

ARGO and Computer-supported Geometry Reasoning

presented by Vesna Marinković

Faculty of Mathematics, University of Belgrade

ARGO seminar, 8.7.2026.

Table of Contents

- 1 Key Goals within Geometric Reasoning
- 2 Intelligent Dynamic Geometry Tools
- 3 Geometric Constructions
- 4 Automated Theorem Proving
- 5 Formalizations

Central Research Directions in the Field of Geometry Reasoning

- Solving different type of geometric problems
- Proving geometry statements **automatically**
- **Understanding** automatically obtained proofs
- Obtain proofs **we can trust** with complete confidence
- Improve process of learning and teaching geometry

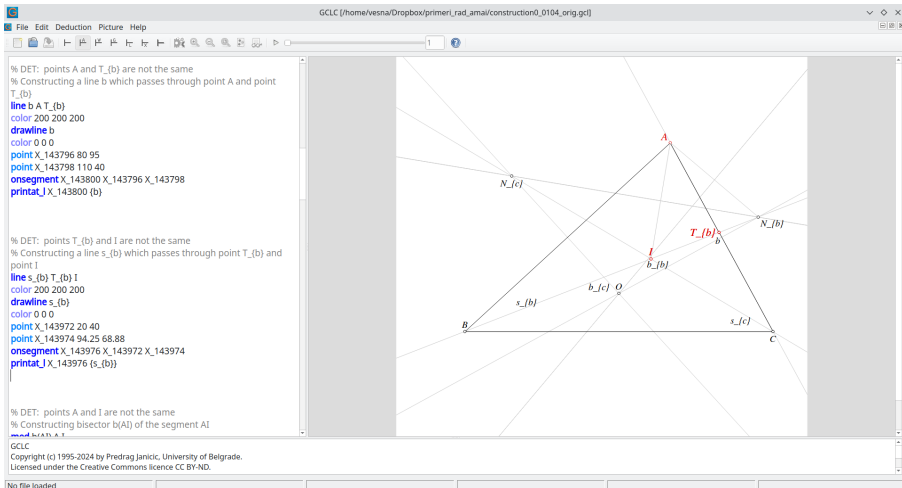
Table of Contents

- 1 Key Goals within Geometric Reasoning
- 2 Intelligent Dynamic Geometry Tools
- 3 Geometric Constructions
- 4 Automated Theorem Proving
- 5 Formalizations

Specifying and Visualizing Geometrical Constructions

- **GCLC** – a C++ tool for visualizing geometrical objects (Janičić, 1995)
- Unlike point-and-click tools, it is equipped with a language for specifying construction in a **formal**, textual form
- Offers ability of moving fixed points and generating animations
- It features built-in **geometric provers**:
 - area method (Janičić, Quaresma), GB method and Wu's method (Predović)
- It can serve as a tool for making digital illustrations of high quality

Example of Specifying and Visualisation of Construction in GCLC



Dynamic Visualization of Constructions

- **ArgoDG** – lightweight JavaScript library for precise specification and dynamic visualization of geometric constructions (Marić, 2021)
- Constructions are specified as **linear programs** in JavaScript
- Construction steps are defined as **functions** in JavaScript

```
function segment_bisector(A, B) {  
  c1 = circle(A, B);  
  c2 = circle(B, A);  
  [M1, M2] = intersectCC(c1, c2).both();  
  return line(M1, M2);  
}
```

- **Predicates** are used to select objects and enable branching constructions

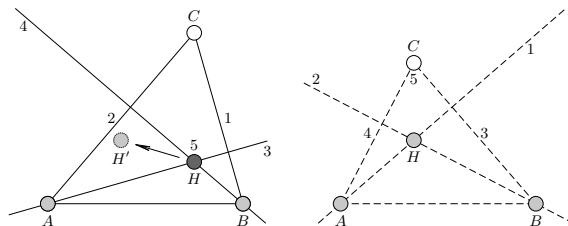
```
abs = circle(O, I); // absolute of the Poincare disc  
X = intersectCC(l1, l2).select(P => abs.inDisc(P));
```

Dynamic Geometry Tools and Typical Scenario

- Usual scenario:
 - ▶ user chooses several free points
 - ▶ using them user constructs other geometric objects
 - ▶ user can move free point and explore how other (constructed) objects change
- Example of Visualizing Construction using ArgoDG: A, B, H

