Coq Survival Kit

David Pichardie, Sandrine Blazy
ENS Rennes - IRISA - Université Rennes 1

Table of contents

Tactics used in Lecture3.v

destruct

on a hypothesis of the form P/\Q

```
H : P /\ Q
======================
R
```

destruct H

```
H : P
H0 : Q
======================
R
```

```
H : P /\ Q
======================
R
```

destruct H
as [H1 H2]

```
H1 : P
H2 : Q
======================
R
```

destruct

on a hypothesis of the form P\/Q

```
H : P \/ Q
======================
R
```

destruct H

first subgoal

```
H : P
======================
R
```

second subgoal

```
H : Q
======================
R
```

`destruct`

on a hypothesis of the form False

```
H : False
======================
P
```

`destruct H`

no more subgoal

destruct

on a term with an inductive type

```
n : nat
=====================
P n
```

destruct n

first subgoal

```
=====================
P 0
```

second subgoal

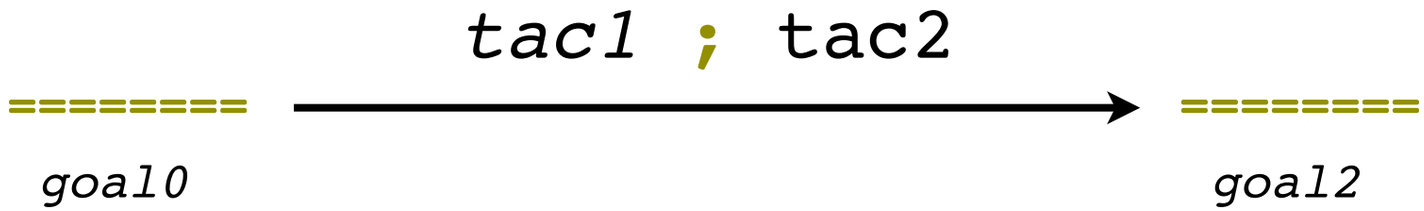
```
m : nat
=====================
P (S m)
```

left / right

```
            left
=====================  ------->  =====================
 P \/ Q                           P

            right
=====================  ------->  =====================
 P \/ Q                           Q
```

tac1 ; tac2



If *tac1* generates several subgoals, *tac2* is applied on each of them.

simpl

see also simpl in *

```
Definition pred (n:nat) :=
  match n with
    | O => O
    | S m => m
  end.
```

```
======================          simpl          ======================
 ... pred 0 ...                  ------>          ... 0 ...

======================          simpl          ======================
 ... pred (S n) ...              ------>          ... n ...
```

But the behavior of the command is not always that simple ...

intros

```
======================
 forall (a : A), P
```

intros

```
a : A
======================
 P
```

You should think about a term of type **Prop** as a logical property

```
P : Prop
======================
 P -> Q
```

intros

```
P : Prop
H : P
======================
 Q
```

```
======================
 P -> Q -> R
```

intros

```
H : P
H0 : Q
======================
 R
```

In Coq, we often use the form $P \Rightarrow Q \Rightarrow R$ instead of $P \wedge Q \Rightarrow R$

intros *names*

```
=====================          intros a          a : A
                            ------------->       =====================
 forall (a : A), P                                P
```

```
                                                 P : Prop
 P : Prop                      intros H          H : P
=====================       ------------->       =====================
 P -> Q                                           Q
```

```
                                                 H : P
                               intros H H0       H0 : Q
=====================       ------------->       =====================
 P -> Q -> R                                      R
```

```
                               intros H          H : P
=====================       ------------->       =====================
 not P                                            False
```

not P is a macro for P -> False

admit

```
======================
P
```

admit →

no more subgoal

- solve the current subgoal with an axiom
- *this is cheating!*

congruence

It solves automatically a subgoal using only the following deduction rules

$$\frac{}{x = x} \qquad \frac{x = y \quad (P\ x)}{(P\ y)} \qquad \frac{}{(C\ x) \neq (C'\ y)} \qquad \frac{(C\ x) = (C\ y)}{x = y}$$

where C and C' are constructors

Examples

```
n : nat
H : S n = S m
======================
plus n p = plus m p
```

congruence →

no more subgoal

```
n : nat
H : S n = O
======================
False
```

congruence →

no more subgoal

induction

on a term with an inductive type

```
n : nat
=====================
P n
```

induction n

first subgoal

```
=====================
P 0
```

second subgoal

```
m : nat
IHm : P m
=====================
P (S m)
```

assert

```
==========================

Q
```

assert (H: P)

first subgoal

```
==========================

P
```

second subgoal

```
H : P
==========================
Q
```

rewrite

```
H : a = b
======================
 ... a ...
```

rewrite H →

```
H : a = b
======================
 ... b ...
```

```
H : forall x y, f x y = x
=========================
 ... (f a b) ...
```

rewrite H →

```
H : forall x y, f x y = x
=========================
 ... a ...
```

Coq guesses how to instantiate the quantifiers

```
H : a = b
======================
 ... b ...
```

rewrite <- H →

```
H : a = b
======================
 ... a ...
```

omega

(do a `Require Import ZArith` before using it)

It solves automatically a subgoal using only arithmetic reasoning on `nat` and `Z`.
Beware, this is only for *linear arithmetic*: multiplication is only understood if one of the arguments is a numerical constant.

Examples

```
H : x <= y + 1
H0 : 2 * y <= z - 3
======================
2 * x +1 <= z
```

omega →

no more subgoal

```
============================
 x + (y + z) = (x + y) + z
```

omega →

no more subgoal

apply

```
H : P -> Q
======================
 Q
```

apply H

```
H : P -> Q
======================
 P
```

```
H : forall x y, P y -> Q x y
============================
 Q a (f a)
```

apply H

Coq guesses how to instantiate the quantifiers

```
H : forall x y, P y -> Q x y
============================
 P (f a)
```

```
H : forall x y, P x y -> Q y
============================
 Q (f a)
```

apply H with a

We have to help Coq and give him the missing instantiation

```
H : forall x y, P x y -> Q y
============================
 P a (f a)
```

Other useful tactics

assumption

```
H1 : P
H2 : Q
H3 : R
======================
P
```

P appears in the current hypotheses

assumption →

no more subgoal

unfold

replace a name by its definition

```
Definition succ (n:nat) :=  S n.
```

```
======================
 ... succ x ...
```

unfold succ →

```
======================
  ... S x ...
```

```
======================
 not P
```

unfold not →

```
======================
  P -> False
```

split

```
=====================
  P /\ Q
```

split

first subgoal

```
=====================
  P
```

second subgoal

```
=====================
  Q
```

```
=====================
  P <-> Q
```

split

first subgoal

```
=====================
  P -> Q
```

second subgoal

```
=====================
  Q -> P
```

exists

```
exists x, P x
```

exists t

```
P t
```

inv

can be loaded with the library MSMLib given

or by inserting `Ltac inv H := inversion H; clear H; try subst.`

```
Inductive le (n : nat) : nat -> Prop :=
  | le_n :
   (* ==== *)
     le n n
  | le_S m
     (Hle: n <= m):
   (* ==== *)
     le n (S m).
```

```
H: le n O
======================
P n
```

inv H →

```
======================
P O
```

```
H: le n m
======================
Q n m
```

inv H →

first subgoal

```
======================
Q m m
```

second subgoal

```
m' : nat
H' : le n m'
======================
Q n (S m')
```