Automated Generation of Formal and Readable Proofs in Geometry Using Coherent Logic

Sana Stojanović, Vesna Pavlović, Predrag Janičić
Faculty of Mathematics, Belgrade, Serbia
{sana,vesnap,janicic}@matf.bg.ac.rs

*Eighth International Workshop on
Automated Deduction in Geometry,
Munich, 22.07.2010.*

Agenda

- Motivation and goals
- Axiomatization and formalization of geometry
- Coherent logic
- ArgoCLP prover
- Conclusions and future work

Motivation and goals

- Two main directions in computer theorem proving in geometry:
  - Interactive theorem proving (using proof assistants)
  - Automated theorem proving (e.g, using algebraic methods)
- These directions have different motivations, but can get closer
- Goals:
  - development of a prover that automatically generates formal proofs, but also traditional, human readable proofs
  - proving automatically theorems that are a current subject of manual formal proving

Axiomatizations of Geometry

- Euclid, “Elements”
- Hilbert, “The Foundations of Geometry”
  - three sorts of primitive objects
  - the set of axioms is divided into five groups
- Borsuk, Szmielev
- Tarski
  - one sort of primitive objects
  - only two predicates and eleven axioms

Formalizations of Geometry

- Formalization of Hilbert's axiomatics:
  - Dehlinger/Dufourd/Schreck using Coq (2000)
  - Fleuriot/Meikle using Isabelle/Isar (2003)
- Formalization of Tarski's axiomatics:
  - Narboux using Coq (2006)
- Formalization of projective plane geometry:
  - Narboux/Magaud/Schreck using Coq (2008)
- Avigad/Dean/Mumma (2008), development of new axiomatization for Euclid's "Elements" (not formalized yet within a proof assistant)

Coherent Logic

A fragment of first-order logic, consisting of formulae of the following form:

$$A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}_1\ B_1(\vec{x}, \vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\ B_m(\vec{x}, \vec{y}_m)$$

CL deals with sets of facts (ground atomic expressions).

The only inference rules used are:

$$\frac{A_1(\vec{a}) \wedge \ldots \wedge A_n(\vec{a})}{A_1(\vec{a}), \ldots, A_n(\vec{a})}\ \wedge E \qquad \frac{A_1 \vee \ldots \vee A_n \quad \begin{array}{c}[A_1]\\ \vdots \\ B\end{array} \quad \ldots \quad \begin{array}{c}[A_n]\\ \vdots \\ B\end{array}}{B}\ \vee E \qquad \frac{\bot}{A}\ efq$$

$$\frac{A_1(\vec{a}), \ldots, A_n(\vec{a}) \quad A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}_1\ B_1(\vec{x}, \vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\ B_m(\vec{x}, \vec{y}_m)}{B_1(\vec{a}, \vec{w}_1) \vee \ldots \vee B_m(\vec{a}, \vec{w}_m)}\ ax$$

A formula is a CL-theorem if from its premises all conjuncts of a formula $B_j(\vec{x}, \vec{w})$ can be derived for some j and for some vector of constants $\vec{w}$.

A breadth-first proof procedure for coherent logic is sound and complete.

ArgoCLP Proof Procedures

- A generic proof procedure for coherent logic
- Sorts can be used
- Negations can be used in limited way

  $R(\vec{x}) \vee nonR(\vec{x})$

  $R(\vec{x}) \wedge nonR(\vec{x}) \Rightarrow \bot$

- The first axiom schema brings us out of intuitionistic setting

Basic Proof Procedure

- Simple proof procedure (forward chaining, iterative deepening)
- Sets of facts are maintained
- The axioms are applied in waterfall manner
- A dedicated counter that controls applications of axioms
  - Initially equals the number of constants appearing in the premises of the conjecture
  - Increases once no axiom can be applied
- The procedure is sound and complete, but inefficient

Improved Proof Procedure
(techniques that preserve completeness)

- Ordering of axioms
  - non-productive non-branching axioms:
    $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x})$,
    $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow B(\vec{x})$
  - non-productive branching axioms:
    $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow B_1(\vec{x}) \vee \ldots \vee B_m(\vec{x})$
  - productive non-branching axioms:
    $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}\ B(\vec{x}, \vec{y})$
  - productive branching axioms:
    $A_1(\vec{x}) \wedge \ldots \wedge A_n(\vec{x}) \Rightarrow \exists \vec{y}_1\ B_1(\vec{x}, \vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\ B_m(\vec{x}, \vec{y}_m)$
  - strongly productive non-branching axioms:
    $\exists \vec{y}\ B(\vec{x}, \vec{y})$
  - strongly productive branching axioms:
    $\exists \vec{y}_1\ B_1(\vec{x}, \vec{y}_1) \vee \ldots \vee \exists \vec{y}_m\ B_m(\vec{x}, \vec{y}_m)$

Improved Proof Procedure (techniques that preserve completeness)

- Early pruning in unifications used in axiom applications
- Lemma generation for axioms that introduce several witnesses
- Dealing with equality (Tarjan's union-find algorithm)
- Dealing with symmetrical predicate symbols
- Reuse of proved theorems

Improved Proof Procedure (techniques that don't preserve completeness)

- Restriction on branching axioms
  - axioms of the form $R(\vec{x}) \vee nonR(\vec{x})$ are generated and used only for primitive predicates
- Restriction on axioms used
  - all the predicates from the axiom occur in the conjecture
  - at least one predicate from the axiom occurs in the conjecture
- The restrictions are irrelevant for a concrete formula if it was proved

Implementation

- Generic implementation
- The prover can be used for any coherent theory
- Implemented in C++ (around 5000 lines of code)
- Freely available

Program input

- Conjecture, axioms and definitions are of the form:

```
point(1) point(2) ~eq_point(1,2) =>
 line(3) inc_po_l(1,3) inc_po_l(2,3)
```

- The user can configure the prover to use some of the additional techniques

Program output

- "Clean" proof trace with all irelevant inference steps eliminated

- Proof in natural language (in English, in latex format)

- Formal proof (in Isabelle/Isar)

- Example: *Isabelle*, *Natural language*

Applications

- Four axiom systems for Euclidean (space) geometry
  - Hilbert
  - Borsuk
  - Janičić
  - Tarski
- Proved dozens of theorems (mostly simple)

Related Work

- Coherent logic was initially defined by Skolem and in recent years it was popularized by Bezem

- Janičić/Kordić, first automated theorem prover using CL

- Bezem/Coquand, CL prover that generates proof objects in Coq, implemented in Prolog

- Bezem/Berghofer, internal prover for CL in Isabelle, implemented in ML

Future work

- Improvement of the search procedure
  - Techniques used in other automated reasoning systems (SAT solvers)
- Improvement of other components
  - Support for TPTP input format
  - Automated detection of symmetrical relations and lemmas
  - Output in Coq
- Applications
  - Analyzing relationships between different axiomatic systems
  - Formalization of geometry knowledge (e.g, within Avigad's system)
  - Assistant for proving subgoals of larger theorems

Conclusions

ArgoCLP prover:

- produces formal, machine verifiable proofs (Isar)
- produces readable proofs given in a natural language form
- applicable for different geometries
- moreover, applicable for any theory with coherent axioms and for conjectures in the coherent form