# Tactiques

David Delahaye
David.Delahaye@lirmm.fr

Université de Montpellier
Faculté des Sciences

Licence Informatique L3 2022-2023

# La tactique `apply` : la connaît-on vraiment ?

## Application non dépendante

```
1 subgoal
  H : A -> B
  H0 : A
  ============================
  B
```

## Remarques

- La conclusion doit être la même que la conclusion de l'hypothèse.
- Ici, on peut donc appliquer $H$.

# La tactique apply : la connaît-on vraiment ?

## Application non dépendante

```
Coq < apply H.
1 subgoal
  H : A -> B
  H0 : A
  ============================
  A
```

## Remarques

- L'application est possible mais il faut démontrer la prémisse de $H$.
- En effet, ne pas oublier la règle du modus ponens : si on a $A \rightarrow B$, on peut en déduire $B$ uniquement si on a $A$.
- On peut aussi voir $H$ comme une fonction : elle rend du $B$ mais on doit lui donner du $A$ pour cela.

# La tactique `apply` : la connaît-on vraiment ?

## Application non dépendante

```
1 subgoal
  H : A -> B
  H0 : A
  ============================
  B

Coq < apply (H H0).
No more subgoals.
```

## Remarques

- Si on voit $H$ comme une fonction, on peut directement lui donner son argument de type $A$, ici l'hypothèse $H0$.

# La tactique `apply` : la connaît-on vraiment ?

## Application dépendante

```
1 subgoal
  H : forall x : E, P x
  ============================
  P a
```

## Remarques

- Il faut voir l'hypothèse $H$ comme une implication.
- Sauf qu'ici, l'implication est dépendante (variable $x$).
- En théorie des types, on parle de produit : produit non dépendant pour l'implication ($\rightarrow$) et produit dépendant (*forall*).
- L'implication est un cas dégénéré du produit dépendant où la variable n'a pas d'occurrence dans le reste de la formule.

# La tactique apply : la connaît-on vraiment ?

## Application dépendante

```
1 subgoal
 H : forall x : E, P x
 ============================
 P a
```

## Remarques

- Pour appliquer $H$, les conclusions doivent maintenant s'unifier et non plus seulement être les mêmes (comme dans le cas non dépendant).
- On doit trouver des termes pour toutes les variables de la conclusion de $H$ (ici $x$) permettant de rendre les conclusions identiques.
- Dans ce cas, les conclusions s'unifient ($x$ peut en effet être remplacé par $a$) et on peut appliquer $H$.

# La tactique apply : la connaît-on vraiment ?

## Application dépendante

```
1 subgoal
  H : forall x : E, P x
  ============================
  P a

Coq < apply H.
No more subgoals.
```

## Remarques

- La preuve est terminée immédiatement : pourquoi ne pas demander de démontrer $E$ comme dans le cas non dépendant ?
- Car la valeur de $x$ a été trouvée par unification.
- Toutes les variables quantifiées dans $H$ doivent être trouvées par unification sinon la tactique **apply** ne peut pas s'appliquer.

# La tactique `eapply` : des trous dans les formules ?

## Application en donnant tous les arguments

```
1 subgoal
  H : forall x y z : E, P x y -> P y z -> P x z
  H0 : P a b
  H1 : P b c
  ============================
  P a c
```

## Remarques

- On souhaiterait appliquer $H$ avec la tactique `apply`.
- La conclusion de $H$, à savoir $P\ x\ z$, s'unifie bien avec la conclusion du but $P\ a\ c$ (avec $x$ qui vaut $a$ et $z$ qui vaut $c$).
- Mais $y$ qui est quantifiée dans $H$ reste inconnue et la tactique **apply** va tout simplement échouer.

# La tactique eapply : des trous dans les formules ?

## Application en donnant tous les arguments

```
1 subgoal
  H : forall x y z : E, P x y -> P y z -> P x z
  H0 : P a b
  H1 : P b c
  ============================
  P a c

Coq < apply H.
Toplevel input, characters 6-7:
> apply H.
>       ^
Error: Unable to find an instance for the variable y.
```

## Remarques

- Pour résoudre le problème, on peut donner plus d'informations à $H$.

# La tactique eapply : des trous dans les formules ?

## Application en donnant tous les arguments

```
1 subgoal
  H : forall x y z : E, P x y -> P y z -> P x z
  H0 : P a b
  H1 : P b c
  ============================
  P a c

Coq < Unnamed_thm < apply (H a b c H0 H1).
No more subgoals.
```

## Remarques

- On peut donner tous les arguments à l'hypothèse $H$.

