Syntax of Coq V8

B. Barras

April 16, 2004

1 Meta notations used in this document

Non-terminals are printed between angle brackets (e.g. ⟨*non-terminal*⟩) and terminal symbols are printed in bold font (e.g. **terminal**). Lexemes are displayed as non-terminals.

The usual operators on regular expressions:

notation	meaning
$regexp*$	repeat $regexp$ 0 or more times
$regexp+$	repeat $regexp$ 1 or more times
$regexp?$	$regexp$ is optional
$regexp_1$ | $regexp_2$	alternative

Parenthesis are used to group regexps. Beware to distinguish this operator ( ) from the terminals **( )**, and | from terminal **|**.

Rules are optionaly annotated in the right margin with:

- a precedence and associativity (L for left, R for right and N for no associativity), indicating how to solve conflicts; lower levels are tighter;
- a rule name.

In order to solve some conflicts, a non-terminal may be invoked with a precedence (notation: $\langle entry \rangle_{prec}$), meaning that rules with higher precedence do not apply.

2 Lexical conventions

Lexical categories are:

$$
\begin{array}{rcl}
\langle \textit{ident} \rangle & ::= & \big(\langle \textit{letter} \rangle \mid \_\big) * \big(\langle \textit{letter} \rangle \mid \langle \textit{digit} \rangle \mid \text{'} \mid \_\big)* \\
\langle \textit{field} \rangle & ::= & \textbf{.}\langle \textit{ident} \rangle \\
\langle \textit{meta-ident} \rangle & ::= & \textbf{?}\langle \textit{ident} \rangle \\
\langle \textit{num} \rangle & ::= & \langle \textit{digit} \rangle + \\
\langle \textit{int} \rangle & ::= & \langle \textit{num} \rangle \mid \textbf{-}\langle \textit{num} \rangle \\
\langle \textit{digit} \rangle & ::= & \mathbf{0} - \mathbf{9} \\
\langle \textit{letter} \rangle & ::= & \mathbf{a} - \mathbf{z} \mid \mathbf{A} - \mathbf{Z} \mid \langle \textit{unicode-letter} \rangle \\
\langle \textit{string} \rangle & ::= & \text{"} \ \big(\text{""} \mid \langle \textit{unicode-char-but-"} \rangle\big) * \ \text{"}
\end{array}
$$

Reserved identifiers for the core syntax are:

as, **cofix**, **else**, **end**, **fix**, **for**, **forall**, **fun**, **if**, **in**, **let**, **match**, **Prop**, **return**, **Set**, **then**, **Type**, **with**

Symbols used in the core syntax:

( ) { } : , ⇒ → := _ | @ % .(

Note that **struct** is not a reserved identifier.

3 Syntax of terms

3.1 Core syntax

The main entry point of the term grammar is $\langle \textit{constr} \rangle_9$. When no conflict can appear, $\langle \textit{constr} \rangle_{200}$ is also used as entry point.

⟨*constr*⟩ ::=	⟨*binder-constr*⟩	$200R$	(BINDERS)
|	⟨*constr*⟩ **:** ⟨*constr*⟩	$100R$	(CAST)
|	⟨*constr*⟩ **:** ⟨*binder-constr*⟩	$100R$	(CAST')
|	⟨*constr*⟩ → ⟨*constr*⟩	$80R$	(ARROW)
|	⟨*constr*⟩ → ⟨*binder-constr*⟩	$80R$	(ARROW')
|	⟨*constr*⟩ ⟨*appl-arg*⟩+	$10L$	(APPLY)
|	**@** ⟨*reference*⟩ ⟨*constr*⟩$_9$∗	$10L$	(EXPL-APPLY)
|	⟨*constr*⟩ **.(** ⟨*reference*⟩ ⟨*appl-arg*⟩ ∗ **)**	$1L$	(PROJ)
|	⟨*constr*⟩ **.(** **@** ⟨*reference*⟩ ⟨*constr*⟩$_9$ ∗ **)**	$1L$	(EXPL-PROJ)
|	⟨*constr*⟩ **%** ⟨*ident*⟩	$1L$	(SCOPE-CHG)
|	⟨*atomic-constr*⟩	0	
|	⟨*match-expr*⟩	0	
|	**(** ⟨*constr*⟩ **)**	0	

⟨*binder-constr*⟩ ::=	**forall** ⟨*binder-list*⟩ **,** ⟨*constr*⟩$_{200}$	(PROD)
|	**fun** ⟨*binder-list*⟩ ⇒ ⟨*constr*⟩$_{200}$	(LAMBDA)
|	⟨*fix-expr*⟩	
|	**let** ⟨*ident-with-params*⟩ **:=** ⟨*constr*⟩$_{200}$ **in** ⟨*constr*⟩$_{200}$	(LET)
|	**let** ⟨*single-fix*⟩ **in** ⟨*constr*⟩$_{200}$	(REC-LET)
|	**let (** ⟨*let-pattern*⟩? **)** ⟨*return-type*⟩? **:=** ⟨*constr*⟩$_{200}$ **in** ⟨*constr*⟩$_{200}$	(LET-CASE)
|	**if** ⟨*if-item*⟩ **then** ⟨*constr*⟩$_{200}$ **else** ⟨*constr*⟩$_{200}$	(IF-CASE)

⟨*appl-arg*⟩ ::=	**(** ⟨*ident*⟩ **:=** ⟨*constr*⟩$_{200}$ **)**	(IMPL-ARG)
|	**(** ⟨*num*⟩ **:=** ⟨*constr*⟩$_{200}$ **)**	(IMPL-ARG)
|	⟨*constr*⟩$_9$	

⟨*atomic-constr*⟩ ::=	⟨*reference*⟩	(VARIABLES)
|	⟨*sort*⟩	(CIC-SORT)
|	⟨*num*⟩	(NUMBER)
|	_	(HOLE)
|	⟨*meta-ident*⟩	(META/EVAR)

⟨*ident-with-params*⟩ ::= ⟨*ident*⟩ ⟨*binder-let*⟩ ∗ ⟨*type-cstr*⟩

⟨*binder-list*⟩ ::= ⟨*binder*⟩ ⟨*binder-let*⟩∗
| ⟨*name*⟩ + **:** ⟨*constr*⟩

⟨*binder*⟩ ::= ⟨*name*⟩ (INFER)
| **(** ⟨*name*⟩ + **:** ⟨*constr*⟩ **)** (BINDER)

⟨*binder-let*⟩ ::= ⟨*binder*⟩
| **(** ⟨*name*⟩ ⟨*type-cstr*⟩ **:=** ⟨*constr*⟩ **)**

⟨*let-pattern*⟩ ::= ⟨*name*⟩
| ⟨*name*⟩ **,** ⟨*let-pattern*⟩

⟨*type-cstr*⟩ ::= (**:** ⟨*constr*⟩)?

