100, February 2020. (ThEdu'19), Natal, Brazil, 25th August 2019,. doi:10.4204/EPTCS.313.6.

[7] Francisco Botana, Markus Hohenwarter, Predrag Janičić, Zoltán Kovács, Ivan Petrović, Tomás Recio, and Simon Weitzhofer. Automated Theorem Proving in GeoGebra: Current

Achievements. *Journal of Automated Reasoning*, 55(1):39–59, mar 2015. doi:10.1007/s10817-015-9326-4.

[8] S.C. Chou. *Proving and discovering geometry theorems using Wu's method*. PhD thesis, The University of Texas, Austin, 1985.

[9] Shang-Ching Chou and Xiao-Shan Gao. Automated reasoning in geometry. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 707–749. Elsevier Science Publishers B.V., 2001. doi:10.1016/B978-044450813-3/50013-8.

[10] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated production of traditional proofs for constructive geometry theorems. In Moshe Vardi, editor, *Proceedings of the Eighth Annual IEEE Symposium on Logic in Computer Science LICS*, pages 48–56. IEEE Computer Society Press, June 1993. doi:10.1109/LICS.1993.287601.

[11] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. A collection of 110 geometry theorems and their machine produced proofs using full-angles. Technical Report TR-94-4, Department of Computer Science, The Wichita State University, November 1994.

[12] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. *Machine Proofs in Geometry*. World Scientific, apr 1994. doi:10.1142/2196.

[13] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants, I. multiple and shortest proof generation. *Journal of Automated Reasoning*, 17(3):325–347, dec 1996. doi:10.1007/BF00283133.

[14] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. Automated generation of readable proofs with geometric invariants, II. theorem proving with full-angles. *Journal of Automated Reasoning*, 17(3):349–370, dec 1996. doi:10.1007/BF00283134.

[15] Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning*, 25(3):219–246, 2000. doi:10.1023/A:1006171315513.

[16] H. Coelho and L. M. Pereira. Automated reasoning in geometry theorem proving with Prolog. *Journal of Automated Reasoning*, 2(4):329–390, dec 1986. doi:10.1007/BF00248249.

[17] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition: a synopsis. *SIGSAM Bull.*, 10(1):10–12, February 1976. doi:10.1145/1093390.1093393.

[18] E. W. Elcock. Representation of knowledge in geometry machine. *Machine Intelligence*, 8:11–29, 1977.

[19] Herve Gallaire, Jack Minker, and Jean-Marie Nicolas. Logic and databases: A deductive approach. *ACM Computing Surveys*, 16(2):153–185, June 1984. doi:10.1145/356924.356929.

[20] X. S. Gao. Building dynamic mathematical models with Geometry Expert, III. a geometry deductive database. In *Proc. Of ATCM'99*, pages 153–162. ATCM Inc., USA, 1999.

[21] H. Gelernter. *Computers & Thought*, chapter Realization of a geometry-theorem proving machine, pages 134–152. MIT Press, Cambridge, MA, USA, 2nd edition, 1995.

[22] J.G. Greeno, M. E. Magone, and S. Chaiklin. Theory of constructions and set in problem solving. *Memory and Cognition*, 7(6):445–461, nov 1979. doi:10.3758/BF03198261.

[23] Gila Hanna. Proof, explanation and exploration: An overview. *Educational Studies in Mathematics*, 44(1-2):5–23, 2000. doi:10.1023/A:1012737223465.

[24] Gila Hanna and Michael de Villiers, editors. *Proof and Proving in Mathematics Education*. Springer Netherlands, 2012. doi:10.1007/978-94-007-2129-6.

[25] Gila Hanna, David Reid, and Michael de Villiers, editors. *Proof Technology in Mathematics Research and Teaching*. Springer, 2019. doi:10.1007/978-3-030-28483-1.

[26] Li Hongbo. Clifford algebra approaches to mechanical geometry theorem proving. In X.-S. Gao and D. Wang, editors, *Mathematics Mechanization and Applications*, pages 205–299. Elsevier, San Diego, CA, 2000. doi:10.1016/B978-012734760-8/50009-0.

[27] Predrag Janičić. GCLC — A tool for constructive euclidean geometry and more than that. In Andrés Iglesias and Nobuki Takayama, editors, *Mathematical Software - ICMS 2006*, volume 4151 of *Lecture Notes in Computer Science*, pages 58–73. Springer, 2006. doi:10.1007/11832225_6.

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

[29] Predrag Janičić and Pedro Quaresma. Automatic verification of regular constructions in dynamic geometry systems. In Francisco Botana and Tomás Recio, editors, *Automated Deduction in Geometry*, volume 4869 of *Lecture Notes in Computer Science*, pages 39–51. Springer, 2007. doi:10.1007/978-3-540-77356-6_3.

[30] Deepak Kapur. Geometry Theorem Proving using Hilbert's Nullstellensatz. In *SYMSAC '86: Proceedings of the fifth ACM symposium on Symbolic and algebraic computation*, pages 202–208, New York, NY, USA, 1986. ACM Press. doi:10.1145/32439.32479.

[31] Deepak Kapur. Using Gröbner bases to reason about geometry problems. *Journal of Symbolic Computation*, 2(4):399–408, 1986. doi:10.1016/S0747-7171(86)80007-4.

[32] Zoltán Kovács, Tomás Recio, and M. Pilar Vélez. Automated reasoning tools in GeoGebra discovery. *ACM Communications in Computer Algebra*, 55(2):39–43, jun 2021. doi:10.1145/3493492.3493495.

