1 Syntax

The minimal syntax, may extend someday.

letter	::=	`a..z` | `A..Z`
ident	::=	*letter* {*letter*}
term	::=	`forall` *binder* {*binder*}, *term*
	|	`fun` {*binder*} `=>` *term*
	|	`fix` *ident* *binder* {*binder*} `:` *term* `:=` *term*
	|	`let` *ident* {*binder*} `:`*term* `:=` *term* `in` *term*
	|	*term* `->` *term*
	|	*term* *arg* {*arg*}
	|	`match` *term* `with`
		{ | *equation*}
		`end`
	|	*sort*
	|	*(term)*
arg	::=	*term*
binder	::=	*(ident* `:` *term)*
sort	::=	`Prop` | `Set` | `Type`
equation	::=	*pattern* `=>` *term*
pattern	::=	*ident* {*ident*}

sentence	::=	*axiom*
	|	*definition*
	|	*inductive*
	|	*fixpoint*
	|	*assertion proof*
axiom	::=	`Axiom` *ident* `:` *term* .
definition	::=	`Definition` *ident* {*binder*} `:` *term* `:=` *term* .
inductive	::=	`Inductive` *ident* {*binder*} `:` *term* `:=`
		{ | *ident* `:` *term*} .
fixpoint	::=	`Fixpoint` *ident* {*binder*} `:` *term* `:=` *term* .
assertion	::=	`Theorem` *ident* {*binder*} `:` *term* .
proof	::=	`Proof` . {*tactic* .} `Qed` .

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

2.2 Typing Rule

2.2.1 Notation

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

2.2.2 Typing Rules

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

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

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

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

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

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

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

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

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

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

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

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

2.3 Conversion Rule

2.3.1 Notation

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

2.3.2 Conversion Rules

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

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

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

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

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

*Later in **Sugar and Desugar*** (ι-Conv)

Definition 1 (Convertibility)**.** $\mathtt{t}_1$ *and* $\mathtt{t}_2$ *are convertible iff there exists* $\mathtt{u}_1$ *and* $\mathtt{u}_2$ *such that* $E[\Gamma] \vdash \mathtt{t}_1 \overset{*}{\triangleright} \mathtt{u}_1$ *and* $E[\Gamma] \vdash \mathtt{t}_2 \overset{*}{\triangleright} \mathtt{u}_2$ *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*

- $\mathtt{T} \equiv (\mathtt{I}\ \mathtt{t_1}\ \cdots\ \mathtt{t_n})$
- $\mathtt{T} \equiv \forall \mathtt{x} : \mathtt{U}, \mathtt{T}'$, 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*

- $\mathtt{T} \equiv (\mathtt{X}\ \mathtt{t_1}\ \cdots\ \mathtt{t_n})$ and $\mathtt{X}$ does not occur free in $\mathtt{t_i}$
- $\mathtt{T} \equiv \forall \mathtt{x} : \mathtt{U}, \mathtt{V}$ 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 $\mathtt{T}$
- $\mathtt{T} \rhd^* (\mathtt{X}\ \mathtt{t_1}\ \cdots\ \mathtt{t_n})$ and $\mathtt{X}$ does not occur in $\mathtt{t_i}$
- $\mathtt{T} \rhd^* \forall \mathtt{x} : \mathtt{U}, \mathtt{V}$ and $\mathtt{X}$ does not occur in $\mathtt{U}$ but occurs *strictly positively* in $\mathtt{V}$
- $\mathtt{T} \rhd^* (\mathtt{I}\ \mathtt{a_1}\ \cdots\ \mathtt{a_m}\ \mathtt{t_1}\ \cdots\ \mathtt{t_p})$, where $\mathtt{Ind}[m](\mathtt{I} : \mathtt{A} := \mathtt{c_1} : \forall \mathtt{p_1} : \mathtt{P_1}, \ldots \forall \mathtt{p_m} : \mathtt{P_m}, \mathtt{C_1}; \cdots ; \mathtt{c_n} : \forall \mathtt{p_1} : \mathtt{P_1}, \ldots, \forall \mathtt{p_m} : \mathtt{P_m}, \mathtt{C_n})$, and $\mathtt{X}$ does not occur in t_i, and the types of constructor $\mathtt{C_i}\{\mathtt{p_j}/\mathtt{a_j}\}_{\mathtt{j=1..m}}$ 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*

- $\mathtt{T} \equiv (\mathtt{I}\ \mathtt{b_1}\ \cdots\ \mathtt{b_m}\ \mathtt{u_1}\ \cdots\ \mathtt{u_p})$, 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 Sugar and Desugar

3.1 Match

Definition 6 (Desugar of `match`).