⟨*reference*⟩ ::= ⟨*ident*⟩ (SHORT-IDENT)
| ⟨*ident*⟩ ⟨*field*⟩+ (QUALID)

⟨*sort*⟩ ::= **Prop** | **Set** | **Type**

⟨*name*⟩ ::= ⟨*ident*⟩ | _

⟨*fix-expr*⟩ ::= ⟨*single-fix*⟩
| ⟨*single-fix*⟩ (**with** ⟨*fix-decl*⟩) + **for** ⟨*ident*⟩

⟨*single-fix*⟩ ::= ⟨*fix-kw*⟩ ⟨*fix-decl*⟩

⟨*fix-kw*⟩ ::= **fix** | **cofix**

⟨*fix-decl*⟩ ::= ⟨*ident*⟩ ⟨*binder-let*⟩ ∗ ⟨*annot*⟩? ⟨*type-cstr*⟩ **:=** ⟨*constr*⟩$_{200}$

⟨*annot*⟩ ::= **{ struct** ⟨*ident*⟩ **}**

$\langle match\text{-}expr \rangle$ ::= **match** $\langle match\text{-}items \rangle$ $\langle return\text{-}type \rangle$? **with** **|**? $\langle branches \rangle$? **end** (MATCH)

$\langle match\text{-}items \rangle$ ::= $\langle match\text{-}item \rangle$ **,** $\langle match\text{-}items \rangle$
| $\langle match\text{-}item \rangle$

$\langle match\text{-}item \rangle$ ::= $\langle constr \rangle_{100}$ (**as** $\langle name \rangle$)? (**in** $\langle constr \rangle_{100}$)?

$\langle return\text{-}type \rangle$ ::= **return** $\langle constr \rangle_{100}$

$\langle if\text{-}item \rangle$ ::= $\langle constr \rangle$ ((**as** $\langle name \rangle$)? $\langle return\text{-}type \rangle$)?

$\langle branches \rangle$ ::= $\langle eqn \rangle$ **|** $\langle branches \rangle$
| $\langle eqn \rangle$

$\langle eqn \rangle$ ::= $\langle pattern \rangle$ (**,** $\langle pattern \rangle$) $*$ $\Rightarrow$ $\langle constr \rangle$

$\langle pattern \rangle$ ::= $\langle reference \rangle$ $\langle pattern \rangle+$	$1L$	(CONSTRUCTOR)
$\langle pattern \rangle$ **as** $\langle ident \rangle$	$1L$	(ALIAS)
$\langle pattern \rangle$ **%** $\langle ident \rangle$	$1L$	(SCOPE-CHANGE)
$\langle reference \rangle$	0	(PATTERN-VAR)
_	0	(HOLE)
$\langle num \rangle$	0	
($\langle tuple\text{-}pattern \rangle$ **)**		

$\langle tuple\text{-}pattern \rangle$ ::= $\langle pattern \rangle$
| $\langle tuple\text{-}pattern \rangle$ **,** $\langle pattern \rangle$ (PAIR)

3.2 Notations of the prelude (logic and basic arithmetic)

Reserved notations:

Symbol	precedence
_, _	$250L$
IF _ **then** _ **else** _	$200R$
_ : _	$100R$
_ $\leftrightarrow$ _	$95N$
_ $\rightarrow$ _	$90R$
_ $\vee$ _	$85R$
_ $\wedge$ _	$80R$
~ _	$75R$
_ = _ _ = _:>_ _ = _ = _ _ $\neq$ _ _ $\neq$ _:>_	$70N$
_ < _ _ > _ _ $\leq$ _ _ $\geq$ _ _ < _ < _ _ < _ $\leq$ _ _ $\leq$ _ < _ _ $\leq$ _ $\leq$ _	
_ + _ _ − _ − _	$50L$
_ * _ _/_ /_	$40L$

Existential quantifiers follows the **forall** notation (with same precedence 200), but only one quantified variable is allowed.

⟨*binder-constr*⟩ ::= ...
| ⟨*quantifier-kwd*⟩ ⟨*name*⟩ ⟨*type-cstr*⟩ **,** ⟨*constr*⟩$_{200}$

⟨*quantifier-kwd*⟩ ::= **exists** (EX)
| **exists2** (EX2)

Symbol	precedence	
_ + {_}	50	(SUMOR)
{_ : _ | _}	0	(SIG)
{_ : _ | _&_}	0	(SIG2)
{_ : _ & _}	0	(SIGS)
{_ : _ & _&_}	0	(SIGS2)
{_} + {_}	0	(SUMBOOL)

4 Grammar of tactics

Additional symbols are:

’ ; () || ⊢ [ ] ←

Additional reserved keywords are:

at using

4.1 Basic tactics

⟨*simple-tactic*⟩ ::= **intros until** ⟨*quantified-hyp*⟩
| **intros** ⟨*intro-patterns*⟩
| **intro** ⟨*ident*⟩? (**after** ⟨*ident*⟩)?
| **assumption**
| **exact** ⟨*constr*⟩$_9$
| **apply** ⟨*constr-with-bindings*⟩
| **elim** ⟨*constr-with-bindings*⟩ ⟨*eliminator*⟩?
| **elimtype** ⟨*constr*⟩$_9$
| **case** ⟨*constr-with-bindings*⟩
| **casetype** ⟨*constr*⟩$_9$
| **fix** ⟨*ident*⟩? ⟨*num*⟩
| **fix** ⟨*ident*⟩ ⟨*num*⟩ **with** ⟨*fix-spec*⟩+
| **cofix** ⟨*ident*⟩?
| **cofix** ⟨*ident*⟩ ⟨*fix-spec*⟩+
| **cut** ⟨*constr*⟩$_9$
| **assert** ⟨*constr*⟩$_9$
| **assert (** ⟨*ident*⟩ **:** ⟨*constr*⟩$_{200}$ **)**
| **assert (** ⟨*ident*⟩ **:=** ⟨*constr*⟩$_{200}$ **)**
| **pose** ⟨*constr*⟩$_9$
| **pose (** ⟨*ident*⟩ **:=** ⟨*constr*⟩$_{200}$ **)**
| **generalize** ⟨*constr*⟩$_9$+
| **generalize dependent** ⟨*constr*⟩$_9$
| **set** ⟨*constr*⟩$_9$ ⟨*clause*⟩?
| **set (** ⟨*ident*⟩ **:=** ⟨*constr*⟩$_{200}$ **)** ⟨*clause*⟩?
| **instantiate (** ⟨*num*⟩ **:=** ⟨*constr*⟩$_{200}$ **)** ⟨*clause*⟩?
| **specialize** ⟨*num*⟩? ⟨*constr-with-bindings*⟩
| **lapply** ⟨*constr*⟩$_9$
| **simple induction** ⟨*quantified-hyp*⟩
| **induction** ⟨*induction-arg*⟩ ⟨*with-names*⟩? ⟨*eliminator*⟩?
| **double induction** ⟨*quantified-hyp*⟩ ⟨*quantified-hyp*⟩
| **simple destruct** ⟨*quantified-hyp*⟩
| **destruct** ⟨*induction-arg*⟩ ⟨*with-names*⟩? ⟨*eliminator*⟩?
| **decompose record** ⟨*constr*⟩$_9$
| **decompose sum** ⟨*constr*⟩$_9$
| **decompose** [⟨*reference*⟩ +] ⟨*constr*⟩$_9$
| ...

⟨*simple-tactic*⟩ ::= ...
| **trivial** ⟨*hint-bases*⟩?
| **auto** ⟨*num*⟩? ⟨*hint-bases*⟩?
| **auto** ⟨*num*⟩? **decomp** ⟨*num*⟩?
| **clear** ⟨*ident*⟩+
| **clearbody** ⟨*ident*⟩+
| **move** ⟨*ident*⟩ **after** ⟨*ident*⟩
| **rename** ⟨*ident*⟩ **into** ⟨*ident*⟩
| **left** ⟨*with-binding-list*⟩?
| **right** ⟨*with-binding-list*⟩?
| **split** ⟨*with-binding-list*⟩?
| **exists** ⟨*binding-list*⟩?
| **constructor** ⟨*num*⟩ ⟨*with-binding-list*⟩?
| **constructor** ⟨*tactic*⟩?
| **reflexivity**
| **symmetry** (**in** ⟨*ident*⟩)?
| **transitivity** ⟨*constr*⟩$_9$
| ⟨*inversion-kwd*⟩ ⟨*quantified-hyp*⟩ ⟨*with-names*⟩? ⟨*clause*⟩?
| **dependent** ⟨*inversion-kwd*⟩ ⟨*quantified-hyp*⟩ ⟨*with-names*⟩? (**with** ⟨*constr*⟩$_9$)?
| **inversion** ⟨*quantified-hyp*⟩ **using** ⟨*constr*⟩$_9$ ⟨*clause*⟩?
| ⟨*red-expr*⟩ ⟨*clause*⟩?
| **change** ⟨*conversion*⟩ ⟨*clause*⟩?

⟨*red-expr*⟩ ::= **red** | **hnf** | **compute**
| **simpl** ⟨*pattern-occ*⟩?
| **cbv** ⟨*red-flag*⟩+
| **lazy** ⟨*red-flag*⟩+
| **unfold** ⟨*unfold-occ*⟩ (**,** ⟨*unfold-occ*⟩)∗
| **fold** ⟨*constr*⟩$_9$+
| **pattern** ⟨*pattern-occ*⟩ (**,** ⟨*pattern-occ*⟩)∗

⟨*conversion*⟩ ::= ⟨*pattern-occ*⟩ **with** ⟨*constr*⟩$_9$
| ⟨*constr*⟩$_9$

⟨*inversion-kwd*⟩ ::= **inversion** | **invesion_clear** | **simple inversion**

Conflicts exists between integers and constrs.

⟨*quantified-hyp*⟩ ::= ⟨*int*⟩ | ⟨*ident*⟩

⟨*induction-arg*⟩ ::= ⟨*int*⟩ | ⟨*constr*⟩$_9$

⟨*fix-spec*⟩ ::= **(** ⟨*ident*⟩ ⟨*binder*⟩$*$ ⟨*annot*⟩? **:** ⟨*constr*⟩$_{200}$ **)**

⟨*intro-patterns*⟩ ::= ⟨*intro-pattern*⟩$*$

⟨*intro-pattern*⟩ ::= ⟨*name*⟩
| **[** ⟨*intro-patterns*⟩ (**|** ⟨*intro-patterns*⟩)$*$ **]**
| **(** ⟨*intro-pattern*⟩ (**,** ⟨*intro-pattern*⟩)$*$ **)**

⟨*with-names*⟩ ::= **as** ⟨*intro-pattern*⟩

⟨*eliminator*⟩ ::= **using** ⟨*constr-with-bindings*⟩

⟨*constr-with-bindings*⟩ ::= ⟨*constr*⟩$_9$ ⟨*with-binding-list*⟩?