[33] Zoltán Kovács, Tomás Recio, and M. Pilar Vélez. *Automated Reasoning Tools with GeoGebra: What Are They? What Are They Good For?*, chapter Automated Reasoning Tools with GeoGebra: What are they? What are they good for?, pages 23–44. Springer Nature, 2022. doi:10.1007/978-3-030-86909-0_2.

[34] N. Leduc. *QED-Tutrix : système tutoriel intelligent pour l'accompagnement d'élèves en situation de résolution de problèmes de démonstration en géométrie plane*. PhD thesis, École polytechnique de Montréal., 2016.

[35] Émile Lemoine. *Géométrographie ou Art des constructions géométriques*, volume 18 of *Phys-Mathématique*. Scentia, 1902. URL: http://catalogue.bnf.fr/ark:/12148/cb36049032t.

[36] Fou-Lai Lin, Feng-Jui Hsieh, Gila Hanna, and Michael de Villiers, editors. *Proceedings of the ICMI Study 19 conference: Proof and Proving in Mathematics Education*, volume 1. The Department of Mathematics, National Taiwan Normal University, 2009.

[37] Fou-Lai Lin, Feng-Jui Hsieh, Gila Hanna, and Michael de Villiers, editors. *Proceedings of the ICMI Study 19 conference: Proof and Proving in Mathematics Education*, volume 2. The Department of Mathematics, National Taiwan Normal University, 2009.

[38] J. S. Mackay. The geometrography of Euclid's problems. *Proceedings of the Edinburgh Mathematical Society*, 12:2–16, 1893. doi:10.1017/S0013091500001565.

[39] A.J. Nevins. Plane geometry theorem proving using forward chaining. *Artificial Intelligence*, 6(1):1–23, 1975. doi:10.1016/0004-3702(75)90013-2.

[40] Pedro Quaresma. Thousands of Geometric problems for geometric Theorem Provers (TGTP). In Pascal Schreck, Julien Narboux, and Jürgen Richter-Gebert, editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*, pages 169–181. Springer, 2011. doi:10.1007/978-3-642-25070-5_10.

[41] Pedro Quaresma. *Proofs & Dynamics in Geometry*, chapter Topic Group 3. Dynamic Geometry for Mathematics Education, pages 25–27. Editorial de la Universidad de Cantabria, 2020.

[42] Pedro Quaresma. *Evolution of Automated Deduction and Dynamic Constructions in Geometry*, volume 17 of *Mathematics Education in the Digital Era*, chapter 1, pages 3–22. Springer, 2022. doi:10.1007/978-3-030-86909-0.

[43] Pedro Quaresma and Nuno Baeta. Current status of the I2GATP common format. In Francisco Botana and Pedro Quaresma, editors, *Automated Deduction in Geometry*, volume 9201 of *Lecture Notes in Artificial Intelligence*, pages 119–128. Springer, 2015. doi:10.1007/978-3-319-21362-0_8.

[44] Pedro Quaresma and Vanda Santos. Four geometry problems to introduce automated deduction in secondary schools. In *Proceedings 10th International Workshop on Theorem Proving Components for Educational Software*, volume 354 of *Electronic Proceedings in Theoretical Computer Science*, pages 27–42. Open Publishing Association, feb 2022. doi:10.4204/eptcs.354.3.

[45] Pedro Quaresma, Vanda Santos, and Nuno Baeta. Exchange of geometric information between applications. *Electronic Proceedings in Theoretical Computer Science*, 267:108–119, mar 2018. doi:10.4204/eptcs.267.7.

[46] Pedro Quaresma, Vanda Santos, and Seifeddine Bouallegue. The Web Geometry Laboratory project. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *CICM 2013*, volume 7961 of *Lecture Notes in Computer Science*, pages 364–368. Springer, 2013. doi:10.1007/978-3-642-39320-4_30.

[47] Pedro Quaresma, Vanda Santos, Pierluigi Graziani, and Nuno Baeta. Taxonomy of geometric problems. *Journal of Symbolic Computation*, 97:31–55, mar 2020. doi:10.1016/j.jsc.2018.12.004.

[48] Pedro Quaresma, Vanda Santos, and Milena Marić. WGL, a web laboratory for geometry. *Education and Information Technologies*, 23(1):237–252, Jan 2018. doi:10.1007/s10639-017-9597-y.

[49] Philippe R. Richard, M. Pilar Vélez, and Steven Van Vaerenbergh, editors. *Mathematics Education in the Age of Artificial Intelligence*, volume 17 of *Mathematics Education in the Digital Era*. Springer International Publishing, 2022. doi:10.1007/978-3-030-86909-0.

[50] E. Santiago, Maxim Hendriks, Yves Kreis, Ulrich Kortenkamp, and Daniel Marquès. I2G Common File Format Final Version. Technical Report D3.10, The Intergeo Consortium, 2010. URL: http://i2geo.net/xwiki/bin/view/I2GFormat/.

[51] Vanda Santos, Nuno Baeta, and Pedro Quaresma. Geometrography in dynamic geometry. *International Journal for Technology in Mathematics Education*, 26(2):89–96, 2019. doi:10.1564/tme_v26.2.06.

[52] Vanda Santos and Pedro Quaresma. Exploring geometric conjectures with the help of a learning environment - a case study with pre-service teachers. *The Electronic Journal of Mathematics and Technology*, 2(1), 2021.

[53] Sana Stojanović, Vesna Pavlović, and Predrag Janičić. A coherent logic based geometry theorem prover capable of producing formal and readable proofs. In Pascal Schreck, Julien Narboux, and Jürgen Richter-Gebert, editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*, pages 201–220. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-25070-5_12.

