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` *name* *fix_body*
	|	`let` *ident* {*binder*} : *term* := *term* `in` *term*
	|	*term* `->` *term*
	|	*term* *arg* {*arg*}
	|	`match` *term* `with`
		{ | *equation*}
		`end`
	|	*sort*
	|	(*term*)
arg	::=	*term*
binder	::=	*name* : *term*
name	::=	_
	|	*ident*
sort	::=	`Prop` | `Set` | `Type`
fix_body	::=	*ident* {*binder*} : *term* := *term*
equation	::=	*pattern* `=>` *term*
pattern	::=	*ident* {*name*}

sentence	::=	*axiom*
	|	*definition*
	|	*inductive*
	|	*fixpoint*
	|	*assertion proof*
assumption	::=	`Axiom` *ident* : *term* .
definition	::=	`Definition` *ident* {*binder*} : *term* := *term* .
inductive	::=	`Inductive` *ident* {*binder*} : *term* :=
		{ | *ident* {*binder*} : *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` `.`