1 Basic tactics

1.1 Apply Theorem

- **exact** term :
  exact P russit si pour un but t, et P de type u, t et u sont convertibles
- **assumption** :
- **refine** term :
- **apply** term [**with**$(ref_1 := term_1)...(ref_n := term_n)|term_1...term_n]$
-

2 tacticals

expr ::= expr ; expr
| expr ; [expr | ... | expr]
| tacexpr3

tacexpr3 ::= **do** (natural | ident) tacexpr3
| **progress** tacexpr3
| **repeat** tacexpr3
| **try** tacexpr3
| **timeout** (natural | ident) tacexpr3
| tacexpr2

tacexpr2 ::= tacexpr1 || tacexpr3
| tacexpr1

tacexpr1 ::= **fun** name name → atom
| **let** [**rec**] let_clause **with** **with** let_clause **in** atom
| **match goal with** context_rule | | context_rule **end**
| **match reverse goal with** context_rule | | context_rule **end**
| **match** expr **with** match_rule | | match_rule **end**
| **lazymatch goal with** context_rule | | context_rule **end**
| **lazymatch reverse goal with** context_rule | | context_rule **end**
| **lazymatch** expr **with** match_rule | | match_rule **end**
| **abstract** atom
| **abstract** atom **using** ident
| **first** [expr | | expr]
| **solve** [expr | | expr]
| **idtac** [message_token message_token]
| **fail** [natural] [message_token message_token]
| **fresh** | **fresh** string
| **context** ident [term]
| **eval** redexpr **in** term
| **type of** term
| **external** string string tacarg tacarg
| **constr** : term
| atomic_tactic
| qualid tacarg tacarg
| atom

atom ::= qualid
| ()
| integer
| (expr)

message_token ::= string | ident | integer

tacarg ::= qualid
| ()
| **ltac** : atom
| term

let_clause ::= ident [name name] := expr

context_rule ::= context_hyp , , context_hyp |-cpattern ⇒ expr
| |- cpattern ⇒ expr
| _⇒ expr

context_hyp ::= name : cpattern
| name := cpattern [: cpattern]

match_rule ::= cpattern ⇒ expr
| **context** [ident] [cpattern] ⇒ expr
| **appcontext** [ident] [cpattern] ⇒ expr
| _=¿ expr

top ::= [Local] **Ltac** ltac_def **with** **with** ltac_def

ltac_def ::= ident [ident ident] := expr
| qualid [ident ident] ::=expr