⟨*with-binding-list*⟩ ::= **with** ⟨*binding-list*⟩

⟨*binding-list*⟩ ::= ⟨*constr*⟩$_9$+
| ⟨*simple-binding*⟩+

⟨*simple-binding*⟩ ::= **(** ⟨*quantified-hyp*⟩ **:=** ⟨*constr*⟩$_{200}$ **)**

⟨*red-flag*⟩ ::= **beta** | **iota** | **zeta** | **delta** | **delta -**? **[** ⟨*reference*⟩ + **]**

⟨*clause*⟩ ::= **in ***
| **in *** ⊢ ⟨*concl-occ*⟩?
| **in** ⟨*hyp-ident-list*⟩? ⊢ ⟨*concl-occ*⟩?
| **in** ⟨*hyp-ident-list*⟩?

⟨*hyp-ident-list*⟩ ::= ⟨*hyp-ident*⟩
| ⟨*hyp-ident*⟩ **,** ⟨*hyp-ident-list*⟩

⟨*hyp-ident*⟩ ::= ⟨*ident*⟩
| **(type of** ⟨*ident*⟩ **)**
| **(value of** ⟨*ident*⟩ **)**

⟨*concl-occ*⟩ ::= ***** ⟨*occurrences*⟩

⟨*pattern-occ*⟩ ::= ⟨*constr*⟩$_9$ ⟨*occurrences*⟩

⟨*unfold-occ*⟩ ::= ⟨*reference*⟩ ⟨*occurrences*⟩

⟨*occurrences*⟩ ::= (**at** ⟨*int*⟩ +)?

⟨*hint-bases*⟩ ::= **with ***
| **with** ⟨*ident*⟩+

⟨*auto-args*⟩ ::= ⟨*num*⟩? (**adding [** ⟨*reference*⟩ + **]**)? **destructuring**? (**using tdb**)?

4.2 Ltac

⟨*tactic*⟩	::=	⟨*tactic*⟩ **;** ⟨*tactic*⟩	5	(THEN)
	|	⟨*tactic*⟩ **;** [⟨*tactic-seq*⟩?]	5	(THEN-SEQ)
	|	**try** ⟨*tactic*⟩	$3R$	(TRY)
	|	**do** ⟨*int-or-var*⟩ ⟨*tactic*⟩		
	|	**repeat** ⟨*tactic*⟩		
	|	**progress** ⟨*tactic*⟩		
	|	**info** ⟨*tactic*⟩		
	|	**abstract** ⟨*tactic*⟩$_2$ (**using** ⟨*ident*⟩)?		
	|	⟨*tactic*⟩ **||** ⟨*tactic*⟩	$2R$	(ORELSE)
	|	**fun** ⟨*name*⟩ + ⇒ ⟨*tactic*⟩	1	(FUN-TAC)
	|	**let** ⟨*let-clauses*⟩ **in** ⟨*tactic*⟩		
	|	**let rec** ⟨*rec-clauses*⟩ **in** ⟨*tactic*⟩		
	|	**match reverse**? **goal with** ‖? ⟨*match-goal-rules*⟩? **end**		
	|	**match** ⟨*tactic*⟩ **with** ‖? ⟨*match-rules*⟩? **end**		
	|	**first** [⟨*tactic-seq*⟩]		
	|	**solve** [⟨*tactic-seq*⟩]		
	|	**idtac**		
	|	**fail** ⟨*num*⟩? ⟨*string*⟩?		
	|	**constr :** ⟨*constr*⟩$_9$		
	|	**ipattern :** ⟨*intro-pattern*⟩		
	|	⟨*term-ltac*⟩		
	|	⟨*reference*⟩ ⟨*tactic-arg*⟩∗		(CALL-TACTIC)
	|	⟨*simple-tactic*⟩		
	|	⟨*tactic-atom*⟩	0	(ATOMIC)
	|	**(** ⟨*tactic*⟩ **)**		
⟨*tactic-arg*⟩	::=	**ltac :** ⟨*tactic*⟩$_0$		
	|	**ipattern :** ⟨*intro-pattern*⟩		
	|	⟨*term-ltac*⟩		
	|	⟨*tactic-atom*⟩		
	|	⟨*constr*⟩$_9$		
⟨*term-ltac*⟩	::=	**fresh** ⟨*string*⟩?		
	|	**context** ⟨*ident*⟩ [⟨*constr*⟩$_{200}$]		
	|	**eval** ⟨*red-expr*⟩ **in** ⟨*constr*⟩$_9$		
	|	**type** ⟨*constr*⟩$_9$		
⟨*tactic-atom*⟩	::=	⟨*reference*⟩		
	|	()		
⟨*tactic-seq*⟩	::=	⟨*tactic*⟩ **|** ⟨*tactic-seq*⟩		
	|	⟨*tactic*⟩		

⟨*let-clauses*⟩ ::= ⟨*let-clause*⟩ (**with** ⟨*let-clause*⟩)∗

⟨*let-clause*⟩ ::= ⟨*ident*⟩ ⟨*name*⟩ ∗ **:=** ⟨*tactic*⟩

⟨*rec-clauses*⟩ ::= ⟨*rec-clause*⟩ **with** ⟨*rec-clauses*⟩
| ⟨*rec-clause*⟩

⟨*rec-clause*⟩ ::= ⟨*ident*⟩ ⟨*name*⟩ + **:=** ⟨*tactic*⟩

⟨*match-goal-rules*⟩ ::= ⟨*match-goal-rule*⟩
| ⟨*match-goal-rule*⟩ ‖ ⟨*match-goal-rules*⟩

⟨*match-goal-rule*⟩ ::= ⟨*match-hyps-list*⟩ ⊢ ⟨*match-pattern*⟩ ⇒ ⟨*tactic*⟩
| [⟨*match-hyps-list*⟩ ⊢ ⟨*match-pattern*⟩] ⇒ ⟨*tactic*⟩
| _ ⇒ ⟨*tactic*⟩

⟨*match-hyps-list*⟩ ::= ⟨*match-hyps*⟩ **,** ⟨*match-hyps-list*⟩
| ⟨*match-hyps*⟩

