

# All Coq Rules in One Place

Xiaohong Chen

Lucas Peña

May 19, 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  $x, y, \dots$ . Let us fix a countably infinite set  $C$  of *constants*, denoted  $c, d, \dots$ .

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

1.  $SProp, Prop, Set \in Term$ ;  $Type(i) \in Term$  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  $(tu) \in Term$ , called *application*.
7. If  $x \in V$  and  $t, T, u \in Term$ , then  $\text{let } x := t : T \text{ in } u \in Term$ .

where  $\forall x:T, U$  binds  $x$  to  $U$  and  $\lambda x:T.u$  binds  $x$  to  $u$ . We use  $FV(T) \subseteq V$  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  $\alpha$ -renaming happens implicitly to prevent variable capture.

**Definition 1.2.** We define the set  $Sort = \{SProp, Prop, Set\} \cup \{Type(i) \mid i \in \mathbb{N}\}$ . 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  $\Gamma$ , 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  $\Gamma$  be a local context. We write  $x \in \Gamma$  to mean that  $x$  is declared in  $\Gamma$ . We write  $(x:T) \in \Gamma$  to mean that the local assumption  $x:T$  is in  $\Gamma$ , or that the local definition  $x := u:T$  is in  $\Gamma$  for some  $u \in Term$ . We write  $(x := u:T) \in \Gamma$  to mean that the local definition  $x := u:T$  is in  $\Gamma$ . We write  $\Gamma :: (x:T)$  to denote the local context that enriches  $\Gamma$  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  $\Gamma$  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  $\Gamma_1$  and  $\Gamma_2$ , with the implicit requirement that all variables declared in  $\Gamma_2$  are not declared in  $\Gamma_1$ .

**Definition 1.5.** A *global assumption* is written  $(c:T)$ , with the parentheses, where  $c \in C$  and  $T \in \text{Term}$ . A *global definition* is written  $c := u:T$ , where  $c \in C$  and  $u, T \in \text{Term}$ . 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  $u \in \text{Term}$ . 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  $\Gamma$ . We write  $\mathcal{WF}(E)[\Gamma]$  to mean that the global environment  $E$  is *well-formed* and  $\Gamma$  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  $\Gamma$  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)	$\frac{E[\Gamma] \vdash T:s \quad \textcolor{green}{s \in S} \quad \textcolor{green}{x \notin \Gamma}}{\mathcal{WF}(E)[\Gamma :: (x:T)]}$	The side condition $x \notin \Gamma$ needs not to be specified because it is implicit in the notation $\Gamma :: (x:T)$ ; see Notation 1.4.

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

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.

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

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

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

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

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

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

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

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

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

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

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

(PROD-SET)	$\frac{E[\Gamma] \vdash T : s \quad s \in \{\text{SProp}, \text{Prop}, \text{Set}\} \quad E[\Gamma :: (x : T)] \vdash U : \text{Set}}{E[\Gamma] \vdash \forall x : T, U : \text{Set}}$
(PROD-TYPE)	$\frac{E[\Gamma] \vdash T : s \quad s \in \{\text{SProp}, \text{Type}(i)\} \quad E[\Gamma :: (x : T)] \vdash U : \text{Type}(i)}{E[\Gamma] \vdash \forall x : T, U : \text{Type}(i)}$
(LAM)	$\frac{E[\Gamma] \vdash \forall x : T, U : s \quad s \in \mathcal{S} \quad E[\Gamma :: (x : T)] \vdash t : U}{E[\Gamma] \vdash \lambda x : T. t : \forall x : T, U}$
(APP)	$\frac{E[\Gamma] \vdash t : \forall x : U, T \quad E[\Gamma] \vdash u : U}{E[\Gamma] \vdash (tu) : T[u/x]}$
(LET)	$\frac{E[\Gamma] \vdash t : T \quad E[\Gamma :: (x := t : T)] \vdash u : U}{E[\Gamma] \vdash \text{let } x := t : T \text{ in } u : U[t/x]}$

## 2.2 Conversion Rules

In this section, we define what it means for two Coq programs to be *intentionally equal*, or *convertible*.