[54] Alfred Tarski. A decision method for elementary algebra and geometry. Technical report, RAND Corporation, 1951.

[55] Joana Teles, Vanda Santos, and Pedro Quaresma. A rule based theorem prover: an introduction to proofs in secondary schools. *Electronic Proceedings in Theoretical Computer Science*, 375:24–37, mar 2023. doi:10.4204/EPTCS.375.3.

[56] Dongming Wang. Reasoning about geometric problems using an elimination method. In J. Pfalzgraf and D. Wang, editors, *Automated Pratical Reasoning*, pages 147–185. Springer, New York, 1995. doi:10.1007/978-3-7091-6604-8_8.

[57] Wu Wen-Tsun. Basic principles of mechanical theorem proving in elementary geometries. *Journal of Automated Reasoning*, 2:221–252, 1986. doi:10.1007/BF02328447.

[58] W.-K. Wong, S.-K. Yin, H.-H. Yang, and Y.-H. Cheng. Using computer-assisted multiple representations in learning geometry proofs. *Educational Technology and Society*, 14(3):43–54, 2011. URL: http://www.jstor.org/stable/jeductechsoci.14.3.43.

[59] Wen-Tsun Wu. *Automated Theorem Proving: After 25 Years*, volume 29 of *Contemporary Mathematics*, chapter On the decision problem and the mechanization of theorem proving in elementary geometry, pages 213–234. American Mathematical Society, 1984.

[60] Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. Visually dynamic presentation of proofs in plane geometry, part 1. *Journal of Automated Reasoning*, 45:213–241, October 2010. doi:10.1007/s10817-009-9162-5.

[61] Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. Visually dynamic presentation of proofs in plane geometry, part 2. *Journal of Automated Reasoning*, 45:243–266, October 2010. doi:10.1007/s10817-009-9163-4.

[62] Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. An introduction to Java Geometry Expert. In Thomas Sturm and Christoph Zengler, editors, *Automated Deduction in Geometry*, volume 6301 of *Lecture Notes in Computer Science*, pages 189–195. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-21046-4_10.

[63] Zheng Ye, Shang-Ching Chou, and Xiao-Shan Gao. An introduction to Java Geometry Expert. In Thomas Sturm and Christoph Zengler, editors, *Automated Deduction in Geometry*, volume 6301 of *Lecture Notes in Computer Science*, pages 189–195. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-21046-4_10.

Appendix A

GCLC Prover, Wu's Method, Conjecture "Ceva"

Construction and prover internal state

Construction commands:

- Point A
- Point B
- Point C
- Point P
- Line a: $B\ C$
- Line b: $A\ C$
- Line c: $A\ B$
- Line pa: $P\ A$
- Line pb: $P\ B$
- Line pc: $P\ C$
- Intersection of lines, D: $a\ pa$
- Intersection of lines, E: $b\ pb$
- Intersection of lines, F: $c\ pc$

Coordinates assigned to the points:

- $A = (0, 0)$
- $B = (u_1, 0)$
- $C = (u_2, u_3)$
- $P = (u_4, u_5)$
- $D = (x_2, x_1)$
- $E = (x_4, x_3)$
- $F = (x_6, 0)$

Conjecture(s):

1. Given conjecture

   - **GCLC code:**

```
equal
      { mult
  { mult
    { sratio A F F B }
    { sratio B D D C }
  }
  { sratio C E E A }
       }
{  1 }
```

   - **Expression:**
     $\left(\left(\frac{\overrightarrow{AF}}{\overrightarrow{FB}} \cdot \frac{\overrightarrow{BD}}{\overrightarrow{DC}}\right) \cdot \frac{\overrightarrow{CE}}{\overrightarrow{EA}}\right) = 1$
   - **Expression after rationalization:**
     $(((x(A) - x(F)) \cdot (y(B) - y(D))) \cdot (y(C) - y(E))) = (((1 \cdot (x(F) - x(B))) \cdot (y(D) - y(C))) \cdot (y(E) - y(A)))$

Resolving constructed lines

- $a \ni B, C, D$
- $b \ni A, C, E$
- $c \ni A, B, F$; line is horizontal (i.e., $y(A) = y(B)$)
- $pa \ni P, A, D$
- $pb \ni P, B, E$
- $pc \ni P, C, F$

Creating polynomials from hypotheses

- Point A

  no condition
- Point B

  no condition
- Point C

  no condition
- Point P

  no condition
- Line a: B C
  - point B is on the line (B, C)

    no condition
  - point C is on the line (B, C)

    no condition
- Line b: A C
  - point A is on the line (A, C)

    no condition
  - point C is on the line (A, C)

    no condition
- Line c: A B

  - point A is on the line (A, B)
    no condition
  - point B is on the line (A, B)
    no condition

- Line *pa*: P A
  - point P is on the line (P, A)
    no condition
  - point A is on the line (P, A)
    no condition

- Line *pb*: P B
  - point P is on the line (P, B)
    no condition
  - point B is on the line (P, B)
    no condition

- Line *pc*: P C
  - point P is on the line (P, C)
    no condition
  - point C is on the line (P, C)
    no condition

- Intersection of lines, D: a pa
  - point D is on the line (B, C)

$$p_1 = -u_3x_2 + (u_2 - u_1)x_1 + u_3u_1$$

  - point D is on the line (P, A)

$$p_2 = u_5x_2 - u_4x_1$$

- Intersection of lines, E: b pb
  - point E is on the line (A, C)

$$p_3 = -u_3x_4 + u_2x_3$$

  - point E is on the line (P, B)

$$p_4 = u_5x_4 + (-u_4 + u_1)x_3 - u_5u_1$$

- Intersection of lines, F: $c\ pc$
  - point F is on the line (A, B) — true by the construction
    no condition
  - point F is on the line (P, C)

$$p_5 = (u_5 - u_3)x_6 + (-u_5u_2 + u_4u_3)$$

Creating polynomial from the conjecture

- Processing given conjecture(s).

