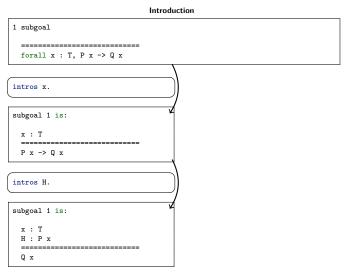
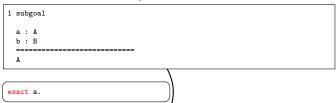
# Aide-mémoire des tactiques de preuve Coq

## Logique propositionnelle minimale

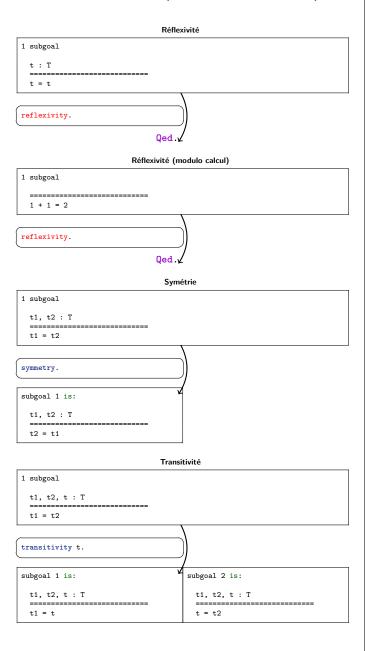


#### Terme de preuve fourni exactement



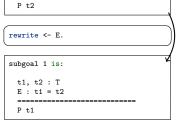
#### Application (d'hypothèse ou de lemme)

## Propriétés de l'égalité (relation d'équivalence)



### Utilisation d'une égalité

## 



#### Réécriture dans une hypothèse

### Types inductifs

p : T1

T21 p

.

#### Soit le code suivant définissant un type inductif

```
Inductive t (p : T1) :=
| C1 (p11 : T11 p) (p12 : T12 p p11)
| C2 (p21 : T21 p) (p22 : t p)
.
```

#### Application d'un constructeur



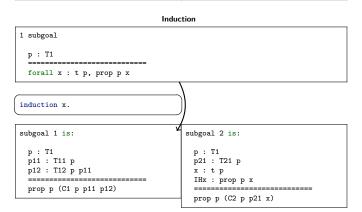
### Analyse par cas

p : T1

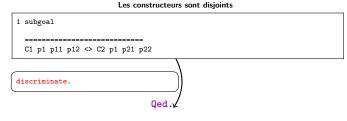
.

```
1 subgoal
p: T1
-------
forall x: t p, prop p x
```

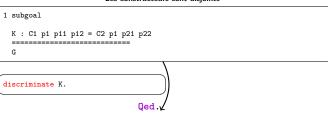




## Égalité et propriétés des constructeurs

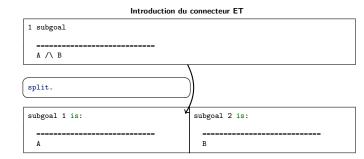


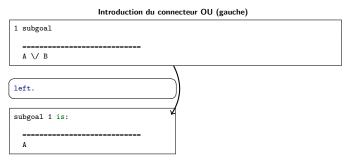
#### Les constructeurs sont disjoints

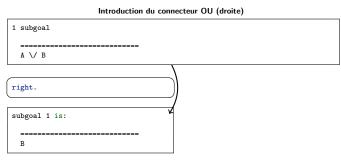


#### Chaque constructeur est injectif

## Connecteurs et logique propositionnelle







### Introduction du connecteur existentiel



```
subgoal 1 is:

e: T

Pe
```

#### Élimination du connecteur ET

#### Élimination du connecteur OU

#### Élimination du connecteur existentiel

```
1 subgoal

H: exists x: T, P x

G

destruct H as [x Hx].

subgoal 1 is:

x: T

Hx: P x
```

### Réduction

```
Dépliage de définition (unfold)
1 subgoal
 a : A
 -----
 ~ ~ A
unfold not.
subgoal 1 is:
 _____
 (A -> False) -> False
intros H.
subgoal 1 is:
 H : A -> False
 False
apply H.
subgoal 1 is:
 a : A
 H : A -> False
 _____
exact a.
                        Qed.
                     Simplification du but
1 subgoal
 1 : list A
 _____
 [a] ++ 1 = a :: 1
simpl.
subgoal 1 is:
 1 : list A
 _____
 a :: 1 = a :: 1
```

### **Automatisation**

### Arithmétique linéaire

```
Require Import Omega.

Lemma example:
forall (n p : nat), 2 * (n + 1) - n + p > n.

intros n p; omega.
```

#### Tautologies propositionnelles

```
Lemma iff_implies:
    forall (P1 P2 : Prop), (P1 <-> P2) <-> ((P1 -> P2) /\ (P2 -> P1)).

tauto.

Qed./
```

#### Tautologies du premier ordre

```
(* Aucun violoniste n'est punk ;
   Or quelques punks sont musiciens ;
   Donc quelques musiciens ne sont pas violonistes. *)
Lemma Fresison :
   forall (T : Type) (violoniste punk musicien : T -> Prop),
   (forall v, violoniste v -> - punk v) ->
   (exists p, punk p /\ musicien p) ->
   exists m, musicien m /\ - violoniste m.
```



1 subgoal

G