⟨*match-hyps*⟩ ::= ⟨*name*⟩ **:** ⟨*match-pattern*⟩

⟨*match-rules*⟩ ::= ⟨*match-rule*⟩
| ⟨*match-rule*⟩ ‖ ⟨*match-rules*⟩

⟨*match-rule*⟩ ::= ⟨*match-pattern*⟩ ⇒ ⟨*tactic*⟩
| _ ⇒ ⟨*tactic*⟩

⟨*match-pattern*⟩ ::= **context** ⟨*ident*⟩? [⟨*constr-pattern*⟩] (SUBTERM)
| ⟨*constr-pattern*⟩

⟨*constr-pattern*⟩ ::= ⟨*constr*⟩$_9$

4.3 Other tactics

⟨*simple-tactic*⟩ ::= ...
| **rewrite** ⟨*orient*⟩ ⟨*constr-with-bindings*⟩ (**in** ⟨*ident*⟩)?
| **replace** ⟨*constr*⟩$_9$ **with** ⟨*constr*⟩$_9$ (**in** ⟨*ident*⟩)?
| **replace** ⟨*orient*⟩? ⟨*constr*⟩$_9$ (**in** ⟨*ident*⟩)?
| **symplify_eq** ⟨*quantified-hyp*⟩?
| **discriminate** ⟨*quantified-hyp*⟩?
| **injection** ⟨*quantified-hyp*⟩?
| **conditional** ⟨*tactic*⟩ **rewrite** ⟨*orient*⟩ ⟨*constr-with-bindings*⟩ (**in** ⟨*ident*⟩)?
| **dependent rewrite** ⟨*orient*⟩ ⟨*ident*⟩
| **cutrewrite** ⟨*orient*⟩ ⟨*constr*⟩$_9$ (**in** ⟨*ident*⟩)?
| **absurd** ⟨*constr*⟩$_9$
| **contradiction**
| **autorewrite** ⟨*hint-bases*⟩ (**using** ⟨*tactic*⟩)?
| **refine** ⟨*constr*⟩$_9$
| **setoid_replace** ⟨*constr*⟩$_9$ **with** ⟨*constr*⟩$_9$
| **setoid_rewrite** ⟨*orient*⟩ ⟨*constr*⟩$_9$
| **subst** ⟨*ident*⟩$*$
| **decide equality** (⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$)?
| **compare** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$
| **eexact** ⟨*constr*⟩$_9$
| **eapply** ⟨*constr-with-bindings*⟩
| **prolog** [⟨*constr*⟩$_9$ $*$] ⟨*quantified-hyp*⟩
| **eauto** ⟨*quantified-hyp*⟩? ⟨*quantified-hyp*⟩? ⟨*hint-bases*⟩
| **eautod** ⟨*quantified-hyp*⟩? ⟨*quantified-hyp*⟩? ⟨*hint-bases*⟩
| **tauto**
| **simplif**
| **intuition** ⟨*tactic*⟩$_0$?
| **linearintuition** ⟨*num*⟩?
| **cc**
| **field** ⟨*constr*⟩$_9*$
| **ground** ⟨*tactic*⟩$_0$?
| **ground** ⟨*tactic*⟩$_0$? **with** ⟨*reference*⟩+
| **ground** ⟨*tactic*⟩$_0$? **using** ⟨*ident*⟩+
| **gintuition** ⟨*tactic*⟩$_0$?
| **fourierZ**
| **functional induction** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$+
| **jp** ⟨*num*⟩?
| **omega**
| **quote** ⟨*ident*⟩ ([⟨*ident*⟩ +])?
| **ring** ⟨*constr*⟩$_9*$
| **romega**

⟨*orient*⟩ ::= → | ←

5 Grammar of commands

New symbols:

. .. >-> :> <:

New keyword:

where

5.1 Classification of commands

$$
\begin{array}{rcll}
\langle \textit{vernac} \rangle & ::= & \textbf{Time}\ \langle \textit{vernac} \rangle & 2\ \text{(TIMING)} \\
& | & \langle \textit{gallina} \rangle\ \textbf{.} & 1 \\
& | & \langle \textit{command} \rangle\ \textbf{.} & \\
& | & \langle \textit{syntax} \rangle\ \textbf{.} & \\
& | & [\ \langle \textit{vernac} \rangle +\ ]\ \textbf{.} & \\
& | & \big(\langle \textit{num} \rangle\ \textbf{:}\big)?\ \langle \textit{subgoal-command} \rangle\ \textbf{.} & 0 \\
\\
\langle \textit{subgoal-command} \rangle & ::= & \langle \textit{check-command} \rangle & \\
& | & \langle \textit{tactic} \rangle\ \textbf{..}? &
\end{array}
$$

5.2 Gallina and extensions

$$
\begin{array}{rcl}
\langle \textit{gallina} \rangle & ::= & \langle \textit{thm-token} \rangle\ \langle \textit{ident} \rangle\ \langle \textit{binder-let} \rangle *\ \textbf{:}\ \langle \textit{constr} \rangle \\
& | & \langle \textit{def-token} \rangle\ \langle \textit{ident} \rangle\ \langle \textit{def-body} \rangle \\
& | & \langle \textit{assum-token} \rangle\ \langle \textit{assum-list} \rangle \\
& | & \langle \textit{finite-token} \rangle\ \langle \textit{inductive-definition} \rangle\ \big(\textbf{with}\ \langle \textit{inductive-definition} \rangle\big)* \\
& | & \textbf{Fixpoint}\ \langle \textit{fix-decl} \rangle\ \big(\textbf{with}\ \langle \textit{fix-decl} \rangle\big)* \\
& | & \textbf{CoFixpoint}\ \langle \textit{fix-decl} \rangle\ \big(\textbf{with}\ \langle \textit{fix-decl} \rangle\big)* \\
& | & \textbf{Scheme}\ \langle \textit{scheme} \rangle\ \big(\textbf{with}\ \langle \textit{scheme} \rangle\big)* \\
& | & \langle \textit{record-tok} \rangle >?\ \langle \textit{ident} \rangle\ \langle \textit{binder-let} \rangle *\ \textbf{:}\ \langle \textit{constr} \rangle\ \textbf{:=}\ \langle \textit{ident} \rangle?\ \textbf{\{}\ \langle \textit{field-list} \rangle\ \textbf{\}} \\
& | & \textbf{Ltac}\ \langle \textit{ltac-def} \rangle\ \big(\ \textbf{with}\ \langle \textit{ltac-def} \rangle\big)*
\end{array}
$$

