Types inductifs

David Delahaye
David.Delahaye@lirmm.fr

Université de Montpellier
Faculté des Sciences

Licence Informatique L3 2021-2022

Spécifications inductives et preuves par induction

L'induction à la base de la formalisation

- On peut tout formaliser à base de types inductifs ;
- Types inductifs pour les types de données ;
- Relations inductives pour spécifier des comportements ;
- Fonctions récursives pour les programmes ;
- Preuves par induction pour l'adéquation prog./spéc. ;
- Moyen idiomatique de formalisation de beaucoup d'outils.

Support pour l'induction

- Générer les schémas d'induction automatiquement ;
- Pouvoir en générer de nouveaux au besoin ;
- Gérer les lemmes d'inversion automatiquement.

Spécifications inductives et preuves par induction

Systèmes formels et outils

- Induction présente en théorie des ensembles ;
- Théories des types dédiées :
  - Système T de Gödel, théorie des types de Martin-Löf, calcul des constructions inductives de Coquand-Huet-Paulin.
- Outils dédiés : Coq, Lego, Alfa, etc.

Historiquement

- Formulation explicite de l'induction au 17ème siècle ;
- Auparavant : utilisation de l'induction mathématique ;
- Pascal : « Traité du triangle arithmétique » ;
- Fermat : descente infinie.

Preuve du théorème de Fermat pour $n = 4$

Théorème

Il n'existe pas d'entiers non nuls x, y, et z, tels que :

$$x^4 + y^4 = z^4$$

Le théorème se déduit aisément de la preuve du 20ème problème de Diophante : est-ce qu'un triangle rectangle dont les côtés sont mesurés par des entiers peut avoir une surface mesurée par un carré ?

Fermat a résolu la question par la négative et il a démontré qu'il n'existe pas d'entiers naturels non nuls tels que :

$$x^2 + y^2 = z^2 \text{ et } xy = 2t^2$$

Preuve du théorème de Fermat pour $n = 4$

Principe de la descente infinie

- Preuve par l'absurde : démontrer que résoudre le problème au rang n revient à le résoudre au rang $n-1$, ce qui n'est pas possible car on est borné par 0 ;
- La descente infinie résout des propositions $\nexists x.P(x)$;
- Mais équivalent au principe habituel (contraposée).

Preuve de Fermat chargée d'histoire

- Première utilisation de l'induction ;
- Résolution d'un problème de Diophante (250 apr. J.-C.) ;
- Utilisation des triplets pythagoriciens (Euclide, 300 av. J.-C. ; Babyloniens, 1900-1600 av. J.-C.).

Un premier exemple : les entiers naturels

Définition inductive

On peut définir les entiers naturels comme le plus petit ensemble $\mathcal{N}$ vérifiant les propriétés suivantes :

- $O \in \mathcal{N}$
- Pour tout $n \in \mathcal{N}$, on a $(S\ n) \in \mathcal{N}$

Éléments de cet ensemble

- Syntaxiquement : $\{O, (S\ O), (S\ (S\ O)), \ldots\}$
- Ensemble isomorphe à $\mathbb{N}$

Questions

- En quoi la définition est-elle inductive ?
- Pourquoi « le plus petit ensemble » ?

Un premier exemple : les entiers naturels

Fonction récursive

On peut définir la fonction d'addition *plus* de type $\mathcal{N} \times \mathcal{N} \to \mathcal{N}$ de la façon suivante :

- Pour tout $n \in \mathcal{N}$, $plus(O, n) = n$
- Pour tout $p, n \in \mathcal{N}$, $plus(S\ p, n) = S\ (plus(p, n)$

La récursion se fait sur le premier argument.

Exemple d'évaluation

- $plus(S\ (S\ O), S\ (S\ O)) = S\ (plus(S\ O, S\ (S\ O))) =$
  $S\ (S\ (plus(O, S\ (S\ O)))) = S\ (S\ (S\ (S\ O)))$

Un premier exemple : les entiers naturels

Exemple de preuve

On veut démontrer que : $\forall n \in \mathcal{N}.plus(O, n) = n$.

