

Natural Deduction in Coq

(Based on [Pfenning 2004], [Bertot and Castéran 2004], and [Coq Reference Manual])

Yih-Kuen Tsay

Dept. of Information Management
National Taiwan University



Overview

- 🌐 (Intuitionistic) Natural Deduction
- 🌐 Proof Terms
- 🌐 Typing/Inference Rules in Coq
- 🌐 Inductive Definitions in Coq
- 🌐 Introduction and Elimination in Coq



Natural Deduction in the Sequent Form

$$\frac{}{\Gamma, A \vdash A} \text{ (Hyp)}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \text{ (\wedge I)}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \text{ (\wedge E}_1\text{)}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \text{ (\wedge E}_2\text{)}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \text{ (\vee I}_1\text{)}$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \text{ (\vee E)}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \text{ (\vee I}_2\text{)}$$

Note: *I*: Introduction; *E*: Elimination.



Natural Deduction (cont.)

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} (\rightarrow I)$$

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} (\rightarrow E)$$

$$\frac{\Gamma, A \vdash B \wedge \neg B}{\Gamma \vdash \neg A} (\neg I)$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash \neg A}{\Gamma \vdash B} (\neg E)$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \neg \neg A} (\neg \neg I)$$

$$\frac{\Gamma \vdash \neg \neg A}{\Gamma \vdash A} (\neg \neg E)$$



Intuitionistic Natural Deduction

$$\frac{}{A_1, \dots, A_i, \dots, A_n \vdash A_i} \text{ (Hyp}^i\text{)}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \text{ (}\wedge I\text{)}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \text{ (}\wedge E_1\text{)}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \text{ (}\wedge E_2\text{)}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \text{ (}\vee I_1\text{)}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \text{ (}\vee I_2\text{)}$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \text{ (}\vee E\text{)}$$



Intuitionistic Natural Deduction (cont.)

$$\frac{}{\Gamma \vdash \top} (\top I)$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} (\rightarrow I)$$

$$\frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} (\rightarrow E)$$

$$\frac{\Gamma \vdash \perp}{\Gamma \vdash A} (\perp E)$$

Note:

- 🌐 The last rule says that, if we can deduce \perp (representing a contradiction), then we can deduce anything.
- 🌐 $\neg A$ is then represented (i.e., defined) as $A \rightarrow \perp$.
- 🌐 With the \rightarrow -elimination rule, one can deduce a contradiction (and hence anything) from $\neg A$ and A .



Quantifier Rules

$$\frac{\Gamma \vdash B[y/x]}{\Gamma \vdash \forall x B} (\forall I)$$

$$\frac{\Gamma \vdash \forall x B}{\Gamma \vdash B[t/x]} (\forall E)$$

$$\frac{\Gamma \vdash B[t/x]}{\Gamma \vdash \exists x B} (\exists I)$$

$$\frac{\Gamma \vdash \exists x B \quad \Gamma, B[y/x] \vdash C}{\Gamma \vdash C} (\exists E)$$

Note: In the quantifier rules above, we assume that all substitutions are admissible and y does not occur free in Γ or A .



Equality Rules

Let t, t_1, t_2 be arbitrary terms and again assume all substitutions are admissible.

$$\frac{}{\Gamma \vdash t = t} (= I) \qquad \frac{\Gamma \vdash t_1 = t_2 \quad \Gamma \vdash A[t_1/x]}{\Gamma \vdash A[t_2/x]} (= E)$$

Note: The $=$ sign is part of the object language, not a meta symbol.



Intuitionistic ND with Proof Terms

$$\frac{}{x_1 : A_1, \dots, x_i : A_i, \dots, x_n : A_n \vdash x_i : A_i} \text{ (Hyp}^i\text{)}$$

$$\frac{\Gamma \vdash t_1 : A \quad \Gamma \vdash t_2 : B}{\Gamma \vdash \mathbf{pair}(t_1, t_2) : A \wedge B} \text{ (}\wedge I\text{)}$$

$$\frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash \mathbf{fst}(t) : A} \text{ (}\wedge E_1\text{)}$$

$$\frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash \mathbf{snd}(t) : B} \text{ (}\wedge E_2\text{)}$$



Intuitionistic ND with Proof Terms (cont.)