Moving Constructed Points

- Rather new scenario: **moving constructed point**
 - ▶ necessary to specify which points stay fixed
 - ▶ user can move only one of the significant points in triangle
 - ▶ reconstruction of location of all other objects in the triangle is performed by reconstruction of the coordinates of triangle vertices
- Reconstruction of triangle vertices given positions of significant points in triangle boils down to solving triangle construction problem, which can be done using ArgoTriCS



- This scenario is implemented in tool **Touch&Drag** for touch screens (Selaković, Marinković, Janičić, 2018)

Table of Contents

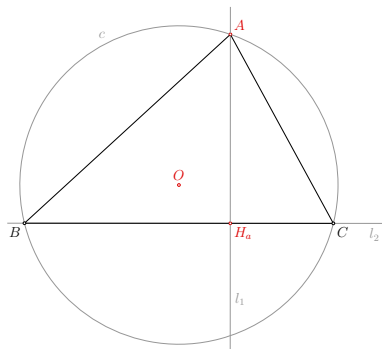
- 1 Key Goals within Geometric Reasoning
- 2 Intelligent Dynamic Geometry Tools
- 3 Geometric Constructions
- 4 Automated Theorem Proving
- 5 Formalizations

Solving Geometry Construction Problems

- **Geometric construction solving** is a special case of geometric constraint solving
- The task is to describe a construction of a geometric figure that meets given constraints
- Constructions are **procedures**
- The most commonly used tools: straightedge and compass
- The main problem in solving is **combinatorial explosion**
 - ▶ many different construction steps available
 - ▶ plenty of objects that each step could be applied to
- Some instances are unsolvable (e.g. angle trisection)

Location Triangle Construction Problems

- Primarily focused on **triangle location construction problems**: constructing $\triangle ABC$ if locations of three significant points in the triangle are given
- Example**: Construct a triangle ABC given locations of
 - its vertex A
 - its altitude foot H_a
 - its circumcenter O



- 1 Construct the line l_1 through points A and H_a
- 2 Construct the perpendicular l_2 to line l_1 through H_a
- 3 Construct the circle c centered at O passing through A
- 4 Construct the intersection points B and C of l_2 and c

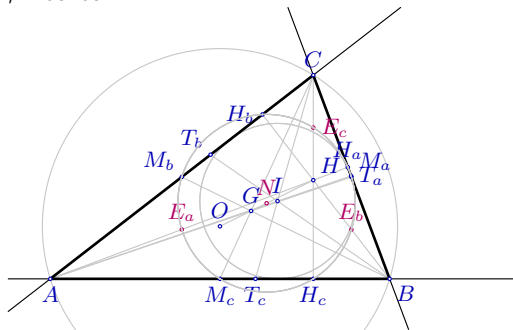
Wernick's Corpus of Triangle Location Construction Problems

- Significant points considered:

- ▶ vertices A, B, C ,
- ▶ side midpoints M_a, M_b, M_c ,
- ▶ feet of altitudes H_a, H_b, H_c ,
- ▶ feet of internal angle bisectors T_a, T_b, T_c ,
- ▶ orthocenter H , centroid G , circumcenter O , incenter I

- Possible status:

- ▶ solvable – S (A, B, M_a)
- ▶ unsolvable – U (T_a, T_b, T_c)
- ▶ redundant – R (A, B, M_c)
- ▶ locus-dependent – L (A, B, O)



Automated Construction Solving

- **ArgoTriCS** – **Argo Triangle Construction Solver** (Marinković, Janičić, 2015)
- System for automated solving of location construction problems **from the given corpus**
- Based on systematization of background geometrical knowledge needed
- It can export construction in NL form, formal specification of construction enabling visualization and reasoning about its correctness
- Solved 66 out of 74 significantly different solvable problems from Wernick's corpus
- Automatically generated compendium of solutions to problems from two different corpora available online (Marinković, 2015)

<http://poincare.matf.bg.ac.rs/~vesnap/animations/compendiums.html>

Automatically Generated Construction in NL Form

Problem: Construct a triangle ABC given the vertex A , altitude foot H_a , and circumcenter O .

Construction:

- 1 Construct the line h_a through points A and H_a ;
- 2 Construct the perpendicular a to line h_a through point H_a ;
- 3 Construct the circle $\mathcal{C}_c$ with center at point O passing through point A ;
- 4 Construct the intersection points B and C of the circle $\mathcal{C}_c$ and the line a .