# La tactique `eapply` : des trous dans les formules ?

## Application en donnant les arguments non instanciés

```
1 subgoal
  H : forall x y z : E, P x y -> P y z -> P x z
  H0 : P a b
  H1 : P b c
  ============================
  P a c
```

## Remarques

- On peut aussi donner juste la valeur de $y$ qui ne peut être devinée par unification mais cela réclame de savoir où l'on va dans la preuve.

# La tactique eapply : des trous dans les formules ?

## Application en donnant les arguments non instanciés

```
Coq < apply H with b.
2 subgoals
  H : forall x y z : E, P x y -> P y z -> P x z
  H0 : P a b
  H1 : P b c
  ============================
  P a b

subgoal 2 is:
 P b c
```

## Remarques

- On donne directement la valeur de $y$ après le **with**.

# La tactique `eapply` : des trous dans les formules ?

## Application en donnant aucun argument

```
1 subgoal
  H : forall x y z : E, P x y -> P y z -> P x z
  H0 : P a b
  H1 : P b c
  ============================
  P a c
```

## Remarques

- Mais on peut faire encore mieux.
- On peut retarder l'instanciation de $y$ et essayer de la deviner plus tard lorsqu'on aura plus avancé dans la preuve.

# La tactique eapply : des trous dans les formules ?

## Application en donnant aucun argument

```
Coq < eapply H.
2 focused subgoals
(shelved: 1)
  H : forall x y z : E, P x y -> P y z -> P x z
  H0 : P a b
  H1 : P b c
  ============================
  P a ?y

subgoal 2 is:
 P ?y c
```

## Remarques

- On a introduit une métavariable $?y$, c'est un trou dans la formule.

# La tactique eapply : des trous dans les formules ?

## Application en donnant aucun argument

```
Coq < apply H0.
1 subgoal
  H : forall x y z : E, P x y -> P y z -> P x z
  H0 : P a b
  H1 : P b c
  ============================
  P b c
```

## Remarques

- On peut trouver la valeur de $?y$ par unification en utilisant $H0$.
- Cette valeur est ensuite remplacée partout ailleurs dans les autres buts.
- On a bien retardé l'instanciation de $y$ dans $H$.
- À noter qu'on ne peut pas instancier manuellement une métavariable, on l'instancie uniquement par unification (donc indirectement).

# Les tactiques d'induction : plusieurs tactiques ?

## La tactique de base `elim`

```
1 subgoal
  n : nat
  ============================
  P n
```

## Remarques

- La tactique de base pour déclencher une induction est `elim`.
- Elle prend en argument une hypothèse dont le type est un type inductif (type de données ou relation inductive).

# Les tactiques d'induction : plusieurs tactiques ?

## La tactique de base `elim`

```
Coq < elim n.
2 subgoals
  n : nat
  ============================
  P 0

subgoal 2 is:
 forall n0 : nat, P n0 -> P (S n0)
```

## Remarques

- Les buts en question sont les prémisses du schéma d'induction.
- Aucune introduction (**intro(s)**) n'est faite.

# Les tactiques d'induction : plusieurs tactiques ?

## Une alternative : la tactique `induction`

```
1 subgoal
______________________________________
forall n : nat, P n
```

## Remarques

- La tactique **induction** est une alternative à la tactique **elim**.
- Mais elle se fait sur une variable quantifiée universellement du but et dont le type est un type inductif.

# Les tactiques d'induction : plusieurs tactiques ?

## Une alternative : la tactique `induction`

```
Coq < induction n.
2 subgoals
  ============================
   P 0

subgoal 2 is:
 P (S n)
```

## Remarques

- On obtient les mêmes buts qu'auparavant.
- Sauf que la tactique **intros** est appliquée à tous ces buts.

# Les tactiques d'induction : plusieurs tactiques ?

## Une alternative : la tactique `induction`

```
Coq < Show 2.
subgoal 2 is:
  n : nat
  IHn : P n
  ============================
  P (S n)
```

## Remarques

- On voit bien que l'hypothèse d'induction *IHn* a été introduite.
- Au passage, la commande **Show** affiche le but dont le numéro est indiqué en paramètre de la commande (Coq n'affiche entièrement qu'un seul but ; pour les autres, il affiche seulement la conclusion).

