Section 3 Solutions – Theorem Proving in Lean 4

Chun Ding • 20 May 2025

```
variable (p q r : Prop)

-- commutativity of ∧ and ∨
example : p ∧ q ↔ q ∧ p :=
  Iff.intro
    (fun ⟨hp, hq⟩ => ⟨hq, hp⟩)
    (fun ⟨hq, hp⟩ => ⟨hp, hq⟩)

example : p ∨ q ↔ q ∨ p :=
  Iff.intro
    (fun | Or.inl hp => Or.inr hp | Or.inr hq =>
        Or.inl hq)
    (fun | Or.inl hq => Or.inr hq | Or.inr hp =>
        Or.inl hp)

-- associativity of ∧ and ∨
example : (p ∧ q) ∧ r ↔ p ∧ (q ∧ r) :=
  Iff.intro
    (fun ⟨⟨hp, hq⟩, hr⟩ => ⟨hp, ⟨hq, hr⟩⟩)
    (fun ⟨hp, ⟨hq, hr⟩⟩ => ⟨⟨hp, hq⟩, hr⟩)
example : (p ∨ q) ∨ r ↔ p ∨ (q ∨ r) :=
  Iff.intro
    (fun | Or.inl (Or.inl hp) => Or.inl hp | Or.inl
        (Or.inr hq) => Or.inr (Or.inl hq) | Or.inr hr
        => Or.inr (Or.inr hr))
    (fun | Or.inl hp => Or.inl (Or.inl hp) | Or.inr
        (Or.inl hq) => Or.inl (Or.inr hq) | Or.inr
        (Or.inr hr) => Or.inr hr)

-- distributivity
example : p ∧ (q ∨ r) ↔ (p ∧ q) ∨ (p ∧ r) :=
  Iff.intro
    (fun | ⟨hp, Or.inl hq⟩ => Or.inl ⟨hp, hq⟩ | ⟨hp,
        Or.inr hr⟩ => Or.inr ⟨hp, hr⟩)
    (fun | Or.inl ⟨hp, hq⟩ => ⟨hp, Or.inl hq⟩ | Or.inr
        ⟨hp, hr⟩ => ⟨hp, Or.inr hr⟩)
```

```
example : p ∨ (q ∧ r) ↔ (p ∨ q) ∧ (p ∨ r) :=
  Iff.intro
    (fun | Or.inl hp => ⟨Or.inl hp, Or.inl hp⟩ |
        Or.inr ⟨hq, hr⟩ => ⟨Or.inr hq, Or.inr hr⟩)
    (fun | ⟨Or.inl hp, _ ⟩ => Or.inl hp | ⟨Or.inr hq,
        Or.inr hr⟩ => Or.inr ⟨hq, hr⟩ | ⟨_, Or.inl hp⟩
        => Or.inl hp)

-- other properties
example : (p → (q → r)) ↔ (p ∧ q → r) :=   Iff.intro
    (fun h ⟨hp, hq⟩ => h hp hq) -- h : p → (q → r),
        hp : p, hq : q
    (fun h hp hq => h ⟨hp, hq⟩)

example : ((p ∨ q) → r) ↔ (p → r) ∧ (q → r) :=
  Iff.intro
    (fun h => ⟨fun hp => h (Or.inl hp), fun hq => h
        (Or.inr hq)⟩)
    (fun ⟨hp, hq⟩ hr => Or.elim hr hp hq)

example : ¬(p ∨ q) ↔ ¬p ∧ ¬q := Iff.intro
    (fun h => ⟨fun hp => h (Or.inl hp), fun hq => h
        (Or.inr hq)⟩)
    (fun ⟨hp, hq⟩ hr => Or.elim hr hp hq)

example : ¬p ∨ ¬q → ¬(p ∧ q) :=
  fun h ⟨hp, hq⟩ => Or.elim h (fun hnp => hnp hp) (fun
        hnq => hnq hq)

example : ¬(p ∧ ¬p) :=
  fun ⟨hp, hnp⟩ => hnp hp

example : p ∧ ¬q → ¬(p → q) :=
  fun ⟨hp, hq⟩ h => hq (h hp)

example : ¬p → (p → q) :=
  fun hnp hp => False.elim (hnp hp)

example : (¬p ∨ q) → (p → q) :=
  fun h => Or.elim h (fun hnp hp => False.elim (hnp
        hp)) (fun hq _ => hq)

example : p ∨ False ↔ p := Iff.intro
```

```
    (fun | Or.inl hp => hp | Or.inr h => False.elim h)
    (fun hp => Or.inl hp)

example : p ∧ False ↔ False := Iff.intro
    (fun | ⟨_, h⟩ => False.elim h)
    (fun h => ⟨False.elim h, h⟩)

example : (p → q) → (¬q → ¬p) :=
  fun hpq hnq hp => hnq (hpq hp)
```

```
open Classical

variable (p q r : Prop)

example : (p → q ∨ r) → ((p → q) ∨ (p → r)) :=
  λ h => Or.elim (Classical.em q)
    (λ hq => Or.inl (λ _ => hq))
    (λ hnq => Or.inr (λ hp => Or.resolve_left (h hp)
        hnq))

example : ¬(p ∧ q) → ¬p ∨ ¬q :=
  fun h => Or.elim (Classical.em p)
    (λ hp => Or.inr (λ hq => h ⟨hp, hq⟩))
    (λ hnp => Or.inl hnp)

example : ¬(p → q) → p ∧ ¬q :=
  fun h => Or.elim (Classical.em p)
    (λ hp => Or.elim (Classical.em q)
      (λ hq => False.elim (h (λ _ => hq)))
      (λ hnq => ⟨hp, hnq⟩))
    (λ hnp => False.elim (h (λ hp => absurd hp hnp)))

example : (p → q) → (¬p ∨ q) :=
  fun h => Or.elim (Classical.em p)
    (λ hp => Or.inr (h hp))
    (λ hnp => Or.inl hnp)

example : (¬q → ¬p) → (p → q) :=
  fun h => Or.elim (Classical.em p)
    (λ hp => Or.elim (Classical.em q)
      (λ hq => λ _ => hq)
      (λ hnq => False.elim (h hnq hp)))
```

```
    (λ hnp => λ hp => False.elim (hnp hp))

example : p ∨ ¬p := em p

example : (((p → q) → p) → p) :=
  fun h => Or.elim (Classical.em p)
    (λ hp => hp)
    (λ hnp => absurd (h (λ hp => absurd hp hnp)) hnp)
```