NDG conditions: line a and circle $\mathcal{C}_c$ intersect; points A and O are not the same.

Determination conditions: points A and H_a are not the same.

Solution: A, H_a, O

Extending ArgoTriCS to Hyperbolic Geometry

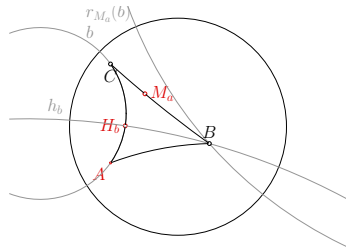
- ArgoTriCS was initially focused on Euclidean geometry
- By adapting **solely** geometrical knowledge ArgoTriCS can be adjusted to solve problems in **absolute** and **hyperbolic geometry** (Marinković, Šukilović, Marić, 2023)
- Challenges:
 - ▶ The centroid of a triangle does not divide the median in $2 : 1$ ratio,
 - ▶ The inscribed angle subtended by a diameter need not be right, ...
- We applied it to Wernick's corpus and solved 31 solvable problems, 1 redundant and 11 locus dependent
- Automatically generated compendium of solutions to problems in hyperbolic geometry:
http://poincare.matf.bg.ac.rs/~vesnap/animations_hyp/compendium_wernick_hyperbolic.html

Example Construction in Hyperbolic Geometry

Problem: Construct the triangle ABC given the vertex A , side midpoint M_a , and altitude foot H_b .

Construction:

- 1 Construct the line b through points A and H_b ;
- 2 Construct the line h_b perpendicular to the line b through H_b ;
- 3 Construct the line $s_{M_a}(b)$ that is image of the line b under the reflection wrt. point M_a ;
- 4 Construct the intersection point B of the lines $s_{M_a}(b)$ and h_b ;
- 5 Construct the point C symmetric to B wrt. point M_a .



Extending ArgoTriCS to Origami

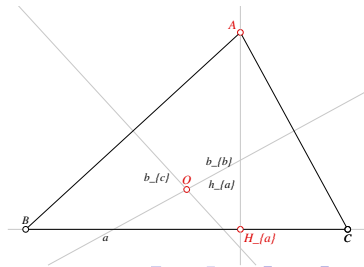
- When we talk about constructions, we implicitly consider ruler and compass constructions
- However, alternative tools can be used, for instance **origami**
- By **solely** adjusting geometrical knowledge, ArgoTriCS can be adapted to generate origami constructions (Milojković, Marinković, 2026)
- Challenges:
 - ▶ Only points and lines are supported primitive objects
 - ▶ Primitive constructions that use circles has to be replaced with a combination of reflections and alignment constraints, ...
- Obtained constructions are often significantly different

Example Construction using Origami

Problem: Construct the triangle ABC given the vertex A , circumcenter O , and altitude foot H_a .

Construction:

- 1 Construct the fold line h_a through the points A and H_a ;
- 2 Construct the fold line a that passes through the point H_a and is perpendicular to the line h_a ;
- 3 Construct the fold line b_c that passes through the point O and superposes the point A with the line a , and the point B obtained from the point O by folding;
- 4 Construct the fold line b_b that passes through the point O and superposes the point A with the line a , and the point C obtained from the point O by folding.



Next Step Guidance for Solving Construction Problems

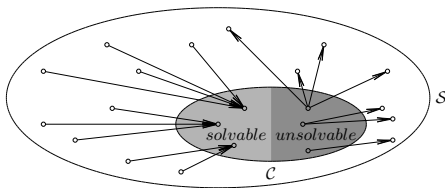
- Help students once they get stuck and do not know how to proceed with the construction (Marinković, Marić, 2020)
- Different types of help:
 - ▶ suggest the next construction step
 - ▶ suggest an object (point, line, circle) that should be constructed
 - ▶ suggest a lemma that the construction relies on
- There can be **many** possible constructions for the same problem
- We need to select **target construction** closest to the partial construction given by student
- There are different approaches to choose it:
 - ▶ ArgoTriCS can find several possible constructions (in advance), compare them to the partial construction, and select the most adequate one
 - ▶ ArgoTriCS can be restarted given all the objects constructed by the partial construction

Proving Unconstructibility

- Most of the times we are focused on solving construction problems
- What about **unsolvable** construction problems?
- Proofs of unconstructibility are typically algebraic and not human understandable
- Determining the status of one problem can be **reduced** to determining the status of another (Schreck, Marinković, Janičić, 2016)
- Then we need to determine explicitly the status of just a few problems
- There is a **reduction** from triangle construction problem $P(A_1, A_2, A_3)$ to problem $P'(A'_1, A'_2, A'_3)$ if each point from P' **can be constructed** using the points from P
 - ▶ if P' is **S**, then P is also **S**; if P is **U**, then P' is either **U** or **R** or **L**
 - ▶ if P is **L**, then P' is **R** or **L**; if P is **R**, then P' is also **R**
- How to effectively find reductions between triangle construction problems from Wernick's problems?