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
Coq < Inductive mul3 : nat → Prop :=
| T0 : mul3 0
| T3 : forall n, mul3 n → mul3 (3 + n).
mul3 is defined
mul3_ind is defined
```

## Remarques

- Les relations inductives sont dans la sorte **Prop** (formules).
- Elles ne sont donc pas nommées (produit non dépendant).
- On ne peut donc pas utiliser **induction** avec un nom de variable.
- Il faut introduire l'hypothèse inductive en question et utiliser **elim**.

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
Coq < Goal forall n : nat, mul3 n ->
           exists p : nat, n = 3 * p.
1 subgoal
  ============================
   forall n : nat, mul3 n -> exists p : nat, n = 3 * p

Coq < intros.
1 subgoal
  n : nat
  H : mul3 n
  ============================
   exists p : nat, n = 3 * p
```

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
Coq < elim H.
2 subgoals
  n : nat
  H : mul3 n
  ============================
  exists p : nat, 0 = 3 * p

subgoal 2 is:
 forall n0 : nat,
 mul3 n0 -> (exists p : nat, n0 = 3 * p) ->
 exists p : nat, 3 + n0 = 3 * p
```

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
2 subgoals
  n : nat
  H : mul3 n
  ============================
  exists p : nat, 0 = 3 * p

Coq < exists 0.
2 subgoals
  n : nat
  H : mul3 n
  ============================
  0 = 3 * 0

Coq < lia.
```

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
1 subgoal
  n : nat
  H : mul3 n
  ============================
  forall n0 : nat,
  mul3 n0 -> (exists p : nat, n0 = 3 * p) ->
  exists p : nat, 3 + n0 = 3 * p
```

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
Coq < intros.
1 subgoal
  n : nat
  H : mul3 n
  n0 : nat
  H0 : mul3 n0
  H1 : exists p : nat, n0 = 3 * p
  ============================
  exists p : nat, 3 + n0 = 3 * p
```

## Remarques

- Il faut trouver le $p$ tel que $3 + n0 = 3 * p$.
- Pour ce faire, il faut d'abord exhiber celui de $n0$.
- On utilise l'hypothèse de récurrence ($H1$) pour cela.

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
Coq < elim H1; intros.
1 subgoal
  n : nat
  H : mul3 n
  n0 : nat
  H0 : mul3 n0
  H1 : exists p : nat, n0 = 3 * p
  x : nat
  H2 : n0 = 3 * x
  ============================
  exists p : nat, 3 + n0 = 3 * p
```

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
Coq < rewrite H2.
1 subgoal
  n : nat
  H : mul3 n
  n0 : nat
  H0 : mul3 n0
  H1 : exists p : nat, n0 = 3 * p
  x : nat
  H2 : n0 = 3 * x
  ============================
   exists p : nat, 3 + 3 * x = 3 * p
```

## Remarques

- Le $p$ à exhiber est donc $x + 1$.

# Les tactiques d'induction : plusieurs tactiques ?

## Induction sur les relations inductives

```
Coq < exists (x + 1).
1 subgoal
  n : nat
  H : mul3 n
  n0 : nat
  H0 : mul3 n0
  H1 : exists p : nat, n0 = 3 * p
  x : nat
  H2 : n0 = 3 * x
  ============================
  3 + 3 * x = 3 * (x + 1)

Coq < lia.
No more subgoals.
```

# Les tactiques d'induction : plusieurs tactiques ?

## Alternative à **elim** pour les relations inductives

```
Coq < Goal forall n : nat, mul3 n →
           exists p : nat, n = 3 * p.
1 subgoal
  ============================
   forall n : nat, mul3 n → exists p : nat, n = 3 * p
```

## Remarques

- En fait, même si *mul*3 *n* n'est pas nommé, on peut utiliser **induction**.
- Comme, on n'a pas de nom, on utilise la position.
- On va indiquer à **induction** la position de l'inductif (non nommé) sur lequel on veut fait l'induction (ici, la position est 1).