- On suppose un $n \in \mathcal{N}$
- Puis on part de $plus(O, n)$
- On utilise le cas de base de *plus* pour dire que $plus(O, n) = n$

La preuve est directe en utilisant la définition de *plus*.

Un autre exemple de preuve

Et si on veut démontrer que : $\forall n \in \mathcal{N}.plus(n, O) = n$?
Il nous faut faire une preuve par induction.
Mais quel est le schéma d'induction ?

Un premier exemple : les entiers naturels

Exemple de preuve

On veut démontrer que : $\forall n \in \mathcal{N}.plus(O, n) = n$.

- On suppose un $n \in \mathcal{N}$
- Puis on part de $plus(O, n)$
- On utilise le cas de base de *plus* pour dire que $plus(O, n) = n$

La preuve est directe en utilisant la définition de *plus*.

Un autre exemple de preuve

Et si on veut démontrer que : $\forall n \in \mathcal{N}.plus(n, O) = n$?

Il nous faut faire une preuve par induction.

Mais quel est le schéma d'induction ?

Un premier exemple : les entiers naturels

Exemple de preuve

On veut démontrer que : $\forall n \in \mathcal{N}.plus(O, n) = n$.

- On suppose un $n \in \mathcal{N}$
- Puis on part de $plus(O, n)$
- On utilise le cas de base de *plus* pour dire que $plus(O, n) = n$

La preuve est directe en utilisant la définition de *plus*.

Un autre exemple de preuve

Et si on veut démontrer que : $\forall n \in \mathcal{N}.plus(n, O) = n$?
Il nous faut faire une preuve par induction.
Mais quel est le schéma d'induction ?

Un premier exemple : les entiers naturels

Exemple de preuve

On veut démontrer que : $\forall n \in \mathcal{N}.plus(O, n) = n$.

- On suppose un $n \in \mathcal{N}$
- Puis on part de $plus(O, n)$
- On utilise le cas de base de *plus* pour dire que $plus(O, n) = n$

La preuve est directe en utilisant la définition de *plus*.

Un autre exemple de preuve

Et si on veut démontrer que : $\forall n \in \mathcal{N}.plus(n, O) = n$?
Il nous faut faire une preuve par induction.
Mais quel est le schéma d'induction ?

Un premier exemple : les entiers naturels

Schéma d'induction

Pour $\mathcal{N}$, on peut se donner le schéma d'induction structurelle suivant :

- $\forall P \in \mathcal{N} \to Prop. P(O) \Rightarrow (\forall n \in \mathcal{N}. P(n) \Rightarrow P(S\ n)) \Rightarrow \forall n \in \mathcal{N}. P(n)$ où *Prop* est l'ensemble des propositions (ou formules)
- Ce schéma est d'ordre 2 (non exprimable au premier ordre)

Autres schémas d'induction

- Le schéma précédent suit strictement la syntaxe de la définition de $\mathcal{N}$, d'où le nom de schéma d'induction structurelle
- Il existe d'autres schémas, plus généraux ou plus appropriés pour certaines démonstrations (nous les verrons plus tard)

Un premier exemple : les entiers naturels

Retour sur l'autre exemple de preuve

Preuve de : $\forall n \in \mathcal{N}.plus(n, O) = n$.
On fait une preuve par induction avec $P(n) = (plus(n, O) = n)$:

- Cas de base : $plus(O, O) = O$
  Démontré en utilisant le cas de base de la fonction *plus*
- Cas inductif : $plus(S\ n, O) = S\ n$
  Sachant que : $plus(n, O) = n$ (hypothèse d'induction)
  $plus(S\ n, O) = S\ (plus(n, O))$ (cas inductif de la fonction *plus*)
  $= S\ n$ (hypothèse d'induction)

Un deuxième exemple : les listes

Définition inductive

On se donne un ensemble $\mathcal{A}$ (éléments de la liste).
On peut définir les listes d'éléments de $\mathcal{A}$ comme le plus petit ensemble $\mathcal{L}$ vérifiant les propriétés suivantes :

- $nil \in \mathcal{L}$
- Pour tout $e \in \mathcal{A}$ et $l \in \mathcal{L}$, on a $e :: l \in \mathcal{L}$

Cette définition est polymorphe en ce sens qu'elle ne dépend pas de la structure des éléments de $\mathcal{A}$.

