Université de Strasbourg

An Automated Approach towards Constructivizing the GeoCoq Library

Alexandre Jean, Pierre Boutry, Nicolas Magaud

University of Strasbourg

2 august 2025

Geocoq

- The only library to have formalized the arithmetization of geometry.
- Partially translated manually into Isabelle and Lean.
- About 150 kloc.

geocoq.github.io/GeoCoq/

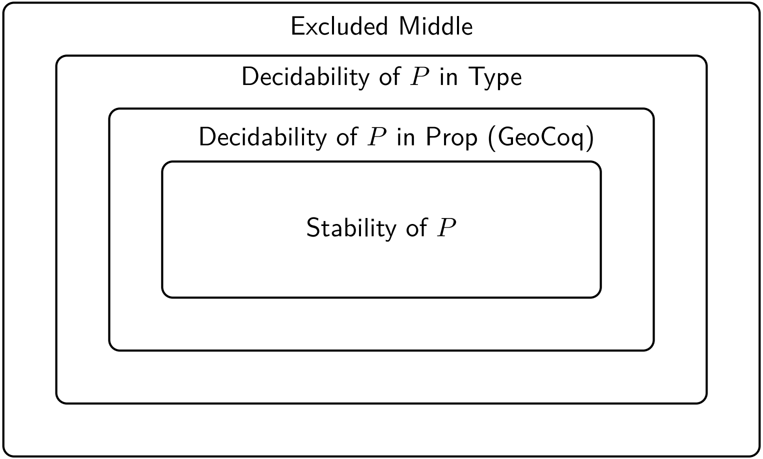
Axioms

Identity for betweenness	$A \text{—} B \text{—} A \Rightarrow A = B$
Transitivity for congruence	$AB \equiv CD \wedge AB \equiv EF \Rightarrow CD \equiv EF$
Reflexivity for congruence	$AB \equiv BA$
Identity for congruence	$AB \equiv CC \Rightarrow A = B$
Segment Construction	$\exists E, A \text{—} B \text{—} E \wedge BE \equiv CD$
Pasch	$A \text{—} P \text{—} C \wedge B \text{—} Q \text{—} C \Rightarrow \exists X, P \text{—} X \text{—} B \wedge Q \text{—} X \text{—} A$
Five-Segment	$AB \equiv A'B' \wedge BC \equiv B'C' \wedge$ $AD \equiv A'D' \wedge BD \equiv B'D' \wedge$ $A \text{—} B \text{—} C \wedge A' \text{—} B' \text{—} C' \wedge A \neq B \Rightarrow CD \equiv C'D'$
Lower 2-Dimensional	$\exists ABC, \neg A \text{—} B \text{—} C \wedge \neg B \text{—} C \text{—} A \wedge \neg C \text{—} A \text{—} B$
Upper 2-Dimensional	$AP \equiv AQ \wedge BP \equiv BQ \wedge CP \equiv CQ \wedge P \neq Q \Rightarrow$ $A \text{—} B \text{—} C \vee B \text{—} C \text{—} A \vee C \text{—} A \text{—} B$
Euclid	$A \text{—} D \text{—} T \wedge B \text{—} D \text{—} C \wedge A \neq D \Rightarrow$ $\exists XY, A \text{—} B \text{—} X \wedge A \text{—} C \text{—} Y \wedge X \text{—} T \text{—} Y$
Continuity	$\forall \Xi \Upsilon, (\exists A, (\forall XY, \Xi X \wedge \Upsilon Y \Rightarrow A \text{—} X \text{—} Y)) \Rightarrow$ $\exists B, (\forall XY, \Xi X \wedge \Upsilon Y \Rightarrow X \text{—} B \text{—} Y)$
Point equality decidability	$X = Y \vee X \neq Y$

Axioms

Identity for betweenness	$A–B–A \Rightarrow A = B$
Transitivity for congruence	$AB \equiv CD \wedge AB \equiv EF \Rightarrow CD \equiv EF$
Reflexivity for congruence	$AB \equiv BA$
Identity for congruence	$AB \equiv CC \Rightarrow A = B$
Segment Construction	$\exists E, A–B–E \wedge BE \equiv CD$
Pasch	$A–P–C \wedge B–Q–C \Rightarrow \exists X, P–X–B \wedge Q–X–A$
Five-Segment	$AB \equiv A'B' \wedge BC \equiv B'C' \wedge$ $AD \equiv A'D' \wedge BD \equiv B'D' \wedge$ $A–B–C \wedge A'–B'–C' \wedge A \neq B \Rightarrow CD \equiv C'D'$
Lower 2-Dimensional	$\exists ABC, \neg A–B–C \wedge \neg B–C–A \wedge \neg C–A–B$
Upper 2-Dimensional	$AP \equiv AQ \wedge BP \equiv BQ \wedge CP \equiv CQ \wedge P \neq Q \Rightarrow$ $A–B–C \vee B–C–A \vee C–A–B$
Euclid	$A–D–T \wedge B–D–C \wedge A \neq D \Rightarrow$ $\exists XY, A–B–X \wedge A–C–Y \wedge X–T–Y$
Continuity	$\forall \Xi \Upsilon, (\exists A, (\forall XY, \Xi X \wedge \Upsilon Y \Rightarrow A–X–Y)) \Rightarrow$ $\exists B, (\forall XY, \Xi X \wedge \Upsilon Y \Rightarrow X–B–Y)$
Point equality decidability	$X = Y \vee X \neq Y$