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathbf{inl}(B, t) : A \vee B} (\vee I_1)$$

$$\frac{\Gamma \vdash t : B}{\Gamma \vdash \mathbf{inr}(A, t) : A \vee B} (\vee I_2)$$

$$\frac{\Gamma \vdash t : A \vee B \quad \Gamma, x : A \vdash t_1 : C \quad \Gamma, y : B \vdash t_2 : C}{\Gamma \vdash \mathbf{case}(t, x.t_1, y.t_2) : C} (\vee E)$$

Note: $\mathbf{case}(\mathbf{inl}(B, t), x.t_1, y.t_2) \stackrel{\Delta}{=} t_1[t/x]$;
 $\mathbf{case}(\mathbf{inr}(A, t), x.t_1, y.t_2) \stackrel{\Delta}{=} t_2[t/y]$.



Intuitionistic ND with Proof Terms (cont.)

$$\frac{}{\Gamma \vdash \mathbf{unit} : \top} (\top I)$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \mathbf{fun}(x : A) \Rightarrow t : A \rightarrow B} (\rightarrow I)$$

$$\frac{\Gamma \vdash t_1 : A \rightarrow B \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1 t_2 : B} (\rightarrow E)$$

Note: In the $(\rightarrow I)$ rule above, it is assumed that B does not contain x (so B is independent of A).

$$\frac{\Gamma \vdash t : \perp}{\Gamma \vdash \mathbf{abort}(A, t) : A} (\perp E)$$



An Example Proof

α :

$$\frac{\frac{}{x : (A \vee B) \rightarrow C, y : A \vdash y : A} (Hyp^2)}{x : (A \vee B) \rightarrow C, y : A \vdash \text{inl}(B, y) : A \vee B} (\vee I)$$

$$\frac{\frac{\frac{}{x : (A \vee B) \rightarrow C, y : A \vdash x : (A \vee B) \rightarrow C} (Hyp^1)}{x : (A \vee B) \rightarrow C, y : A \vdash x \text{ inl}(B, y) : C} (\rightarrow E)}{x : (A \vee B) \rightarrow C \vdash \text{fun}(y : A) \Rightarrow x \text{ inl}(B, y) : A \rightarrow C} (\rightarrow I)$$

$$\vdash \text{fun}(x : (A \vee B) \rightarrow C) \Rightarrow \text{fun}(y : A) \Rightarrow x \text{ inl}(B, y) : ((A \vee B) \rightarrow C) \rightarrow (A \rightarrow C) (\rightarrow)$$



Intuitionistic ND with Proof Terms (cont.)

$$\frac{\Gamma, y : A \vdash t : B[y/x]}{\Gamma \vdash \mathbf{fun}(x : A) \Rightarrow t : \forall x : A, B} (\forall I)$$

$$\frac{\Gamma \vdash t_1 : \forall x : A, B \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1 t_2 : B[t_2/x]} (\forall E)$$

$$\frac{\Gamma \vdash t_1 : A \quad \Gamma \vdash t_2 : B[t_1/x]}{\Gamma \vdash \mathbf{pair}(t_1, t_2) : \exists x : A, B} (\exists I)$$

$$\frac{\Gamma \vdash t' : \exists x : A, B \quad \Gamma, y : A, z : B[y/x] \vdash t : C}{\Gamma \vdash \mathbf{open}(t', y.z.t) : C} (\exists E)$$

Note: All substitutions are admissible and y does not occur free in Γ or A ; $\mathbf{open}(\mathbf{pair}(t_1, t_2), y.z.t) \triangleq t[t_2/z][t_1/y]$.