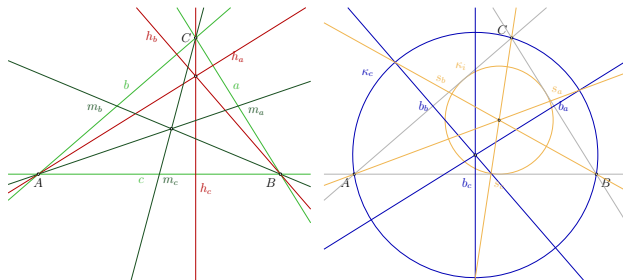
Identifying Key Problems

- Using ArgoTriCS we identify points constructible from a small set of points
- The goal is to find **minimal** subset $\mathcal{C}$ of the set of construction problems $\mathcal{S}$ that satisfies:
 - ▶ for each problem $P \in \mathcal{S}$ that is **S**, there is a problem $P' \in \mathcal{C}$ such that each point in P' is constructible from P
 - ▶ for each problem $P \in \mathcal{S}$ that is **U**, there is a problem $P' \in \mathcal{C}$ such that each point in P is constructible from P'
- The problem reduces to a set covering problem, solvable by our system URSA (Janičić)
- Problems from $\mathcal{C}$ are **key problems**



Dealing with Locus Dependent Problems

- Construction problems whose solvability depends on certain conditions are rarely studied
- We made a thorough analysis of different types of L problems (Marinković, Šukilović, Marić, 2026)
- A new corpus of construction problems – **DTCP corpus** is proposed which provides a number of problems solvable only when the initial elements satisfy certain conditions



- Solutions to problems from DTCP corpus visualized using ArgoDG:

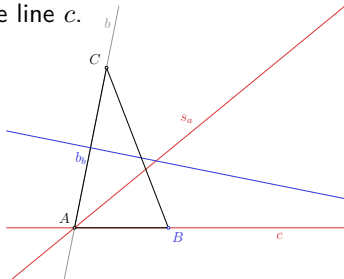
https://poincare.matf.bg.ac.rs/~filip.marić/locus_dependent_corpus/

Example of Solution to Locus Dependent Problem

Problem: Construct the triangle ABC given its side c , an internal angle bisector s_a , and a perpendicular bisector b_b .

Construction:

- 1 Construct the intersection point A of the lines c and s_a ;
- 2 Construct the line b symmetrical to the line c with respect to the line s_a ;
- 3 Choose **freely** a line b_b perpendicular to the line b ;
- 4 Construct the point C symmetrical to the point A with respect to the line b_b ;
- 5 Choose **freely** a point B on the line c .



Solving Construction Problems as Constraint Solving

- Solving geometric constructions can be seen as **automated planning problem** (Banković, 2025)
- We employ existing artificial intelligence tools in search for a solution of construction problem
- By solving problem in this way one can rely on the efficient search that is already implemented within the tool and focus on geometric aspects of the problem
- Often a shorter construction plans are found thanks to the optimization capabilities offered by the modern planners and constraint solvers

Table of Contents

- 1 Key Goals within Geometric Reasoning
- 2 Intelligent Dynamic Geometry Tools
- 3 Geometric Constructions
- 4 Automated Theorem Proving**
- 5 Formalizations

Automated Theorem Proving

- Geometry theorems arise from different contexts
- Our work is organized along two main directions:
 - ▶ Development of our implementations of well known proving methods or custom provers for concrete tasks
 - ▶ Using existing proving methods and evaluate success rate on a set of problems

Open-source Implementation of Algebraic Provers

- **OpenGeoProver** is an improved Java implementation of the simple Wu's method integrated within GeoGebra, along with other provers (Botana, Hohenwarter, Janičić, Kovacs, Petrović, Recio, Weitzhofer, 2015)
- It produces detailed report with steps that are taken in proving process
- A list of NDG conditions obtained during proving process, transformed to a readable, geometry form is returned