Notation

- On utilisera la notation [] pour *nil*
- Et si $a, b, c \in \mathcal{A}$, on utilisera la notation $[a; b; c]$ pour $a :: b :: c :: nil$

Un deuxième exemple : les listes

Fonction récursive

On peut définir la fonction de concaténation app de type $\mathcal{L} \times \mathcal{L} \to \mathcal{L}$ de la façon suivante :

- Pour tout $l \in \mathcal{L}$, $app(nil, l) = l$
- Pour tout $e \in \mathcal{A}$ et $l_1, l_2 \in \mathcal{L}$, $app(a :: l_1, l_2) = a :: (app(l_1, l_2))$

La récursion se fait sur le premier argument.

Exemple d'évaluation

- $app([1;2],[3;4;5]) = 1 :: app([2],[3;4;5]) =$
  $1 :: 2 :: app([],[3;4;5]) = 1 :: 2 :: [3;4;5] = [1;2;3;4;5]$

Un deuxième exemple : les listes

Exemple de preuve

On veut démontrer que : $\forall l \in \mathcal{L}.app(nil, l) = l$.

- On suppose un $l \in \mathcal{L}$
- Puis on part de $app(nil, l)$
- On utilise le cas de base de app pour dire que $app(nil, l) = l$

La preuve est directe en utilisant la définition de app.

Un autre exemple de preuve

Et si on veut démontrer que : $\forall l \in \mathcal{L}.app(l, nil) = l$?
Il nous faut faire une preuve par induction.
Mais quel est le schéma d'induction ?

Un deuxième exemple : les listes

Exemple de preuve

On veut démontrer que : $\forall l \in \mathcal{L}.app(nil, l) = l$.

- On suppose un $l \in \mathcal{L}$
- Puis on part de $app(nil, l)$
- On utilise le cas de base de app pour dire que $app(nil, l) = l$

La preuve est directe en utilisant la définition de app.

Un autre exemple de preuve

Et si on veut démontrer que : $\forall l \in \mathcal{L}.app(l, nil) = l$?

Il nous faut faire une preuve par induction.

Mais quel est le schéma d'induction ?

Un deuxième exemple : les listes

Exemple de preuve

On veut démontrer que : $\forall l \in \mathcal{L}.app(nil, l) = l$.

- On suppose un $l \in \mathcal{L}$
- Puis on part de $app(nil, l)$
- On utilise le cas de base de app pour dire que $app(nil, l) = l$

La preuve est directe en utilisant la définition de app.

Un autre exemple de preuve

Et si on veut démontrer que : $\forall l \in \mathcal{L}.app(l, nil) = l$?
Il nous faut faire une preuve par induction.
Mais quel est le schéma d'induction ?

Un deuxième exemple : les listes

Exemple de preuve

On veut démontrer que : $\forall l \in \mathcal{L}.app(nil, l) = l$.

- On suppose un $l \in \mathcal{L}$
- Puis on part de $app(nil, l)$
- On utilise le cas de base de app pour dire que $app(nil, l) = l$

La preuve est directe en utilisant la définition de app.

Un autre exemple de preuve

Et si on veut démontrer que : $\forall l \in \mathcal{L}.app(l, nil) = l$?
Il nous faut faire une preuve par induction.
Mais quel est le schéma d'induction ?

Un deuxième exemple : les listes

Schéma d'induction

Pour $\mathcal{L}$, on peut se donner le schéma d'induction structurelle suivant :

- $\forall P \in \mathcal{L} \to Prop.P(nil) \Rightarrow (\forall e \in \mathcal{A}.\forall l \in \mathcal{L}.P(l) \Rightarrow P(e :: l)) \Rightarrow \forall l \in \mathcal{L}.P(l)$

Autres schémas d'induction

- Comme précédemment, il s'agit là du schéma d'induction structurelle (il existe d'autres schémas)

Un deuxième exemple : les listes

Retour sur l'autre exemple de preuve

Preuve de : $\forall l \in \mathcal{L}.app(l, nil) = l$.
On fait une preuve par induction avec $P(l) = (app(l, nil) = l)$:

