

Chassez le naturel dans la formalisation des mathématiques

Thomas Portet

JFLA Janvier 2025

Inria

Contexte

- ▶ Utiliser des outils de preuves pour l'enseignement
- ▶ Capacités du langage, documentation, message d'erreur
 - ▶ Inspiration : Waterproof
 - ▶ Expériences similaires sur Lean (Lean Verbose)
- ▶ Le contenu des preuves joue un rôle primordial pas seulement leur présentation
 - ▶ Plusieurs types de nombres, plusieurs versions d'opérateurs
 - ▶ Des coercions peuvent être cachées bloquant certaines opérations
 - ▶ La théorie des types oblige les développeurs à définir des fonctions là où elles ne devraient pas être définies.
- ▶ Le typage aide les jeunes mathématiciens, mais pas le type des entiers naturels

Caractéristiques des entiers naturels

- ▶ Aspects positifs
 - ▶ Un type inductif
 - ▶ Calcul par réduction (plus rapide que la réécriture)
 - ▶ Les définitions récursives sont le plus souvent avec des entiers naturels
- ▶ Aspects négatifs
 - ▶ La soustraction est étrange : la valeur de $3 - 5$ est contre-intuitive
 - ▶ Le statut de la fonction/constructeur S est un obstacle pour les étudiants
 - ▶ Dans Coq, $S\ 4$ et 5 sont interchangeables, mais $S\ x$ et $x + 1$ ne le sont pas.
 - ▶ Le temps consacré à l'apprentissage du pattern matching n'est pas consacré aux mathématiques
 - ▶ La charge cognitive est trop importante

Proposition

- ▶ N'utiliser qu'un seul type de nombres : les nombres réels
 - ▶ Choisi pour être intuitif pour les élèves à la fin du lycée
 - ▶ Incluant la relation d'ordre
- ▶ Considérer les autres types connus comme des sous-ensembles
- ▶ Inclure les lois de stabilité dans l'automatisation des preuves
- ▶ Inspiration forte : l'approche PVS
 - ▶ Cependant, PVS est trop agressif en matière d'automatisation pour l'éducation.
- ▶ Entiers naturels, relatifs sont encore utilisés de manière cachée

Plan

- ▶ Revue des usages des entiers naturels et relatifs
- ▶ Définir des sous-ensembles de $\mathbb{R}$ pour les types inductifs
- ▶ De $\mathbb{Z}$ et $\mathbb{N}$ à $\mathbb{R}$ et vice-versa
- ▶ Preuves ad hoc d'appartenance
- ▶ Définitions récursives et fonctions itérées
- ▶ Exemple pratique : autour de Fibonacci

Utilisation des entiers naturels et relatifs

- ▶ Une base pour les preuves par récurrence
- ▶ Définition d'une suite récursive
- ▶ Itérer une opération un certain nombre de fois $f^n(k)$
- ▶ La séquence $0, \dots, n$
- ▶ Indices pour les opérations itérées $\sum_{i=m}^n f(i)$
- ▶ Dans les nombres réels de Coq, les nombres $0, 1, \dots, 37, \dots$ s'appuient sur le type inductif des entiers relatifs pour la représentation.
 - ▶ En Coq, vous pouvez définir `Zfact` comme un équivalent efficace de la factorielle et calculer `100!`

Définir des sous-ensembles de $\mathbb{R}$ pour les types inductifs

- ▶ Approche inductive des prédicats
 - ▶ Hériter du principe de récurrence
 - ▶ Prouver l'existence d'un entier naturel ou relatif correspondant

Principe de récurrence en Coq

```
Require Import Reals.  
Open Scope R_scope.
```

```
Inductive Rnat : R -> Prop :=  
  Rnat0 : Rnat 0  
| Rnat_succ : forall n, Rnat n -> Rnat (n + 1).
```

Principe de récurrence généré :

```
nat_ind  
  : forall P : R -> Prop,  
    P 0 ->  
    (forall n : R, Rnat n -> P n -> P (n + 1)) ->  
    forall r : R, Rnat r -> P r
```

De $\mathbb{N}$ et $\mathbb{Z}$ vers $\mathbb{R}$ et vice-versa

- ▶ Rappel : les types $\mathbb{N}$ (nat) et $\mathbb{Z}$ (Z), ne doivent pas être exposés
- ▶ Les injections INR et IZR existent déjà
- ▶ Nouvelles fonctions IRN et IRZ
- ▶ Définissables à l'aide d'outils de logique classiques comme l'axiome de choix d'Hilbert ou la partie entière.

