Basic commands

Require Import Bool Arith List.
Load libraries Bool, Arith and List

Definition name (arg : type):= body.
Defines a new constant called name.

Lemma name : statement.
State a lemma

Proof.
Start the proof of a lemma

Qed.
Terminate the proof of a lemma

SearchAbout (_ + _ = _ + _).
Search lemmas by pattern.

About lem.
Print the statement of lemma lem

Syntax for the statements

<table>
<thead>
<tr>
<th>Mathematics</th>
<th>Coq</th>
</tr>
</thead>
<tbody>
<tr>
<td>\neg p</td>
<td>\neg p</td>
</tr>
<tr>
<td>p \land q</td>
<td>p \land q</td>
</tr>
<tr>
<td>p \lor q</td>
<td>p \lor q</td>
</tr>
<tr>
<td>a = b</td>
<td>a = b</td>
</tr>
<tr>
<td>a \neq b</td>
<td>a \neq b</td>
</tr>
<tr>
<td>p \Rightarrow q</td>
<td>p \Rightarrow q</td>
</tr>
<tr>
<td>\forall x \in A. p(x)</td>
<td>\forall x \in A. p(x)</td>
</tr>
<tr>
<td>\exists x \in A. p(x)</td>
<td>\exists x \in A. p(x)</td>
</tr>
</tbody>
</table>

Example of a lemma

Lemma good_name :
forall (n m : nat), n - m + m = n.
Proof.
(* your proof *)
Qed.

Basic proof commands

reflexivity.
Prove an equational goal by trivial means, or fail

\[ 2 + 1 = 3 \rightarrow \]

exact p.
Prove a goal by using an assumption, or fail

\[ x : \text{nat} \]
\[ p : x = 0 \rightarrow \]
\[ x = 0 \]

intros x px.
Introduce x and P x naming them x and px

\[ \forall x, p x \rightarrow \]
\[ \forall x : T \]
\[ \forall x : P x \]
\[ P x \rightarrow Q x \rightarrow G \rightarrow \]
\[ Q x \rightarrow G \]

apply H.
Apply H to the current goal

\[ H : A \rightarrow B \rightarrow \]
\[ B \rightarrow \]

right

 rewrite \rightarrow Eab.
Rewrite with Eab left to right

\[ Eab : a = b \rightarrow \]
\[ P a \rightarrow P b \rightarrow \]

rewrite \leftarrow Eab.
Rewrite with Eab right to left

\[ Eab : a = b \rightarrow \]
\[ P b \rightarrow P a \rightarrow \]

unfold name.
If name is a Definition then it replaces name by its body.

Good practices

• Choose meaningful names for intros, not H1, H2.
• Structure the proof using indentation.