lean-mode

emacs mode for Lean Theorem Prover

Soonho Kong
soonhok@cs.cmu.edu

Leonardo de Moura
leonardo@microsoft.com

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)

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

```
-- Copyright (c) 2014 Microsoft Corporation. All rights reserved.
-- Released under Apache 2.0 license as described in the file LICENSE.
-- Author: Leonardo de Moura

import logic.axioms.hilbert logic.axioms.funext
open eq.ops nonempty inhabited

-- Diaconescu's theorem
-- Show that Excluded middle follows from
--   Hilbert's choice operator, function extensionality and Prop extensionality
context
hypothesis propext {a b : Prop} : (a → b) → (b → a) → a = b
parameter  p : Prop

private definition u [reducible] := epsilon (λx, x = true ∨ p)

private definition v [reducible] := epsilon (λx, x = false ∨ p)

private lemma u_def : u = true ∨ p :=
epsilon_spec (exists.intro true (or.inl rfl))

private lemma v_def : v = false ∨ p :=
epsilon_spec (exists.intro false (or.inl rfl))

private lemma uv_implies_p : ¬(u = v) ∨ p :=
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 ∨ p) = (λ x, x = false ∨ p), from
    funext (take x : Prop,
      have Hl : (x = true ∨ p) → (x = false ∨ p), from
        assume A, or.inr Hp,
      have Hr : (x = false ∨ p) → (x = true ∨ p), from
        assume A, or.inr Hp.
```

```
-U:---  diaconescu.lean   Top (23,9)   Git  (Lean Hi ElDoc company MMM FlyC GitGutter Projectile[lean] MRev guru Fill) 04:56
epsilon_spec : ∀ (Hex : ∃ (x : Prop), x = false ∨ p), epsilon (λ (x : Prop), x = false ∨ p) = false ∨ p
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

```
private definition u [reducible] := epsilon (λx, x = true ∨ p)

private definition v [reducible] := epsilon (λx, x = false ∨ p)

private lemma u_def : u = true ∨ p :=
epsilon_spec (exists.intro true (or.inl rfl))

private lemma v_def : v = false ∨ p :=
epsilon_spec (exists.intro false (or.inl rfl))
```

Show type information of a sub-term in parens
(put a cursor on a open-paren)

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

theorem em : p ∨ ¬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
```

```
-U:---  diaconescu.lean    Bot (28,4)    Git  (Lean ElDoc company LeanDebug*-2 MMM FlyC GitGutter Projectile[lean] MRev guru Fi
: v p = false → ¬ u p = v p ∨ p
```

On-the-fly syntax check

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

```
-- Copyright (c) 2014 Microsoft Corporation. All rights reserved.
-- Released under Apache 2.0 license as described in the file LICENSE.
-- Author: Leonardo de Moura

import logic.axioms.hilbert logic.axioms.funext
open eq.ops nonempty inhabited

-- Diaconescu's theorem
-- Show that Excluded middle follows from
--   Hilbert's choice operator, function extensionality and Prop extensionality
context
hypothesis propext {a b : Prop} : (a → b) → (b → a) → a = b
parameter  p : Prop

private definition u [reducible] := epsilon (λx, x = true ∨ p)

private definition v [reducible] := epsilon (λx, x = false ∨ p)

private lemma u_def : u = true ∨ p :=
epsilon_spec (exists.intro true (or.inl rfl))

private lemma v_def : v = false ∨ p :=
epsilon_spec (exists.intro false (or.inl rfl))

private lemma uv_implies_p : ¬(u = v) ∨ p :=
!or.elim u_
  (assume     : u = true, or.elim v_def
    (assu   vf : v = false,
      have H    ¬(u = v), from Hvf⁻¹ ▸ Hut⁻¹ ▸ true_ne_false,
      or.inl H   )
    (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 ∨ p) = (λ x, x = false ∨ p), from
    funext (take x : Prop,
      have Hl : (x = true ∨ p) → (x = false ∨ p), from
        assume A, or.inr  ,
      have Hr : (x = f   ∨ p) → (x = true ∨ p), from
        assume A, o    Hp,
```

```
-UU:**-  diaconescu.le   Top (26,9)    Git  (Lean Hi ElDoc company MMM FlyC:2/0 GitGutter Projectile[lean] MRev guru Fill) 04
unknown identifier 'u_'
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

