HIGH-ORDER LOGIC IN MATCHING μ-LOGIC

XIAOHONG CHEN AND GRIGORE ROSU

Contents

1. Simply-Typed λ-Calculus

2. Calculus of Inductive Constructions

The following is a compact digest of the comprehensive document of Cic at the reference website of the Coq theorem prover at `https://coq.inria.fr/distrib/current/refman/language/cic.html`.

2.1. The formal language of Cic.

Definition 2.1. The set of *sorts*, denoted as $\mathcal{S}$, is defined as:

$$\mathcal{S} \equiv \{\mathsf{Prop}, \mathsf{Set}, \mathsf{Type}(i) \mid i \in \mathbb{N}\} \tag{1}$$

Definition 2.2. The set of *terms* is inductively defined as:

- the sorts $\mathsf{Set}, \mathsf{Prop}, \mathsf{Type}(i)$ are terms.
- variables $x, y, \ldots$ are terms.
- constants $c, d, \ldots$ are terms.
- $\forall x : T\ , U$ is a term, where x is a variable and T, U are terms. If x does not occur in U, we write $T \to U$ instead of $\forall x : T, U$.
- $\lambda x : T\ .\ u$ is a term, where x is a variable and T, u are terms.
- (tu) is a term, where t, u are terms.
- *let* $x := t : T$ *in* u is a term, where x is a variable and t, T, u are terms.

Remark 2.3. I didn't find the definition of *types*. I quote from the reference document that "from a syntactic point of view, types cannot be distinguished from terms, except that they cannot start by an abstraction or a constructor."

2.2. **Typing rules in Cic.**

Definition 2.4. A *local context*, denoted as Γ, is an ordered list of *local declarations*, which can be either a *local assumption* with the form $x : T$ where T is a type, or a *local definition* with the form $x := t : T$. Variables in a local context must be distinct. If x is declared in Γ, we write $x \in \Gamma$. If $x : T$ is in Γ, or there exists a t such that $x := t : T$ is in Γ, we write $x : T \in \Gamma$. In the latter case, we also write $x := t : T \in \Gamma$. The empty local context is denoted as $[]$. We write $\Gamma :: (x : T)$ and $\Gamma :: (x := t : T)$ to mean the local context Γ extended with a local assumption/definition. We use $\Gamma_1; \Gamma_2$ to mean the concatenation of two local contexts.

Definition 2.5. A *global environment*, denoted as E, is an ordered list of *global declarations*, which can be either a *global assumption* or a *global definition*, but also a *inductive definition*, which we define later. A global assumption has the form $c : T$ for a constant c. A global definition has the form $c := t : T$ for a constant c. The empty global environment is denoted as $[]$. We write $E; c : T$ and $E; c := t : T$ to mean the global environment extended with a global assumption/definition.

Definition 2.6. The following two forms of judgments are defined simultaneously in Figure 1:

- $E[\Gamma] \vdash t : T$, which means that the term t is well-typed and has type T in the global environment E and local context Γ.
- $\mathcal{WF}(E)[\Gamma]$, which means that the global environment E is well-formed and Γ is valid in E.

W-Empty

$$\frac{}{\mathcal{WF}([])[]}$$

W-Local-Assum

$$\frac{E[\Gamma] \vdash T : s \qquad s \in \mathcal{S} \qquad x \notin \Gamma}{\mathcal{WF}(E)[\Gamma :: (x : T)]}$$

W-Local-Def

$$\frac{E[\Gamma] \vdash t : T \qquad x \notin \Gamma}{\mathcal{WF}(E)[\Gamma :: (x := t : T)]}$$

W-Global-Assum

$$\frac{E[] \vdash T : s \qquad s \in \mathcal{S} \qquad c \notin \mathcal{E}}{\mathcal{WF}(E; c : T)[]}$$

W-Global-Def

$$\frac{E[] \vdash t : T \qquad c \notin E}{\mathcal{WF}(E; c := t : T)[]}$$

Ax-Prop

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

Ax-Set

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

Ax-Type

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

Var

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

Const

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

Prod-Prop

$$\frac{E[\Gamma] \vdash T : s \qquad s \in \mathcal{S} \qquad E[\Gamma :: (x : T)] \vdash U : \mathsf{Prop}}{E[\Gamma] \vdash \forall\, x : T, U : \mathsf{Prop}}$$

Prod-Set

$$\frac{E[\Gamma] \vdash T : s \qquad s \in \{\mathsf{Prop}, \mathsf{Set}\} \qquad E[\Gamma :: (x : T)] \vdash U : \mathsf{Set}}{E[\Gamma] \vdash \forall\, x : T, U : \mathsf{Set}}$$

Prod-Type

$$\frac{E[\Gamma] \vdash T : \mathsf{Type}(i) \qquad E[\Gamma :: (x : T)] \vdash U : \mathsf{Type}(i)}{E[\Gamma] \vdash \forall\, x : T, U : \mathsf{Type}(i)}$$

Lam

$$\frac{E[\Gamma] \vdash \forall\, x : T, U : s \qquad E[\Gamma :: (x : T)] \vdash t : U}{E[\Gamma] \vdash \lambda x : T.\ t : \forall x : T, U}$$

App

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

Let

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

FIGURE 1. Cic proof system