Proving Constructions Correct

- Most of existing systems for automated solving geometric constructions **do not** consider proving generated constructions correct
- This is not an easy task, since in some cases **the proof is quite different from the analysis**
- Proving triangle construction correct means proving that the points given by problem setting are indeed the corresponding points of the constructed triangle
- We equipped ArgoTriCS with ability to generate correctness conjectures that can be passed to automated provers (Marinković, Janičić, 2015)
- We compared four different algebraic and semi-synthetic provers
 - ▶ area method, Wu's method, and GB method provers within GCLC
 - ▶ OpenGeoProver
- We worked also on **portfolio approach**, by choosing one particular solver for a specific correctness conjecture (Nikolić, Marinković, Janičić, 2018)

Readability and Understandability of Proofs

- We need explanation why some statement holds
- None of the proofs obtained by considered provers are traditional, human-readable proofs
- How can readable correctness proofs be **automatically** obtained?
- Are there automated provers that can
 - ▶ prove these types of conjectures?
 - ▶ produce readable proofs?
 - ▶ possibly produce formal proofs?

Coherent Logic

- **Coherent logic** (CL) is a fragment of FOL that consists of formulae of a form:

$$A_1(\vec{x}) \wedge \dots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}_1 \mathcal{B}_1(\vec{x}, \vec{y}_1) \vee \dots \vee \exists \vec{y}_m \mathcal{B}_m(\vec{x}, \vec{y}_m)$$

- Reasoning in CL is constructive
- CL is suitable for generation of readable and formal proofs
- Many theories like geometry can be directly formulated in CL
- Having prover capable of efficient automatic generation of formal, human-readable proofs would be beneficial
- There are not many CL provers

CL Provers Developed by ARGO Group

- **Euklid** (Janičić, Kordić, 1995)
 - ▶ the first CL prover
 - ▶ uses forward chaining and iterative deepening
 - ▶ has fixed set of geometrical axioms
 - ▶ export proofs in NL form
- **ArgoCLP** (Stojanović Đurđević, Marinković, Janičić, 2011)
 - ▶ generic CL prover that can be used for any coherent theory
 - ▶ uses forward chaining and iterative deepening
 - ▶ exploits different techniques that improves efficiency and preserve completeness, but it can also use some techniques that do not preserve completeness
 - ▶ exports both readable proofs in NL form, formal proofs in Isabelle, and proofs in XML that can be exported into different formats

CL provers developed by ARGO group

- **Larus** (Janičić, Narboux, 2023)
 - ▶ it searches for a proof of a given length by encoding proofs as sequences of natural numbers satisfying certain conditions
 - ▶ suitable SAT or SMT solver can find such a sequence
 - ▶ it exports readable proofs and formal proofs in Isabelle and Coq
- **GCProuver** (Novaković, 2025)
 - ▶ a prover based on fragment of CL without disjunctions and existential quantifiers
 - ▶ it also uses forward chaining
 - ▶ it export proofs in NL form

Example of Automatically Generated Readable Proof by ArgoCLP

Theorem TH_geo:

Show that the plane α and the plane β do not intersect or there exist a line p such that the line p is incident to the plane α and the line p is incident to the plane β .

Proof:

1. It holds that the plane α and the plane β intersect or the plane α and the plane β do not intersect (by axiom ax_D8a).
2. Assume that the plane α and the plane β intersect.
 3. From the facts that the plane α and the plane β intersect, there exist a point A such that the point A is incident to the plane α , the point A is incident to the plane β and $\alpha \neq \beta$ (by axiom ax_D8).
 4. From the facts that $\alpha \neq \beta$, the point A is incident to the plane α , and the point A is incident to the plane β , there exist a point B such that $A \neq B$, the point B is incident to the plane α and the point B is incident to the plane β (by axiom ax_I7).
 5. From the facts that $A \neq B$, there exist a line p such that the point A is incident to the line p and the point B is incident to the line p (by axiom ax_I1).
 6. From the facts that the point A is incident to the line p , $A \neq B$, the point B is incident to the line p , the point A is incident to the plane α , and the point B is incident to the plane α , it holds that the line p is incident to the plane α (by axiom ax_I6).
 7. From the facts that the point A is incident to the line p , $A \neq B$, the point B is incident to the line p , the point A is incident to the plane β , and the point B is incident to the plane β , it holds that the line p is incident to the plane β (by axiom ax_I6).