Names	Rules	Comments
(BETA)	$\overline{E[\Gamma] \vdash ((\lambda x : T. t)u) \triangleright_{\beta} t[x/u]}$	We say that $t[x/u]$ is the $\beta$ -contraction of $((\lambda x : T. t)u)$ , and that $((\lambda x : T. t)u)$ is the $\beta$ -expansion of $t[x/u]$
		$\iota$ -reduction rules to be defined later
(DELTA-LOCAL)	$\frac{\mathcal{WF}(E)[\Gamma] \quad (x := t : T) \in \Gamma}{E[\Gamma] \vdash x \triangleright_{\Delta} t}$	Reducing variable defined in local context

(DELTA-LOCAL)	$\frac{\mathcal{WF}(E)[\Gamma] \quad (c := t : T) \in E}{E[\Gamma] \vdash c \triangleright_{\delta} t}$	Reducing constant defined in global context
(ZETA)	$\frac{\mathcal{WF}(E)[\Gamma] \quad E[\Gamma] \vdash u : U \quad E[\Gamma :: (x := u : U)] \vdash t : T}{E[\Gamma] \vdash \text{let } x := u : U \text{ in } t \triangleright_{\zeta} t[x/x]}$	Remove local definitions occurring in terms

In addition to the above convertibility rules, we also allow identifying a term  $t$  of type  $\forall x:t, U$  with its  $\eta$ -expansion  $\lambda x:T.(tx)$  for  $x$  fresh in  $t$ . Note  $\eta$ -reduction is deliberately not defined.

**Notation 2.1.** We write  $E[\Gamma] \vdash t \triangleright u$  for the contextual closure of the rules defined above. That is,  $t$  reduces to  $u$  with global environment  $E$  and local context  $\Gamma$  with one of the previous reductions  $\beta, \Delta, \delta, \iota$ , or  $\zeta$ .

**Definition 2.2.** Two terms are called *irrelevant* if they share a common type of a strict proposition  $A : \text{SProp}$ . Irrelevant terms can be identified.

**Definition 2.3.** Two terms  $t_1, t_2$  are called  $\beta\delta\iota\zeta\eta$ -convertible, or *convertible*, or *equivalent* in global environment  $E$  and local context  $\Gamma$  iff there exists  $t_1, t_2$  such that

$$E[\Gamma] \vdash t_1 \triangleright \dots \triangleright u_1 \text{ and } E[\Gamma] \vdash t_2 \triangleright \dots \triangleright u_2$$

and either  $u_1$  and  $u_2$  are identical up irrelevant subterms, or they are convertible up to  $\eta$ -exxpansion. We denote this as  $E[\Gamma] \vdash t_1 =_{\beta\delta\iota\zeta\eta} t_2$

## 2.3 Subtyping Rules

The *subtyping* relation is inductively defined as follows:

- if  $E[\Gamma] \vdash t =_{\beta\delta\iota\zeta\eta} u$ , then  $E[\Gamma] \vdash t \leq_{\beta\delta\iota\zeta\eta} u$
- if  $i \leq j$ , then  $E[\Gamma] \vdash \text{Type}(i) \leq_{\beta\delta\iota\zeta\eta} \text{Type}(j)$
- $E[\Gamma] \vdash \text{Set} \leq_{\beta\delta\iota\zeta\eta} \text{Type}(i)$  for all  $i$
- $E[\Gamma] \vdash \text{Prop} \leq_{\beta\delta\iota\zeta\eta} \text{Set}$
- if  $E[\Gamma] \vdash T =_{\beta\delta\iota\zeta\eta} U$  and  $E[\Gamma :: (x : T)] \vdash T' \leq_{\beta\delta\iota\zeta\eta} U'$ , then  $E[\Gamma] \vdash \forall x : T, T' \leq_{\beta\delta\iota\zeta\eta} \forall x : U, U'$

## 2.4 Conversion/Subtyping: Polymorphic Universes