- Cas de base : $app(nil, nil) = nil$
  Démontré en utilisant le cas de base de la fonction app
- Cas inductif : $app(e :: l, nil) = e :: l$
  Sachant que : $app(l, nil) = l$ (hypothèse d'induction)
  $app(e :: l, nil) = e :: (app(l, nil))$ (cas inductif de la fonction app)
  $= e :: l$ (hypothèse d'induction)

Relations inductives

Somme des n premiers entiers naturels

On peut définir cette fonction comme la relation inductive is_sum de type $\mathcal{N} \times \mathcal{N} \to Prop$ de la façon suivante :

- On a : $is\_sum(O, O)$;
- Pour $n, s \in \mathcal{N}$, si $is\_sum(n, s)$, alors on a : $is\_sum(S\ n, s + (S\ n))$.

Le premier paramètre est l'entier n dont on souhaite calculer la somme des n premiers entiers. Le deuxième paramètre est le somme calculée.

On peut voir cette relation comme une spécification mais elle est très calculatoire. En fait, c'est comme cela qu'on l'écrirait en Prolog.

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_sum(S\ (S\ (S\ O)), S\ (S\ (S\ (S\ (S\ O)))))$.

- On utilise le cas inductif de *is_sum* et on se retrouve à démontrer que : $is\_sum(S\ (S\ O), S\ (S\ (S\ O)))$
- On utilise le cas inductif de *is_sum* et on se retrouve à démontrer que : $is\_sum(S\ O, S\ O)$
- On utilise le cas inductif de *is_sum* et on se retrouve à démontrer que : $is\_sum(O, O)$
- Démontré en utilisant le cas de base de *is_sum*

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_sum(S\ (S\ (S\ O)), S\ (S\ (S\ (S\ (S\ O)))))$.

- On utilise le cas inductif de *is_sum* et on se retrouve à démontrer que : $is\_sum(S\ (S\ O), S\ (S\ (S\ O)))$
- On utilise le cas inductif de *is_sum* et on se retrouve à démontrer que : $is\_sum(S\ O, S\ O)$
- On utilise le cas inductif de *is_sum* et on se retrouve à démontrer que : $is\_sum(O, O)$
- Démontré en utilisant le cas de base de *is_sum*

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_sum(S\ (S\ (S\ O)), S\ (S\ (S\ (S\ (S\ O)))))$.

- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(S\ (S\ O), S\ (S\ (S\ O)))$
- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(S\ O, S\ O)$
- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(O, O)$
- Démontré en utilisant le cas de base de is_sum

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_sum(S\ (S\ (S\ O)), S\ (S\ (S\ (S\ (S\ O)))))$.

- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(S\ (S\ O), S\ (S\ (S\ O)))$
- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(S\ O, S\ O)$
- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(O, O)$
- Démontré en utilisant le cas de base de is_sum

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_sum(S\ (S\ (S\ O)), S\ (S\ (S\ (S\ (S\ O)))))$.

- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(S\ (S\ O), S\ (S\ (S\ O)))$
- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(S\ O, S\ O)$
- On utilise le cas inductif de is_sum et on se retrouve à démontrer que : $is\_sum(O, O)$
- Démontré en utilisant le cas de base de is_sum

Relations inductives

Parité des entiers naturels

On peut définir cette fonction comme la relation inductive is_even de type $\mathcal{N} \to Prop$ de la façon suivante :

- On a : $is\_even(O)$;
- Pour $n \in \mathcal{N}$, si $is\_even(n)$, alors on a : $is\_even(S\ (S\ n))$.

La récursion se fait ici avec une profondeur de 2.

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_even(S\ (S\ (S\ (S\ O))))$.

- On utilise le cas inductif de is_even et on se retrouve à démontrer que : $is\_even(S\ (S\ O))$
- On utilise le cas inductif de is_even et on se retrouve à démontrer que : $is\_even(O)$
- Démontré en utilisant le cas de base de is_even

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_even(S\ (S\ (S\ (S\ O))))$.

- On utilise le cas inductif de is_even et on se retrouve à démontrer que : $is\_even(S\ (S\ O))$
- On utilise le cas inductif de is_even et on se retrouve à démontrer que : $is\_even(O)$
- Démontré en utilisant le cas de base de is_even

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_even(S\ (S\ (S\ (S\ O))))$.

