Quick reference to Coq commands and syntax

1 Programming

Loading preamble

```
Require Import ZArith List Bool Lia.
Open Scope Z_scope.
```

Loading preample for mathematical components

```
From mathcomp
  Require Import all_ssreflect.
```

Plain definition

```
Definition myfun (x : nat) :=
  x + 2.
```

Recursive definition

```
Fixpoint addl (l : list nat) :=
  match l with
  | nil => 0
  | a :: s => a + addl s
  end.
```

Unnamed function

```
fun x : nat => x + 2
```

Finding predefined functions

```
Search (Z -> Z).
Search (list _ -> list _).
```

Notations for list concatenation `l1 ++ l2`

Notations for numbers

addition	`x + y`
subtraction	`x - y`
multiplication	`x * y`
division	`x / y`
modulo	`x mod y`

Testing you own code

```
Check fun x => myfun x.
Compute addl (1 :: 2 :: nil).
```

2 Logical formulas

Example formula in Prop

```
exists l : list nat, l <> nil /\
  forall x,
    In x l -> x = 2 \/ In x + 3
```

3 Proofs

Starting a proof

```
Lemma addl_ge n l :
   In n l -> n <= addl l.
Proof.
```

Ending a proof.

```
Qed.
```

Finding an existing theorem

```
Search "sub".
Search "sub" (_ < _).
Search "sub" (_ < _) -positive.
Search "sub" (_ < _) outside ZMicromega.
```

4 Tactics

Tactics to interact with logical connectives

	⇒	∀	∧
Hypothesis `H`	`apply H`	`apply H`	`destruct H as [H1 H2]`
conclusion	`intros H`	`intros H`	`split`
	¬	∃	∨
Hypothesis `H`	exfalso; apply H	`destruct H as [x H1]`	`destruct H as [H1 | H2]`
conclusion	`intros H`	`exists` v	`left` or `right`
	=	`False`	
Hypothesis `H`	`rewrite H` `rewrite <- H`	`destruct H`	
conclusion	`reflexivity` `ring`		

Basic all-purpose tactics
`assert`, `exact e`, `assumption`
Tactics to control program computation

Expose function code : `unfold f`
Provoke recursive execution `simpl`,
`simpl expr`,`cbn[function_name]`
Choose final value : `change expr`,
`change e1 with e2`,
`replace e1 with e2`

Tactics to interact with data

Case based reasoning : `case e` or `destruct e as [ | a l]`
Case based reasoning : `destruct e as [ | a l] eqn:e_eq`
with memo. hypothesis
Data contradiction : `discriminate` or `discriminate H`
Decompose equality : `injection H`

5 Automatic tactics

Permanently available tactics `auto`, `eauto`, `tauto`, `intuition`, `firstorder`

Specialized tactics (available when loading a package)

equalities between expressions : `ring`
inequalities (linear) : `lia` (also uses hypotheses)