Conjecture 1:

$$p_6 = -2x_6x_3x_1 + u_3x_6x_3 + u_3x_6x_1 + u_1x_3x_1 - u_3u_1x_3$$

Invoking the theorem prover

The used proving method is Wu's method.

The input system is:

$$\begin{aligned}
p_0 &= -u_3x_2 + (u_2 - u_1)x_1 + u_3u_1 \\
p_1 &= u_5x_2 - u_4x_1 \\
p_2 &= -u_3x_4 + u_2x_3 \\
p_3 &= u_5x_4 + (-u_4 + u_1)x_3 - u_5u_1 \\
p_4 &= (u_5 - u_3)x_6 + (-u_5u_2 + u_4u_3)
\end{aligned}$$

Triangulation, step 1

Choosing variable: Trying the variable with index 6.

Variable x_6 selected: The number of polynomials with this variable is 1.

Single polynomial with chosen variable: No reduction needed.

The triangular system has not been changed.

Triangulation, step 2

Choosing variable: Trying the variable with index 5.

Choosing variable: Trying the variable with index 4.

Variable x_4 selected: The number of polynomials with this variable is 2.

Minimal degrees: 3 polynomials with degree 1 and 2 polynomials with degree 1.

Polynomial with linear degree: Removing variable x_4 from all other polynomials by reducing them with polynomial p_3.

Finished a triangulation step, the current system is:

$$
\begin{aligned}
p_0 &= -u_3x_2 + (u_2 - u_1)x_1 + u_3u_1 \\
p_1 &= u_5x_2 - u_4x_1 \\
p_2 &= (u_5u_2 - u_4u_3 + u_3u_1)x_3 - u_5u_3u_1 \\
p_3 &= u_5x_4 + (-u_4 + u_1)x_3 - u_5u_1 \\
p_4 &= (u_5 - u_3)x_6 + (-u_5u_2 + u_4u_3)
\end{aligned}
$$

Triangulation, step 3

Choosing variable: Trying the variable with index 3.

Variable x_3 selected: The number of polynomials with this variable is 1.

Single polynomial with chosen variable: No reduction needed.

The triangular system has not been changed.

Triangulation, step 4

Choosing variable: Trying the variable with index 2.

Variable x_2 selected: The number of polynomials with this variable is 2.

Minimal degrees: 1 polynomials with degree 1 and 0 polynomials with degree 1.

Polynomial with linear degree: Removing variable x_2 from all other polynomials by reducing them with polynomial p_1.

Finished a triangulation step, the current system is:

$$
\begin{aligned}
p_0 &= (u_5u_2 - u_5u_1 - u_4u_3)x_1 + u_5u_3u_1 \\
p_1 &= u_5x_2 - u_4x_1 \\
p_2 &= (u_5u_2 - u_4u_3 + u_3u_1)x_3 - u_5u_3u_1 \\
p_3 &= u_5x_4 + (-u_4 + u_1)x_3 - u_5u_1 \\
p_4 &= (u_5 - u_3)x_6 + (-u_5u_2 + u_4u_3)
\end{aligned}
$$

Triangulation, step 5

Choosing variable: Trying the variable with index 1.

Variable x_1 selected: The number of polynomials with this variable is 1.

Single polynomial with chosen variable: No reduction needed.

The triangular system has not been changed.

The triangular system is:

$$
\begin{aligned}
p_0 &= (u_5u_2 - u_5u_1 - u_4u_3)x_1 + u_5u_3u_1 \\
p_1 &= u_5x_2 - u_4x_1 \\
p_2 &= (u_5u_2 - u_4u_3 + u_3u_1)x_3 - u_5u_3u_1 \\
p_3 &= u_5x_4 + (-u_4 + u_1)x_3 - u_5u_1 \\
p_4 &= (u_5 - u_3)x_6 + (-u_5u_2 + u_4u_3)
\end{aligned}
$$

Final remainder

Final remainder for conjecture 1

Calculating final remainder of the conclusion:

$$g = -2x_6x_3x_1 + u_3x_6x_3 + u_3x_6x_1 + u_1x_3x_1 - u_3u_1x_3$$

with respect to the triangular system.

1. Pseudo remainder with p_4 over variable x_6:

$$g = (-2u_5u_2 + u_5u_1 + 2u_4u_3 - u_3u_1)x_3x_1 +$$

$$(u_5u_3u_2 - u_5u_3u_1 - u_4u_3^2 + u_3^2u_1)x_3 + (u_5u_3u_2 - u_4u_3^2)x_1$$

2. Pseudo remainder with p_3 over variable x_4:

$$g = (-2u_5u_2 + u_5u_1 + 2u_4u_3 - u_3u_1)x_3x_1 +$$

$$(u_5u_3u_2 - u_5u_3u_1 - u_4u_3^2 + u_3^2u_1)x_3 + (u_5u_3u_2 - u_4u_3^2)x_1$$

3. Pseudo remainder with p_2 over variable x_3:

$$g = ($$

$$u_5^2u_3u_2^2 - 2u_5^2u_3u_2u_1 + u_5^2u_3u_1^2 - 2u_5u_4u_3^2u_2 +$$
$$2u_5u_4u_3^2u_1 + u_5u_3^2u_2u_1 - u_5u_3^2u_1^2 + u_4^2u_3^3 - u_4u_3^3u_1)x_1 +$$
$$(u_5^2u_3^2u_2u_1 - u_5^2u_3^2u_1^2 - u_5u_4u_3^3u_1 + u_5u_3^3u_1^2)$$

