Features

- Show type/overload information at point
- On-the-fly syntax check
- Auto completion
- Jump to definition
- Set Lean options
- Eval Lean commands
- and More to come!
Configuration

https://github.com/leanprover/lean/blob/master/src/emacs/README.md
Show information at point (type, overloading, casting, etc)
Show type information of a sub-term in parens
(put a cursor on a open-paren)
On-the-fly syntax check
on-the-fly syntax check
C-c ! 1: show list of errors
Auto-completion with type tab
Auto-completion with type tab

private lemma p_implies_uv : p → u = v :=
  assume H : p,
  have Hpred : (∀ x, x = true v p) = (∀ x, x = false v p), from
  funext (take x : Prop, 
  have HL : (x = true v p) → (x = false v p), from
  assume A, or.inr Hp, 
  have Hr : (x = false v p) → (x = true v p), from
  assume A, or.inr Hp, 
  show (x = true v p) = (x = false v p), from
  propext Hl Hr),
  show u = v, from 
  Hpred → (eq.refl (epsilon (λ x, x = true v p))")

theorem em : p v ¬p :=
  have H : ¬(u = v) → ¬p, from mt p_implies_uv,
  or.elim uv_implies_p
    (assume Hne : ¬(u = v), or.inr (H Hne))

or.inr : b → a v b
or.intro_left : Π (b : Prop), a → a v b
or.intro_right : Π (a : Prop) {b : Prop}, b → a v b
or.induction_on : a v b → (a → c) → (b → c) → c
bool.induction_on : Π (n : bool), C bool.ff → C bool.tt → C n
tactic.expr.induction_on : Π (n : tactic.expr), C tactic.expr.builtin → C n
prod.rprod.intro : Ra a1 a2 → Rb b1 b2 → prod.rprod Ra Rb (prod.mk a1 b1) (prod.mk a2 b2)
char.induction_on : Π (n : char), (Π (a a a a a a a a : bool), C (char.mk a a a a a a a a)) → C n
not.intro : (a → false) → ¬a
option.induction_on : Π (n : option A), C option.none → (Π (a : A), C (option.some a)) → C n
prod.induction_on : Π (n : prod A B), (Π (pr1 : A) (pr2 : B), C (prod.mk pr1 pr2)) → C n
prod.rprod.induction_on : prod.rprod Ra Rb a a → (Π {a1 : A} {b1 : B} {a2 : A} {b2 : B}, Ra a1 a2 →...
have Hne : -(u = v), from Hv^{-1} \cdot Hut^{-1} \rightarrow true_ne_false, 
   or.inl Hne) 
   (assume Hp : p, or.inr Hp)) 
   (assume Hp : p, or.inr Hp)

private lemma p_implies_uv : p \rightarrow u = v := 
   assume Hp : p, 
   have Hpred : (A x, x = true v p) = (A x, x = false v p), from 
   funext (take x : Prop, 
   have Hl : (x = true v p) \rightarrow (x = false v p), from 
   assume A, or.inr Hp, 
   show (x = true v p) = (x = false v p), from 
   propext Hl Hr), 
   show u = v, from 
   Hpred \rightarrow (eq.refl (epsilon (A x, x = true v p)))

theorem em : p \lor \neg p := 
   have H : -(u = v) \rightarrow \neg p, from mt p_implies_uv, 
   or.elm uv_implies_p 
   (assume Hne : -(u = v), or.inr (H Hne)) 
   (assume Hp : p, or.inl Hp) 
   end
Can take a few seconds for the first time.
Jump to definition M-. 

M-* will pop you back!
Set lean options

C-c C-o
Set lean options
C-c C-o
Evaluate lean commands
C-c C-e
or.elim u_def
(assume Hut : u = true, or.elim v_def
 (assume Hvf : v = false,
  have Hne : ¬(u = v), from Hvf⁻¹ ∘ Hut⁻¹ ∘ true_ne_false,
  or.inl Hne)
 (assume Hp : p, or.inr Hp))
(assume Hp : p, or.inr Hp)

private lemma p_implies_uv : p → u = v :=
assume Hp : p,
 have Hpred : (λ x, x = true v p) = (λ x, x = false v p), from
 funext (take x : Prop,
  have Hl : (x = true v p) → (x = false v p), from
  assume A, or.inr Hp,
  have Hr : (x = false v p) → (x = true v p), from
  assume A, or.inr Hp,
  show (x = true v p) = (x = false v p), from
  propext Hl Hr),
 show u = v, from
 Hpred ∘ (eq.refl (epsilon (λ x, x = true v p)))

theorem em : p v ¬p :=
have H : ¬(u = v) → ¬p, from mt p_implies_uv,
or.elim uv_implies_p
 (assume Hne : ¬(u = v), or.inr (H Hne))
 (assume Hp : p, or.inl Hp)
end

Evaluate lean commands
C-c C-e
FAQs
Q: How can I type this symbol ‘▸’?
A: Press ‘C-c C-k’!
Q: It seems that nothing is working! What should I do?
A: Keep calm and run 
“M-x lean-server-restart-process”
then please file a bug report! 
(with reproducible steps)
Bug Reports, Feature Requests

https://github.com/leanprover/lean/issues/new

Contributions are Welcome!