1 Syntax

The minimal syntax, may extend someday.

$$
\begin{array}{lcl}
\mathit{letter} & ::= & \texttt{a..z} \mid \texttt{A..Z} \\
\mathit{ident} & ::= & \mathit{letter}\ \{\mathit{letter}\} \\
\mathit{term} & ::= & \texttt{forall}\ \mathit{binder}\ \{\mathit{binder}\},\ \mathit{term} \\
 & \mid & \texttt{fun}\ \{\mathit{binder}\}\ \texttt{=>}\ \mathit{term} \\
 & \mid & \texttt{fix}\ \mathit{fix\_body} \\
 & \mid & \texttt{let}\ \mathit{ident}\ \{\mathit{binder}\}\ \texttt{:}\mathit{term}\ \texttt{:=}\ \mathit{term}\ \texttt{in}\ \mathit{term} \\
 & \mid & \mathit{term}\ \texttt{->}\ \mathit{term} \\
 & \mid & \mathit{term}\ \mathit{arg}\ \{\mathit{arg}\} \\
 & \mid & \texttt{match}\ \mathit{term}\ \texttt{with} \\
 & & \quad \{\ \mid \mathit{equation}\} \\
 & & \texttt{end} \\
 & \mid & \mathit{sort} \\
 & \mid & \mathit{(term)} \\
\mathit{arg} & ::= & \mathit{term} \\
\mathit{binder} & ::= & \mathit{(ident}\ \texttt{:}\ \mathit{term)} \\
\mathit{sort} & ::= & \texttt{Prop} \mid \texttt{Set} \mid \texttt{Type} \\
\mathit{fix\_body} & ::= & \mathit{ident}\ \{\mathit{binder}\}\ \texttt{:}\mathit{term}\ \texttt{:=}\ \mathit{term} \\
\mathit{equation} & ::= & \mathit{pattern}\ \texttt{=>}\ \mathit{term} \\
\mathit{pattern} & ::= & \mathit{ident}\ \{\mathit{ident}\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{sentence} & ::= & \mathit{axiom} \\
 & \mid & \mathit{definition} \\
 & \mid & \mathit{inductive} \\
 & \mid & \mathit{fixpoint} \\
 & \mid & \mathit{assertion}\ \mathit{proof} \\
\mathit{axiom} & ::= & \texttt{Axiom}\ \mathit{ident}\ \texttt{:}\ \mathit{term}\ . \\
\mathit{definition} & ::= & \texttt{Definition}\ \mathit{ident}\ \{\mathit{binder}\}\ \texttt{:}\ \mathit{term}\ \texttt{:=}\ \mathit{term}\ . \\
\mathit{inductive} & ::= & \texttt{Inductive}\ \mathit{ident}\ \{\mathit{binder}\}\ \texttt{:}\ \mathit{term}\ \texttt{:=} \\
 & & \quad \{\ \mid \mathit{ident}\ \{\mathit{binder}\}\ \texttt{:}\ \mathit{term}\}\ . \\
\mathit{assertion} & ::= & \texttt{Theorem}\ \mathit{ident}\ \{\mathit{binder}\}\ \texttt{:}\ \mathit{term}\ . \\
\mathit{proof} & ::= & \texttt{Proof}\ .\ \{\mathit{tactic}\ .\}\ \texttt{Qed}\ .
\end{array}
$$

tactic	::=	*applying*
	|	*context_managing*
	|	*case_analyzing*
	|	*rewriting*
	|	*computing*
	|	*equality*
applying	::=	`exact` *term*
	|	`apply` *term [*`in` *ident]*
context_managing	::=	`intro` *[ident]*
	|	`intros`
case_analyzing	::=	`destruct` *term*
	|	`induction` *term*
rewriting	::=	`rewrite` *[* `<-` | `->`*]* *term [* `in` *term]*
computing	::=	`simpl`
equality	::=	`reflexivity`
	|	`symmetry`

helper	::=	*printing*
	|	*proof_handling*
printing	::=	`Print` *ident .*
	|	`Check` *term .*
proof_handling	::=	`Undo` .
	|	`Restart` .
	|	`Admitted` .
	|	`Abort` .

