Categorical Logic in Lean

Jacob Neumann[1]

University of Nottingham
jacob.neumann@nottingham.ac.uk

Categorical Logic is a branch of mathematical logic which uses the concepts and tools of category theory to investigate logical systems and deductive calculi, following in the example of Lawvere's pioneering work on functorial semantics for algebraic theories[Law63]. In this talk, we'll provide a progress report on a formalization of categorical logic in the Lean proof assistant [dMKA+15][1]. Lean is an interactive theorem prover and dependently-typed functional programming language, based on the Calculus of Inductive Constructions. Proofs in Lean are done using *proof tactics*, making use of Lean's powerful and flexible `tactic` monad. In Lean, we can define new tactics – allowing for abstraction and reuse of common reasoning patterns–, and also make use of various tactic combinators to automate and simplify proofs.

As a simple proof-of-concept for categorical logic in Lean, we'll discuss the formalization of the *syntactic category* construction for the *positive propositional calculus (PPC)*, which is the following fragment of intuitionistic propositional logic: formulas are given by the grammar

$$\varphi, \psi \quad ::= \quad \mathsf{p} \mid \top \mid \varphi \wedge \psi \mid \varphi \to \psi$$

with the usual natural deduction rules for these connectives. By quotienting the set of formulas by the *inter-derivability* relation $\dashv\vdash$, we obtain the *syntactic poset* or *Lindenbaum-Tarski algebra* [Tar83] of the PPC. Viewing this poset as a category, we obtain the syntactic category of PPC. With the right Lean tactics, we're able to prove in just a few lines that this syntactic category forms a *cartesian closed category* (a key step in the proof of the completeness of the PPC with respect to Kripke semantics), with this extra categorical structure arising from the deductive rules of PPC. The full proof can be seen in Figure 1. Take for instance this line,

```
pr2 := by LiftT `[ apply And.and_elimr ],
```

which constructs the 'pairing' operation in a CCC (combining morphisms $f : Z \to X$ and $g : Z \to Y$ into $\langle f, g\rangle : Z \to X \times Y$) by *lifting* the PPC deduction rule of $\wedge$-introduction (if $\Phi \vdash \varphi$ and $\Phi \vdash \psi$, then $\Phi \vdash \varphi \wedge \psi$). The `LiftT` tactic defined as part of this project – whose operation we'll seek to describe – allows us to perform these kinds of 'liftings' of deduction rules onto constructions in the syntactic category. Time permitting, we will also discuss extensions to this basic PPC framework – such as the addition of modal operators $\Box$ and $\diamond$ to the logic, which correspond to (co)monads on the syntactic category – as well as the role this construction plays in soundness and completeness proofs.

The (work-in-progresss) documentation for this formalization project can be found at lean-catLogic.github.io.

[1]This formalization is done in Lean 3, and partially uses the accompanying mathematical library [mC20].

```

instance syn_FP_cat {Form : Type} [And : has_and Form] : FP_cat (Form _eq) :=
{
  unit := syn_obj And.top,
  term := by LiftT `[ apply And.truth ],
  unit_η := λ X f, by apply thin_cat.K,
  prod := and_eq,
  pr1 := by LiftT `[ apply And.and_eliml ],
  pr2 := by LiftT `[ apply And.and_elimr ],
  pair := by LiftT `[ apply And.and_intro ],
  prod_β1 := λ X Y Z f g, by apply thin_cat.K,
  prod_β2 := λ X Y Z f g, by apply thin_cat.K,
  prod_η :=  λ X Y, by apply thin_cat.K
}
instance syn_CC_cat {Form : Type} [Impl : has_impl Form] : CC_cat (Form _eq) :=
{
  exp := impl_eq,
  eval := by LiftT `[ apply cart_x.modus_ponens ],
  curry := by LiftT `[ apply cart_x.impl_ε],
  curry_β := λ {X Y Z} u, by apply thin_cat.K,
  curry_η := λ {X Y Z} v, by apply thin_cat.K,
```

Figure 1: For any deductive calculus with truth, conjunction, and implication (satisfying the usual rules), its syntactic category is a CCC. As discussed, the uses of `LiftT` are instances where deduction rules of the PPC are lifted to constructions on the syntactic category. The lines invoking `thin_cat.K` are appeals to the fact that the syntactic category is a poset in order to prove that certain diagrams commute.

References

[dMKA+15] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pages 378–388. Springer, 2015.

[Law63] F William Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences*, 50(5):869–872, 1963.

[mC20] The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, page 367–381, New York, NY, USA, 2020. Association for Computing Machinery.

[Tar83] Alfred Tarski. *Logic, semantics, metamathematics: papers from 1923 to 1938*. Hackett Publishing, 1983.