Theorem Proving as Constraint Solving with Coherent Logic

Predrag Janičić
University of Belgrade, Serbia

Julien Narboux
University of Strasbourg, France

Dagstuhl Seminar 21472: "Geometric Logic, Constructivisation, and Automated Theorem Proving", November 21-27, 2021.

Coherent Logic / Finitary Geometric Implications

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

where universal closure is assumed, A_i denotes an atomic formula, and B_j denotes a conjunction of atomic formulae.

Inference System for Coherent Logic

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

where ax is

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

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

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

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

Starting Ideas

- The pure forward chaining approach to ATP does not take the goal into account.
- SAT/SMT solvers have been progressing a lot in the recent years.
- Encoding the problem of finding a Coherent Logic proof into SAT/SMT theories can restore 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*:
  - the search is performed globally over a set of possible proofs;
  - a proof of a given formula can be represented by a sequence of natural numbers, meeting some constraints;
  - a proof is found by a solver that finds a sequence that meets these conditions.

Related work

Surprisingly (as far as we know) this approach has not been studied extensively. Only:

- 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.

Inference System for Coherent Logic: Example

Consider the following set of axioms:
ax1: $\forall x\ (p(x) \Rightarrow r(x) \vee q(x))$
ax2: $\forall x\ (q(x) \Rightarrow \bot)$

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

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

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 $r(a), q(a)$)

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;*/
```

Constraints

- We add constraints expressing that a sequence of integers represents a valid proof.
- The proofs by cases are encoded by associating nesting information to each proof step.
- The absence of function symbols in CL allows a trivial encoding of matching of axiom arguments (no need to encode the unification problem).

From CL to CL2 and Back

- For convenience, we consider CL2, all formuae are of the form:

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

  where
  - $m = 1$ or $m = 2$;
  - each formula B_i consists of only one conjunct.
- The axioms and the conjecture can be relatively simply translated from CL to CL2 (by introducing new predicate symbols)
- The proof obtained for CL2 can be simply transformed to a proof over the original CL language
- Each proof has a form:
  $\text{Proof} \ ::= \ \text{As}^* \ \text{MP}^* \ \left(\text{QEDcs}\left(\text{Proof}^2\right) \ \mid \text{QEDas} \mid \text{QEDefq}\ \right)$

Overview

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 can be constructed.

Optimizations

1. Symmetry breaking (for instance, if step $s+1$ does not use step the result of step s, then we order the two steps by the lexicographic order (number of premises, number of the lemma used)).
2. Memoization (use variables instead of duplicating constraints).

CL: a good framework for obtaining readable proofs

Gentzen: "I wanted to set up a formalism that comes as close as possible to actual reasoning"

In CL:

- no need for normalization to clausal form.
- a better level of granularity compared to natural deduction.

Extension: inline lemmas

1. "ABCD is a rectangle because ABCD is square"
2. "ABC are collinear because BAC are collinear"
3. "ABCD is a parallelogram because BCDA is a parallelogram"

Should these proof steps be implicit ?

It depends on the context: in high-school 1) should be explicit, 3) is equivalent to the parallel postulate.

In Larus an option is available to consider all lemmas with at most one assumption as implicit.

Extension: inline lemmas

1. "ABCD is a rectangle because ABCD is square"
2. "ABC are collinear because BAC are collinear"
3. "ABCD is a parallelogram because BCDA is a parallelogram"

Should these proof steps be implicit ?
It depends on the context: in high-school 1) should be explicit, 3) is equivalent to the parallel postulate.
In Larus an option is available to consider all lemmas with at most one assumption as implicit.

Extension: inline lemmas

1. "ABCD is a rectangle because ABCD is square"
2. "ABC are collinear because BAC are collinear"
3. "ABCD is a parallelogram because BCDA is a parallelogram"

Should these proof steps be implicit ?
It depends on the context: in high-school 1) should be explicit, 3) is equivalent to the parallel postulate.
In Larus an option is available to consider all lemmas with at most one assumption as implicit.

Example

Euclid Book I, Proposition 4: Side-Angle-Side
Euclid Book I, Proposition 5: In isosceles triangles the angles at the base equal one another. Or, in formal terms:

$$\forall\, A, B, C\ (isosceles(A, B, C) \Rightarrow congA(A, B, C, A, C, B)\ )$$

Output example

Consider arbitrary a, b, c such that: $isosceles(a, b, c)$. It should be proved that $congA(a, b, c, a, c, b)$.

1. $col(c, a, b) \vee \neg col(c, a, b)$ (by MP, using axiom cn_col1b; instantiation: $A \mapsto c$, $B \mapsto a$, $C \mapsto b$)
2. Case $col(c, a, b)$:
3. $\perp$ (by MP, from $col(c, a, b)$, $isosceles(a, b, c)$ using axiom nnncolNegElim; instantiation: $A \mapsto a$, $B \mapsto b$, $C \mapsto c$)
4. Contradiction! (by QEDefq)
5. Case $\neg col(c, a, b)$:
6. $congA(a, b, c, a, c, b)$ (by MP, from $isosceles(a, b, c)$, $isosceles(a, b, c)$, $\neg col(c, a, b)$ using axiom proposition_04; instantiation: $A \mapsto a$, $B \mapsto c$, $C \mapsto b$, $Xa \mapsto a$, $Xb \mapsto b$, $Xc \mapsto c$)
7. Proved by assumption! (by QEDas)
8. Proved by case split! (by QEDcs, by $col(c, a, b), \neg col(c, a, b)$)

Implementation

The C++ implementation — Larus — is available at:
https://github.com/janicicpredrag/Larus

Experimental Results

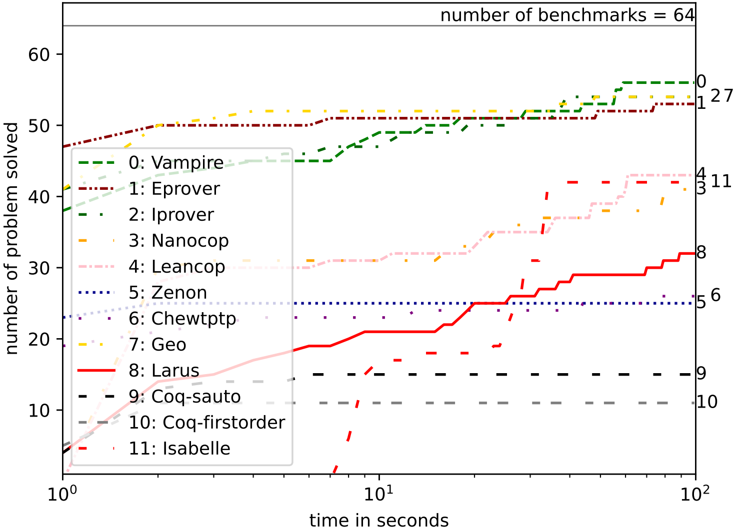
We experimented with our implementation of the approach using four sets of problems:

1. a corpus of 64 problems coming from the CL community
2. a corpus of 234 problems coming from the formalization of Book 1 of Euclid's Elements
3. some crafted examples
4. a corpus of lemma about pseudo transitivity of collinearity

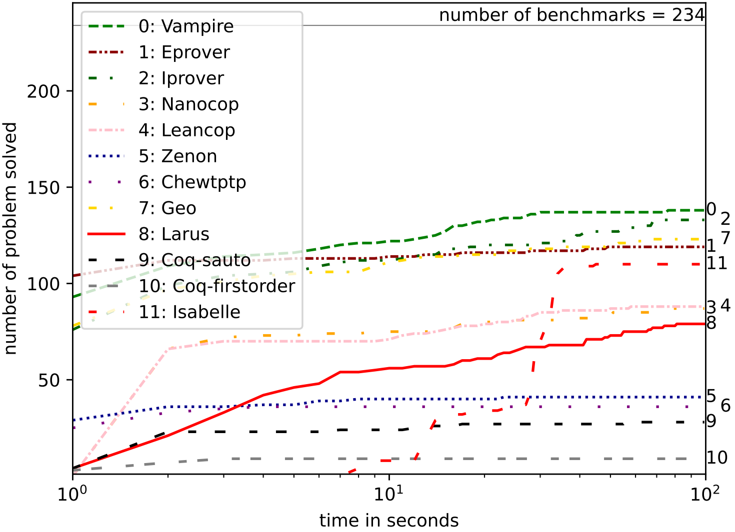
and compared to:

- State of the art provers: Vampire, Eprover, Iprover
- Small (prolog based) provers: LeanCop, NanoCop
- Provers generating Coq proofs or Coq's tactics: Zenon, Coq-firstorder, Coq-sauto
- A Coherent Logic prover: Geo
- Similar approach for FOL: ChewTPTP
- Portfolio approach: Isabelle SledgeHammer (without proof reconstruction)

Experimental Results (CL benches)



Experimental Results (Euclid)



Proof Hints

- 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))`
- Hints could be used within proof assistant for turning proof sketches into formal proofs.

- Using tactics rather than λ-terms.
- Using tactics we can mimic CL rules, preserve readability and maintainability.
- The generated Coq's proof is in declarative style.
- Inlined lemmas are kept implicit.

Example

Coq output (without inline lemmas)