Curry-Howard Correspondence

Logic	Programming (Typed λ -Calculus)
proof	program (λ term)
proposition	type
proof checking	type checking



Typing/Inference Rules in Coq/pCIC

- 🌐 A type is expressed as a term.
- 🌐 Every term has a type.
- 🌐 The type of a type is called a *sort*.
- 🌐 Sorts in predicative Calculus of Inductive Constructions (pCIC, the type theory behind Coq):

$$\mathcal{S} = \{\text{Prop}, \text{Set}, \text{Type}(0), \text{Type}(1), \text{Type}(2), \dots\}$$

- ☀️ $\text{Prop} : \text{Type}(i)$, for every i .
- ☀️ $\text{Set} : \text{Type}(i)$, for every i .
- ☀️ $\text{Type}(i) : \text{Type}(j)$, for $i < j$.

Typing/Inference Rules in Coq (cont.)

Prod(\mathcal{S} , Prop, Prop)

$$\frac{E, \Gamma \vdash A : s \quad E, \Gamma :: (a : A) \vdash B : \text{Prop}}{E, \Gamma \vdash \forall a : A, B : \text{Prop}}$$

where $s \in \mathcal{S}$.

Prod({Prop, Set}, Set, Set) /* Set is predicative in pCIC. */

$$\frac{E, \Gamma \vdash A : s \quad E, \Gamma :: (a : A) \vdash B : \text{Set}}{E, \Gamma \vdash \forall a : A, B : \text{Set}}$$

where $s \in \{\text{Prop}, \text{Set}\}$.

Prod(Type, Type, Type)

$$\frac{E, \Gamma \vdash A : \text{Type}(i) \quad E, \Gamma :: (a : A) \vdash B : \text{Type}(j)}{E, \Gamma \vdash \forall a : A, B : \text{Type}(k)}$$

where $i \leq k$ and $j \leq k$.

Typing/Inference Rules in Coq (cont.)

$$\mathbf{Lam} \frac{E, \Gamma \vdash A \rightarrow B : s \quad E, \Gamma :: (v : A) \vdash e : B}{E, \Gamma \vdash \text{fun } v : A \Rightarrow e : A \rightarrow B}$$

Note: B does not depend on v .

$$\mathbf{App} \frac{E, \Gamma \vdash e_1 : A \rightarrow B \quad E, \Gamma \vdash e_2 : A}{\Gamma \vdash e_1 e_2 : B}$$

$$\mathbf{Lam} \frac{E, \Gamma \vdash \forall v : A, B : s \quad E, \Gamma :: (v : A) \vdash t : B}{E, \Gamma \vdash \text{fun } v : A \Rightarrow t : \forall v : A, B}$$

Note: “ $A \rightarrow B$ ” is just a shorthand for “ $\forall v : A, B$ ”, when B does not depend on v .

$$\mathbf{App} \frac{E, \Gamma \vdash t_1 : \forall v : A, B \quad E, \Gamma \vdash t_2 : A}{\Gamma \vdash t_1 t_2 : B[t_2/v]}$$



The Example Proof in Coq

- 🌐 The proposition to prove: $((A \vee B) \rightarrow C) \rightarrow (A \rightarrow C)$.
Formalization in Coq:

Variables A B C: Prop.

Lemma t0: $((A \vee B) \rightarrow C) \rightarrow (A \rightarrow C)$.

- 🌐 A proof in Coq:

```
intro x; intro y; apply x; left; assumption.
```

- 🌐 The proof term in Coq:

```
fun (x : A \vee B -> C) (y : A) => x (or_introl B y)
  : (A \vee B -> C) -> A -> C
```

cf.: $\text{fun}(x : (A \vee B) \rightarrow C) \Rightarrow \text{fun}(y : A) \Rightarrow x \text{ inl}(B, y)$
: $((A \vee B) \rightarrow C) \rightarrow (A \rightarrow C)$



\wedge -Introduction in Coq

🌐 Definitions for \wedge :

```
Inductive and (A : Prop) (B : Prop) : Prop :=  
  conj : A -> B -> A /\ B
```

Note: “ $A \ /\ \ B$ ” is a convenient notation for “and $A \ B$ ”.