4. Pseudo remainder with p_1 over variable x_2:

$$g = ($$

$$u_5^2u_3u_2^2 - 2u_5^2u_3u_2u_1 + u_5^2u_3u_1^2 - 2u_5u_4u_3^2u_2 +$$
$$2u_5u_4u_3^2u_1 + u_5u_3^2u_2u_1 - u_5u_3^2u_1^2 + u_4^2u_3^3 - u_4u_3^3u_1)x_1 +$$
$$(u_5^2u_3^2u_2u_1 - u_5^2u_3^2u_1^2 - u_5u_4u_3^3u_1 + u_5u_3^3u_1^2)$$

5. Pseudo remainder with p_0 over variable x_1:

$$g = 0$$

Prover report

Status: The conjecture has been proved.

Space Complexity: The biggest polynomial obtained during proof process contained 13 terms.

Time Complexity: Time spent by the prover: 0.002 seconds.

NDG conditions are:

- $P_{FBF} \neq 0$ i.e., points F and B are not identical (conjecture based assumption).
- $P_{DCD} \neq 0$ i.e., points D and C are not identical (conjecture based assumption).
- $P_{EAE} \neq 0$ i.e., points E and A are not identical (conjecture based assumption).

Appendix B

GCLC Prover, Area Method, Conjecture "Ceva"

$$\left(\left(\frac{\overrightarrow{AF}}{\overrightarrow{FB}} \cdot \frac{\overrightarrow{BD}}{\overrightarrow{DC}}\right) \cdot \frac{\overrightarrow{CE}}{\overrightarrow{EA}}\right) = 1 \quad \text{by the statement} \quad (0)$$

$$\left(\left(\left(-1 \cdot \frac{\overrightarrow{AF}}{\overrightarrow{BF}}\right) \cdot \frac{\overrightarrow{BD}}{\overrightarrow{DC}}\right) \cdot \frac{\overrightarrow{CE}}{\overrightarrow{EA}}\right) = 1 \quad \text{by geometric simplifications} \quad (1)$$

$$\left(-1 \cdot \left(\frac{\overrightarrow{AF}}{\overrightarrow{BF}} \cdot \left(\frac{\overrightarrow{BD}}{\overrightarrow{DC}} \cdot \frac{\overrightarrow{CE}}{\overrightarrow{EA}}\right)\right)\right) = 1 \quad \text{by algebraic simplifications} \quad (2)$$

$$\left(-1 \cdot \left(\frac{S_{APC}}{S_{BPC}} \cdot \left(\frac{\overrightarrow{BD}}{\overrightarrow{DC}} \cdot \frac{\overrightarrow{CE}}{\overrightarrow{EA}}\right)\right)\right) = 1 \quad \text{by Lemma 8 (point } F \text{ eliminated)} \quad (3)$$

$$\left(-1 \cdot \left(\frac{S_{APC}}{S_{BPC}} \cdot \left(\frac{\overrightarrow{BD}}{\overrightarrow{DC}} \cdot \left(-1 \cdot \frac{\overrightarrow{CE}}{\overrightarrow{AE}}\right)\right)\right)\right) = 1 \quad \text{by geometric simplifications} \quad (4)$$

$$\frac{\left(S_{APC} \cdot \left(\frac{\overrightarrow{BD}}{\overrightarrow{DC}} \cdot \frac{\overrightarrow{CE}}{\overrightarrow{AE}}\right)\right)}{S_{BPC}} = 1 \quad \text{by algebraic simplifications} \quad (5)$$

$$\frac{\left(S_{APC} \cdot \left(\frac{\overrightarrow{BD}}{\overrightarrow{DC}} \cdot \frac{S_{CPB}}{S_{APB}}\right)\right)}{S_{BPC}} = 1 \quad \text{by Lemma 8 (point } E \text{ eliminated)} \quad (6)$$

$$\frac{\left(S_{APC} \cdot \left(\left(-1 \cdot \frac{\overrightarrow{BD}}{\overrightarrow{CD}}\right) \cdot \frac{S_{CPB}}{S_{APB}}\right)\right)}{(-1 \cdot S_{CPB})} = 1 \quad \text{by geometric simplifications} \quad (7)$$

$$\frac{\left(S_{APC} \cdot \frac{\overrightarrow{BD}}{\overrightarrow{CD}}\right)}{S_{APB}} = 1 \quad \text{by algebraic simplifications} \quad (8)$$

$\frac{\left(S_{APC} \cdot \frac{S_{BPA}}{S_{CPA}}\right)}{S_{APB}}$	=	1	by Lemma 8 (point D eliminated) (9)
$\frac{\left(S_{APC} \cdot \frac{S_{BPA}}{(-1 \cdot S_{APC})}\right)}{(-1 \cdot S_{BPA})}$	=	1	by geometric simplifications (10)
1	=	1	by algebraic simplifications (11)

Q.E.D.

