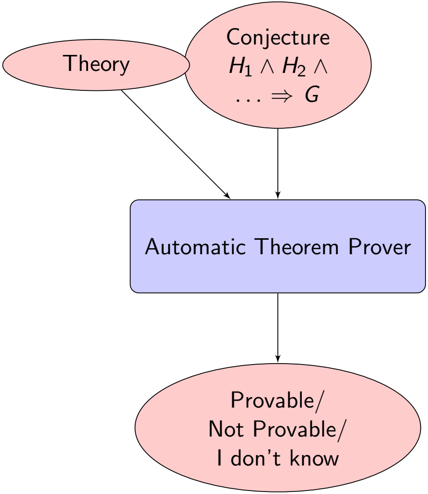
Automated Completion of Statements and Proofs in Synthetic Geometry: an Approach based on Constraint Solving

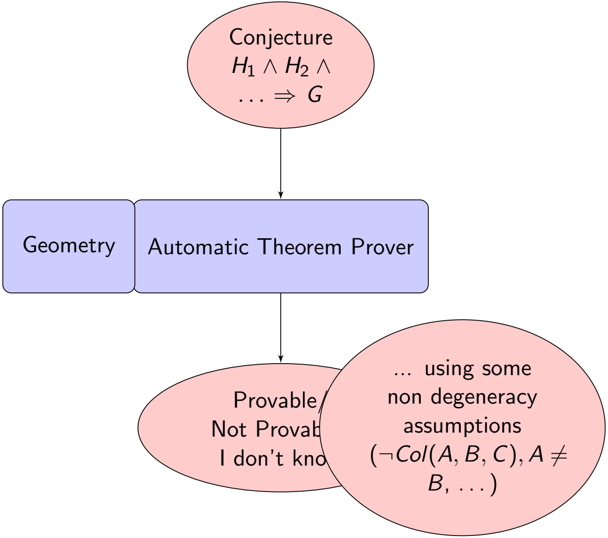
Salwa Tabet Gonzalez   Predrag Janičić   Julien Narboux
University of Belgrade, Serbia   University of Strasbourg, France

Automated Deduction in Geometry, September 2023, Beograd.

Automated deduction in general



Automated deduction in geometry

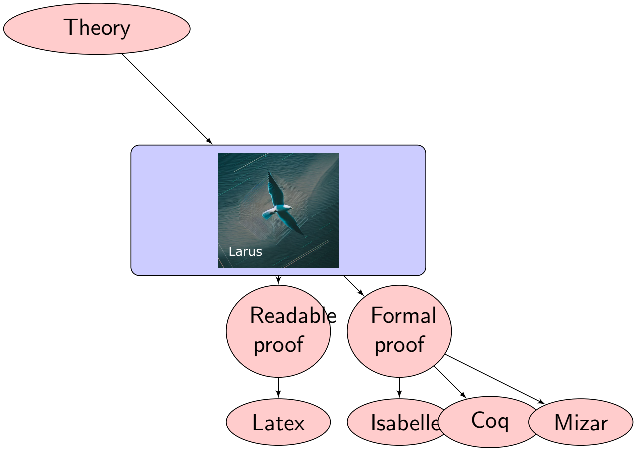


Conjecturing and proving

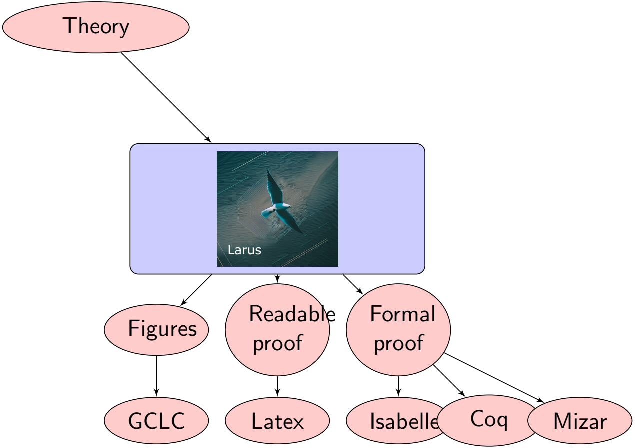
- But real mathematical activity does not fit into this picture.
- Conjecturing/refuting/proving/producing lemmas, theories or definitions are interlaced activities. See Lakatos [1]

[1] *Proofs and Refutations* (1976).