🌐 Proposition to prove:

```
Lemma and_intro: A -> B -> (A /\ B).
```

🌐 A proof in Coq:

```
intro t1; intro t2; split; assumption; assumption.
```

🌐 The proof term in Coq:

```
and_intro = fun (t1 : A) (t2 : B) => conj t1 t2  
           : A -> B -> A /\ B
```



\wedge -Elimination in Coq

🌐 Definitions for \wedge :

```
and_ind =
```

```
fun A B P : Prop => and_rect (A:=A) (B:=B) (P:=P)  
  : forall A B P : Prop, (A -> B -> P) -> A /\ B -> P
```

🌐 Proposition to prove:

```
Lemma and_elim: (A /\ B) -> A.
```

🌐 A proof in Coq:

```
intro t; elim t; intro x; intro y; assumption.
```

🌐 The proof term in Coq:

```
fun t : A /\ B => and_ind (fun (x : A) (_ : B) => x) t  
  : A /\ B -> A
```



∨-Introduction in Coq

🌐 Definitions for ∨:

```
Inductive or (A : Prop) (B : Prop) : Prop :=  
  or_introl : A -> A \\/ B | or_intror : B -> A \\/ B
```

Note: “A \\/ B” is a convenient notation for “or A B”.

🌐 Proposition to prove:

```
Lemma or_intro: A -> (A \\/ B).
```

🌐 A proof in Coq:

```
intro x; left; assumption.
```

🌐 The proof term in Coq:

```
or_intro = fun x : A => or_introl B x  
          : A -> A \\/ B
```



∨-Elimination in Coq

🌐 Definitions for ∨:

```
or_ind =
fun (A B P : Prop) (f : A -> P) (f0 : B -> P)
  (o : A ∨ B) =>
match o with
| or_introl x => f x
| or_intror x => f0 x
end

: forall A B P : Prop, (A -> P) -> (B -> P)
  -> A ∨ B -> P
```



∨-Elimination in Coq (cont.)

🌐 Proposition to prove:

Lemma `or_elim`: $(A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow ((A \vee B) \rightarrow C)$.

🌐 A proof in Coq:

```
intro xt1; intro xt2; intro t; elim t;
assumption; assumption.
```

🌐 The proof term in Coq:

```
fun (xt1 : A -> C) (xt2 : B -> C) (t : A ∨ B)
  => or_ind xt1 xt2 t
  : (A -> C) -> (B -> C) -> A ∨ B -> C
```



T-Introduction in Coq

🌐 Definitions for T:

```
Inductive True : Prop := I : True
True_ind =
fun P : Prop => True_rect (P:=P)
      : forall P : Prop, P -> True -> P
```

🌐 Proposition to prove:

```
Lemma True_intro: True.
```

🌐 A proof in Coq:

```
exact I.
```

🌐 The proof term in Coq:

```
True_intro = I
      : True
```



\perp -Elimination in Coq

Definitions for \perp :

```
Inductive False : Prop := /* nothing */  
False_ind = fun P : Prop => False_rect P  
          : forall P : Prop, False -> P
```

Proposition to prove:

```
Lemma False_elim: False -> A.
```

A proof in Coq:

```
intro bot; elim bot.
```

The proof term in Coq:

```
False_elim = fun bot : False => False_ind A bot  
          : False -> A
```

\forall -Introduction and Elimination in Coq

🌐 \forall is a primitive in Coq (pCIC)

🌐 Proposition to prove:

```
Variable D: Set.
```

```
Variables P Q: D -> Prop.
```

```
Section All.
```

```
Hypothesis h: (forall x, P x).
```

```
Lemma all_intro: (forall y, P y).
```

```
End All.
```

