Cheat Sheet — lesson 2

Basic notations

Logic	Prop	bool
$\top, \bot$	`True, False`	`true, false`
$\neg p$	`~ p`	`~~ p`
$p \wedge q$	`p /\ q`	`p && q`
$p \vee q$	`p \/ q`	`p || q`
$a = b$	`a = b`	`a == b`
$a \neq b$	`a <> b`	`a != b`
$p \Rightarrow q$	`p -> q`	`p ==> q`
$\forall x \in A . \forall y . q(x,y)$	`forall (x:A) y, q x y`	
$\exists x \in A . p(x)$	`exists x:A, p x`	

Paper	Coq

Lemma 1. *For all natural numbers n and m if $m \leq n$ then the following equation holds:* $$n - m + m = n$$ *Proof.* Trivial $\square$	`Lemma good_name :`
	` forall n m,`
	`   m <= n ->`
	`   n - m + m = n.`
	`Proof.`
	`(* your proof *)`
	`Qed.`

Basic commands

`Require Import ssreflect ssrbool.`
Load libraries `ssreflect` and `ssrbool`

`Section name.`
Open a section

`Variable name : type.`
Declares a section variable

`Hypothesis name : statement.`
Declares a section assumption

`End name.`
Close a section

`Lemma name : statement.`
State a lemma

`Proof.`
Start the proof of a lemma

`Qed.`
Terminate the proof of a lemma

`Search _ (_ + _) "mul" modn.`
Search lemmas whose name contains the string `mul` and whose statement mentions the infix plus operation and contains the constant `modn`. $\boxed{\text{caveat}}$: always put an underscore after `Search`

`About lem.`
Print the statement of lemma `lem` and some informations associated with it, like the arguments that are marked as implicit

`Print def.`
Prints the body of the definition

Basic proof commands

`done.`
Prove the goal by trivial means, or fail

```
=========
 0 <= n          →
```

`exact: H.`
Apply `H` to the current goal and then assert all remaining goals, if any, are trivial. Equivalent to `by apply: H.`

```
 H : B
=========        →
 B
```

`move=> x px.`
Introduce `x` and `P x` naming them `x` and `px`

```
                         x : T
=========                px : P x
 forall x,          →   =========
  P x -> Q x -> G        Q x -> G
```

`apply: H.`
Apply `H` to the current goal

```
 H : A -> B              =========
=========           →    A
 B
```

`case: ab.`
Eliminate the conjunction or disjunction

```
 ab : A /\ B              =========
=========           →      A -> B -> G
 G
```

```
 ab : A \/ B         =========          =========
=========       →     A -> G             B -> G
 G
```

`case: exg3.`
Eliminate the existential quantification

```
 exg3 : exists n, 3 < n        =========
=========                 →     forall n,
 G                                3 < n -> G
```

`case: (P a).`
Reason by cases on a boolean predicate

```
                                 a : nat
 a : nat                         b : nat
 b : nat                         P : nat -> bool
 P : nat -> bool          →     =========
=========                        true && P b
 P a && P b is_true                  is_true
```

`split.`
Prove a conjunction

```
=========            =========          =========
 A /\ B         →     A                  B
```

`left.`
Prove a disjunction choosing the left part. `right` chooses the right part

```
=========                 =========
 A \/ B              →     A
```

`rewrite Eab.`
Rewrite with `Eab` left to right

```
 Eab : a = b              Eab : a = b
=========            →   =========
 P a                      P b
```

`rewrite -Eab.`
Rewrite with `Eab` right to left

```
 Eab : a = b              Eab : a = b
=========        →       =========
 P b                      P a
```

`have pa : P a.`
Open a new goal for `P a`. Once resolved introduce a new entry in the context for it named `pa`

```
                                    a : T
 a : T            a : T             pa : P a
=========   →    =========         =========
 G                P a               G
```

`suffices pa : P a.`
Open a new goal with an extra item in the context for `P a` named `pa`. When resolved, it asks to prove `P a`

```
                  a : T
 a : T            pa : P a          a : T
=========   →    =========         =========
 G                G                 P a
```

Proof General

`Ctrl-C Ctrl-Enter`
Goto the cursor

`Ctrl-X Ctrl-S`
Save

`Ctrl-X Ctrl-F`
Open a file

`Ctrl-X Ctrl-C`
Quit

Simple notations

```
"[ /\ P1 , P2 & P3 ]" := (and3 P1 P2 P3)
"[ \/ P1 , P2 | P3 ]" := (or3 P1 P2 P3)
"[ && b1 , b2 , .. , bn & c ]" :=
  (b1 && (b2 && .. (bn && c) .. ))
"[ || b1 , b2 , .. , bn | c ]" :=
  (b1 || (b2 || .. (bn || c) .. ))
"b1 (+) b2" := (addb b1 b2)
```