A Coq Library for Mechanised First-Order Logic

Dominik Kirst[1], Johannes Hostert[1], Andrej Dudenhefner[1], Yannick Forster[1], Marc Hermes[1], Mark Koch[1], Dominique Larchey-Wendling[2], Niklas Mück[1], Benjamin Peters[1], Gert Smolka[1], and Dominik Wehr[1]

[1] Saarland University, Saarland Informatics Campus, Germany
[2] Université de Lorraine, CNRS, LORIA, Vandœuvre-lès-Nancy, France

We report about an ongoing collaborative effort to consolidate several Coq developments concerning metamathematical results in first-order logic [1, 2, 11, 10, 8, 7, 6, 15, 12] into a single library. We first describe the framework regarding the representation of syntax, deduction systems, and semantics as well as its instantiation to axiom systems and tools for user-friendly interaction. Next, we summarise the included results mostly connected to completeness, undecidability, and incompleteness. Finally, we conclude by reporting on challenges experienced and anticipated during the integration. The current status of the project can be tracked in a public fork of the Coq Library of Undecidability Proofs [3].

Framework In principle, we follow ideas and suggestions present in various approaches [14, 9, 5, 4, 13] to the representation of first-order logic in CIC. Over the span of our initial projects we tried out several variants and found the final framework to be most suitable. Notably, a previous version used the Autosubst 2 tool [16] to generate the syntax, which we decided to avoid in later versions due to its use of function extensionality. The final framework, however, still follows the same design principles for binding and substitution.

The *syntax* is represented by inductive types for terms $t : \mathbb{T}$ and formulas $\varphi : \mathbb{F}$ depending on signatures of function symbols f and relation symbols P as well as a collection of binary connectives $\Box$ and quantifiers ∇:

$$t : \mathbb{T} ::= x_n \mid f\,\vec{t} \qquad\qquad \varphi, \psi : \mathbb{F} ::= \bot \mid P\,\vec{t} \mid \varphi \,\Box\, \psi \mid \nabla\varphi \qquad\qquad (n : \mathbb{N})$$

The term vectors $\vec{t}$ are required to have length matching the specified arities $|f|$ and $|P|$ of f and P. Binding is implemented using de Bruijn indices, where a bound variable is encoded as the number of quantifiers shadowing its relevant binder. Capture-avoiding instantiation with parallel substitutions $\sigma : \mathbb{N} \to \mathbb{T}$ is defined both for terms as $t[\sigma]$ and formulas as $\varphi[\sigma]$. The availability of $\bot$ is handled with a flag and instances for the negative $(\to, \forall)$-fragment and the full syntax with $(\to, \land, \lor, \forall, \exists)$ are provided. The parameters controlling the syntax are implemented with type classes so they can be automatically inferred from the context.

Deduction systems are represented by inductive predicates relating finite contexts Γ with derivable formulas. We mostly use natural deduction systems $\Gamma \vdash \varphi$ but also consider several sequent calculi $\Gamma \Rightarrow \varphi$. While the rules for binary connectives are straightforward to specify, the rules for quantifiers crucially exploit the de Bruijn encoding. Using the substitution $\uparrow n := x_{n+1}$, in a shifted context $\Gamma[\uparrow]$ the index 0 generates a canonical fresh variable, for instance allowing for the derivation of $\Gamma \vdash \forall\varphi$ from $\Gamma[\uparrow] \vdash \varphi$ without side conditions. However, more traditional rules for quantifiers employing fresh names can be shown equivalent:

$$\frac{\Gamma[\uparrow] \vdash \varphi}{\Gamma \vdash \forall\varphi}\ \text{de Bruijn rule} \qquad\qquad \frac{\Gamma \vdash \varphi[x_n] \quad n \text{ fresh for } \Gamma, \varphi}{\Gamma \vdash \forall\varphi}\ \text{equivalent named rule}$$

The de Bruijn rules trivialise structural properties like weakening while the named rules simplify concrete derivations. Classical deduction is incorporated by switching on Peirce's law $((\varphi \to \psi) \to \varphi) \to \varphi$ via a flag.

Several flavours of *semantics* are provided, in particular model-theoretic Tarski and Kripke semantics, algebraic semantics based on complete Boolean and Heyting algebras, as well as game-theoretic dialogue semantics. The mostly employed notion of Tarski semantics is obtained by interpreting formulas in types D providing functions $D^{|f|} \to D$ for each f and relations $D^{|P|} \to \mathsf{Prop}$ for each P. Given an environment $\rho : \mathbb{N} \to D$ we define term evaluation $\hat{\rho}\,t$ and formula satisfaction $\rho \vDash \varphi$ recursively, ultimately yielding the semantic consequence relation $\Gamma \vDash \varphi$ expressing that φ holds in all interpretations satisfying Γ.

Concrete *axiom systems* can be defined by instantiating to a suitable signature of function and relation systems and collecting the axioms in a predicate $\mathcal{A} : \mathbb{F} \to \mathsf{Prop}$, inducing the deductive and semantic theories $\mathcal{A} \vdash \varphi$ and $\mathcal{A} \vDash \psi$. Among others, the library includes Peano arithmetic (PA) and ZF set theory.