🌐 A proof in Coq:

```
intro; apply h.
```

🌐 The proof term in Coq:

```
all_intro = fun y : D => h y
           : forall y : D, P y
```



\forall -Elimination in Coq

🌐 \forall is a primitive in Coq (pCIC)

🌐 Proposition to prove:

```
Section All_Elim.
```

```
Hypothesis h: (forall x, P x).
```

```
Variable t: D.
```

```
Lemma all_elim: P t.
```

```
End All_Elim.
```

🌐 A proof in Coq:

```
apply h.
```

🌐 The proof term in Coq:

```
all_elim = h t
```

```
: P t
```



∃-Introduction in Coq

🌐 Definitions for ∃:

```
Inductive ex (A : Type) (P : A -> Prop) : Prop :=  
  ex_intro : forall x : A, P x -> ex P
```

```
Notation "'exists' x , p" := (ex (fun x => p))  
  (at level 200, x ident, right associativity)  
  : type_scope.
```

🌐 Proposition to prove:

Variable t: D.

Lemma exists_intro: P t -> exists x, P x.



∃-Introduction in Coq (cont.)

- 🌐 A proof in Coq:

```
intro; exists t; assumption.
```

- 🌐 The proof term in Coq:

```
exists_intro =
```

```
fun H : P t => ex_intro (fun x : D => P x) t H  
  : P t -> exists x : D, P x
```



\exists -Elimination in Coq

Definitions for \exists :

```
ex_ind =  
fun (A : Type) (P : A -> Prop) (P0 : Prop)  
  (f : forall x : A, P x -> P0)  
  (e : ex P) => match e with  
    | ex_intro x x0 => f x x0  
  end  
: forall (A : Type) (P : A -> Prop) (P0 : Prop),  
  (forall x : A, P x -> P0) -> ex P -> P0
```

Proposition to prove:

```
Lemma exist_elim: (exists x, P x /\ Q x)  
  -> (exists x, P x).
```

∃-Elimination in Coq (cont.)

🌐 A proof in Coq:

```
intro; elim H; intro; intro; elim H0;  
intro; intro; exists x; assumption.
```

🌐 The proof term in Coq:

```
exists_elim =  
fun H : exists x : D, P x /\ Q x =>  
ex_ind  
  (fun (x : D) (H0 : P x /\ Q x) =>  
    and_ind (fun (H1 : P x) (_ : Q x) =>  
      ex_intro (fun x0 : D => P x0) x H1) H0) H  
  : (exists x : D, P x /\ Q x) -> exists x : D, P x
```



=-Introduction in Coq

🌐 Definitions for =:

```
Inductive eq (A : Type) (x : A) : A -> Prop :=  
  refl_equal : x = x
```

Note: “ $x = x$ ” is a convenient notation for “`eq x x`”.

🌐 Proposition to prove:

```
Lemma eq_intro: t1=t1.
```

🌐 A proof in Coq:

```
reflexivity.
```

🌐 The proof term in Coq:

```
eq_intro = refl_equal t1  
          : t1 = t1
```



=-Elimination in Coq

🌐 Definitions for =:

```
eq_ind =  
fun (A : Type) (x : A) (P : A -> Prop)  
  => eq_rect x P  
  : forall (A : Type) (x : A) (P : A -> Prop),  
    P x -> forall y : A, x = y -> P y
```

🌐 Proposition to prove:

```
Lemma eq_elim: t1=t2 -> t2=t1.
```

🌐 A proof in Coq:

```
intro; rewrite <- H; reflexivity.
```



=-Elimination in Coq (cont.)

🌐 The proof term in Coq:

```
eq_elim =  
fun H : t1 = t2 => eq_ind t1 (fun d : D => d = t1)  
                           (refl_equal t1) t2 H  
  : t1 = t2 -> t2 = t1
```