NDG conditions are:
$S_{BPA} \neq S_{CPA}$ i.e., lines BC and PA are not parallel (construction based assumption)
$S_{APB} \neq S_{CPB}$ i.e., lines AC and PB are not parallel (construction based assumption)
$S_{APC} \neq S_{BPC}$ i.e., lines AB and PC are not parallel (construction based assumption)
$P_{FBF} \neq 0$ i.e., points F and B are not identical (conjecture based assumption)
$P_{DCD} \neq 0$ i.e., points D and C are not identical (conjecture based assumption)
$P_{EAE} \neq 0$ i.e., points E and A are not identical (conjecture based assumption)

Number of elimination proof steps: 3
Number of geometric proof steps: 6
Number of algebraic proof steps: 23
Total number of proof steps: 32
Time spent by the prover: 0.000 seconds

Appendix C

Geometric Inference Rules

Some of the geometric inference rules proposed in [15] present syntatic inconsistencies [5] and modifications were made. In the following list, for each proposed rule, the modified rule, as well as the non-degenerate (NDG) conditions, are presented whenever needed. Moreover, the *C++* methods used to implement each rule are also listed.

D1. **coll**(A,C,B) :– **coll**(A,B,C)

Method(s): `ruleD01`

D2. **coll**(B,A,C) :– **coll**(A,B,C)

Method(s): `ruleD02`

D3. **coll**(C,D,A) :– **coll**(A,B,C), **coll**(A,B,D)

Method(s): `ruleD03`

D4. **para**(A,B,D,C) :– **para**(A,B,C,D)

Method(s): `ruleD04`

D5. **para**(C,D,A,B) :– **para**(A,B,C,D)

Method(s): `ruleD05`

D6. **para**(A,B,E,F) :– **para**(A,B,C,D), **para**(C,D,E,F)

Method(s): `ruleD06`

D7. **perp**(A,B,D,C) :– **perp**(A,B,C,D)

Method(s): `ruleD07`

D8. **perp**(C,D,A,B) :– **perp**(A,B,C,D)

Method(s): `ruleD08`

D9. **para**(A,B,E,F) :– **perp**(A,B,C,D), **perp**(C,D,E,F)

Method(s): `ruleD09`

D10. **perp**(A,B,E,F) :– **para**(A,B,C,D), **perp**(C,D,E,F)

Method(s): `ruleD10para`, `ruleD10perp`

D11. **midp**(M,A,B) :– **midp**(M,B,A)

Method(s): `ruleD11`

D12. **circle**(O,A,B,C) :– **cong**(O,A,O,B), **cong**(O,A,O,C)

Method(s): `ruleD12`

D13. **cyclic**(A,B,C,D) :– **cong**(O,A,O,B), **cong**(O,A,O,C), **cong**(O,A,O,D)

Method(s): `ruleD13`

D14. **cyclic**(A,B,D,C) :– **cyclic**(A,B,C,D)

Method(s): `ruleD14`

D15. **cyclic**(A,C,B,D) :– **cyclic**(A,B,C,D)

Method(s): `ruleD15`

D16. **cyclic**(B,A,C,D) :– **cyclic**(A,B,C,D)

Method(s): `ruleD16`

D17. **cyclic**(B,C,D,E) :– **cyclic**(A,B,C,D), **cyclic**(A,B,C,E)

Method(s): `ruleD17`

D18. **eqangle**(B,A,C,D,P,Q,U,V) :– **eqangle**(A,B,C,D,P,Q,U,V)

Method(s): `ruleD18`

D19. **eqangle**(C,D,A,B,U,V,P,Q) :– **eqangle**(A,B,C,D,P,Q,U,V)

Method(s): `ruleD19`

D20. **eqangle**(P,Q,U,V,A,B,C,D) :– **eqangle**(A,B,C,D,P,Q,U,V)

Method(s): `ruleD20`

D21. **eqangle**(A,B,P,Q,C,D,U,V) :– **eqangle**(A,B,C,D,P,Q,U,V)

Method(s): `ruleD21`

D22. **eqangle**(A,B,C,D,E,F,G,H) :– **eqangle**(A,B,C,D,P,Q,U,V), **eqangle**(P,Q,U,V,E,F,G,H)

Method(s): `ruleD22`

D23. **cong**(A,B,D,C) :– **cong**(A,B,C,D)

Method(s): `ruleD23`

D24. **cong**(C,D,A,B) :– **cong**(A,B,C,D)

Method(s): `ruleD24`

D25. **cong**(A,B,E,F) :– **cong**(A,B,C,D), **cong**(C,D,E,F)

Method(s): `ruleD25`

D26. **eqratio**(B,A,C,D,P,Q,U,V) :– **eqratio**(A,B,C,D,P,Q,U,V)

Method(s): `ruleD26`

D27. **eqratio**(C,D,A,B,U,V,P,Q) :– **eqratio**(A,B,C,D,P,Q,U,V)

Method(s): `ruleD27`

D28. **eqratio**(P,Q,U,V,A,B,C,D) :– **eqratio**(A,B,C,D,P,Q,U,V)

Method(s): `ruleD28`

D29. **eqratio**(A,B,P,Q,C,D,U,V) :– **eqratio**(A,B,C,D,P,Q,U,V)

Method(s): `ruleD29`

D30. **eqratio**(A,B,C,D,E,F,G,H) :– **eqratio**(A,B,C,D,P,Q,U,V), **eqratio**(P,Q,U,V,E,F,G,H)

Method(s): `ruleD30`

D31. **simtri**(A,B,C,P,Q,R) :– **simtri**(A,C,B,P,R,Q)

Method(s): `ruleD31`

D32. **simtri**(A,B,C,P,Q,R) :– **simtri**(B,A,C,Q,P,R)

