Formalizing and Proving Theorems in Coq — Lecture 6

Tobias Kappé

Today's lecture

You will learn about...

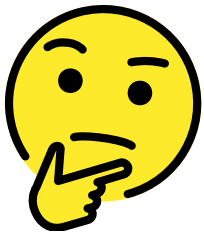
- the Curry-Howard isomorphism as an idea
- how this correspondence manifests in Coq

Making you wonder

Interesting coincidences:

- *"is of type"* and *"is a proof of"* are denoted by `:`
- *implication* and *functions* both use the symbol `->`

This is completely intentional!



A logical pun

In Coq's underlying logic

- Proofs are expressions (programs)
- Propositions are types
- Simplification is execution

Demo

Exploiting the correspondence

We have just...

- constructed proofs via syntax.
- constructed definitions in proof mode.

Computational content

The Brouwer-Heyting-Kolmogorov interpretation of logic:

- A proof of $P \to Q$ *transforms* a proof of P into a proof of Q.
- A proof of $P \wedge Q$ is a proof of P *and* a proof of Q.
- A proof of $P \vee Q$ is a proof of P *or* a proof of Q.

Let's see how this works in Coq!

One step further

What about the first-order connective $\forall$?

For instance, what is $\forall n : \texttt{nat}.\ P(n)$ as a type?

Alternatively, what is a value of type $\forall n : \texttt{nat}.\ P(n)$?

A function that, for every n, produces a value of type $P(n)$!

One step further

What about the first-order connective $\exists$?

For instance, what is $\exists n : \texttt{nat}.\ P(n)$ as a type?

Alternatively, what is a value of type $\exists n : \texttt{nat}.\ P(n)$?

A pair of an n and a value of type $P(n)$!

It's up to you now

Start working on your projects!