A Coq Library for Mechanised First-Order Logic

Dominik Kirst, Johannes Hostert, Andrej Dudenhefner, Yannick Forster, Marc Hermes, Mark Koch, Dominique Larchey-Wendling*, Niklas Mück, Benjamin Peters, Gert Smolka, Dominik Wehr

Saarland University, Saarland Informatics Campus, Germany
*Université de Lorraine, CNRS, LORIA, Vandœuvre-lès-Nancy, France

The Coq Workshop'22, Haifa, Israel, August 12th

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Loria

cnrs

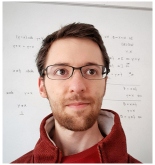
Contributors

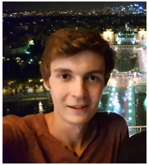

Dominik Kirst


Johannes Hostert


Andrej Dudenhefner


Yannick Forster


Marc Hermes

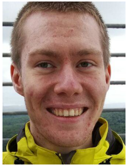

Mark Koch


Dominique Larchey-Wendling


Niklas Mück


Benjamin Peters


Gert Smolka


Dominik Wehr

Background

- Merge of several developments concerned with first-order logic
- Published at several venues (CPP, ITP, IJCAR, LFCS, FSCD, TYPES, JAR, JLC, LMCS)
- Design of a core framework general enough to accommodate all results
- Import of developments based on earlier versions of the framework
- Developed in a fork of the Coq library of undecidability proofs (Forster et al. (2020))

`https://github.com/dominik-kirst/coq-library-undecidability/tree/fol-library/theories/FOL`

Framework

Emerged over several projects with ideas from various contributors:

- Deep embedding of syntax, deduction systems, and semantics
- Combination of well-known techniques, most notably de Bruijn indices
- Tool support for easy interaction by external users

> Took most inspiration from O'Connor (2009); Ilik (2010); Herbelin and Lee (2014); Han and van Doorn (2020); Laurent (2021)

Framework: Syntax

Terms and formulas are represented as inductive types $\mathfrak{T}$ and $\mathfrak{F}$ over a signature $\Sigma = (\mathcal{F}_\Sigma, \mathcal{P}_\Sigma)$:

$$t : \mathfrak{T} \ ::= \ x_n \mid f\,\vec{t} \qquad\qquad (n : \mathbb{N}, f : \mathcal{F}_\Sigma, \vec{t} : \mathfrak{T}^{|f|})$$

$$\varphi, \psi : \mathfrak{F} \ ::= \ \bot \mid P\,\vec{t} \mid \varphi \to \psi \mid \varphi \land \psi \mid \varphi \lor \psi \mid \forall \varphi \mid \exists \varphi \qquad\qquad (P : \mathcal{P}_\Sigma, \vec{t} : \mathfrak{T}^{|P|})$$

- Syntax modular in type classes for binary connectives and quantifiers
- Common instances $(\to, \forall)$ and $(\to, \land, \lor, \forall, \exists)$ provided
- Availability of $\bot$ regulated via type class flag
- De Bruijn indices encode the number of quantifiers shadowing their relevant binder
- Capture-avoiding instantiation $t[\sigma]$ and $\varphi[\sigma]$ for parallel substitutions $\sigma : \mathbb{N} \to \mathfrak{T}$

Framework: Syntax (Coq)

```
Context {sig_funcs : funcs_signature}.

Inductive term  : Type :=
  | var : nat -> term
  | func : forall (f : syms), vec term (ar_syms f) -> term.

Context {sig_preds : preds_signature}.

Inductive falsity_flag := falsity_off | falsity_on.
Existing Class falsity_flag.

Class operators := {binop : Type ; quantop : Type}.
Context {ops : operators}.

Inductive form : falsity_flag -> Type :=
  | falsity : form falsity_on
  | atom {b} : forall (P : preds), vec term (ar_preds P) -> form b
  | bin {b} : binop -> form b -> form b -> form b
  | quant {b} : quantop -> form b -> form b.
```