```
private definition v [reducible] := epsilon (λx, x = false ∨ p)

private lemma u_def : u = true ∨ p :=
epsilon_spec (exists.intro true (or.inl rfl))

private lemma v_def : v = false ∨ p :=
epsilon_spec (exists.intro false (or.inl rfl))

private lemma uv_implies_p : ¬(u = v) ∨ p :=
or.elim u_
  (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,
```

On-the-fly syntax check

C-c ! l : show list of errors

```
U:---  diaconescu.lean   33% (26,9)   Git  (Lean Hi ElDoc company MMM FlyC:2/0 GitGutter Projectile[lean] MRev guru Fill) 05
 Line Col Level     ID      Message (Checker)
   26   9 error             unknown identifier 'u_'... (lean-checker)
   49  11 error             unknown identifier 'uv_implies_p'... (lean-checker)

U:%%-  *Flycheck errors* for buffer diaconescu.lean   All (1,0)   (Flycheck errors Projectile[lean] MRev guru Fill) 05:03
unknown identifier 'u_'
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

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

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

Auto-completion with type tab

```
U:---  diaconescu.lean   Bot (50,33)   Git  (Lean ElDoc company MMM FlyC:1/0 GitGutter Projectile[lean] MRev guru Fill) 05:08
Quit
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

Auto-completion with type tab

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

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

```
or.inr : b → a ∨ b
or.inl : a → a ∨ b
or.intro_left : Π (b : Prop), a → a ∨ b
or.intro_right : ∀ (a : Prop) {b : Prop}, b → a ∨ b
or.induction_on : a ∨ 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 →...
```

ΠU:--- diaconescu.lean   Bot (50,33)   Git (Lean ElDoc company-lean--findp MMM FlyC:1/0 GitGutter Projectile[lean] MRev guru

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

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

theorem em : p ∨ ¬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
```

Jump to definition
M-.

```
-U:---  diaconescu.lean   Bot (49,7)   Git  (Lean Hi ElDoc company MMM FlyC GitGutter Projectile[lean] MRev guru Fill) 05:14
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

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

theorem em : p ∨ ¬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
```

Jump to definition
M-.

Can take a few seconds
for the first time.

```
U:---  diaconescu.lean    Bot (49,7)    Git  (Lean Hi ElDoc company MMM FlyC GitGutter Projectile[lean] MRev guru Fill) 05:12
Generating TAGS...
```

/Users/soonhok/work/lean/library/init/logic.lean

Jump to definition
M-.

M-* will pop you back!

```
calc_trans heq.of_heq_of_eq
calc_trans heq.of_eq_of_heq
calc_symm  heq.symm

/- and -/

notation a /\ b := and a b
notation a ∧ b  := and a b

variables {a b c d : Prop}

theorem and.elim (H₁ : a ∧ b) (H₂ : a → b → c) : c :=
and.rec H₂ H₁

/- or -/

notation a
notation a ∨ b :

namespace or
  theorem elim (H₁ : a ∨ b) (H₂ : a → c) (H₃ : b → c) : c :=
  rec H₂ H₃ H₁
end or

/- iff -/

definition iff (a b : Prop) := (a → b) ∧ (b → a)

notation a <-> b := iff a b
notation a ↔ b := iff a b

namespace iff
  definition intro (H₁ : a → b) (H₂ : b → a) : a ↔ b :=
  and.intro H₁ H₂

  definition elim (H₁ : (a → b) → (b → a) → c) (H₂ : a ↔ b) : c :=
  and.rec H₁ H₂

  definition elim_left (H : a ↔ b) : a → b :=
  elim (assume H₁ H₂, H₁) H
```

```
-U:---  logic.lean     26% (169,0)   Git  (Lean ElDoc company MMM FlyC GitGutter Projectile[lean] MRev guru Fill) 05:14
```

Minibuf-1

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

theorem em : p ∨ ¬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
```

Set lean options
C-c C-o