- On utilise le cas inductif de is_even et on se retrouve à démontrer que : $is\_even(S\ (S\ O))$
- On utilise le cas inductif de is_even et on se retrouve à démontrer que : $is\_even(O)$
- Démontré en utilisant le cas de base de is_even

Relations inductives

Un exemple de preuve

On veut démontrer que : $is\_even(S\ (S\ (S\ (S\ O))))$.

- On utilise le cas inductif de is_even et on se retrouve à démontrer que : $is\_even(S\ (S\ O))$
- On utilise le cas inductif de is_even et on se retrouve à démontrer que : $is\_even(O)$
- Démontré en utilisant le cas de base de is_even

Relations inductives

Listes d'entiers naturels pairs

On veut définir la relation qui dit si une liste d'entiers naturels est uniquement constituées d'entiers pairs.
Pour ce faire, on utilise les listes $\mathcal{L}$ avec $\mathcal{A} = \mathcal{N}$.

On peut définir cette relation comme la relation inductive is_even_list de type $\mathcal{L} \to Prop$ de la façon suivante :

- On a : $is\_even\_list(nil)$;
- Pour $n \in \mathcal{N}$ et $l \in \mathcal{L}$, si $is\_even(n)$ et $is\_even\_list(l)$, alors on a : $is\_even\_list(n :: l)$.

Relations inductives

Un exemple de preuve

On veut démontrer que :
$is\_even\_list([O; (S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$.

- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(O)$ (cas de base de is_even)
  - $is\_even\_list([(S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ O))$ (cas inductif + cas de base de is_even)
  - $is\_even\_list([(S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ (S\ (S\ O))))$ ($2 \times$ cas inductif + cas de base de is_even)
  - $is\_even\_list(nil)$
- On utilise le cas de base de is_even_list

Relations inductives

Un exemple de preuve

On veut démontrer que :
$is\_even\_list([O; (S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$.

- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(O)$ (cas de base de is_even)
  - $is\_even\_list([(S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ O))$ (cas inductif + cas de base de is_even)
  - $is\_even\_list([(S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ (S\ (S\ O))))$ ($2 \times$ cas inductif + cas de base de is_even)
  - $is\_even\_list(nil)$
- On utilise le cas de base de is_even_list

Relations inductives

Un exemple de preuve

On veut démontrer que :
$is\_even\_list([O; (S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$.

- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(O)$ (cas de base de is_even)
  - $is\_even\_list([(S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ O))$ (cas inductif + cas de base de is_even)
  - $is\_even\_list([(S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ (S\ (S\ O))))$ ($2 \times$ cas inductif + cas de base de is_even)
  - $is\_even\_list(nil)$
- On utilise le cas de base de is_even_list

Relations inductives

Un exemple de preuve

On veut démontrer que :
$is\_even\_list([O; (S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$.

- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(O)$ (cas de base de is_even)
  - $is\_even\_list([(S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ O))$ (cas inductif + cas de base de is_even)
  - $is\_even\_list([(S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ (S\ (S\ O))))$ ($2 \times$ cas inductif + cas de base de is_even)
  - $is\_even\_list(nil)$
- On utilise le cas de base de is_even_list

Relations inductives

Un exemple de preuve

On veut démontrer que :
$is\_even\_list([O; (S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$.

- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(O)$ (cas de base de is_even)
  - $is\_even\_list([(S\ (S\ O)); (S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ O))$ (cas inductif + cas de base de is_even)
  - $is\_even\_list([(S\ (S\ (S\ (S\ O))))])$
- On utilise le cas inductif de is_even_list et on se retrouve à démontrer que :
  - $is\_even(S\ (S\ (S\ (S\ O))))$ ($2 \times$ cas inductif + cas de base de is_even)
  - $is\_even\_list(nil)$
- On utilise le cas de base de is_even_list

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < Print nat.
Inductive nat : Set :=  O : nat | S : nat -> nat
For S: Argument scope is [nat_scope]

Coq < Check (S O).
1
     : nat

Coq < Check 1.
1
     : nat
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < Print plus.
plus =
fix plus (n m : nat) {struct n} : nat :=
  match n with
  | O => m
  | S p => S (plus p m)
  end
     : nat -> nat -> nat
Argument scopes are [nat_scope nat_scope]
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < Goal forall x : nat, 0 + x = x.
1 subgoal