$$
\begin{array}{rcl}
\langle \textit{thm-token} \rangle & ::= & \textbf{Theorem}\ |\ \textbf{Lemma}\ |\ \textbf{Fact}\ |\ \textbf{Remark} \\
\langle \textit{def-token} \rangle & ::= & \textbf{Definition}\ |\ \textbf{Let}\ |\ \textbf{Local}?\ \textbf{SubClass} \\
\langle \textit{assum-token} \rangle & ::= & \textbf{Hypothesis}\ |\ \textbf{Variable}\ |\ \textbf{Axiom}\ |\ \textbf{Parameter} \\
\langle \textit{finite-token} \rangle & ::= & \textbf{Inductive}\ |\ \textbf{CoInductive} \\
\langle \textit{record-tok} \rangle & ::= & \textbf{Record}\ |\ \textbf{Structure}
\end{array}
$$

⟨*def-body*⟩ ::= ⟨*binder-let*⟩ ∗ ⟨*type-cstr*⟩ **:=** ⟨*reduce*⟩? ⟨*constr*⟩
| ⟨*binder-let*⟩ ∗ **:** ⟨*constr*⟩

⟨*reduce*⟩ ::= **Eval** ⟨*red-expr*⟩ **in**

⟨*ltac-def*⟩ ::= ⟨*ident*⟩ ⟨*name*⟩ ∗ **:=** ⟨*tactic*⟩

⟨*rec-definition*⟩ ::= ⟨*fix-decl*⟩ ⟨*decl-notation*⟩?

⟨*inductive-definition*⟩ ::= ⟨*string*⟩? ⟨*ident*⟩ ⟨*binder-let*⟩ ∗ **:** ⟨*constr*⟩ **:=** ‖? ⟨*constructor-list*⟩? ⟨*decl-notation*⟩?

⟨*constructor-list*⟩ ::= ⟨*constructor*⟩ ‖ ⟨*constructor-list*⟩
| ⟨*constructor*⟩

⟨*constructor*⟩ ::= ⟨*ident*⟩ ⟨*binder-let*⟩ ∗ (⟨*coerce-kwd*⟩ ⟨*constr*⟩)?

⟨*decl-notation*⟩ ::= **where** ⟨*string*⟩ **:=** ⟨*constr*⟩

⟨*field-list*⟩ ::= ⟨*field*⟩ **;** ⟨*field-list*⟩
| ⟨*field*⟩

⟨*field*⟩ ::= ⟨*ident*⟩ (⟨*coerce-kwd*⟩ ⟨*constr*⟩)?
| ⟨*ident*⟩ ⟨*type-cstr-coe*⟩ **:=** ⟨*constr*⟩

⟨*assum-list*⟩ ::= (**(** ⟨*simple-assum-coe*⟩ **)**)+
| ⟨*simple-assum-coe*⟩

⟨*simple-assum-coe*⟩ ::= ⟨*ident*⟩ + ⟨*coerce-kwd*⟩ ⟨*constr*⟩

⟨*coerce-kwd*⟩ ::= **:>** | **:**

⟨*type-cstr-coe*⟩ ::= (⟨*coerce-kwd*⟩ ⟨*constr*⟩)?

⟨*scheme*⟩ ::= ⟨*ident*⟩ **:=** ⟨*dep-scheme*⟩ **for** ⟨*reference*⟩ **Sort** ⟨*sort*⟩

⟨*dep-scheme*⟩ ::= **Induction** | **Minimality**

5.3 Modules and sections

⟨*gallina*⟩ ::= **Module** ⟨*ident*⟩ ⟨*mbinder*⟩ ∗ ⟨*of-mod-type*⟩? (**:=** ⟨*mod-expr*⟩)?
| **Module Type** ⟨*ident*⟩ ⟨*mbinder*⟩ ∗ (**:=** ⟨*mod-type*⟩)?
| **Declare Module** ⟨*ident*⟩ ⟨*mbinder*⟩ ∗ ⟨*of-mod-type*⟩? (**:=** ⟨*mod-expr*⟩)?
| **Section** ⟨*ident*⟩
| **Chapter** ⟨*ident*⟩
| **End** ⟨*ident*⟩
| **Require** ⟨*export-token*⟩? ⟨*specif-token*⟩? ⟨*reference*⟩+
| **Require** ⟨*export-token*⟩? ⟨*specif-token*⟩? ⟨*string*⟩
| **Import** ⟨*reference*⟩+
| **Export** ⟨*reference*⟩+

⟨*export-token*⟩ ::= **Import** | **Export**

⟨*specif-token*⟩ ::= **Implementation** | **Specification**

⟨*mod-expr*⟩ ::= ⟨*reference*⟩
| ⟨*mod-expr*⟩ ⟨*mod-expr*⟩ *L*
| **(** ⟨*mod-expr*⟩ **)**

⟨*mod-type*⟩ ::= ⟨*reference*⟩
| ⟨*mod-type*⟩ **with** ⟨*with-declaration*⟩

⟨*with-declaration*⟩ ::= **Definition** ⟨*ident*⟩ **:=** ⟨*constr*⟩
| **Module** ⟨*ident*⟩ **:=** ⟨*reference*⟩