Decidability



Stability

Definition

A predicate P is stable if

$$\forall x, \neg\neg P(x) \to P(x).$$

It is trivial to show that a decidable predicate P is also stable.

Constructive or

We know that:

$$\forall A\ B,\ \mathrm{stable}\, A \Rightarrow \mathrm{stable}\, B \Rightarrow \mathrm{stable}\, A \wedge B$$

However, this is not true for $A \vee B$.
To have some form of disjunction, we introduce the following formula: $\neg(\neg A \wedge \neg B)$, noted $A \sqcup B$.
This disjunction preserves the stability of propositions.

Stability of equality, congruence and betweeness

We assumes the stability of point equality, $\neg\neg X = Y \Rightarrow X = Y$
Using this, we can deduce the stability of the congruence predicate Cong, but not of the betweenesss predicate Bet, we can only prove its stability under a non-degeneracy assumption:
$\forall A\ B\ C, A \neq B \Rightarrow B \neq C \Rightarrow \neg\neg \,\mathrm{Bet}\, A\ B\ C \Rightarrow \mathrm{Bet}\, A\ B\ C$
We define a new predicate of betweeness with this non-degeneracy assumption and name it BetS.
We then define BetL as follow:

$$A = B \sqcup B = C \sqcup \mathrm{BetS}\, A\ B\ C$$

Wholesale importation

Now that all the primitive predicates are stable.

We call formulas that do not involve $\exists$ or $\vee$ negative, the others are called positive.

Negative formulas are stable.

If a formula F is classically provable, then $\neg\neg F$ is provable intuitionistically.

As a result, all negative formulas remain provable without relying on the decidability of point equality.

Stable modus ponens

The stable *modus ponens* is stated as follows:

$$\mathrm{stable}\, B \wedge \neg\neg A \wedge (A \Rightarrow B) \Rightarrow B$$

Reasoning by decidability to prove a stable property is a corollary of this rule.
Indeed, for any proposition P, we have:

$$\neg\neg(P \vee \neg P)$$

By inner pasch

$$A\text{—}P\text{—}C \wedge B\text{—}Q\text{—}C \Rightarrow$$

$$\exists X, P\text{—}X\text{—}B \wedge Q\text{—}X\text{—}A$$

$$\begin{aligned}
&A\text{—}P\text{—}C \wedge B\text{—}Q\text{—}C \wedge \\
&A \neq P \wedge P \neq C \wedge \\
&B \neq Q \wedge Q \neq C \wedge \\
&\neg\left(\neg A\text{—}B\text{—}C \wedge \neg B\text{—}C\text{—}A \wedge \neg C\text{—}A\text{—}B\right) \Rightarrow \\
&\exists X, P\text{—}X\text{—}B \wedge Q\text{—}X\text{—}A
\end{aligned}$$

```
Lemma by_inner_pasch : forall A B C P Q SP,
  stable SP ->
  BetL A P C -> BetL B Q C ->
  ((exists X, BetL P X B /\ BetL Q X A) -> SP) ->
  SP.
```

Proof differences after constructivization

```
Lemma between_equality : forall A B C,
  Bet A B C -> Bet B A C ->
  A = B.
Proof.
 intros A B C HB1 HB2.
 destruct (inner_pasch A B C B A HB1 HB2) as [I [HB3 HB4]].
 apply between_identity in HB3, HB4; congruence.
Qed.
```

```
Lemma between_equality : forall A B C,
  BetL A B C -> BetL B A C ->
  A = B.
Proof.
intros A B C HB1 HB2.
stab_destruct (inner_pasch A B C B A HB1 HB2) as [I [HB3 HB4]].
apply between_identity in HB3, HB4; congruence.
Qed.
```

Rocq-ditto

- OCaml library for rewriting Rocq ASTs[1].
- Use *rocq-lsp* to extract a Rocq AST from a `.v` file.
- Allows for easy Rocq AST rewriting by automatically moving other AST nodes when adding, removing or replacing a node.
- Compatible with Ocaml standard library functions: *filter*, *fold*, *map*, etc.
- Dual proof representation: tree-based and linear.
- Allow for speculative execution.
- Provides quoting and unquoting functions.

