

Tactics for Natural Deduction in Coq

The Coq proof assistant is based on (predicative) Calculus of Inductive Constructions (CIC), a combination of a higher-order logic and a richly-typed functional programming language. CIC is very expressive and can encode the whole of (intuitionistic) first-order logic. Proof construction using Coq can also be carried out in a manner very much like that using natural deduction in the sequent form.

	\wedge \wedge	\vee \vee	\rightarrow \rightarrow	\top <i>True</i>	\perp <i>False</i>	\neg \sim
Introduction	<code>split</code>	<code>left</code> <code>right</code>	<code>intro</code> <code>intro H</code>	<code>exact I</code>		<code>intro</code> <code>intro H</code>
Elimination	<code>apply H</code> <code>elim H</code>	<code>case H</code> <code>elim H</code>	<code>apply H</code>		<code>elim H</code>	<code>apply H</code> <code>elim H</code>

	\forall <code>forall</code>	\exists <code>exists</code>	$=$ $=$
Introduction	<code>intro</code> <code>intro H</code>	<code>exists v</code>	<code>reflexivity</code>
Elimination	<code>apply H</code>	<code>apply H</code> <code>elim H</code>	<code>rewrite <- H</code> <code>rewrite H</code>

(other tactics: `assumption`, `cut`, and `assert`, explained below)

In the above, H names, or gives name to, an assumption (which is a “higher-order term”) in the environment (i.e., the antecedent of the current sequent/goal); same for v (in `exists v`). In Coq, $\neg A$ is written as $\sim A$, which is defined to be $A \rightarrow \text{False}$ (i.e., $A \rightarrow \perp$).

Always end a tactic with a period “.” so that Coq knows the proof command is completed. Say “`assumption`” when you find the current goal to be an axiom as in the rules of Natural Deduction (appended below). It may occur that you want to apply the \rightarrow -Elimination rule (see the appendix), while $A \rightarrow B$ is not immediately available in the set Γ of assumptions. Use “`cut A`” in this case; enclose A in parentheses if it is a compound formula. A similar tactic is “`assert A,`” which you may find more convenient.

Appendix

Intuitionistic Natural Deduction

Below are the inference rules in the sequent form for intuitionistic first-order logic with equality.

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

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} (\wedge I) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} (\wedge E_1)$$

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

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} (\vee I_1) \qquad \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} (\vee E)$$

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

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

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

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

Note: the last rule says that, if we can deduce \perp (or *False*, representing a contradiction), then we can deduce anything.

Note: with the \rightarrow -Elimination rule, one can deduce a contradiction (and hence anything) from $\neg A$ (or $A \rightarrow \perp$) and A .

$$\frac{\Gamma \vdash A[y/x]}{\Gamma \vdash \forall x A} (\forall I) \qquad \frac{\Gamma \vdash \forall x A}{\Gamma \vdash A[t/x]} (\forall E)$$

$$\frac{\Gamma \vdash A[t/x]}{\Gamma \vdash \exists x A} (\exists I) \qquad \frac{\Gamma \vdash \exists x A \quad \Gamma, A[y/x] \vdash B}{\Gamma \vdash B} (\exists E)$$

Note: in the quantifier rules above, we assume that all substitutions are admissible. In Rule $\forall I$, y does not occur free in Γ or A , and in Rule $\exists E$, y does not occur free in Γ , A , or B .

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.