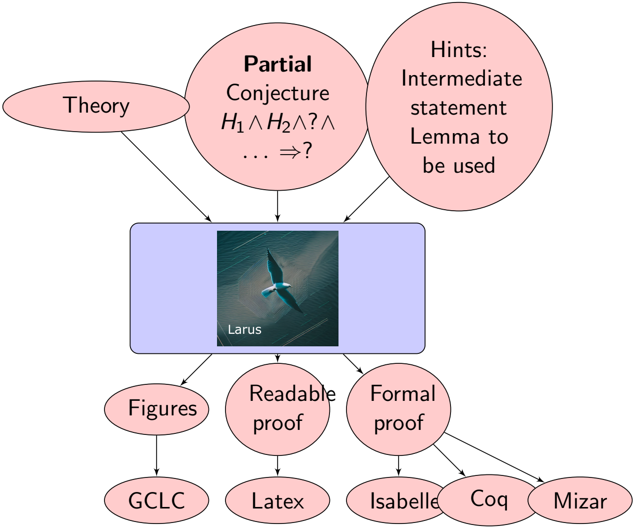
Our approach



Our approach



Our approach



Coherent Logic / Finitary Geometric Implications

- A formula of coherent logic (universally closed):

$$A_0(\vec{x}) \wedge \ldots \wedge A_{n-1}(\vec{x}) \Rightarrow \exists \vec{y}(B_0(\vec{x}, \vec{y}) \vee \ldots \vee \; B_{m-1}(\vec{x}, \vec{y}))$$

  where universal closure is assumed, A_i denotes an atomic formula, and B_j denotes a conjunction of atomic formulae.
- No function symbols of arity > 0 and no negations
- Many theories can be simply formulated in CL
- Every FOL theory can be translated into CL, possible with additional predicate symbols
- For instance, for each predicate symbol R, a new symbol $\overline{R}$ is introduced for $\neg R$, and the axioms: $\forall \vec{x}(R(\vec{x}) \wedge \overline{R}(\vec{x}) \Rightarrow \bot)$, $\forall \vec{x}(R(\vec{x}) \vee \overline{R}(\vec{x}))$

Inference System for Coherent Logic

$$\frac{\Gamma, ax, A_0(\vec{a}), \ldots, A_{n-1}(\vec{a}), \underline{B_0(\vec{a}, \vec{b}) \vee \ldots \vee B_{m-1}(\vec{a}, \vec{b})} \vdash P}{\Gamma, ax, A_0(\vec{a}), \ldots, A_{n-1}(\vec{a}) \vdash P} \text{ MP}$$

where ax is
$A_0(\vec{x}) \wedge \ldots \wedge A_{n-1}(\vec{x}) \Rightarrow \exists \vec{y}(B_0(\vec{x}, \vec{y}) \vee \ldots \vee B_{m-1}(\vec{x}, \vec{y}))$

$$\frac{\Gamma, \underline{B_0(\vec{c})} \vdash P \quad \ldots \quad \Gamma, \underline{B_{m-1}(\vec{c})} \vdash P}{\Gamma, B_0(\vec{c}) \vee \ldots \vee B_{m-1}(\vec{c}) \vdash P} \text{ QEDcs (case split)}$$

$$\frac{}{\Gamma, \underline{B_i(\vec{a}, \vec{b})} \vdash \exists \vec{y}(B_0(\vec{a}, \vec{y}) \vee \ldots \vee B_{m-1}(\vec{a}, \vec{y}))} \text{ QEDas (assumption)}$$

$$\frac{}{\Gamma, \bot \vdash P} \text{ QEDefq (ex falso quodlibet)}$$

Inference System for Coherent Logic: Example

Consider the following two axioms:

ax1: $\forall x\ (p(x) \Rightarrow r(x) \vee q(x))$ ax2: $\forall x\ (q(x) \Rightarrow \bot)$

and the conjecture: $\forall x\ (p(x) \Rightarrow r(x))$

$$\dfrac{\dfrac{\dfrac{}{ax1, ax2, p(a), r(a) \vdash r(a)}\ \text{QEDas} \qquad \dfrac{\dfrac{}{ax1, ax2, p(a), q(a), \bot \vdash r(a)}\ \text{QEDefq}}{ax1, ax2, p(a), q(a) \vdash r(a)}\ \text{MP(ax2)}}{ax1, ax2, p(a), r(a) \vee q(a) \vdash r(a)}\ \text{QEDcs}}{ax1, ax2, p(a) \vdash r(a)}\ \text{MP(ax1)}$$

The same proof in a forward manner, in a natural language form:

Consider an arbitrary a such that: $p(a)$. It should be proved that $r(a)$.

1. $r(a) \vee q(a)$ (by MP, from $p(a)$ using axiom ax1; instantiation: $X \mapsto a$)
    2. Case $r(a)$:
        3. Proved by assumption! (by QEDas)
    4. Case $q(a)$:
        5. $\bot$ (by MP, from $q(a)$ using axiom ax2; instantiation: $X \mapsto a$)
        6. Contradiction! (by QEDefq)
7. Proved by case split! (by QEDcs, by $r(a), q(a)$)

Starting Ideas

- The pure forward chaining approach to ATP does not take the goal into account.
- SAT/SMT solvers have seen huge progress in the recent years.
- Encoding the problem of finding a Coherent Logic proof into SAT/SMT theories can give a form of multidirectional reasoning.

Theorem Proving as Constraint Solving

- In traditional automated proving:
  - the search is performed over a set of formulae, and it terminates once the goal formula or contradiction is found.
  - a proof can then be reconstructed as a byproduct of this process.
- In our approach, *proving as constraint solving*:
  - a proof of a given formula can be represented by a sequence of natural numbers, meeting some constraints;
  - the search is performed globally over a set of possible proofs (i.e., over a set of possible sequences of natural numbers);
  - a proof is found by a solver that finds a sequence that meets these conditions.
  - a proper proof can be reconstructed from the found sequence.

Encoded Proof: Example

```
0.   1  0 0    2 0      /* Nesting: 1; Step kind:0 = Assumption;
                           Branching: no; p2(a) */
1.   1 13 1    4 0 6 0 /* Nesting: 1; Step kind:13 = MP-axiom:13;
                           Branching: yes; p4(a) or p6(a) */
                     0 /* From steps: (0) */
                     0 /* Instantiation */
2.   2  2 0    4 0      /* Nesting: 2; Step kind:2 = First case;
                           Branching: no; p4(a) */
3.   2 10               /* Nesting: 2; Step kind:10 =
                                            QED by assumption; */
4.   3  3 0    6 0      /* Nesting: 3; Step kind:3 = Second case;
                           Branching: no; p6(a) */
5.   3 14 0    0        /* Nesting: 3; Step kind:14=MP-axiom:14);
                           Branching: no; p0() */
                     4 /* From steps: (4) */
                     0 /* Instantiation */
6.   3 11               /* Nesting: 3; Step kind:11 = QED by EFQ;*/
7.   1  9               /* Nesting: 1; Step kind:9 = QED by cases;*/
```

Related work

Surprisingly (as far as we know), this approach has hardly been studied extensively. Only, partly related:

- Todd Deshane, Wenjin Hu, Patty Jablonski, Hai Lin, Christopher Lynch, and Ralph Eric McGregor. *Encoding First Order Proofs in SAT*, CADE-21, 2007.
- Jeremy Bongio, Cyrus Katrak, Hai Lin, Christopher Lynch, and Ralph Eric McGregor. *Encoding First Order Proofs in SMT*. ENTCS, 198(2):71–84, 2008.