[1] Abstract Syntax Trees

Defining a transformation with Rocq-ditto

Definition

Transformation: A transformation is a function f that takes a proof as input and returns a list of transformation steps from the set:

$$\{Remove(id),\ Replace(id,\ new\_node),\ Add(new\_node),\ Attach(new\_node,\ attach\_position,\ anchor\_id)\}$$

- **Remove**(id) : removes the node identified by id.
- **Replace**(id, *new_node*) : replaces the node identified by id with *new_node*
- **Add**(*new_node*) : adds a new node to the AST
- **Attach**(*new_node*, *attach_position*, *anchor_id*) : places new_node at a position relative to the node identified by $anchor\_id$.

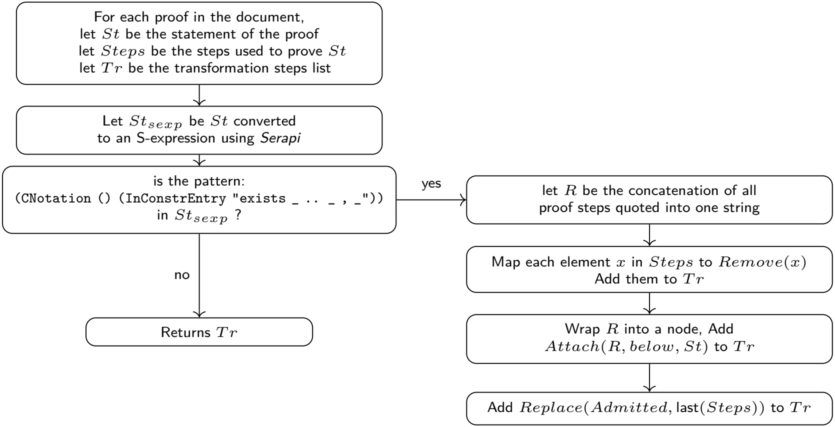
Current transformations for constructivization

- A transformation to admit proofs involving *exists* in the proof statement.
- A transformation to replace `\/` by its constructive version, `\_/`.
- A transformation to replace `Bet` by `BetL`.
- A transformation to replace classical tactics like `left` with constructive alternatives, here `stab_left`.

```
Lemma by_left : forall A B : Prop,
  A -> A \_/ B.
Proof. unfold or_dM; tauto. Qed.

Ltac stab_left :=
  match goal with
  | |- ?A \_/ ?B => apply (by_left A B)
  end.
```

Zoom on a Transformation: Admitting Proofs Using *exists*



Difficulties

During development with Rocq-ditto, we encountered a major blocker, which we call *small-scale proof repair*.

Definition (Proof Repair (Talia Ringer))

The problem of automatically updating proofs in response to changes in programs or specifications.

Initially, we assumed that most transformations would leave proofs intact.
In practice, however, many proofs required small adjustments to align the new proof state with the expected next proof state.

Small scale proof repair

We believe our problem is simpler than the general proof repair problem, due to the following properties:

- The proof can be represented as a tree, making dependencies explicit.
- We have access to the proof state both before and after the change.

This allows us to identify which branches of the proof are affected by a change.
As a result, the scope of the repair remains limited.

Additional difficulties

Additional difficulties arose during the writing of the paper but have since been resolved, such as:

- **Importing errors:** Loading our new constructive definitions required unloading previously imported modules and reloading new ones. This was addressed by implementing a custom loader that uses the existing `_CoqProject` configuration.

Results

- Manual constructivization of the first chapters of GeoCoq.
- Creation of multiple tactics to help automate the constructivization of Geocoq.
- Development of *Rocq-ditto*, a Rocq AST rewriting library.
- Automated syntactic replacement of tactics and predicates in Geocoq.

Future work

The objective of our future work is a complete constructivization of the GeoCoq library.
To do that, we currently aim to solves the following problems:

- Small scale proof repair.
- We aim to proves lemmas that uses existential quantifiers in their statement by checking if they are only used in negative proofs.

Conclusion

Summary

- Objective: constructize Geocoq, a Rocq geometry library.
- Current results: manual constructivization of the first chapters of Geocoq, developpement of a library to mechanize this transformation.
- Future work: small scale proof repair, work on a subset of positive lemmas.

Thank you for your attention, any questions ?