2 Calculus

2.1 Term

1. $\texttt{Set}, \texttt{Prop}$ are terms.
2. Variables $\texttt{x}, \texttt{y}$, etc., are terms.
3. Constants $\texttt{c}, \texttt{d}$, etc., are terms.
4. If $\texttt{x}$ is a variable and $\texttt{T}, \texttt{U}$ are terms, then $\forall \texttt{x} : \texttt{T}, \texttt{U}$ is a term.
5. If $\texttt{x}$ is a variable and $\texttt{T}, \texttt{u}$ are terms, then $\lambda \texttt{x} : \texttt{T}.\ \texttt{u}$ is a term.
6. If $\texttt{x}$ and $\texttt{u}$ are terms, then $(\texttt{t}\ \texttt{u})$ is a term.
7. If $\texttt{x}$ is a variable and $\texttt{t}, \texttt{T}, \texttt{u}$ are terms, then $\texttt{let}\ \ \texttt{x} := \texttt{t} : \texttt{T}\ \ \texttt{in}\ \ \texttt{u}$ is a term.

2.2 Typing Rule

2.2.1 Notation

- $\mathcal{S}$: $\{\texttt{Prop}, \texttt{Set}\}$.
- E : global environment.
- Γ : local context.
- $\texttt{u}\{\texttt{x}/\texttt{t}\}$: substitute free occurrence of variable $\texttt{x}$ to term $\texttt{t}$ in term $\texttt{u}$.
- $\mathcal{WF}(E)[\Gamma]$: E is well-formed and Γ is valid in E.

2.2.2 Typing Rules

$$\mathcal{WF}([])[] \tag{T-Empty}$$

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

$$\frac{E[] \vdash \texttt{t} : \texttt{T} \qquad \texttt{c} \notin E}{\mathcal{WF}(E : \texttt{c} := \texttt{t} : \texttt{T})} \tag{T-Local-Def}$$

$$\frac{\mathcal{WF}(E)[\Gamma] \qquad (\texttt{x} : \texttt{T}) \in \Gamma}{E[\Gamma] \vdash \texttt{x} : \texttt{T}} \tag{T-Var1}$$

$$\frac{\mathcal{WF}(E)[\Gamma] \qquad (\texttt{x} := \texttt{t} : \texttt{T}) \in \Gamma}{E[\Gamma] \vdash \texttt{x} : \texttt{T}} \tag{T-Var2}$$

$$\frac{\mathcal{WF}(E)[\Gamma] \qquad (\texttt{c} : \texttt{T}) \in E}{E[\Gamma] \vdash \texttt{c} : \texttt{T}} \tag{T-Const1}$$

$$\frac{\mathcal{WF}(E)[\Gamma] \qquad (\texttt{c} := \texttt{t} : \texttt{T}) \in E}{E[\Gamma] \vdash \texttt{c} : \texttt{T}} \tag{T-Const2}$$

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

$$\frac{E[\Gamma] \vdash \texttt{T} : \texttt{s} \qquad \texttt{s} \in \mathcal{S} \qquad E[\Gamma :: (\texttt{x} : \texttt{T})] \vdash \texttt{U} : \texttt{Set}}{E[\Gamma] \vdash \forall \texttt{x} : \texttt{T}, \texttt{U} : \texttt{Set}} \tag{T-Prod-Set}$$

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

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

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

2.3 Conversion Rule

2.3.1 Notation

- $E[\Gamma] \vdash \mathtt{t} \rhd \mathtt{u}$: $\mathtt{t}$ reduces to $\mathtt{u}$ in E, Γ with one of the $\beta, \iota, \delta, \zeta$ reductions.
- $E[\Gamma] \vdash \mathtt{t} \overset{*}{\rhd} \mathtt{u}$: $E[\Gamma] \vdash \mathtt{t} \rhd \cdots \rhd \mathtt{u}$.
- $\mathtt{u} \equiv \mathtt{v}$: $\mathtt{u}$ and $\mathtt{v}$ are identical.

