All Coq Rules in One Place

Xiaohong Chen   Lucas Peña

March 25, 2020

Abstract

This document summarizes all the proof rules of the Coq proof assistant, as listed in `https://coq.inria.fr/distrib/current/refman/language/cic.html`.

1 Syntax

Let us fix a countably infinite set V of *variables*, denoted $x, y, \ldots$. Let us fix a countably infinite set C of *constants*, denoted $c, d, \ldots$.

Definition 1.1. We define the set *Term* to be the smallest set that satisfies the following conditions:

1. $\mathsf{SProp}, \mathsf{Prop}, \mathsf{Set} \in Term$; $\mathsf{Type}(i) \in Term$ for every $i \in \mathbb{N}_{\geq 1}$.

2. $V \subseteq Term$.

3. $C \subseteq Term$.

4. If $x \in V$ and $T, U \in Term$, then $\forall x : T, U \in Term$.

5. If $x \in V$ and $T, u \in Term$, then $\lambda x : T.u \in Term$.

6. If $t, u \in Term$, then $(t\,u) \in Term$, called *application*.

7. If $x \in V$ and $t, T, u \in Term$, then $\mathsf{let}\, x := t : T \,\mathsf{in}\, u \in Term$.

where $\forall x : T, U$ binds x to U and $\lambda x : T.u$ binds x to u. We use $\mathrm{FV}(T) \subseteq V$ to denote the set of free variables in $T \in Term$. For $T, U \in Term$ and $x \in V$, we use $T[U/x]$ to denote the result of substituting U for x in T, where α-renaming happens implicitly to prevent variable capture.

Definition 1.2. We define the set $Sort = \{\mathsf{SProp}, \mathsf{Prop}, \mathsf{Set}\} \cup \{\mathsf{Type}(i) \mid i \in \mathbb{N}\}$. Note that $Sort \subseteq Term$. Elements in $Sort$ are called *sorts* and denoted as s, possibly with subscripts.

Definition 1.3. A *local assumption* is written $x : T$, where $x \in V$ and $T \in Term$. A *local definition* is written $x := u : T$, where $x \in V$ and $u, T \in Term$. In both cases, we call x the *declared variable*. A *local context* is an ordered list of local assumptions and local definitions, such that the declared variables are all distinct. We use Γ, possibly with subscripts, to denote local contexts.

Notation 1.4. We use the notation $[x : T \,;\, y := u : U \,;\, z : V]$ to denote the local context that consists of the local assumption $x : T$, the local definition $y := u : U$ and the local assumption $z : V$, with the implicit requirement that x, y, z are all distinct. The empty local context is written as $[]$. Let Γ be a local context. We write $x \in \Gamma$ to mean that x is declared in Γ. We write $(x : T) \in \Gamma$ to mean that the local assumption $x : T$ is in Γ, or that the local definition $x := u : T$ is in Γ for some $u \in Term$. We write $(x := u : T) \in \Gamma$ to mean that the local definition $x := u : T$ is in Γ. We write $\Gamma :: (x : T)$ to denote the local context that enriches Γ with $x : T$, with the implicit requirement that $x \notin \Gamma$. Similarly, we write $\Gamma :: (x := u : T)$ to denote the local context that enriches Γ with $x := u : T$, with the implicit requirement that $x \notin \Gamma$. We write $\Gamma_1 \,;\, \Gamma_2$ to mean the local context obtained by concatenating Γ_1 and Γ_2, with the implicit requirement that all variables declared in Γ_2 are not declared in Γ_1.

Definition 1.5. A *global assumption* is written $(c:T)$, with the parentheses, where $c \in C$ and $T \in Term$. A *global definition* is written $c := u:T$, where $c \in C$ and $u, T \in Term$. In both cases, we call c the *declared constant*. A *global environment* is an ordered list of global assumptions and global definitions, and also *declarations of inductive objects*, which are defined later. We use E, possibly with subscripts, to denote global environments.

Notation 1.6. We use the notation $c_1:T \,;\, c_2 := u:U \,;\, c_3:V$ to denote the local context that consists of the global assumption $c_1:T$, the global definition $c_2 := u:U$ and the global assumption $c_3:V$. The empty global context is written as $[]$. Let E be a local context. We write $c \in E$ to mean that c is declared in E. We write $(c:T) \in E$ to mean that the global assumption $c:T$ is in E, or that the global definition $c := u:T$ is in E for some $u \in Term$. We write $(c := u:T) \in E$ to mean that the global definition $c := u:T$ is in E. We write $E \,;\, c:T$ to denote the global context that enriches E with $c:T$. Similarly, we write $E \,;\, c := u:T$ to denote the global context that enriches E with $(c := u:T)$.

Notation 1.7. We write $E[\Gamma] \vdash u:T$ to mean that u is *well-typed* with type T in global environment E and local environment Γ. We write $\mathcal{WF}(E)[\Gamma]$ to mean that the global environment E is *well-formed* and Γ is a *valid local context* in E.