Proof encoding and constraints

- We generate constraints that a sequence of natural numbers represents a valid proof.
- Proofs by cases are encoded by associating `nesting` information to each proof step.
- Each proof consists of steps of the following types: `Assumption`, `MP`, `FirstCase`, `SecondCase`, `QEDbyCases`, `QEDbyAssumption`, `QEDbyEFQ`
- `Contents` corresponds to a disjunction in a proof step, `Goal` is *true* iff `Contents` is the goal...
- There are also global constraints

Example: Constraints for steps QEDbyEFQ

- Each proof step has one of the above sorts and meets some constraints
- For instance, if the step s is of the kind QEDbyEFQ, then the following conditions hold:

1. $\mathrm{StepKind}(s) = \mathrm{QEDbyEFQ}$
2. $s > 0$
3. $\mathrm{contents}(s-1)(0) = \bot$
4. step s is the goal
5. $\mathrm{Nesting}(s) = \mathrm{Nesting}(s-1)$

Pipeline

1. A maximal proof length M is given.
2. Proof steps and the constraints are encoded by natural numbers.
3. A constraint solver (for linear arithmetic, for instance), is invoked to find a model.
4. There is a proof of length $\leq M$ iff there is a model for the constraints.
5. If there is a model, then a proof can be reconstructed from it.
6. A proof for a proof assistant, a readable proof, an illustrated proof then can be constructed from the proof.

Completing incomplete premises: Abduction (1/5)

Given a theory $\mathcal{T}$ and a conjecture G, assuming that $\mathcal{T} \not\models G$ and $\mathcal{T} \not\models \neg G$, the objective is to find a set of atomic formulae F, such that it holds:

- $\mathcal{T}, F \vdash G$
- the set $\{\mathcal{T}, F\}$ is consistent

The formulas in F are called the abducts.

Completing incomplete premises: Abduction (2/5)

There can be additional conditions. In Larus (an abduct makes the step i):

1. StepKind(i) =Assumption
2. Nesting(i) $= 1$
3. Cases(i) $=$ *false*
4. ContentsPredicate($i, 0$) $<$ *sizeof*(*Signature*)
5. for each argument j (up to maximal arity): ContentsArgument($i, 0, j$) $<$ *sizeof*(*Constants*)
6. Goal(i) $=$ *false*
7. ContentsPredicate(i,0) $\neq \perp$

1. ax0 : $\forall X\ (p(X) \Rightarrow q(X))$
2. ax1 : $\forall X\ (q(X) \Rightarrow r(X) \vee s(X))$
3. ax2 : $\forall X\ (r(X) \Rightarrow \bot)$

Conjecture: $\forall X\ (s(X))$

The conjecture cannot be proved, but Larus offers two abducts:

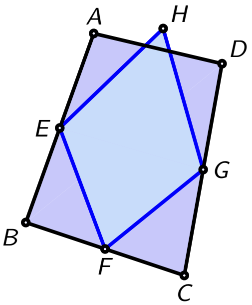
```
1. set:
  1. ((q(b)))
Abducts CONSISTENT!
```

Conjecture: $\forall X\ (\ q(X) \Rightarrow\ s(X))$

```
2. set:
  1. ((p(b)))
Abducts CONSISTENT!
```

Conjecture: $\forall X\ (\ p(X) \Rightarrow\ s(X))$

Completing incomplete premises: Abduction (4/5)



Let ABCD be a quadrilateral. Let E, F, G the midpoints of AB, BC et CD respectively. Let H be a point.
Under which assumption the quadrilateral EFGH is a parallelogram ?

$\longrightarrow$ H should be the midpoint of segment AD.

Consider arbitrary a, b, c, d, e, f, g, h such that:

- $\neg col(b, d, a)$,
- $\neg col(b, d, c)$,
- $\neg col(a, c, b)$,
- $\neg col(a, c, d)$,
- $\neg col(e, f, g)$,
- $b \neq d$,
- $a \neq c$,
- $midpoint(a, e, b)$,
- $midpoint(b, f, c)$,
- $midpoint(c, g, d)$.