Framework: Deduction Systems

Proof rules are represented as inductive predicates relating a context Γ to a formula φ:

$$\dots$$

$$\frac{\Gamma[\uparrow] \vdash \varphi}{\Gamma \vdash \forall\varphi}\ \mathsf{AI} \qquad \frac{\Gamma \vdash \forall\varphi}{\Gamma \vdash \varphi[t]}\ \mathsf{AE} \qquad \frac{\Gamma \vdash \varphi[t]}{\Gamma \vdash \exists\varphi}\ \mathsf{EI} \qquad \frac{\Gamma \vdash \exists\varphi \quad \Gamma[\uparrow], \varphi \vdash \psi[\uparrow]}{\Gamma \vdash \psi}\ \mathsf{EE}$$

$$\dots$$

- Quantifier rules use shifted contexts $\Gamma[\uparrow]$ so that x_0 acts as canonical free variable
- Trivialises structural properties like substitutivity and weakening
- Availability of classical rules regulated via type class flag
- Similar representation of sequent calculi and other systems

Framework: Deduction Systems (Coq)

```
Context {sig_funcs : funcs_signature}.
Context {sig_preds : preds_signature}.

Reserved Notation 'A ⊢ phi' (at level 61).

Inductive peirce := class | intu.
Existing Class peirce.

Inductive prv : forall (ff : falsity_flag) (p : peirce), list form -> form -> Prop :=
  | II {ff} {p} A phi psi : phi::A ⊢ psi -> A ⊢ phi --> psi
  | IE {ff} {p} A phi psi : A ⊢ phi --> psi -> A ⊢ phi -> A ⊢ psi
  | AllI {ff} {p} A phi : map (subst_form ↑) A ⊢ phi -> A ⊢ ∀ phi
  | AllE {ff} {p} A t phi : A ⊢ ∀ phi -> A ⊢ phi[t..]
  | Exp {p} A phi : prv p A falsity -> prv p A phi
  | Ctx {ff} {p} A phi : phi el A -> A ⊢ phi
  | Pc {ff} A phi psi : prv class A (((phi --> psi) --> phi) --> phi)
  where 'A ⊢ phi' := (prv _ A phi).
```

Framework: Semantics

Tarski models $\mathcal{M}$ are represented as a domain type D and symbol interpretations:

$$f^{\mathcal{M}} \; : \; D^{|f|} \to D \qquad\qquad P^{\mathcal{M}} \; : \; D^{|P|} \to \mathbb{P}$$

- Interpretation of terms and formulas based on assignments $\rho : \mathbb{N} \to D$
- Term evaluation $\hat{\rho}\, t$ defined recursively, main rule $\hat{\rho}\,(f\ \vec{t}\,) \; := \; f^{\mathcal{M}}\,(\hat{\rho}\ \vec{t}\,)$
- Formula satisfaction $\rho \vDash \varphi$ defined recursively, main rule $\rho \vDash P\ \vec{t} \; := \; P^{\mathcal{M}}\,(\hat{\rho}\ \vec{t}\,)$
- Induces the logical entailment relation $\Gamma \vDash \varphi$

Framework: Semantics (Coq)

```
Context {domain : Type}.

Class interp := B_I
  { i_func : forall f : syms, vec domain (ar_syms f) -> domain ;
    i_atom : forall P : preds, vec domain (ar_preds P) -> Prop ; }.

Definition env := nat -> domain.

Context {I : interp}.

Fixpoint eval (rho : env) (t : term) : domain := match t with
  | var s => rho s
  | func f v => i_func (Vector.map (eval rho) v) end.

Fixpoint sat {ff : falsity_flag} (rho : env) (phi : form) : Prop := match phi with
  | atom P v => i_atom (Vector.map (eval rho) v)
  | falsity => False
  | bin Impl phi psi => sat rho phi -> sat rho psi
  | quant All phi => forall d : domain, sat (d .: rho) phi end.
```

Framework: Axiom Systems

Concrete axiom systems $\mathcal{A}$ are modelled as predicates of formulas over a specific signature.

For the example of Peano arithmetic (PA), we instantiate to the arithmetical signature

$$(O,\, S\,\_\,,\ \_ + \_\,,\ \_ \times \_\,;\ \_ \equiv \_)$$

and collect the usual axioms, with the induction scheme represented as all instances of

$$\varphi[O] \to (\forall x.\, \varphi[x] \to \varphi[S\,x]) \to \forall x.\, \varphi[x].$$

- Include fragments of PA like Robinson's Q, also several variants of ZF set theory
- Equality $\equiv$ seen as axiomatised symbol of the signature rather than a logical primitive
- Axiom systems $\mathcal{A}$ induce relatives deductive and semantic theories $\mathcal{A} \vdash \varphi$ and $\mathcal{A} \vDash \varphi$

