A Coq Library for Mechanised First-Order Logic

Dominik Kirst¹, Johannes Hostert¹, Andrej Dudenhefner¹, Yannick Forster¹, Marc Hermes¹, Mark Koch¹, Dominique Larchey-Wendling², Niklas Mück¹, Benjamin Peters¹, Gert Smolka¹, and Dominik Wehr¹

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 \( t : T \) and formulas \( \varphi : F \) depending on signatures of function symbols \( f \) and relation symbols \( P \) as well as a collection of binary connectives \( \square \) and quantifiers \( \nabla \):

\[
t : T ::= x_n \mid f \vec{t} \quad \varphi, \psi : F ::= \bot \mid P \vec{t} \mid \varphi \square \psi \mid \nabla \varphi \quad (n : \mathbb{N})
\]

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 \( \sigma : \mathbb{N} \to T \) 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 \((\rightarrow, \forall)\)-fragment and the full syntax with \((\rightarrow, \land, \lor, \forall, \exists)\) 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 \( \Gamma \) 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:

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

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 \( ((\varphi \rightarrow \psi) \rightarrow \varphi) \rightarrow \varphi \) 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 \( D^{[P]} \to \text{Prop} \) for each \( P \). Given an environment \( \rho : \mathbb{N} \to D \) we define term evaluation \( \rho t \) and formula satisfaction \( \rho \models \varphi \) recursively, ultimately yielding the semantic consequence relation \( \Gamma \models \varphi \) expressing that \( \varphi \) holds in all interpretations satisfying \( \Gamma \).

Concrete axiom systems can be defined by instantiating to a suitable signature of function and relation symbols and collecting the axioms in a predicate \( \mathcal{A} : \mathcal{F} \to \text{Prop} \), inducing the deductive and semantic theories \( \mathcal{A} \vdash \varphi \) and \( \mathcal{A} \models \psi \). 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.
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 $\varphi$ 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.

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