  ============================
   forall x : nat, 0 + x = x

Coq < intro.
1 subgoal

  x : nat
  ============================
   0 + x = x
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < simpl.
1 subgoal

  x : nat
  ============================
   x = x

Coq < reflexivity.
No more subgoals.
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < Goal forall x : nat, x + 0 = x.
1 subgoal

  ============================
   forall x : nat, x + 0 = x

Coq < intro.
1 subgoal

  x : nat
  ============================
   x + 0 = x
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < elim x.
2 subgoals

  x : nat
  ============================
   0 + 0 = 0
subgoal 2 is:
 forall n : nat, n + 0 = n -> S n + 0 = S n
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < simpl.
2 subgoals

  x : nat
  ============================
   0 = 0
subgoal 2 is:
 forall n : nat, n + 0 = n -> S n + 0 = S n
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < reflexivity.
1 subgoal

  x : nat
  ============================
   forall n : nat, n + 0 = n -> S n + 0 = S n
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < intros.
1 subgoal

  x : nat
  n : nat
  H : n + 0 = n
  ============================
   S n + 0 = S n
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < simpl.
1 subgoal

  x : nat
  n : nat
  H : n + 0 = n
  ============================
   S (n + 0) = S n
```

Types inductifs en Coq

Exemples de preuves

- Entiers naturels :

```
Coq < rewrite H.
1 subgoal

  x : nat
  n : nat
  H : n + 0 = n
  ============================
   S n = S n

Coq < reflexivity.
No more subgoals.
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < Print list.
Inductive list (A : Type) : Type :=
    nil : list A | cons : A -> list A -> list A
For nil: Argument A is implicit and maximally inserted
For cons: Argument A is implicit
For list: Argument scope is [type_scope]
For nil: Argument scope is [type_scope]
For cons: Argument scopes are [type_scope _ _]

Coq < Open Scope list.

Coq < Check (0 :: 1 :: nil).
0 :: 1 :: nil
     : list nat
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < Open Scope list.

Coq < Print app.
app =
fun A : Type =>
fix app (l m : list A) {struct l} : list A :=
  match l with
  | nil => m
  | a :: l1 => a :: app l1 m
  end
     : forall A : Type, list A -> list A -> list A
Argument A is implicit
Argument scopes are [type_scope list_scope list_scope]
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < Goal forall (E : Type) (l : list E), l ++ nil = l.
1 subgoal

  ============================
   forall (E : Type) (l : list E), l ++ nil = l

Coq < intros.
1 subgoal

  E : Type
  l : list E
  ============================
   l ++ nil = l
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < elim l.
2 subgoals

  E : Type
  l : list E
  ============================
   nil ++ nil = nil
subgoal 2 is:
 forall (a : E) (l0 : list E),
 l0 ++ nil = l0 -> (a :: l0) ++ nil = a :: l0
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < simpl.
2 subgoals

  E : Type
  l : list E
  ============================
   nil = nil
subgoal 2 is:
 forall (a : E) (l0 : list E),
 l0 ++ nil = l0 -> (a :: l0) ++ nil = a :: l0
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < reflexivity.
1 subgoal

  E : Type
  l : list E
  ============================
   forall (a : E) (l0 : list E),
   l0 ++ nil = l0 -> (a :: l0) ++ nil = a :: l0
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < intros.
1 subgoal

  E : Type
  l : list E
  a : E
  l0 : list E
  H : l0 ++ nil = l0
  ============================
   (a :: l0) ++ nil = a :: l0
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < simpl.
1 subgoal

  E : Type
  l : list E
  a : E
  l0 : list E
  H : l0 ++ nil = l0
  ============================
   a :: l0 ++ nil = a :: l0
```

Types inductifs en Coq

Exemples de preuves

- Listes :

```
Coq < rewrite H.
1 subgoal

  E : Type
  l : list E
  a : E
  l0 : list E
  H : l0 ++ nil = l0
  ============================
   a :: l0 = a :: l0

Coq < reflexivity.
No more subgoals.
```