Framework: Tool Support

Tools presented at last year's Coq Workshop (Hostert et al. (2021)):

- HOAS-input language
  - Concrete formulas can be written with Coq binders instead of de Bruijn indices
  - Eases interaction with the syntax
- Proof mode (inspired by Iris proof mode, Krebbers et al. (2017))
  - Tactic and notation layer hiding the proof rules
  - Eases interaction with the deduction systems
- Reification tactic (employing MetaCoq, Sozeau et al. (2020))
  - Extracts first-order formulas from Coq predicates
  - Eases interaction with the semantics

Framework: Tool Support (Proof Mode)

```
    frewrite (ax_add_zero y).
    fapply ax_refl.
  - fintros "x" "IH" "y".
    frewrite (ax_add_rec (σ y) x).
    frewrite ("IH" y).
    frewrite (ax_add_rec y x). fapply ax_refl.
Qed.

Lemma add_comm :
  FAI ⊢ << ∀' x y, x ⊕ y == y ⊕ x.
Proof.
  fstart. fapply ((ax_induction (<< Free x, ∀' y, x ⊕ y == y ⊕ x))).
  - fintros.
    frewrite (ax_add_zero x).
    frewrite (add_zero_r x).
    fapply ax_refl.
  - fintros "x" "IH" "y".
    frewrite (add_succ_r y x).
    frewrite <- ("IH" y).
    frewrite (ax_add_rec y x).
    fapply ax_refl.
Qed.

Lemma pa_eq_dec :
  FAI ⊢ << ∀' x y, (x == y) ∨ ¬ (x == y).
Proof.
  fstart.
  fapply ((ax_induction (<< Free x, ∀' y, (x == y) ∨ ¬ (x == y)))).
```

```
1 goal
p : peirce
x, y : term
______________________________________(1/1)
 FAI
 "IH" : ∀  x0, x`[↑] ⊕  x0 ==  x0 ⊕ x`[↑]
______________________________________
 (σ x ⊕ y == y ⊕ σ x)
```

Messages ↗ Errors ↗ Jobs ↗

https://github.com/dominik-kirst/coq-library-undecidability/blob/fol-library/theories/FOL/Proofmode/DemoPA.v

Framework: Tool Support (Reification Tactic)

```
Proof.
elim a using PA_induction.
- represent.
- eapply ieq_trans. 1:apply (add_zero_l (iS b)).
  apply ieq_congr_succ, ieq_sym, add_zero_l.
- intros d IH.
  eapply ieq_trans. 1:apply (add_succ_l d (iS b)).
  apply ieq_congr_succ. eapply ieq_trans.
  + apply IH.
  + apply ieq_sym, add_succ_l.
Qed.

Lemma add_comm a b : a i⊕ b i= b i⊕ a.
Proof.
elim a using PA_induction.
- represent.
- eapply ieq_trans.
  + apply (add_zero_l b).
  + apply ieq_sym, (add_zero_r b).
- intros a' IH.
  eapply ieq_trans. 2:eapply ieq_trans.
  + apply (add_succ_l a' b).
  + apply ieq_congr_succ, IH.
  + apply ieq_sym, add_succ_r.
Qed.
```

```
1 goal
D' : Type
I : interp D'
D_fulfills : forall (f : form) (rho : env D'),
             PAeq f -> rho ⊨ f
a, b : D'
______________________________________(1/1)
representableP 1 (fun a0 : D => a0 i⊕ b i= b i⊕ a0)
```

Messages | Errors | Jobs

https://github.com/dominik-kirst/coq-library-undecidability/blob/fol-library/theories/FOL/Reification/DemoPA.v

Framework: Evolution

Forster, Kirst, Smolka (2019) at CPP'19:

- Concrete signature, small logical fragment, named variables
- Among the initial projects constituting the undecidability library

Forster, Kirst, and Wehr (2021) at LFCS'20/JLC'21:

- Arbitrary signature, both logical fragments, de Bruijn encoding
- Use of Autosubst 2 (Stark et al. (2019)) for de Bruijn boilerplate

Kirst and Larchey-Wendling (2020) at IJCAR'20/LMCS'22:

- Parametric in logical fragment, merged into undecidability library
- Refrains from Autosubst 2 mostly due to dependency on function extensionality

Kirst and Hermes (2021) at ITP'21/JAR'22:

- Compromise of previous developments, merged into undecidability library
- Still no explicit code generation with Autosubst 2 but identical design