This proves the conjecture.

8. Assume that the plane α and the plane β do not intersect.

This proves the conjecture.

Example of Automatically Generated Readable Correctness Proof by Larus

Axioms:

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

Theorem: th_A_Ha_O0 :

$inc(pA, ha1) \wedge inc(pHa1, ha1) \wedge perp(ha1, a1) \wedge inc(pHa1, a1) \wedge inc_c(pA, cc1) \wedge center(pOc1, cc1) \wedge inc_c(pB, cc1) \wedge inc(pB, a1) \wedge inc_c(pC, cc1) \wedge inc(pC, a1) \Rightarrow pHa = pHa1$

Proof:

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

Applications of CL Provers

- ArgoCLP is tested on simple theorems from standard geometry textbooks using four different axiom systems for Euclidean geometry: it succeeded to solve most of them
- Larus and GCPProver are tested on the subset of generated correctness conjectures of solvable problems from Wernick's corpus: out of 35 problems Larus proved 20 problems, while GCPProver proved them all

Solving Construction Problems using Larus

- We reduce the problem of geometry constructions to FOL theorem proving (Janičić, Marinković, 2026)
- We designed FOL axioms that mimic construction steps and geometry properties
- We use Larus to get readable proofs, from which constructions are extracted
- Afterwards, for the generated construction we use Larus to prove correctness
- Larus prove theorems by using an underlying SMT solver

Example of Construction Generated by Larus

thm_ABH : $\forall A \forall B \forall C \forall H (triangle(A, B, C) \wedge freepoint(A, null, null) \wedge freepoint(B, null, null) \wedge freepoint(H, null, null) \wedge orthocenter(H, A, B, C) \wedge \neg collinear(A, B, H) \Rightarrow solved_triangle(A, B, C))$

Consider arbitrary b, c, d, e such that:

- $triangle(b, c, d),$
- $freepoint(b, null, null),$
- $freepoint(c, null, null),$
- $freepoint(e, null, null),$
- $orthocenter(e, b, c, d),$
- $\neg collinear(b, c, e).$

It should be proved that $solved_triangle(b, c, d).$

1. $known(b)$ (by MP, using axiom known_free; instantiation: $A \mapsto b, X \mapsto null, Y \mapsto null$)
2. $known(c)$ (by MP, using axiom known_free; instantiation: $A \mapsto c, X \mapsto null, Y \mapsto null$)
3. $known(e)$ (by MP, using axiom known_free; instantiation: $A \mapsto e, X \mapsto null, Y \mapsto null$)
4. $perpendicular(b, e, c, d)$ (by MP, using axiom ax_orthocenter2; instantiation: $A \mapsto b, B \mapsto c, C \mapsto d, H \mapsto e$)
5. $orthocenter(e, c, b, d)$ (by MP, using axiom ax_orthocenter9; instantiation: $A \mapsto b, B \mapsto c, C \mapsto d, H \mapsto e$)
6. $perpendicular(c, e, b, d)$ (by MP, using axiom ax_orthocenter2; instantiation: $A \mapsto c, B \mapsto b, C \mapsto d, H \mapsto e$)
7. Let w be such that
 $rel_fun_random_on_perpendicular_from(w, c, b, e) \wedge nneqnative(w, c) \wedge nneqnative(w, d) \wedge on_line(d, c, w) \wedge known(w)$ (by MP, using axiom ax_perp12; instantiation: $A \mapsto b, B \mapsto e, C \mapsto c, D \mapsto d$)
8. Let $w1$ be such that
 $rel_fun_random_on_perpendicular_from(w1, b, c, e) \wedge nneqnative(w1, b) \wedge nneqnative(w1, d) \wedge on_line(d, b, w1) \wedge known(w1)$ (by MP, using axiom ax_perp12; instantiation: $A \mapsto c, B \mapsto e, C \mapsto b, D \mapsto d$)
9. $collinear(d, c, w)$ (by MP, using axiom ax_coll11; instantiation: $A \mapsto d, B \mapsto c, C \mapsto w$)
0. $on_line(d, w1, b)$ (by MP, using axiom ax_coll12; instantiation: $A \mapsto d, B \mapsto b, C \mapsto w1$)
1. $intersection_line_line(d, c, w, w1, b)$ (by MP, using axiom ax_inter3; instantiation: $X \mapsto d, A \mapsto c, B \mapsto w, C \mapsto w1, D \mapsto b$)
2. $known(d)$ (by MP, using axiom known_inter; instantiation: $X \mapsto d, A \mapsto c, B \mapsto w, C \mapsto w1, D \mapsto b$)
3. $solved_triangle(b, c, d)$ (by MP, using axiom ax_solved_triangle; instantiation: $A \mapsto b, B \mapsto c, C \mapsto d$)
4. Proved by assumption! (by QEDas)