A preliminary amount of *tool support* has been developed [8] with the goal to ease interaction with the library for external users. Currently supported are a HOAS-input language to hide de Bruijn indices, a reification tactic extracting formulas from Coq predicates, and a proof mode simplifying formal derivations.

Results The library is planned to cover all previous developments, currently spanning ca. 25k lines of code. While these developments contain results of a mostly metamathematical character, the design idea is to have a general purpose library helpful also for external users working on other aspects of first-order logic.

A first family of results is concerned with *completeness*, i.e. statements of the form that $\Gamma \vDash \varphi$ implies $\Gamma \vdash \varphi$ dual to the usually much simpler soundness property. For the premise $\Gamma \vDash \varphi$ we consider all forms of semantics mentioned above and analyse in which situations the conclusion $\Gamma \vdash \varphi$ can be achieved constructively [2].

A second focus is on *undecidability*, i.e. the fact that first-order logic admits no effective decision procedure in various aspects. Concretely, we handle the historic "Entscheidungsproblem" [1, 7], stating that the relations $\Gamma \vDash \varphi$ and $\Gamma \vdash \varphi$ are undecidable, Trakhtenbrot's theorem [11, 7], stating that $\Gamma \vDash \varphi$ is undecidable when restricted to finite models, as well as the undecidability of PA and ZF [10], all based on the synthetic approach to computability underlying the Coq Library of Undecidability Proofs [3].

A third area of results is the closely related notion of *incompleteness*, i.e. the phenomenon that expressive axiomatisations $\mathcal{A}$ necessarily admit sentences φ with neither $\mathcal{A} \vdash \varphi$ nor $\mathcal{A} \vdash \neg\varphi$. In a weak form, for PA and ZF this is a direct consequence of undecidability [10] but we also pursue a more sophisticated argument yielding stronger forms of incompleteness [15]. Another form of incompleteness is included with our analysis of Tennenbaum's theorem [6], stating that the standard model over $\mathbb{N}$ is the only computable model of PA.

Challenges The main challenges we can report on so far are the incorporation of previous developments based on different syntax versions and the question how to integrate with the growing Coq Library of Undecidability Proofs. Regarding the former, especially the developments [2] and [11] are costlier to import and currently only their main results are included. For the latter, it would be conceivable to have one library depend on (parts) of the other or have both libraries depend on a common core framework.

References

[1] Yannick Forster, Dominik Kirst, and Gert Smolka. On synthetic undecidability in Coq, with an application to the Entscheidungsproblem. In *8th International Conference on Certified Programs and Proofs*, 2019.

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

[3] Yannick Forster, Dominique Larchey-Wendling, Andrej Dudenhefner, Edith Heiter, Dominik Kirst, Fabian Kunze, Gert Smolka, Simon Spies, Dominik Wehr, and Maximilian Wuttke. A Coq library of undecidable problems. In *CoqPL 2020 The Sixth International Workshop on Coq for Programming Languages*, 2020.

[4] Jesse Han and Floris van Doorn. A formal proof of the independence of the continuum hypothesis. In *9th International Conference on Certified Programs and Proofs*, 2020.

[5] Hugo Herbelin and Gyesik Lee. Formalizing logical meta-theory – semantical cut-elimination using Kripke models for first-order predicate logic. 2014.

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

[7] Johannes Hoster, Andrej Dudenhefner, and Dominik Kirst. Undecidability of dyadic first-order logic in Coq. In *13th International Conference on Interactive Theorem Proving (ITP 2022)*, 2022.

[8] Johannes Hostert, Mark Koch, and Dominik Kirst. A toolbox for mechanised first-order logic. In *The Coq Workshop*, volume 2021, 2021.

[9] Danko Ilik. *Constructive completeness proofs and delimited control.* PhD thesis, Ecole Polytechnique X, 2010.

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

[11] Dominik Kirst and Dominique Larchey-Wendling. Trakhtenbrot's theorem in Coq. In *International Joint Conference on Automated Reasoning*. Springer, 2020.

[12] Dominik Kirst, Niklas Mück, and Yannick Forster. Synthetic versions of the Kleene-Post and Post's theorem. *TYPES 2022*, 2022.

[13] Olivier Laurent. An anti-locally-nameless approach to formalizing quantifiers. In *10th International Conference on Certified Programs and Proofs*, 2021.

[14] Russell O'Connor. Incompleteness & completeness: formalizing logic and analysis in type theory. *PhD thesis, Radboud University of Njimegen*, 2009.

[15] Benjamin Peters and Dominik Kirst. Strong, synthetic, and computational proofs of Gödel's first incompleteness theorem. *TYPES 2022*, 2022.

[16] Kathrin Stark, Steven Schäfer, and Jonas Kaiser. Autosubst 2: reasoning with multi-sorted de Bruijn terms and vector substitutions. In *8th International Conference on Certified Programs and Proofs*, 2019.