Definition 1.8. A term u is *well-typed* in a global environment E if there is a local context Γ and type T such that $E[\Gamma] \vdash u:T$ is derivable with the rules below.

2 Coq Rules

In this section we list all Coq rules. A *rule* consists of a set of *premises* and one *conclusion*, separated by a horizontal bar. For readability, we put *side conditions* alongside the premises. Side conditions are typed in green, to distinguish from the premises.

2.1 Basic Typing Rules

There are 18 basic typing rules, as shown below.

Names	Rules	Comments
(W-Empty)	$$\frac{\cdot}{\mathcal{WF}([])[]}$$	The empty global environment is well-formed, and the empty local context is a valid local context in the empty global environment.
(W-Local-Assum)	$$\frac{E[\Gamma] \vdash T:s \quad s \in S \quad x \notin \Gamma}{\mathcal{WF}(E)[\Gamma :: (x:T)]}$$	The side condition $x \notin \Gamma$ needs not to be specified because it is implicitly implied by the notation $\Gamma :: (x:T)$; see Notation 1.4.

$$(\text{W-Local-Def}) \quad \frac{E[\Gamma] \vdash t : T \quad x \notin \Gamma}{\mathcal{WF}(E)[\Gamma :: (x := t : T)]}$$

The side condition $x \notin \Gamma$ needs not to be specified because it is implicitly implied by the notation $\Gamma :: (x := t : T)$; see Notation 1.4.

$$(\text{W-Global-Assum}) \quad \frac{E[] \vdash T : s \quad s \in S \quad c \notin E}{\mathcal{WF}(E; c : T)[]}$$

$$(\text{W-Global-Def}) \quad \frac{E[] \vdash t : T \quad c \notin E}{\mathcal{WF}(E; c := t : T)[]}$$

$$(\text{Ax-SProp}) \quad \frac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash \mathsf{SProp} : \mathsf{Type}(1)}$$

$$(\text{Ax-Prop}) \quad \frac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash \mathsf{Prop} : \mathsf{Type}(1)}$$

$$(\text{Ax-Set}) \quad \frac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash \mathsf{Set} : \mathsf{Type}(1)}$$

$$(\text{Ax-Type}) \quad \frac{\mathcal{WF}(E)[\Gamma]}{E[\Gamma] \vdash \mathsf{Type}(i) : \mathsf{Type}(i+1)}$$

Here, $i \in \mathbb{N}_{\geq 1}$ is any positive natural number.

$$(\text{Var}) \quad \frac{\mathcal{WF}(E)[\Gamma] \quad (x : T \in \Gamma), \text{ or } (x := t : T) \in \Gamma \text{ for some } t \in \mathit{Term}}{E[\Gamma] \vdash x : T}$$

$$(\text{Const}) \quad \frac{\mathcal{WF}(E)[\Gamma] \quad (c : T \in E), \text{ or } (c := t : T) \in E \text{ for some } t \in \mathit{Term}}{E[\Gamma] \vdash c : T}$$

$$(\text{Prod-SProp}) \quad \frac{E[\Gamma] \vdash T : s \quad s \in S \quad E[\Gamma :: (x : T)] \vdash U : \mathsf{SProp}}{E[\Gamma] \vdash \forall x : T, U : \mathsf{SProp}}$$

$$(\text{Prod-Prop}) \quad \frac{E[\Gamma] \vdash T : s \quad s \in S \quad E[\Gamma :: (x : T)] \vdash U : \mathsf{Prop}}{E[\Gamma] \vdash \forall x : T, U : \mathsf{Prop}}$$

$$(\text{Prod-Set}) \quad \frac{E[\Gamma] \vdash T : s \quad s \in \{\mathsf{SProp}, \mathsf{Prop}, \mathsf{Set}\} \quad E[\Gamma :: (x : T)] \vdash U : \mathsf{Set}}{E[\Gamma] \vdash \forall x : T, U : \mathsf{Set}}$$

$$(\text{Prod-Type}) \quad \frac{E[\Gamma] \vdash T : s \quad s \in \{\mathsf{SProp}, \mathsf{Type}(i)\} \quad E[\Gamma :: (x : T)] \vdash U : \mathsf{Type}(i)}{E[\Gamma] \vdash \forall x : T, U : \mathsf{Type}(i)}$$

$$(\text{Lam}) \quad \frac{E[\Gamma] \vdash \forall x : T, U : s \quad s \in S \quad E[\Gamma :: (x : T)] \vdash t : U}{E[\Gamma] \vdash \lambda x : T.t : \forall x : T, U}$$

$$(\text{App}) \quad \frac{E[\Gamma] \vdash t : \forall x : U, T \quad E[\Gamma] \vdash u : U}{E[\Gamma] \vdash (t\, u) : T[u/x]}$$

$$(\text{Let}) \quad \frac{E[\Gamma] \vdash t : T \quad E[\Gamma :: (x := t : T)] \vdash u : U}{E[\Gamma] \vdash \mathsf{let}\, x := t : T\, \mathsf{in}\, u : U[t/x]}$$

2.2 Conversion Rules