# Constructive Inference Rules in Lean 4

**-> ->-Intro**

Term Mode:

```
theorem imp_intro (P Q : Prop) (h : P -> Q) : P -> Q := h
```

Tactic Mode:

```
theorem imp_intro (P Q : Prop) (h : P -> Q) : P -> Q := by exact h
```

**-> ->-Elim (MP)**

Term Mode:

```
theorem mp (P Q : Prop) (hpq : P -> Q) (hp : P) : Q := hpq hp
```

Tactic Mode:

```
theorem mp (P Q : Prop) (hpq : P -> Q) (hp : P) : Q := by exact hpq hp
```

**-> Modus Tollens (MT)**

Term Mode:

```
theorem mt (P Q : Prop) (hpq : P -> Q) (hnq : ~Q) : ~P := fun hp => hnq (hpq hp)
```

Tactic Mode:

```
theorem mt (P Q : Prop) (hpq : P -> Q) (hnq : ~Q) : ~P := by intro hp; exact hnq (hpq hp)
```

**/\ /\-Intro**

Term Mode:

```
theorem and_intro (P Q : Prop) (hp : P) (hq : Q) : P /\ Q := And.intro hp hq
```

Tactic Mode:

```
theorem and_intro (P Q : Prop) (hp : P) (hq : Q) : P /\ Q := by exact And.intro hp hq
```

**/\ /\-Elim Left**

Term Mode:

```
theorem and_elim_left (P Q : Prop) (h : P /\ Q) : P := h.left
```

Tactic Mode:

```
theorem and_elim_left (P Q : Prop) (h : P /\ Q) : P := by exact h.left
```

**/\ /\-Elim Right**

Term Mode:

```
theorem and_elim_right (P Q : Prop) (h : P /\ Q) : Q := h.right
```

Tactic Mode:

```
theorem and_elim_right (P Q : Prop) (h : P /\ Q) : Q := by exact h.right
```

**\/ \/-Intro Left**

Term Mode:

```
theorem or_intro_left (P Q : Prop) (hp : P) : P \/ Q := Or.inl hp
```

Tactic Mode:

```
theorem or_intro_left (P Q : Prop) (hp : P) : P \/ Q := by exact Or.inl hp
```

**\/ \/-Intro Right**

Term Mode:

```
theorem or_intro_right (P Q : Prop) (hq : Q) : P \/ Q := Or.inr hq
```

Tactic Mode:

```
theorem or_intro_right (P Q : Prop) (hq : Q) : P \/ Q := by exact Or.inr hq
```

**\/ \/-Elim**

Term Mode:

```
theorem or_elim (P Q R : Prop) (h : P \/ Q) (hp : P -> R) (hq : Q -> R) : R := Or.elim h hp hq
```

Tactic Mode:

theorem or_elim (P Q R : Prop) (h : P \/ Q) (hp : P -> R) (hq : Q -> R) : R := by cases h with | inl hp => exact hp _ | inr hq => exact hq _

**~ ~-Intro**

Term Mode:

theorem not_intro (P : Prop) (h : P -> False) : ~P := h

Tactic Mode:

theorem not_intro (P : Prop) (h : P -> False) : ~P := by exact h

**~ ~-Elim**

Term Mode:

theorem not_elim (P : Prop) (h : ~P) (hp : P) : False := h hp

Tactic Mode:

theorem not_elim (P : Prop) (h : ~P) (hp : P) : False := by exact h hp

**False Ex Falso**

Term Mode:

theorem ex_falso (P : Prop) (h : False) : P := False.elim h

Tactic Mode:

theorem ex_falso (P : Prop) (h : False) : P := by exact False.elim h

**<-> <->-Intro**

Term Mode:

theorem iff_intro (P Q : Prop) (h1 : P -> Q) (h2 : Q -> P) : P <-> Q := Iff.intro h1 h2

Tactic Mode:

theorem iff_intro (P Q : Prop) (h1 : P -> Q) (h2 : Q -> P) : P <-> Q := by exact Iff.intro h1 h2

**<-> <->-Elim Left**

Term Mode:

theorem iff_elim_left (P Q : Prop) (h : P <-> Q) : P -> Q := h.mp

Tactic Mode:

theorem iff_elim_left (P Q : Prop) (h : P <-> Q) : P -> Q := by exact h.mp

**<-> <->-Elim Right**

Term Mode:

theorem iff_elim_right (P Q : Prop) (h : P <-> Q) : Q -> P := h.mpr

Tactic Mode:

theorem iff_elim_right (P Q : Prop) (h : P <-> Q) : Q -> P := by exact h.mpr

**forall forall-Intro**

Term Mode:

theorem forall_intro (a : Type) (P : a -> Prop) (h : forall x, P x) : forall x, P x := h

Tactic Mode:

theorem forall_intro (a : Type) (P : a -> Prop) (h : forall x, P x) : forall x, P x := by exact h

**forall forall-Elim**

Term Mode:

theorem forall_elim (a : Type) (P : a -> Prop) (h : forall x, P x) (a : a) : P a := h a

Tactic Mode:

theorem forall_elim (a : Type) (P : a -> Prop) (h : forall x, P x) (a : a) : P a := by exact h a

**exists exists-Intro**

Term Mode:

theorem exists_intro (a : Type) (P : a -> Prop) (a : a) (h : P a) : exists x, P x := Exists.intro a h

Tactic Mode:

theorem exists_intro (a : Type) (P : a -> Prop) (a : a) (h : P a) : exists x, P x := by exact Exists.intro a h

**exists  exists-Elim**

Term Mode:

theorem exists_elim (a : Type) (P : a -> Prop) (R : Prop) (h : exists x, P x) (k : forall a, P a -> R) : R := Exists.elim h k

Tactic Mode:

theorem exists_elim (a : Type) (P : a -> Prop) (R : Prop) (h : exists x, P x) (k : forall a, P a -> R) : R := by cases h with | intro x hx => exact k x hx