2.3.2 Conversion Rules

$$\frac{E[\Gamma] \vdash (\lambda \mathtt{x} : \mathtt{T}.\ \mathtt{t})\ u}{\mathtt{t}\{\mathtt{x}/\mathtt{u}\}} \tag{β-Conv}$$

$$\frac{\mathtt{case}((\mathtt{c_p}\ \mathtt{q_1}\ \cdots\ \mathtt{q_r}\ \mathtt{a_1}\ \cdots\ \mathtt{a_m}), \mathtt{P}, \mathtt{f_1}|\cdots|\mathtt{f_n})}{f_i\ a_1\ \cdots\ a_m} \tag{ι-Conv}$$

$$\frac{E[\Gamma] \vdash \mathtt{x} \qquad (\mathtt{x} := \mathtt{t} : \mathtt{T}) \in \Gamma}{\mathtt{t}} \tag{δ-Conv1}$$

$$\frac{E[\Gamma] \vdash \mathtt{c} \qquad (\mathtt{x} := \mathtt{t} : \mathtt{T}) \in E}{\mathtt{t}} \tag{δ-Conv2}$$

$$\frac{E[\Gamma] \vdash \mathtt{let}\ \ \mathtt{x} := \mathtt{u}\ \ \mathtt{in}\ \ \mathtt{t}}{\mathtt{t}\{\mathtt{x}/\mathtt{u}\}} \tag{ζ-Conv}$$

$$\frac{E[\Gamma] \vdash \mathtt{t} : \forall \mathtt{x} : \mathtt{T}, \mathtt{U} \qquad \mathtt{x}\ \text{fresh in}\ \mathtt{t}}{\lambda \mathtt{x} : \mathtt{T}.\ (\mathtt{t}\ \mathtt{x})} \tag{η-Exp}$$

Definition 1 (Convertibility)**.** $\mathtt{t_1}$ *and* $\mathtt{t_2}$ *are convertible iff there exists* $\mathtt{u_1}$ *and* $\mathtt{u_2}$ *such that* $E[\Gamma] \vdash \mathtt{t_1} \overset{*}{\rhd} \mathtt{u_1}$ *and* $E[\Gamma] \vdash \mathtt{t_2} \overset{*}{\rhd} \mathtt{u_2}$ *and either* $u_1 \equiv u_2$ *or they are convertible up to* η*-expansion.*

2.4 Inductive Definition

2.4.1 Notation

- $\mathtt{Ind}[p](\Gamma_I := \Gamma_C)$: inductive definition.
- Γ_I : names and types of inductive type.
- Γ_C : names and types of constructors of inductive type.
- p : the number of parameters of inductive type.
- Γ_P : the context of parameters.

2.4.2 Typing Rule

$$\frac{\mathcal{WF}(E)[\Gamma] \qquad \mathtt{Ind}[p](\Gamma_I := \Gamma_C) \in E \qquad (\mathtt{a} : \mathtt{A}) \in \Gamma_i}{E[\Gamma] \vdash \mathtt{a} : \mathtt{A}} \tag{T-Ind}$$

$$\frac{\mathcal{WF}(E)[\Gamma] \qquad \mathtt{Ind}[p](\Gamma_I := \Gamma_C) \in E \qquad (\mathtt{c} : \mathtt{C}) \in \Gamma_C}{E[\Gamma] \vdash \mathtt{c} : \mathtt{C}} \tag{T-Constr}$$

$$\frac{(E[\Gamma_P] \vdash \mathtt{A_j} : \mathtt{s'_j})_{j=1..k} \qquad (E[\Gamma_i; \Gamma_P] \vdash \mathtt{C_i} : \mathtt{s_{q_i}})_{i=1..n}}{\mathcal{WF}(E; \mathtt{Ind}[p](\Gamma_I := \Gamma_C))[\Gamma]} \tag{T-Wf-Ind}$$

2.4.3 Well-formed Requirement

To maintain the consistency of the system, we must restrict the inductive definitions to a syntactic criterion of **positivity**, which guarantees the *soundness and safety* of the system.

