Cheat Sheet 1

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

Basic notations

Logic	Prop
$\top, \bot$	`True`, `False`
$\neg p$	`~ p`
$p \wedge q$	`p /\ q`
$p \vee q$	`p \/ q`
$a = b$	`a = b`
$a \neq b$	`a <> b`
$p \Rightarrow q$	`p -> q`
$p \Leftrightarrow q$	`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 □	`Theorem 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`

`Variable name : type.`
  Declares a variable

`Theorem name : statement.`
  State a theorem

`Proof.`
  Start the proof of a theorem

`Qed.`
  Terminate the proof of a theorem

Terminology

```
             ⎧P : nat->Prop
   Context   ⎨x : nat
             ⎩h : P x
   The bar    ===============
   Goal      {forall y, y = x -> P y
              \_____________/  \__/
                   Stack     Conclusion
```

Top is the head of the stack, here `y`

Basic proof commands

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

```
=========
 0 <= n             →
```

`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
```

(`move=> x _.` to throw away `P x` completely)

`move: x px.`
  Generalise `x` and `px`

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

(`move: x (px).` to leave a copy of `px` in the context)

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

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

`case: ab.`
  Eliminate the conjunction, disjunction or absurd

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

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

 ab : False
=========        →
 G
```

`case: exg3.`
  Eliminate the existential quantification

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

`case: x.`
  Perform a case analysis on `x`

```
 x : nat                                 =========
=========        →     =========          forall x,
 P x                    P 0                P (S x)
```

`elim: x.`
  Perform an induction on `x`

```
 x : nat                                 =========
=========        →     =========          forall x,
 P x                    P 0                P x -> P (S x)
```

`split.`
  Prove a conjunction

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

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

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

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

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

`exists n.`
Prove an existence giving a witness.

```
  n : nat               n : nat
 =========          →  =========
  exists x, P x         P n
```

`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
```

Notations for natural numbers: `nat`

```
"n.+1 :: " := (S n)
"n.-1 :: " := (predn n)
m + n := (addn m n)
m * n := (muln m n)
m <= n := (leq m n)
```