```
⊤U:---  diaconescu.lean   Bot (46,0)    Git  (Lean ElDoc company LeanDebug*-1 MMM FlyC GitGutter Projectile[lean] MRev guru Fi
Option name: {class.instance_max_depth | class.trace_instances | class.unique_instances | elaborator.calc_assistant | elabora
tor.fail_if_missing_field | elaborator.flycheck_goals | elaborator.ignore_instances | elaborator.local_instances | find_decl.
expensive | find_decl.max_steps | ...}
```

Minibuf-1

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

theorem em : p ∨ ¬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
```

Set lean options
C-c C-o

```
U:---  diaconescu.lean    Bot (46,0)    Git  (Lean ElDoc company LeanDebug*-1 MMM FlyC GitGutter Projectil[...]an] MRev guru Fi
class.instance_max_depth [(class) max allowed depth in class-instance resolution] : Unsigned Int (32) = 64
```

Minibuf-1

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

theorem em : p ∨ ¬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

```
U:---  diac…cu.lean   Bot (46,0)   Git  (Lean ElDoc company LeanDebug MMM FlyC GitGutter Projectile[lean] MRev guru Fill)
Lean CMD: check 3
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

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

theorem em : p ∨ ¬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

```
-U:  diaconescu.lean    Bot (46,0)    Git  (Lean ElDoc company LeanDebug*-1 MMM FlyC GitGutter Projectile[lean] MRev guru Fi
3 : num
```

FAQs

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

FAQ#1

Q: How can I type this symbol '▸' ?

```
assume Hp : p,
  have Hpred : (λ x, x = true ∨ p) = (λ x, x = false ∨ p), from
    funext (take x : Prop,
      have H1                    (x = false ∨ p), from

                                 e ∨ p), from

      show (                  se ∨ p), from
        propext     Hr),
  show u = v, from
    Hpred ▸ (eq.refl (epsilon (λ x, x = true ∨ p)))

theorem em : p ∨ ¬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
```

```
-U:---  diaconescu.lean   Bot (45,10)   Git  (Lean Hi ElDoc company LeanDebug*-1 MMM FlyC GitGutter Projectile[lean] MRev guru
▸ : (λ (x : Prop), x = true ∨ p) = λ (x : Prop),
  x = false ∨ p →
u p = epsilon (λ (x : Prop), x = true ∨ p) → u p = epsilon (λ (x : Prop), x = false ∨ p)
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

FAQ#1

A: Press ‘C-c C-k’!

```
assume Hp : p,
  have Hpred : (λ x, x = true ∨ p) = (λ x, x = false ∨ p), from
    funext (take x : Prop,
      have H1 … (x = false ∨ p), from
      … e ∨ p), from
      show … se ∨ p), from
        propext … Hr),
  show u = v, from
    Hpred ▸ (eq.refl (epsilon (λ x, x = true ∨ p)))

theorem em : p ∨ ¬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
```

```
U:---  diaconescu.lean    (45,10)   Git  (Lean Hi ElDoc company LeanDebug MMM FlyC GitGutter Projectile[lean] MRev guru Fi
To input `▸', type "\blacktriangleright", "\tb", "\t"
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

FAQ#2

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

theorem em : p ∨ ¬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
```

Q: It seems that nothing is working!
What should I do?

```
-U:---  diaconescu.lean   Bot (45,10)   Git  (Lean Hi ElDoc company LeanDebug*-1 MMM FlyC GitGutter Projectile[lean] MRev guru
▸ : (λ (x : Prop), x = true ∨ p) = λ (x : Prop),
  x = false ∨ p →
u p = epsilon (λ (x : Prop), x = true ∨ p) → u p = epsilon (λ (x : Prop), x = false ∨ p)
```

/Users/soonhok/work/lean/library/logic/axioms/examples/diaconescu.lean

FAQ#2

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

theorem em : p ∨ ¬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
```

A: Keep calm and run
"M-x lean-server-restart-process"

then please file a bug report!
(with reproducible steps)

```
∏U:---  diaconescu.lean   Bot (45,10)   Git  (Lean Hi ElDoc company LeanDebug*-1 MMM FlyC GitGutter Projectile[lean] MRev guru
▸ : (λ (x : Prop), x = true ∨ p) = λ (x : Prop),
  x = false ∨ p →
u p = epsilon (λ (x : Prop), x = true ∨ p) → u p = epsilon (λ (x : Prop), x = false ∨ p)
```

Bug Reports, Feature Requests

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

Contributions are Welcome!