$$\frac{\mathtt{match}\ \mathtt{m}\ \mathtt{with}\ (\mathtt{c_1}\ \mathtt{x_{11}}\ \cdots\ \mathtt{x_{1p_1}}) \Rightarrow \mathtt{f_1}\ |\ \cdots\ |\ (\mathtt{c_n}\ \mathtt{x_{n1}}\ \cdots\ \mathtt{x_{np_n}}) \Rightarrow \mathtt{f_n}\ \mathtt{end}}{\mathtt{case}(\mathtt{m}, \lambda \mathtt{x_{11}} \cdots \mathtt{x_{1p_1}}.\ \mathtt{f_1}\ |\ \cdots\ |\lambda \mathtt{x_{n1}} \cdots \mathtt{x_{np_n}}.\ \mathtt{f_n})} \quad \text{(D-MATCH)}$$

$$\frac{E[\Gamma] \vdash \mathtt{case}(\mathtt{c}, \lambda \mathtt{x_{11}} \cdots \mathtt{x_{1p_1}}.\ \mathtt{f_1}| \cdots |\lambda \mathtt{x_{n1}} \cdots \mathtt{x_{np_n}}.\ \mathtt{f_n}) \qquad E[\Gamma] \vdash \lambda \mathtt{x_{i1}} \cdots \mathtt{x_{ip_i}}.\ \mathtt{f_i} : \mathtt{S_{i1}} \to \cdots \to \mathtt{S_{ip_i}} \to \mathtt{S}}{\mathtt{case}(\mathtt{c}, \lambda \mathtt{x_{11}} \cdots \mathtt{x_{1p_1}}.\ \mathtt{f_1}| \cdots |\lambda \mathtt{x_{n1}} \cdots \mathtt{x_{np_n}}.\ \mathtt{f_n}) : \mathtt{S}}$$
(T-MATCH)

$$\frac{\mathtt{case}((\mathtt{c_p}\ \mathtt{q_1}\ \cdots\ \mathtt{q_r}\ \mathtt{a_1}\ \cdots\ \mathtt{a_m}), \mathtt{f_1}| \cdots |\mathtt{f_n})}{f_i\ a_1\ \cdots\ a_m} \quad (\iota\text{-CONV-MATCH})$$

3.2 Fixpoint

Definition 7 (Desugar of `fix`).

$$\frac{\mathtt{fix}\ \mathtt{f_1}(\Gamma_1) : \mathtt{A_1} := \mathtt{t_1}\ \mathtt{with} \cdots \mathtt{with}\ \mathtt{f_n}(\Gamma_n) : \mathtt{A_n} := \mathtt{t_n} \qquad \mathtt{t'_i} = \lambda\Gamma_{\mathtt{i}}.\ \mathtt{t_i} \qquad \mathtt{A'_i} = \forall\Gamma_{\mathtt{i}}, \mathtt{A_i}}{\mathtt{Fix}\ \mathtt{f_i}\{\mathtt{f_1} : \mathtt{A'_1} := \mathtt{t'_1} \cdots \mathtt{f_n} : \mathtt{A'_n} := \mathtt{t'_n}\}} \quad \text{(D-Fix)}$$

$$\frac{(E[\Gamma] \vdash \mathtt{A_i} : \mathtt{s_i})_{\mathtt{i=1..n}} \qquad (E[\Gamma, \mathtt{f_1} : \mathtt{A_1}, \cdots, \mathtt{f_n} : \mathtt{A_n}] \vdash \mathtt{t_i} : \mathtt{A_i})_{\mathtt{i=1..n}}}{E[\Gamma] \vdash \mathtt{Fix}\ \mathtt{f_i}\{\mathtt{f_1} : \mathtt{A_1} := \mathtt{t_1} \cdots \mathtt{f_n} : \mathtt{A_n} := \mathtt{t_n}\} : \mathtt{A_i}} \quad \text{(T-Fix)}$$

$$\frac{\mathtt{Fix}\ \mathtt{f_i}\{\mathtt{F}\}\ \mathtt{a_1}\ \cdots\ \mathtt{a_{k_i}}}{\mathtt{t_i}\{(\mathtt{f_k}/\mathtt{Fix}\ \mathtt{f_k}\{\mathtt{F}\})_{\mathtt{k=1..n}}\}\ \mathtt{a_1}\ \cdots\ \mathtt{a_{k_i}}} \quad (\iota\text{-Conv-Fix})$$

3.3 Let

Definition 8 (Desugar of `let`).

$$\frac{E[\Gamma] \vdash \mathtt{let}\ \ \mathtt{x} := \mathtt{u}\ \ \mathtt{in}\ \ \mathtt{t}}{\mathtt{t}\{\mathtt{x}/\mathtt{u}\}} \quad (\zeta\text{-Conv-Let})$$