Framework: Comparison

Development	Signature	Binding	(AI)-Rule	Weakening
O'Connor	arbitrary	named	side-condition	n.a.
Ilik	monadic	locally-nameless	co-finite	easy
Herbelin et al.	dyadic	locally-named	side-condition	needs renaming
Han and van Doorn	arbitrary	de Bruijn	shifting	easy
Laurent	full	anti-loc.-namel.	shifting	easy
Our framework	arbitrary	de Bruijn	shifting	easy

Content

Overview:

- Many metamathematical results: completeness, undecidability, incompleteness
- Many interdependencies, based on the Coq library of undecidability proofs
- Many possible projects/collaborations: syntactic cut-elimination, Hilbert systems, Löwenheim-Skolem theorems, resolution, tableaux, constructible hierarchy, ...

Shared methods:

- Constructive meta-theory where possible
- Synthetic approach to computability results

Content: Completeness

In which situations does $\Gamma \vDash \varphi$ imply $\Gamma \vdash \varphi$?

Based on the publication Forster et al. (2021):

- Constructively extremely subtle topic, extensive related literature
- Model-theoretic semantics (Tarski/Kripke) yield connections to MP and LEM
- Fully constructive proofs for algebraic and dialogical semantics

Content: Undecidability

Which decision problems of first-order logic are undecidable?

Library includes all common undecidability results:

- Validity, provability, satisfiability (Forster et al. (2019))
- Finite satisfiability (Kirst and Larchey-Wendling (2020))
- Strongest versions regarding binary signatures (Hostert et al. (2022))
- Several variants of PA and ZF (Kirst and Hermes (2021))
- Post's theorem on the arithmetical hierarchy (Kirst et al. (2022))

Content: Incompleteness

Which axiom systems $\mathcal{A}$ satisfy $\mathcal{A} \vdash \varphi$ or $\mathcal{A} \vdash \neg\varphi$ for all φ?

Library exploiting the connection to undecidability:

- Incompleteness of several variants of PA and ZF (Kirst and Hermes (2021))
- Essential incompleteness of Q (Peters and Kirst (2022))
- Tennenbaum's theorem on computable models of PA (Hermes and Kirst (2022))

Current Status: Overview

- Completed core framework ✓
- Main completeness, undecidability, and incompleteness results imported ✓
- Essential incompleteness, Tennenbaum's theorem, and Post's theorem pending ✓
- Signature transformations and further computability results planned to be imported ✗
- Total: about 25k lines of code (8500 spec, 15500 proofs, 1000 comments), 110 files

Current Status: Structure

mark-koch Fix Proofmode MinZF demo ✓ 2722b67 4 days ago History

..		
Arithmetics	Rename Deduction -> ND to prepare for Sequent	2 months ago
Completeness2	Start using Asimpl in places	5 days ago
Deduction	Finish Kripke Completeness, work on atom substitution	10 days ago
Incompleteness	Tarski Constructions ported	2 months ago
Proofmode	Fix Proofmode MinZF demo	4 days ago
Reification	Tarski Constructions ported	2 months ago
Semantics	Add validity facts, remove "not _strong" preservation	10 days ago
Sets	Rename Deduction -> ND to prepare for Sequent	2 months ago
Syntax	Start using Asimpl in places	5 days ago
Undecidability	Start using Asimpl in places	5 days ago
Utils	Port FOLP reduction, refactor PCP decidabilities	2 months ago
FragmentSyntax.v	Start using Asimpl in places	5 days ago
FullSyntax.v	Start using Asimpl in places	5 days ago

Current Status: Pending Contributions

Filters ▾ | is:pr is:open | Labels 9 | Milestones 0 | **New pull request**

2 Open ✓ 1 Closed — Author ▾ Label ▾ Projects ▾ Milestones ▾ Reviews ▾ Assignee ▾ Sort ▾

WIP: Add further incompleteness results
#4 opened 4 days ago by bn-peters • Draft 4 tasks

Add PrenexNormalForm and ArithmeticalHierarchy
#3 opened 5 days ago by SohnyBohny

Filters ▾ | is:issue is:open | Labels 9 | Milestones 0 | **New issue**

1 Open ✓ 0 Closed — Author ▾ Label ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

Proofmode bugs
#2 opened 8 days ago by bn-peters

Current Status: Activity

History for **coq-library-undecidability** / **theories** / **FOL**

Commits on Jul 22, 2022

Fix Proofmode MinZF demo
mark-koch committed 4 days ago ✓ — 2722b67

Commits on Jul 21, 2022

Remove Require in section
JoJoDeveloping committed 5 days ago ✓ — Verified — f55bae2