# Les tactiques d'induction : plusieurs tactiques ?

## Alternative à **elim** pour les relations inductives

```
Coq < induction 1.
2 subgoals
  ============================
   exists p : nat, 0 = 3 * p

subgoal 2 is:
 exists p : nat, 3 + n = 3 * p

Coq < Show 2.
subgoal 2 is:
  n : nat
  H : mul3 n
  IHmul3 : exists p : nat, n = 3 * p
  ============================
   exists p : nat, 3 + n = 3 * p
```

# L'induction à tous les étages

## Puissance d'expression de l'induction

- Avec l'induction, on peut encoder beaucoup de constructions.
- Ce qui est primitif : les produits, le faux et les inductifs.
- Tout le reste est encodé en majeure partie grâce à l'induction.

## Quelques exemples

```
Inductive and (A B : Prop) : Prop :=
    conj : A -> B -> A /\ B.

Inductive or (A B : Prop) : Prop :=
    or_introl : A -> A \/ B | or_intror : B -> A \/ B.

Inductive ex (A : Type) (P : A -> Prop) : Prop :=
    ex_intro : forall x : A, P x -> exists y, P y.
```

# La tactique **auto** : une tactique mystérieuse ?

## Un exemple simple

```
Coq < Goal forall (P Q : nat → Prop),
  (forall n, Q n → P n) →
  (forall n, Q n) →
  P 2.
```

## Remarques

- La tactique **auto** est capable de résoudre un but qui peut être prouvé à l'aide d'une séquence de plusieurs tactiques, à savoir **intros**, **apply**, **assumption** et **reflexivity**.

# La tactique **auto** : une tactique mystérieuse ?

## Un exemple simple

```
1 subgoal
============================
 forall P Q : nat → Prop,
 (forall n : nat, Q n → P n) → (forall n : nat, Q n) →
 P 2

Coq < auto.
No more subgoals.
```

## Remarques

- Mais qu'a fait **auto** ?
- On aimerait avoir le détail des tactiques élémentaires appliquées.
- Pour ce faire, on utilise la tactique **info_auto**.

# La tactique **auto** : une tactique mystérieuse ?

## Un exemple simple

```
1 subgoal
  ============================
   forall P Q : nat -> Prop,
   (forall n : nat, Q n -> P n) -> (forall n : nat, Q n) ->
   P 2

Coq < info_auto.
(* info auto: *)
intro.
intro.
intro.
intro.
apply H.
 apply H0.
No more subgoals.
```

# La tactique **auto** : une tactique mystérieuse ?

## Utiliser des lemmes avec **auto**

```
Parameters P Q : nat -> Prop.
Axiom ax1 : forall n, Q n -> P n.
Axiom ax2 : forall n, Q n.

Coq < Goal P 2.
1 subgoal
  ============================
  P 2
```

## Remarques

- La tactique **auto** utilise toutes les hypothèses pour prouver le but.
- Mais on a la possibilité de lui indiquer d'utiliser un lemme.

# La tactique **auto** : une tactique mystérieuse ?

## Utiliser des lemmes avec **auto**

```
1 subgoal
  ============================
  P 2

Coq < auto.

Coq < Show.
1 subgoal
  ============================
  P 2
```

## Remarques

- La tactique **auto** échoue ici.
- Quand elle échoue, elle ne dit rien et ne fait rien (le but est inchangé).

# La tactique **auto** : une tactique mystérieuse ?

## Utiliser des lemmes avec **auto**

```
1 subgoal
==============================
P 2

Coq < auto using ax1, ax2.
No more subgoals.
```

## Remarques

- On indique les lemmes que l'on souhaite ajouter à la liste d'hypothèses que la tactique **auto** utilisera dans sa recherche de preuve.

# La tactique **auto** : une tactique mystérieuse ?

## Utiliser des lemmes avec **auto**

```
Parameters P Q : nat → Prop.
Axiom ax1 : forall n, Q n → P n.
Axiom ax2 : forall n, Q n.

Hint Resolve ax1 ax2 : ma_base.
```

## Remarques

- Une autre possibilité est de passer par une base de lemmes.
- On ajoute les axiomes *ax*1 et *ax*2 dans la base *ma_base*.

# La tactique **auto** : une tactique mystérieuse ?

## Utiliser des lemmes avec **auto**

```
Coq < Goal P 2.
1 subgoal
  ============================
  P 2

Coq < auto with ma_base.
No more subgoals.
```

## Remarques

- Ici, on utilise explicitement notre base *ma_base*.
- Par défaut, **auto** utilise la base nommée *core*.
- Il en existe d'autres dans la bibliothèque standard (*arith*, *zarith*, etc.).