```
Theorem proposition_05 : forall A B C : MyT, isosceles A B C -> congA A B C A C
     B.
Proof.
intros a b c.
intros.
assert (cong a b a c) by applying (defisosceles a b c).
assert (triangle a b c) by applying (defisosceles a b c).
assert (col b a c \/ ~ col b a c) by applying (cn_col1b b a c).
by cases on ( ( col b a c ) \/ ( ~ col b a c ) ).
- {
   assert (~ col a b c) by applying (deftriangle a b c).
   assert (col a b c) by applying (lemma_collinearorder b a c).
   assert (False) by contradiction_on (col a b c).
   contradict.
  }
- {
   assert (cong a c a b) by applying (lemma_congruencesymmetric a a b c).
   assert (congA b a c c a b) by applying (lemma_ABCequalsCBA b a c).
   assert (congA a b c a c b) by applying (proposition_04 a b c a c b).
   conclude.
  }
Qed.
```

Classical vs. Intuitionistic Logic

While, for instance, in the resolution method, classical logic and reasoning are deeply built-in, in the presented approach and when using the prover, one can easily choose whether to use excluded middle or not (and, hence, choose between classical and intuitionistic setting), it is just a matter of adding axioms.

Potential extensions/Future work

- Test the prover on problems which are not in CL form and study the impact of anti-skolemization and geometrization.
- Improving the encoding used (e.g., by using some form of incremental encoding, or some meta-theorems);
- Improving the solving process (e.g., by using some other SAT/SMT solvers, or by instructing SAT/SMT solvers to take into account some specifics of input instances);
- Paralellisation

Conclusions

- Larus can generate readable and machine checkable proofs and use proof hints.
- Larus can not compete with state of the art provers, but can compete with others provers generating Coq's proofs.
- The approach could be tried for other logics.