⟨*of-mod-type*⟩ ::= **:** ⟨*mod-type*⟩
| **<:** ⟨*mod-type*⟩

⟨*mbinder*⟩ ::= **(** ⟨*ident*⟩ + **:** ⟨*mod-type*⟩ **)**

⟨*gallina*⟩ ::= **Transparent** ⟨*reference*⟩+
| **Opaque** ⟨*reference*⟩+
| **Canonical Structure** ⟨*reference*⟩ ⟨*def-body*⟩?
| **Coercion Local**? ⟨*reference*⟩ ⟨*def-body*⟩
| **Coercion Local**? ⟨*reference*⟩ **:** ⟨*class-rawexpr*⟩ > − > ⟨*class-rawexpr*⟩
| **Identity Coercion Local**? ⟨*ident*⟩ **:** ⟨*class-rawexpr*⟩ > − > ⟨*class-rawexpr*⟩
| **Implicit Arguments** ⟨*reference*⟩ [⟨*num*⟩ ∗]
| **Implicit Arguments** ⟨*reference*⟩
| **Implicit Type** ⟨*ident*⟩ + **:** ⟨*constr*⟩

⟨*command*⟩ ::= **Comments** ⟨*comment*⟩∗
| **Pwd**
| **Cd** ⟨*string*⟩?
| **Drop** | **ProtectedLoop** | **Quit**
| **Load Verbose**? ⟨*ident*⟩
| **Load Verbose**? ⟨*string*⟩
| **Declare ML Module** ⟨*string*⟩+
| **Dump Universes** ⟨*string*⟩?
| **Locate** ⟨*locatable*⟩
| **Add Rec**? **LoadPath** ⟨*string*⟩ ⟨*as-dirpath*⟩?
| **Remove LoadPath** ⟨*string*⟩
| **Add Rec**? **ML Path** ⟨*string*⟩
| **Type** ⟨*constr*⟩
| **Print** ⟨*printable*⟩
| **Print** ⟨*reference*⟩
| **Inspect** ⟨*num*⟩
| **About** ⟨*reference*⟩
| **Search** ⟨*reference*⟩ ⟨*in-out-modules*⟩?
| **SearchPattern** ⟨*constr-pattern*⟩ ⟨*in-out-modules*⟩?
| **SearchRewrite** ⟨*constr-pattern*⟩ ⟨*in-out-modules*⟩?
| **SearchAbout** ⟨*reference*⟩ ⟨*in-out-modules*⟩?
| **SearchAbout** [⟨*ref-or-string*⟩ ∗]⟨*in-out-modules*⟩?
| **Set** ⟨*ident*⟩ ⟨*opt-value*⟩?
| **Unset** ⟨*ident*⟩
| **Set** ⟨*ident*⟩ ⟨*ident*⟩ ⟨*opt-value*⟩?
| **Set** ⟨*ident*⟩ ⟨*ident*⟩ ⟨*opt-ref-value*⟩+
| **Unset** ⟨*ident*⟩ ⟨*ident*⟩ ⟨*opt-ref-value*⟩∗
| **Print Table** ⟨*ident*⟩ ⟨*ident*⟩
| **Print Table** ⟨*ident*⟩
| **Add** ⟨*ident*⟩ ⟨*ident*⟩? ⟨*opt-ref-value*⟩+
| **Test** ⟨*ident*⟩ ⟨*ident*⟩? ⟨*opt-ref-value*⟩∗
| **Remove** ⟨*ident*⟩ ⟨*ident*⟩? ⟨*opt-ref-value*⟩+

⟨*check-command*⟩ ::= **Eval** ⟨*red-expr*⟩ **in** ⟨*constr*⟩
| **Check** ⟨*constr*⟩

⟨*ref-or-string*⟩ ::= ⟨*reference*⟩
| ⟨*string*⟩

⟨*printable*⟩ ::= **Term** ⟨*reference*⟩
| **All**
| **Section** ⟨*reference*⟩
| **Grammar** ⟨*ident*⟩
| **LoadPath**
| **Module Type**? ⟨*reference*⟩
| **Modules**
| **ML Path**
| **ML Modules**
| **Graph**
| **Classes**
| **Coercions**
| **Coercion Paths** ⟨*class-rawexpr*⟩ ⟨*class-rawexpr*⟩
| **Tables**
| **Hint** ⟨*reference*⟩?
| **Hint ***
| **HintDb** ⟨*ident*⟩
| **Scopes**
| **Scope** ⟨*ident*⟩
| **Visibility** ⟨*ident*⟩?
| **Implicit** ⟨*reference*⟩

⟨*class-rawexpr*⟩ ::= **Funclass** | **Sortclass** | ⟨*reference*⟩

⟨*locatable*⟩ ::= ⟨*reference*⟩
| **File** ⟨*string*⟩
| **Library** ⟨*reference*⟩
| ⟨*string*⟩

