Annals of Mathematics and Artificial Intelligence (2023) 91:851–863
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 / Published online: 24 May 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

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

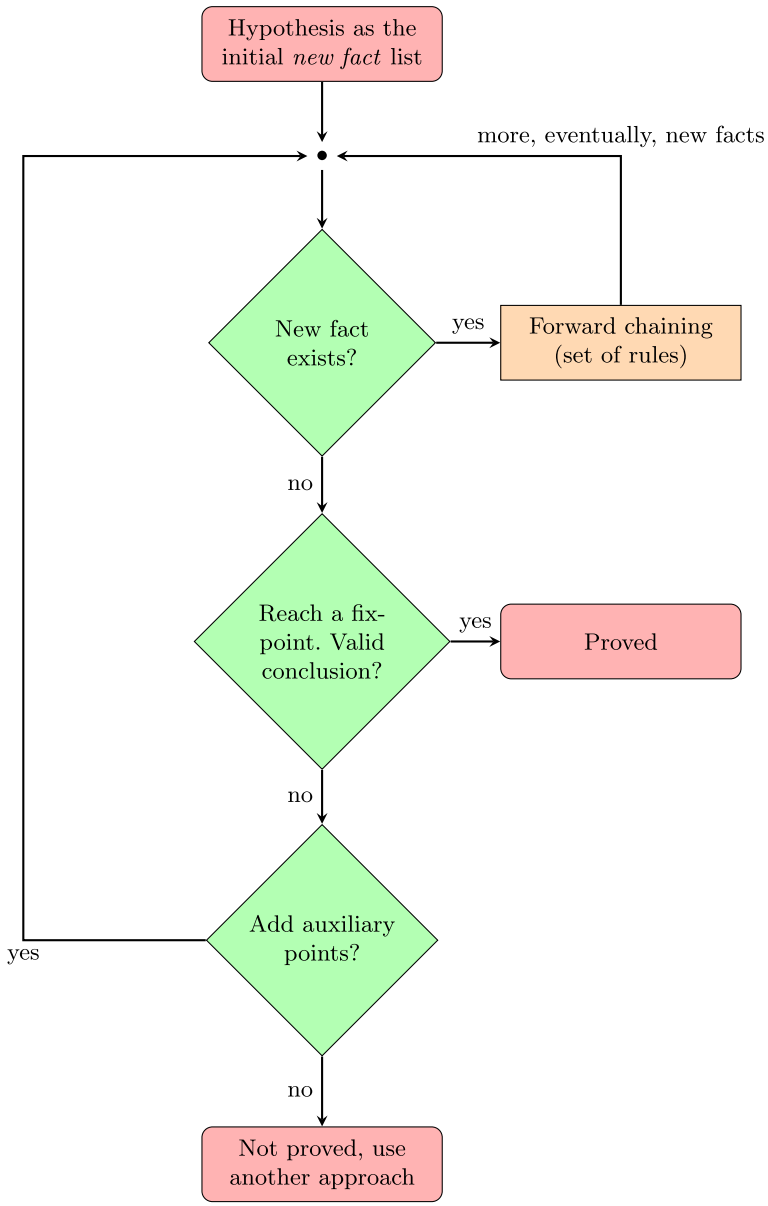


**Fig. 1** GDDM algorithm

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)`.

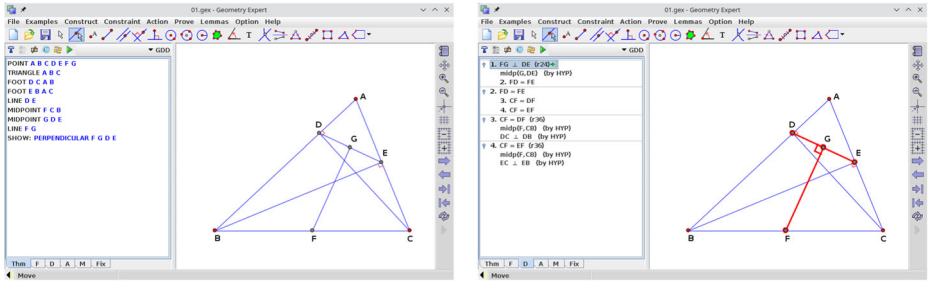

**Fig. 2** Problem 01, *JGEx*, GDD method

– 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

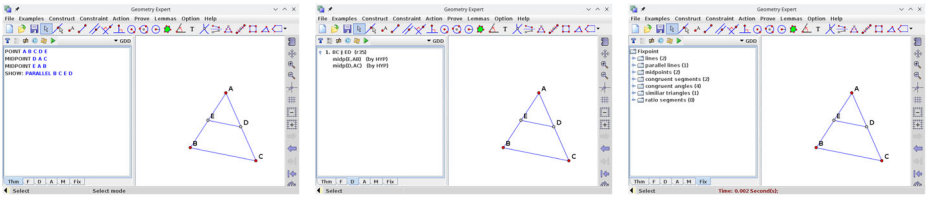

**Fig. 3** *JGEx — TGTP*, GEO0007 problem

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