Start using Asimpl in places
JoJoDeveloping committed 5 days ago ✓ — Verified — 840596b

Merge branch 'fol-library' of github.com:dominik-kirst/coq-library-un... ...
mark-koch committed 5 days ago ✓ — 89d4850

Fix ProofMode
mark-koch committed 5 days ago — 6d9e47b

Commits on Jul 20, 2022

Working and somewhat efficient ASimpl tactic
JoJoDeveloping committed 6 days ago ✓ — Verified — 2922b59

Future Plans

1. Finish importing the remaining developments
2. Possible round of refactoring (proof mode performance, falsity flags)
3. Decide on a plan how to integrate with the undecidability library
4. Follow the release cycle of the undecidability library, itself following Coq
5. Possible timeline: opam package for Coq 8.16, add to Coq CI for Coq 8.17
6. At any time: help new users get started and contribute their developments!

Thanks for listening!

Bibliography I

Forster, Y., Kirst, D., and Smolka, G. (2019). On synthetic undecidability in Coq, with an application to the Entscheidungsproblem. In *8th International Conference on Certified Programs and Proofs*.

Forster, Y., Kirst, D., and Wehr, D. (2021). Completeness theorems for first-order logic analysed in constructive type theory: Extended version. *Journal of Logic and Computation*, 31(1):112–151.

Forster, Y., Larchey-Wendling, D., Dudenhefner, A., Heiter, E., Kirst, D., Kunze, F., Smolka, G., Spies, S., Wehr, D., and Wuttke, M. (2020). A Coq library of undecidable problems. In *CoqPL 2020 The Sixth International Workshop on Coq for Programming Languages*.

Han, J. and van Doorn, F. (2020). A formal proof of the independence of the continuum hypothesis. In *9th International Conference on Certified Programs and Proofs*.

Herbelin, H. and Lee, G. (2014). Formalizing logical meta-theory – semantical cut-elimination using Kripke models for first-order predicate logic.

Hermes, M. and Kirst, D. (2022). An analysis of Tennenbaum's theorem in constructive type theory. In *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*.

Hostert, J., Dudenhefner, A., and Kirst, D. (2022). Undecidability of dyadic first-order logic in Coq. In *13th International Conference on Interactive Theorem Proving (ITP 2022)*.

Hostert, J., Koch, M., and Kirst, D. (2021). A toolbox for mechanised first-order logic. In *The Coq Workshop*, volume 2021.

Bibliography II

Ilik, D. (2010). *Constructive completeness proofs and delimited control*. PhD thesis, Ecole Polytechnique X.

Kirst, D. and Hermes, M. (2021). Synthetic undecidability and incompleteness of first-order axiom systems in Coq. In *12th International Conference on Interactive Theorem Proving (ITP 2021)*.

Kirst, D. and Larchey-Wendling, D. (2020). Trakhtenbrot's theorem in Coq. In *International Joint Conference on Automated Reasoning*. Springer.

Kirst, D., Mück, N., and Forster, Y. (2022). Synthetic versions of the Kleene-Post and Post's theorem. *TYPES 2022*.

Krebbers, R., Timany, A., and Birkedal, L. (2017). Interactive proofs in higher-order concurrent separation logic. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, page 205–217, New York, NY, USA. Association for Computing Machinery.

Laurent, O. (2021). An anti-locally-nameless approach to formalizing quantifiers. In *10th International Conference on Certified Programs and Proofs*.

O'Connor, R. (2009). Incompleteness & completeness: formalizing logic and analysis in type theory. *PhD thesis, Radboud University of Njimegen*.

Peters, B. and Kirst, D. (2022). Strong, synthetic, and computational proofs of Gödel's first incompleteness theorem. *TYPES 2022*.

Bibliography III

Sozeau, M., Anand, A., Boulier, S., Cohen, C., Forster, Y., Kunze, F., Malecha, G., Tabareau, N., and Winterhalter, T. (2020). The MetaCoq Project. *Journal of Automated Reasoning*, 64.

Stark, K., Schäfer, S., and Kaiser, J. (2019). Autosubst 2: reasoning with multi-sorted de Bruijn terms and vector substitutions. In *8th International Conference on Certified Programs and Proofs*.