CL Vernacular

- A simple representation of proofs and statements in CL is proposed and it is named [CL vernacular](#) (Stojanović Đurđević, Narboux, Bezem, Janičić, 2014)
- It is a common representation which can be translated into proofs for different proof assistants (such as Coq and Isabelle)
- [ArgoGeoChecker](#) – a tool for transformation of semi-formal proofs (specified in a TPTP like language) to formal proofs is proposed: proof objects is expanded and expressed in CL vernacular (Stojanović Đurđević, 2018)
- It successfully verified 18 informal proofs found in Serbian high school geometry textbooks, and 4 informal proofs from Euclid's Elements

Proving Theorems in 3D Solid Geometry

- 3D geometry introduces additional complexity due to the increased number of geometric objects and relations holding between them
- Using algebraic methods for proving geometry theorems in 3D is not much explored
- We suggest two distinct algebraization methods for transforming solid geometry statements into algebraic representations (Simić, Stojanović Đurđević, Tanasijević, 2025)
- We propose **ArgoSolid prover** which integrates three third-party algebraic engines: GeoProver, Mathematica, and Maple and incorporate it into dynamic geometry system Stereos (Marić, Marić)

Table of Contents

- 1 Key Goals within Geometric Reasoning
- 2 Intelligent Dynamic Geometry Tools
- 3 Geometric Constructions
- 4 Automated Theorem Proving
- 5 Formalizations

Formalization of Algebraic Methods in Geometry

- We worked on formalization of algebraic methods for geometry theorem proving (Wu's method and the GB method) within Isabelle/HOL (Marić, Petrović, Simić, Janičić, 2011)
- The goal of this project was to promote the wider use of algebraic methods in education and formalization of mathematics
- Once formally verified in a proof assistant, algebraic methods can be used as tactics to automate significant parts of formal geometry proofs

Formalization of Complex Plane Geometry

- We did formalization of the **geometry of the complex plane** within Isabelle/HOL (Simić, Marić, 2014)
- All objects are represented algebraically (using vectors, matrices, ...)
- A ground for formally investigating various non-Euclidean geometries
- We did a formalization of **the Poincare disc model** of hyperbolic geometry (Simić, Marić, Boutry, 2020)

Formalization of Gyrovector Spaces as Models of Hyperbolic Geometry

- We did a formalization of algebraic structures known as gyrogroups and gyrovector spaces in Isabelle/HOL (Marković, Marić, 2025)
- Gyrovector spaces can be used to define models of hyperbolic geometry
- Gyrovector spaces exhibit advantage that definitions show remarkable syntactical similarities to standard Euclidean geometry

Formalization of Tarski's Geometry

- We propose **framework for the formalization of mathematical knowledge** that can produce machine verifiable proofs but also human-readable proofs
- We focus on a book on Tarski's geometry and tried to automatically generate proofs of the theorems from this book using a combination of resolution provers and CL prover (Stojanović Đurđević, Narboux, Janičić, 2015)
- We tried different settings: using only theorems from the book or using additional lemmas from the existing Coq formalization of the book
- We succeeded to prove 37% of the theorems from the book automatically (with readable and machine verifiable proofs generated) without any guidance, while with additional lemmas this percentage rised to 42%

Formalization of Common Geometrical Knowledge

- Obtained correctness proofs rely on high-level lemmas
`fof(ratio23_AGAMa, axiom, (ratio23(pA,pG,pA,pMa)))`.
- Correctness of geometry lemmas is usually **taken for granted**
- To be sure in validity of obtained proofs, correctness of used lemmas should be proved
- We are working on formalization of common geometric lemmas used as axioms within GCProuver using GeoCoq (Boutry, Marinković, 2025)
- We are also working on fixing and verifying axioms used by AlphaGeometry and to produce verifiable proofs (Marković, Doža, Janičić, 2026)