Method(s): `ruleD32`

D33. **simtri**(A,B,C,P,Q,R) :– **simtri**(P,Q,R,A,B,C)

Method(s): `ruleD33`

D34. **simtri**(A,B,C,P,Q,R) :– **simtri**(A,B,C,E,F,G), **simtri**(E,F,G,P,Q,R)

Method(s): `ruleD34`

D35. **contri**(A,B,C,P,Q,R) :– **contri**(A,C,B,P,R,Q)

Method(s): `ruleD35`

D36. **contri**(A,B,C,P,Q,R) :– **contri**(B,A,C,Q,P,R)

Method(s): `ruleD36`

D37. **contri**(A,B,C,P,Q,R) :– **contri**(P,Q,R,A,B,C)

Method(s): `ruleD37`

D38. **contri**(A,B,C,P,Q,R) :– **contri**(A,B,C,E,F,G), **contri**(E,F,G,P,Q,R)

Method(s): `ruleD38`

D39. **para**(A,B,C,D) :– **eqangle**(A,B,P,Q,C,D,P,Q)

Method(s): `ruleD39`

D40. **eqangle**(A,B,P,Q,C,D,P,Q) :– **para**(A,B,C,D)

Method(s): `ruleD40`

D41. **eqangle**(P,A,P,B,Q,A,Q,B) :– **cyclic**(A,B,P,Q)

Method(s): `ruleD41`

D42. **cyclic**(A,B,P,Q) :– **eqangle**(P,A,P,B,Q,A,Q,B), ¬**coll**(P,Q,A,B)

NDG: ¬**coll**(P,Q,A), ¬**coll**(P,Q,B)

Method(s): `ruleD42`

D43. **cong**(A,B,P,Q) :– **cyclic**(A,B,C,P,Q,R), **eqangle**(C,A,C,B,R,P,R,Q)

Rule:

cong(A,B,P,Q) :– **cyclic**(A,B,C,P), **cyclic**(A,B,C,Q),
cyclic(A,B,C,R), **eqangle**(C,A,C,B,R,P,R,Q)

Method(s): `ruleD43cyclic`, `ruleD43eqangle`

D44. **para**(E,F,B,C) :– **midp**(E,A,B), **midp**(F,A,C)

Method(s): `ruleD44`

D45. **midp**(F,A,C) :– **midp**(E,A,B), **para**(E,F,B,C), **coll**(F,A,C)

NDG: ¬**coll**(A,B,C)

Method(s): `ruleD45coll`, `ruleD45midp`, `ruleD45para`

D46. **eqangle**(O,A,A,B,A,B,O,B) :– **cong**(O,A,O,B)

Method(s): `ruleD46`

D47. **cong**(O,A,O,B) :– **eqangle**(O,A,A,B,A,B,O,B), ¬**coll**(O,A,B)

NDG: ¬**coll**(O,A,B)

Method(s): `ruleD47`

D48. **eqangle**(A,X,A,B,C,A,C,B) :– **circle**(O,A,B,C), **perp**(O,A,A,X)

Method(s): `ruleD48circle`, `ruleD48perp`

D49. **perp**(O,A,A,X) :– **circle**(O,A,B,C), **eqangle**(A,X,A,B,C,A,C,B)

Method(s): `ruleD49circle`, `ruleD49eqangle`

D50. **eqangle**(A,B,A,C,O,B,O,M) :– **circle**(O,A,B,C), **midp**(M,B,C)

Method(s): `ruleD50circle`, `ruleD50midp`

D51. **midp**(M,B,C) :– **circle**(O,A,B,C), **coll**(M,B,C),
eqangle(A,B,A,C,O,B,O,M)

Method(s): `ruleD51circle`, `ruleD51coll`, `ruleD51eqangle`

D52. **cong**(A,M,B,M) :– **perp**(A,B,B,C), **midp**(M,A,C)

Method(s): `ruleD52midp`, `ruleD52perp`

D53. **perp**(A,B,B,C) :– **circle**(O,A,B,C), **coll**(O,A,C)

Method(s): `ruleD53circle`, `ruleD53coll`

D54. **eqangle**(A,D,C,D,C,D,C,B) :– **cyclic**(A,B,C,D), **para**(A,B,C,D)

Method(s): `ruleD54cyclic`, `ruleD54para`

D55. **cong**(O,A,O,B) :– **midp**(M.A,B), **perp**(O,M,A,B)

Method(s): `ruleD55midp`, `ruleD55perp`

D56. **perp**(A,B,P,Q) :– **cong**(A,P,B,P), **cong**(A,Q,B,Q)

Method(s): `ruleD56`

D57. **perp**(P,A,A,Q) :– **cong**(A,P,B,P), **cong**(A,Q,B,Q), **cyclic**(A,B,P,Q)

Method(s): `ruleD57cong`, `ruleD57cyclic`

D58. **simtri**(A,B,C,P,Q,R) :– **eqangle**(A,B,B,C,P,Q,Q,R), **eqangle**(A,C,B,C,P,R,Q,R), ¬**coll**(A,B,C)

Method(s): `ruleD58a`, `ruleD58b`

D59. **eqratio**(A,B,A,C,P,Q,P,R) :– **simtri**(A,B,C,P,Q,R)

Method(s): `ruleD59`

D60. **eqangle**(A,B,B,C,P,Q,Q,R) :– **simtri**(A,B,C,P,Q,R)

Method(s): `ruleD60`

D61. **contri**(A,B,C,P,Q,R) :– **simtri**(A,B,C,P,Q,R), AB = PQ

Rule:

contri(A,B,C,P,Q,R) :– **simtri**(A,B,C,P,Q,R), **cong**(A,B,P,Q)