Typage dégradé

- Les lois de stabilité fournissent des preuves automatisables de l'appartenance à un sous-ensemble.

Existing Class Rnat.

```
Lemma Rnat_add x y : Rnat x -> Rnat y -> Rnat (x + y).
```

```
Proof. ... Qed.
```

```
Lemma Rnat_mul x y : Rnat x -> Rnat y -> Rnat (x * y).
```

```
Proof. ... Qed.
```

```
Lemma Rnat_pos : Rnat (IZR (Z.pos _)).
```

```
Proof. ... Qed.
```

Existing Instances Rnat0 Rnat_succ Rnat_add Rnat_mul Rnat_pos.

- typeclasses eauto ou exact _. résoudra automatiquement
Rnat x -> Rnat ((x + 2) * x).

Preuves ad hoc d'appartenance

- ▶ Quand $n, m \in \mathbb{N}$, $m \leq n$, $(n - m) \in \mathbb{N}$ peut aussi être prouvé
- ▶ Cela nécessite une preuve explicite
- ▶ Probablement bon dans un contexte de formation pour les étudiants
- ▶ Idem pour la division

Fonctions récursives

- ▶ Les suites récurrentes sont également un sujet d'introduction typique
- ▶ À titre d'illustration, considérons la suite de Fibonacci. *La suite de Fibonacci est la fonction F telle que $F_0 = 0$, $F_1 = 1$, et $F_{n+2} = F_n + F_{n+1}$ pour tout nombre naturel n .*
- ▶ La preuve par récurrence et les équations de définition suffisent à *étudier* une suite
- ▶ Mais il faut encore la définir
- ▶ Solution : définir une commande de définition récursive à l'aide d'ELPI

Définition de fonctions récursives

- ▶ On peut utiliser un *récurseur*, miroir du récursur sur les entiers naturels
- ▶ $\text{Rnat_rec} : ?A \rightarrow (R \rightarrow ?A \rightarrow ?A) \rightarrow R \rightarrow ?A$
- ▶ La récursivité d'ordre supérieur à 1 peut être mise en œuvre en utilisant des tuples de la bonne taille

```
(* fib 0 = 0   fib 1 = 1                               *)  
(* fib n = fib (n - 1) + fib (n - 2) *)
```

```
Definition fibr := Rnat_rec [0; 1]  
  (fun n l => [nth 1 l 0; nth 0 l 0 + nth 1 l 0]).
```

```
Definition fib n := nth 0 (fibr n) 0.
```

Meta-programmation d'une commande de définition récursive

- ▶ La définition du transparent précédent peut être générée
- ▶ En prenant comme entrée les équations (dans les commentaires)
- ▶ Les résultats de la définition sont en deux parties
 - ▶ La fonction de type $R \rightarrow R$
 - ▶ La preuve, l'énoncé logique de cette fonction

```
Recursive (def fib such that
  fib 0 = 0 /\
  fib 1 = 1 /\
  forall n : R, Rnat (n - 2) ->
    fib n = fib (n - 2) + fib (n - 1)).
```

Calculer avec des nombres réels

- ▶ Compute $42 - 67$. donne une réponse surprenante
 - ▶ Des tas de R1, +, * et de parenthèses.
- ▶ Compute $(42 - 67)\%Z$. donne la bonne réponse
 - ▶ Sauf qu'elle n'a pas le bon type

Calculer avec des nombres réels : notre solution

- ▶ Nouvelle commande `R_compute`.
- ▶ `R_compute (42 - 67)`. réussit et affiche -25
- ▶ `R_compute (fib (42 - 67))`. échoue!
- ▶ `R_compute (fib 42) th_name`. réussit et enregistre la preuve de l'égalité.
- ▶ Implémenté en exploitant une correspondance entre les opérations élémentaires sur $\mathbb{R}$, $\mathbb{Z}$ (avec preuves)
- ▶ Programme en ELPI
- ▶ Miroir automatique d'une définition récursive dans $\mathbb{R}$ vers une définition dans $\mathbb{Z}$
- ▶ Théorème de correction du processus miroir a une précondition `Rnat` sur l'entrée.

Démonstration