

Séminaire Coq

Introduction et utilisation basique

Maxime Folschette

14 février 2013

Définition inductive \rightsquigarrow Nouveaux types

```
Inductive bool : Type :=  
  | true  : bool  
  | false : bool.
```

Définition avec pattern matching \rightsquigarrow Déconstruction

```
Definition negb (b:bool) : bool :=  
  match b with  
    | true => false  
    | false => true  
  end.
```

Définition d'un théorème et écriture de sa preuve

Theorem <nom> : <proposition>.

Proof.

<tactiques et conclusions>

Qed.

Environnement de preuves

Contexte (variables et hypotheses)

=====

Objectif actif

Objectifs en attente

...

⇒ On cherche à prouver l'objectif actif avant de passer aux autres.

Proposition = Assertion, « phrase » non simplifiable.

Exemple

- $2 < 5$
- $\forall x \in \mathbb{R}, x^2 \geq 0$
- $\forall x \in \mathbb{R}, \exists y_1, y_2 \in \mathbb{R}, y_1 \neq y_2 \wedge y_1^2 = y_2^2 = x$
- $5 < 2$

Preuve = Succession de tactiques pour résoudre chaque objectif.

Exemple

- `simpl` → Simplifier l'objectif courant.
- `reflexivity` → Résoudre l'objectif courant s'il a la forme : $X = X$.
- `intros` $a \rightarrow$ On pose a (pour répondre à $\forall a, \dots$).
- `intros` $H \rightarrow$ On suppose H (pour répondre à $H \Rightarrow \dots$).
- `rewrite` $H \rightarrow$ Si $H : x = y$, remplacer tout x par y .
- `apply` $H \rightarrow$ Conclure en appliquant H .

Type paramétré = Type dépendant d'un autre type.

option X — Une option sur le type X est :

- soit un élément qui ne contient aucune valeur $\rightsquigarrow \text{None}$
- soit un élément qui contient une valeur de type X $\rightsquigarrow \text{Some } x$

Type récursif = Type qui fait référence à lui-même.

nat — Un entier naturel nat (dans l'arithmétique de Peano) est :

- soit l'élément nul $\rightsquigarrow 0$
- soit le successeur d'un entier naturel $\rightsquigarrow S\ n$

Exemple de type récursif paramétré : les listes.

list X — Une liste d'éléments de type X est :

- soit la liste vide $\rightsquigarrow []$
- soit un élément de X (tête) et une liste de X (queue) $\rightsquigarrow h :: t$

Fonction récursive = Fonction qui fait référence à elle-même.

→ Problème : quid des fonctions récursives qui ne terminent pas ?
Quel résultat ? Quel type ?

⇒ Mathématiquement non défini

→ Solution : forcer la terminaison

Décroissance structurelle = Une fonction ne peut s'appeler elle-même qu'avec au moins un argument strictement inférieur structurellement
 X est structurellement inférieur à Y = On peut construire Y à partir de X

⇒ On finit toujours dans un cas dégénéré

Exemple

La fonction `length` s'appelle elle-même sur la queue de la liste, qui est structurellement plus petite. (On peut construire la liste de départ à partir de sa tête et de sa queue.)

Récurrence sur \mathbb{N} (scolaire)

Si on parvient à montrer :

- P_0 ,
- $\forall n \in \mathbb{N}, P_n \Rightarrow P_{n+1}$,

Alors on a :

- $\forall n \in \mathbb{N}, P_n$.

Autrement dit : $(P_0 \wedge \forall n \in \mathbb{N}, P_n \Rightarrow P_{n+1}) \Rightarrow \forall n \in \mathbb{N}, P_n$.

Récurrence sur "nat" (structurelle)

$P(0) \Rightarrow (\forall n \in \text{nat}, P(n) \Rightarrow P(S\ n)) \Rightarrow \forall n \in \text{nat}, P(n)$

Récurrence sur "list X" (structurelle)

$P([]) \Rightarrow (\forall x \in X, \forall l \in \text{list } X, P(l) \Rightarrow P(x :: l)) \Rightarrow \forall l \in \text{list } X, P(l)$

avec associativité à droite de " \Rightarrow " :

$$A \Rightarrow B \Rightarrow C \quad \equiv \quad A \Rightarrow (B \Rightarrow C) \quad \equiv \quad (A \wedge B) \Rightarrow C$$

$$H : P \rightarrow Q$$
$$f : P \rightarrow Q$$

Séminaire Coq

Introduction et utilisation basique

Maxime Folschette

<https://github.com/nantes-fp/seance-coq>

14 février 2013

<http://coq.inria.fr/>

<http://www.cis.upenn.edu/~bcpierce/sf/>

<http://www.coursera.org/course/progfun>

<http://www.labri.fr/perso/casteran/CoqArt/index.html>

Licence Beerware / CC0 : si les termes de la licence Beerware ne peuvent s'appliquer ou semblent trop restrictifs, ceux de la licence CC0 s'appliquent. Réutilisation encouragée.