⟨*opt-value*⟩ ::= ⟨*ident*⟩ | ⟨*string*⟩

⟨*opt-ref-value*⟩ ::= ⟨*reference*⟩ | ⟨*string*⟩

⟨*as-dirpath*⟩ ::= **as** ⟨*reference*⟩

⟨*in-out-modules*⟩ ::= **inside** ⟨*reference*⟩+
| **outside** ⟨*reference*⟩+

⟨*comment*⟩ ::= ⟨*constr*⟩
| ⟨*string*⟩

5.4 Other commands

⟨*command*⟩ ::= ...
| **Debug On**
| **Debug Off**
| **Add setoid** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$
| **Add morphism** ⟨*constr*⟩$_9$ **:** ⟨*ident*⟩
| **Derive inversion_clear** ⟨*num*⟩? ⟨*ident*⟩ ⟨*ident*⟩
| **Derive inversion_clear** ⟨*ident*⟩ **with** ⟨*constr*⟩$_9$ (**Sort** ⟨*sort*⟩)?
| **Derive inversion** ⟨*num*⟩? ⟨*ident*⟩ ⟨*ident*⟩
| **Derive inversion** ⟨*ident*⟩ **with** ⟨*constr*⟩$_9$ (**Sort** ⟨*sort*⟩)?
| **Derive dependent inversion_clear** ⟨*ident*⟩ **with** ⟨*constr*⟩$_9$ (**Sort** ⟨*sort*⟩)?
| **Derive dependent inversion** ⟨*ident*⟩ **with** ⟨*constr*⟩$_9$ (**Sort** ⟨*sort*⟩)?
| *Extraction...*
| **Add Field** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*minus-div*⟩?
| **Functional Scheme** ⟨*ident*⟩ **:= Induction for** ⟨*constr*⟩$_9$ (**with** ⟨*constr*⟩$_9$ +)?
| **Add Ring** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ [⟨*constr*⟩$_9$ +]
| **Add Semi Ring** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ [⟨*constr*⟩$_9$ +]
| **Add Abstract Ring** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$
| **Add Abstract Semi Ring** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$
| **Add Setoid Ring** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ [⟨*constr*⟩$_9$ +]
| **Add Setoid Semi Ring** ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ ⟨*constr*⟩$_9$ [*tacconstr* +]

⟨*minus-div*⟩ ::= **with** ⟨*minus-arg*⟩ ⟨*div-arg*⟩
| **with** ⟨*div-arg*⟩ ⟨*minus-arg*⟩

⟨*minus-arg*⟩ ::= **minus :=** ⟨*constr*⟩$_9$

⟨*div-arg*⟩ ::= **div :=** ⟨*constr*⟩$_9$

⟨*command*⟩ ::= ...
| **Write State** ⟨*ident*⟩
| **Write State** ⟨*string*⟩
| **Restore State** ⟨*ident*⟩
| **Restore State** ⟨*string*⟩
| **Reset** ⟨*ident*⟩
| **Reset Initial**
| **Back** ⟨*num*⟩?

5.5 Proof-editing commands

⟨*command*⟩ ::= ...
| **Goal** ⟨*constr*⟩
| **Proof** ⟨*constr*⟩?
| **Proof with** ⟨*tactic*⟩
| **Abort All**?
| **Abort** ⟨*ident*⟩
| **Existential** ⟨*num*⟩ **:=** ⟨*constr-body*⟩
| **Qed**
| **Save** (⟨*thm-token*⟩ ⟨*ident*⟩)?
| **Defined** ⟨*ident*⟩?
| **Suspend**
| **Resume** ⟨*ident*⟩?
| **Restart**
| **Undo** ⟨*num*⟩?
| **Focus** ⟨*num*⟩?
| **Unfocus**
| **Show** ⟨*num*⟩?
| **Show Implicit Arguments** ⟨*num*⟩?
| **Show Node**
| **Show Script**
| **Show Existentials**
| **Show Tree**
| **Show Conjecture**
| **Show Proof**
| **Show Intro**
| **Show Intros**
| **Explain Proof Tree**? ⟨*num*⟩∗
| **Hint Local**? ⟨*hint*⟩ ⟨*inbases*⟩?

⟨*constr-body*⟩ ::= ⟨*type-cstr*⟩ **:=** ⟨*constr*⟩

⟨*hint*⟩ ::= **Resolve** ⟨*constr*⟩$_9$+
| **Immediate** ⟨*constr*⟩$_9$+
| **Unfold** ⟨*reference*⟩+
| **Constructors** ⟨*reference*⟩+
| **Extern** ⟨*num*⟩ ⟨*constr*⟩ ⇒ ⟨*tactic*⟩
| **Destruct** ⟨*ident*⟩ **:=** ⟨*num*⟩ ⟨*destruct-loc*⟩ ⟨*constr*⟩ ⇒ ⟨*tactic*⟩
| **Rewrite** ⟨*orient*⟩ ⟨*constr*⟩$_9$ + (**using** ⟨*tactic*⟩)?

⟨*inbases*⟩ ::= **:** ⟨*ident*⟩+

⟨*destruct-loc*⟩ ::= **Conclusion**
| **Discardable**? **Hypothesis**

5.6 Syntax extensions

⟨*syntax*⟩ ::= **Open Scope** ⟨*ident*⟩
| **Close Scope** ⟨*ident*⟩
| **Delimit Scope** ⟨*ident*⟩ **with** ⟨*ident*⟩
| **Bind Scope** ⟨*ident*⟩ **with** ⟨*class-rawexpr*⟩+
| **Arguments Scope** ⟨*reference*⟩ [⟨*name*⟩ +]
| **Infix Local**? ⟨*string*⟩ **:=** ⟨*reference*⟩ ⟨*modifiers*⟩? ⟨*in-scope*⟩?
| **Notation Local**? ⟨*string*⟩ **:=** ⟨*constr*⟩ ⟨*modifiers*⟩? ⟨*in-scope*⟩?
| **Notation Local**? ⟨*ident*⟩ **:=** ⟨*constr*⟩ **(only parsing)**?
| **Reserved Notation Local**? ⟨*string*⟩ ⟨*modifiers*⟩?
| **Tactic Notation** ⟨*string*⟩ ⟨*tac-production*⟩ ∗ **:=** ⟨*tactic*⟩

⟨*modifiers*⟩ ::= **(** ⟨*mod-list*⟩ **)**

⟨*mod-list*⟩ ::= ⟨*modifier*⟩
| ⟨*modifier*⟩ **,** ⟨*mod-list*⟩

⟨*modifier*⟩ ::= ⟨*ident*⟩ **at** ⟨*num*⟩
| ⟨*ident*⟩ **(,** ⟨*ident*⟩**)** ∗ **at** ⟨*num*⟩
| **at next level**
| **at level** ⟨*num*⟩
| **left associativity**
| **right associativity**
| **no associativity**
| ⟨*ident*⟩ ⟨*syntax-entry*⟩
| **only parsing**
| **format** ⟨*string*⟩

⟨*in-scope*⟩ ::= **:** ⟨*ident*⟩

⟨*syntax-entry*⟩ ::= **ident** | **global** | **bigint**

⟨*tac-production*⟩ ::= ⟨*string*⟩
| ⟨*ident*⟩ **(** ⟨*ident*⟩ **)**