Propositions and Proofs

In this chapter, we introduce the reasoning techniques used in Coq, starting with a very reduced fragment of logic, minimal propositional logic, where formulas are exclusively constructed using propositional variables and implication. For instance, if \( P, Q, R \) are propositions, we may consider the problem of proving the formula below:

\[
(P \Rightarrow Q) \Rightarrow ((Q \Rightarrow R) \Rightarrow (P \Rightarrow R))
\]

Two approaches can be followed to solve this problem.

The first approach, proposed by Tarski [80], consists in assigning to every variable a denotation, a truth value \( t \) (true) or \( f \) (false). One considers all possible assignments, and the formula is true if it is true for all of them. This method can proceed by building a truth table where all the possible combinations of values for the variables are enumerated. If the value of the complete formula is \( t \) in all cases, then the formula is valid. The table in Fig. 3.1 shows the validity of the formula above.

![Fig. 3.1. A truth table](image)

The second approach, proposed by Heyting [49], consists of replacing the question “is the proposition \( P \) true?” with the question “what are the proofs of
"P (if any)?" The Coq system follows this approach, for which a representation is not only needed for statements, but also for proofs. Actually, this need to represent proofs is already fulfilled by the typed λ-calculus we studied in the previous chapter. According to Heyting, a proof of the implication $P \Rightarrow Q$ is a process to obtain a proof of $Q$ from a proof of $P$. In other words, a proof of $P \Rightarrow Q$ is a function that given an arbitrary proof of $P$ constructs a proof of $Q$.

The two approaches correspond to different understandings of logic. Tarski’s approach corresponds to classical logic where the principle of excluded middle "every proposition is either true or false" holds. This justifies the use of truth tables. Heyting’s approach applies to intuitionistic logic where the principle of excluded middle is rejected. In Sect. 3.7.2, we shall see that there are formulas that are true in classical logic and cannot be proved in intuitionistic logic.

Although intuitionistic logic is less powerful than classical logic, it enjoys a property that is important to computer scientists: correct programs can be extracted from proofs. This is the reason why Coq mainly supports the intuitionistic approach of Heyting.

Once we agree with Heyting, it becomes natural to reuse the functional programming techniques seen in the previous chapter. If we consider some proof as an expression in a functional language, then the proven statement is a type (the type of proofs for this statement). Heyting’s implication $P \Rightarrow Q$ becomes the arrow type $P \rightarrow Q$. A proof of an implication can be a simple abstraction with the form \texttt{fun $H$: $P$ \Rightarrow $t$} where $t$ is a proof of $Q$, well-formed in a context where one assumes a hypothesis $H$ stating $P$.

This correspondence between A-calculus as a model of functional programming and proof calculi like natural deduction [77] is called the “Curry–Howard isomorphism.” It has been the subject of many investigations, among which we would like to cite Scott [79], Martin-Löf [62], Girard, Lafont, and Taylor [46] and the seminal papers of Curry and Feys [32] and Howard [52]. Thanks to this correspondence, we can use programming ideas during proof tasks and logical ideas during program design. Moreover, the tools provided by the Coq system can be used for both programming and reasoning.

For a stronger unification of techniques between programming and reasoning, we drop the notation that is specific to logic. To begin with, we drop the specific notation for implication and write $P \rightarrow Q$ to express “$P$ implies $Q$.” The proposition we introduced above is then written in the following manner:

\[
(P \rightarrow Q) \rightarrow (Q \rightarrow R) \rightarrow P \rightarrow R
\]

A proof of this statement is a λ-term whose type is this proposition, for instance the following term:

\[
\text{fun } (H: P \rightarrow Q) (H': Q \rightarrow R) (p: P) \Rightarrow H' (H \ p)
\]

This term expresses how one can construct a proof of $R$ from arbitrary proofs of $P \rightarrow Q$, $Q \rightarrow R$, and $P$, respectively named $H$, $H'$, and $p$. This construction consists in applying $H$ to $p$ to obtain a proof of $Q$ and then applying $H'$ to this proof to obtain a proof of $R$. This kind of reasoning is known as a syllogism.