Method(s): `ruleD61cong`, `ruleD61simtri`

D62. **cong**(A,B,P,Q) :– **contri**(A,B,C,P,Q,R)

Method(s): `ruleD62`

D63. **para**(A,C,B,D) :– **midp**(M,A,B), **midp**(M,C,D)

Method(s): `ruleD63`

D64. **midp**(M,C,D) :– **midp**(M,A,B), **para**(A,C,B,D), **para**(A,D,B,C)

NDG: ¬**coll**(A,B,C), ¬**coll**(A,B,D)

Method(s): `ruleD64midp`, `ruleD64para`

D65. **eqratio**(O,A,A,C,O,B,B,D) :– **para**(A,B,C,D), **coll**(O,A,C), **coll**(O,B,D)

NDG: ¬**coll**(A,B,C)

Method(s): `ruleD65coll`, `ruleD65para`

D66. **coll**(A,B,C) :– **para**(A,B,A,C)

Method(s): `ruleD66`

D67. **midp**(A,B,C) :– **cong**(A,B,A,C), **coll**(A,B,C)

Method(s): `ruleD67coll`, `ruleD67cong`

D68. **cong**(A,B,A,C) :– **midp**(A,B,C)

Method(s): `ruleD68`

D69. **coll**(A,B,C) :– **midp**(A,B,C)

Method(s): `ruleD69`

D70. **eqratio**(M,A,A,B,N,C,C,D) :– **midp**(M,A,B), **midp**(N,C,D)

Method(s): `ruleD70`

D71. **perp**(A,B,C,D) :– **eqangle**(A,B,C,D,C,D,A,B), ¬**para**(A,B,C,D)

NDG: ¬**para**(A,B,C,D)

Method(s): `ruleD71`

D72. **para**(A,B,C,D) :– **eqangle**(A,B,C,D,C,D,A,B), ¬**perp**(A,B,C,D)

NDG: ¬**perp**(A,B,C,D)

Method(s): `ruleD72`

D73. **para**(A,B,C,D) :– **eqangle**(A,B,C,D,P,Q,U,V), **para**(P,Q,U,V)

Method(s): `ruleD73eqangle`, `ruleD73para`

D74. **perp**(A,B,C,D) :– **eqangle**(A,B,C,D,P,Q,U,V), **perp**(P,Q,U,V)

Method(s): `ruleD74eqangle`, `ruleD74perp`

D75. **cong**(A,B,C,D) :– **eqratio**(A,B,C,D,P,Q,U,V), **cong**(P,Q,U,V)

Method(s): `ruleD75cong`, `ruleD75eqratio`

Appendix D

geo0007 Fix-Point

An edited listing of the fix-point found by *OGPGDDM* for conjecture `geo0007` is presented bellow (see Listing D.1). In its current version, the fix-point is the list of all the geometric facts found by the prover, without any attempt to simplify it, e.g. the facts, `midp(D, C, A)` and `midp(D, A, C)` are both in the list, when one of them would be enough. For that reason, even for a simple construction like the conjecture `geo0007`, the fix-point is a 56 pages long listing, the motive to present an edited version of it. It is clear that some strategies to avoid the different meaningless combinations are necessary, even for efficiency sake, e.g. for this example the solution was found after less then 0.016 seconds, but the fix-point took more then 88 seconds (see Listing 4.2).

Listing D.1 Parallel Lines in a Triangle Theorem, OGPGDDM fix-point

```
midp: 4
    midp(D, C, A)
    midp(E, A, B)
    midp(D, A, C)
    midp(E, B, A)
cong: 16
    cong(D, C, D, A)
    cong(E, A, E, B)
    cong(D, A, D, C)
    cong(D, C, A, D)
    cong(E, B, E, A)
    cong(E, A, B, E)
    cong(D, A, C, D)
    cong(A, D, D, C)
    cong(E, B, A, E)
    cong(B, E, E, A)
    cong(C, D, D, A)
    cong(A, D, C, D)
    cong(A, E, E, B)
    cong(B, E, A, E)
    cong(C, D, A, D)
    cong(A, E, B, E)
coll: 12
    coll(D, C, A)
    coll(E, A, B)
```

```
    coll(D, A, C)
    coll(C, D, A)
    coll(E, B, A)
    coll(A, E, B)
    coll(A, D, C)
    coll(C, A, D)
    coll(B, E, A)
    coll(A, B, E)
    coll(A, C, D)
    coll(B, A, E)
para: 8
    para(E, D, B, C)
    para(D, E, C, B)
    para(E, D, C, B)
    para(B, C, E, D)
    para(D, E, B, C)
    para(C, B, D, E)
    para(C, B, E, D)
    para(B, C, D, E)
eqangle: 128
    eqangle(D, C, C, A, C, A, D, A)
    eqangle(E, A, A, B, A, B, E, B)
    eqangle(D, A, A, C, A, C, D, C)
    eqangle(C, D, C, A, C, A, D, A)
    eqangle(C, A, D, C, D, A, C, A)
    eqangle(C, A, D, A, D, C, C, A)
    ...
eqratio: 3008
    eqratio(D, C, C, A, E, A, A, B)
    eqratio(E, A, A, B, D, A, A, C)
    eqratio(D, A, A, C, E, B, B, A)
    eqratio(C, D, C, A, E, A, A, B)
    eqratio(C, A, D, C, A, B, E, A)
    eqratio(E, A, A, B, D, C, C, A)
    ...
```