Definition 2 (Constructor). $\mathtt{T}$ *is a type of constructor of* $\mathtt{I}$ *if*

- $\mathtt{T} \equiv (\mathtt{I}\ \mathtt{t}_1\ \cdots\ \mathtt{t}_\mathtt{n})$
- $\mathtt{T} \equiv \forall \mathtt{x} : \mathtt{U}, \mathtt{T}'$, where T' is a type of constructor of $\mathtt{I}$

Definition 3 (Positivity). *The type of constructor* $\mathtt{T}$ *satisfies the positivity condition for a constant* $\mathtt{X}$ *if*

- $\mathtt{T} \equiv (\mathtt{X}\ \mathtt{t}_1\ \cdots\ \mathtt{t}_\mathtt{n})$ and $\mathtt{X}$ does not occur free in $\mathtt{t}_\mathtt{i}$
- $\mathtt{T} \equiv \forall \mathtt{x} : \mathtt{U}, \mathtt{V}$ and $\mathtt{X}$ occurs only *strictly positively* in $\mathtt{U}$ and $\mathtt{V}$ satisfies *the positivity condition* for $\mathtt{X}$

Definition 4 (Strictly Positivity). *The constant* $\mathtt{X}$ *occurs strictly positively in* $\mathtt{T}$ *if*

- $\mathtt{X}$ does not occur in $\mathtt{T}$
- $\mathtt{T} \rhd^* (\mathtt{X}\ \mathtt{t}_1\ \cdots\ \mathtt{t}_\mathtt{n})$ and $\mathtt{X}$ does not occur in $\mathtt{t}_\mathtt{i}$
- $\mathtt{T} \rhd^* \forall \mathtt{x} : \mathtt{U}, \mathtt{V}$ and $\mathtt{X}$ does not occur in $\mathtt{U}$ but occurs *strictly positively* in $\mathtt{V}$
- $\mathtt{T} \rhd^* (\mathtt{I}\ \mathtt{a}_1\ \cdots\ \mathtt{a}_\mathtt{m}\ \mathtt{t}_1\ \cdots\ \mathtt{t}_\mathtt{p})$, where $\mathtt{Ind}[m](\mathtt{I} : \mathtt{A} := \mathtt{c}_1 : \forall \mathtt{p}_1 : \mathtt{P}_1, \ldots \forall \mathtt{p}_\mathtt{m} : \mathtt{P}_\mathtt{m}, \mathtt{C}_1; \cdots ; \mathtt{c}_\mathtt{n} : \forall \mathtt{p}_1 : \mathtt{P}_1, \ldots, \forall \mathtt{p}_\mathtt{m} : \mathtt{P}_\mathtt{m}, \mathtt{C}_\mathtt{n})$, and $\mathtt{X}$ does not occur in t_i, and the types of constructor $\mathtt{C}_\mathtt{i}\{\mathtt{p}_\mathtt{j}/\mathtt{a}_\mathtt{j}\}_{\mathtt{j}=1..\mathtt{m}}$ satisfies *the nested positivity condition* for $\mathtt{X}$

Definition 5 (Nested Positivity). *The type of constructor* $\mathtt{T}$ *satisfies the nested positivity condition for a constant* $\mathtt{X}$ *if*

- $\mathtt{T} \equiv (\mathtt{I}\ \mathtt{b}_1\ \cdots\ \mathtt{b}_\mathtt{m}\ \mathtt{u}_1\ \cdots\ \mathtt{u}_\mathtt{p})$, where $\mathtt{I}$ is an inductive definition with m parameters and $\mathtt{X}$ does not occur in u_i
- $\mathtt{T} \equiv \forall \mathtt{x} : \mathtt{U}, \mathtt{V}$ and $\mathtt{X}$ occurs *strictly positively* in $\mathtt{U}$ and $\mathtt{V}$ satisfies *the nested positivity condition* for $\mathtt{X}$

3 Destructor

4 Fixpoint

References

[1] The Coq Development Team. *The Coq Proof Assistant Reference Manual*, 8.7.2 edition, February 2018.