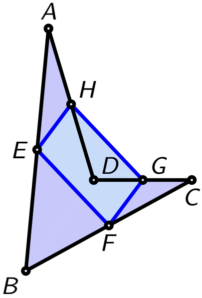
It should be proved that $pG(e, f, g, h)$.

Abducts found:

- $midpoint(d, h, a)$

1. $par(a, c, e, f)$ (by MP, from $\neg col(a, c, b)$, $midpoint(b, f, c)$, $midpoint(a, e, b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto b$, $P \mapsto f$, $Q \mapsto e$)

...



Let ABCD be a quadrilateral. Let E, F, G, H be the midpoints of the segments [AB], [BC], [CD] and [DA] respectively.
What can we say about EFGH ?

$\longrightarrow$ EFGH is a parallelogram.

Completing incomplete goals: Deducts (2/2)

It should be proved that $\_(e, f, g, h)$.

2. $par(a, c, e, f)$ (by MP, from $\neg col(a, c, b)$, $midpoint(b, f, c)$, $midpoint(a, e, b)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto b$, $P \mapsto f$, $Q \mapsto e$)
3. $par(a, c, h, g)$ (by MP, from $\neg col(a, c, d)$, $midpoint(c, g, d)$, $midpoint(a, h, d)$ using axiom triangle_mid_par_strict; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto d$, $P \mapsto g$, $Q \mapsto h$)
4. $par(e, f, g, h)$ (by MP, from $par(a, c, e, f)$, $par(a, c, h, g)$, $\neg col(e, f, g)$ using axiom lemma_par_trans; instantiation: $A \mapsto e$, $B \mapsto f$, $C \mapsto a$, $D \mapsto c$, $E \mapsto g$, $F \mapsto h$)
5. Proved by assumption! (by QEDas)

Completing incomplete proofs: Hints (1/3)

9.3 Lemma. $BaAc \wedge m\in A \wedge Mamc \wedge r\in A \to \forall b[a \underset{r}{\approx} b \to BbAc]$.

(Wenn a und c auf entgegengesetzten Seiten der Geraden A liegen, und zwar spiegelbildlich bezüglich eines Punktes von A, und r auf A liegt, so liegt jeder Punkt b der Halbgeraden H(ra) entgegengesetzt zu c bezüglich A, Abb. 33.)

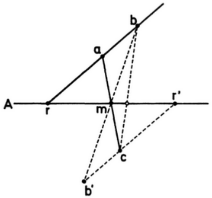


Abb. 33

Beweis: Sei $a \underset{r}{\approx} b$. Nach Def. 6.1(ii) ist $Brba \vee Brab$.

Completing incomplete proofs: Hints (2/3)

- *Hint* can be found in an informal proof (for instance, in a textbook), from machine verifiable proof, or from memory!
- For a proof or a proof step, hint can specify:
  - the predicate symbol
  - arguments in the atomic formula
  - the ordinal of a proof step
  - the axiom applied in the step
  - ...
- In other provers, such hints are extremely difficult to use
- In some cases, hints can lead to significant speed-ups

Completing incomplete proofs: Hints (3/3)

- Using this approach, the user can add constraints either to help the prover or to find a specific proof.
- Examples:
  - predicate r must appear somewhere in the proof:
    `fof(hintname0, hint, r(?,?), _, _)`
  - ax2 must be used in the proof at step 3, instantiating both arguments with the same value
    `fof(hintname0, hint, _, 3, ax2(A,A))`

Future work

- Many generated abducts/deducts are ,,uninteresting“ or mutually similar
- There are different restrictions in abduction considered in the literature and we will consider different criteria for filtering out ,,interesting“ abducts/deducts (for instance, minimal in some sense)

Conclusions

- We have shown that we can extend a prover, which uses constraint solving, so that it can complete:
  - partially specified hypotheses
  - partially specified conclusions
  - partially specified proofs
- All three tasks fit naturally into *proving as constraint solving* paradigm: it is only that some constraints are added or deleted
- To our knowledge, this approach is new, and we are not aware of any other systems that tackle these three completion problems.