Enseignement des mathématiques avec Coq en début de licence et au lycée

Pierre Rousselin

PRofesseur AGrégé au département d'informatique de l'institut Galilée
Université Paris XIII dite « Sorbonne Paris Nord »
Villetaneuse

30 et 31 janvier 2024

Deux expériences d'enseignement des mathématiques avec Coq

- Module « Initiation aux preuves formelles »
  - public : double licence mathématiques et informatique
  - L1, premier semestre
  - $3^{\text{ème}}$ édition cette année
  - 18h de travaux pratiques
- « Du Coq aux maths ou l'art de la preuve »
  - public : 8 élèves *volontaires* de seconde du lycée Joséphine Baker de Pierrefitte-sur-Seine
  - stage « MathC2+ » Toussaint 2022
  - 9h de travaux pratique

Création du module « Initiation aux preuves formelles »

Logique propositionnelle

MathC2+ avec des élèves de seconde

Entiers naturels

Calcul des prédicats et logique classique

Déboires à la maison

Real numbers game

Devoir à la maison final et partiel

Interface utilisateur

Bilan et perspectives

Université Sorbonne Paris Nord (Villetaneuse)

Double licence mathématiques-informatique (DL)

- Recrutement sur dossier via Parcoursup (L1) et « Études en France » (L1 et L2)
- Formation très exigeante donc on doit recruter des étudiants très motivés par les deux disciplines.
- Entre 50 et 60 inscrits en L1, une trentaine en L2, cette année 13 en L3 [1] (record)
- Recrutement difficile car :
  - les « fort·e·s en maths » se tournent vers les classes préparatoires aux grandes écoles
  - Villetaneuse, Seine-Saint-Denis (même avec « Sorbonne » dans le nom !)

1. Ces étudiants étaient nos premiers cobayes !

Génèse du cours

- Il n'y avait aucune différence entre la L1 DL et la L1 informatique
- puis une L2 soudainement très difficile avec des semestres à 420h et 36 ECTS
- Envie de créer un module spécifique DL dès le début de la L1.
- Cahier des charges :
  - maths-info
  - à la place de « méthodologie du travail universitaire » et à moyens constants (donc 18h, 1ECTS)
  - adossé à la recherche
  - défi, même pour les très bons étudiants
- Finalement preuves formelles avec Coq presque par hasard : j'avais entendu Micaela Mayero en parler dans un couloir.
- Micaela pousse alors pour que le projet aboutisse et invite Marie Kerjean à rejoindre l'aventure.

Préparation du cours

- Réunion avec l'ancien responsable de la DL, frileux à cause :
  - des notations (en particulier, `->` pour $\Longrightarrow$) ;
  - des ensembles qui n'existent pas vraiment ;
  - des différences en général avec ce que les étudiants voient au même moment dans « Initiation aux structures mathématiques » (premier semestre de L1, commun maths et info) dont l'objectif est de faire passer les étudiants au « stade rigoureux [2]. »

2. https://terrytao.wordpress.com/career-advice/theres-more-to-mathematics-than-rigour-and-proofs/

Préparation du cours

- Réunion avec l'ancien responsable de la DL, frileux à cause :
  - des notations (en particulier, `->` pour $\Longrightarrow$) ;
  - des ensembles qui n'existent pas vraiment ;
  - des différences en général avec ce que les étudiants voient au même moment dans « Initiation aux structures mathématiques » (premier semestre de L1, commun maths et info) dont l'objectif est de faire passer les étudiants au « stade rigoureux [2]. »
- Été 2021 studieux, passé sur

  `https://softwarefoundations.cis.upenn.edu/lf-current/index.html`.
- Soirs, week-ends, nuits... de septembre à décembre 2021 pour écrire les sujets.

2. `https://terrytao.wordpress.com/career-advice/theres-more-to-mathematics-than-rigour-and-proofs/`

Plan du cours

- Idée de départ : limites de suites numériques
- Donc on a besoin des nombres réels, des entiers naturels, du calcul des prédicats (et en général des connecteurs logiques usuels).
- Ça donne de toute façon quelque chose comme :
  1. Logique propositionnelle
  2. Entiers naturels et récurrence
  3. Calcul des prédicats
  4. Nombres réels
  5. Suites numériques

Aspects pratiques

- Coq sur les machines des salles de TP (Debian et Coq 8.12 en 2021, Coq 8.16 cette année) avec CoqIDE ;
- Installation facile sur les machines des étudiant·e·s, grâce à la « *Coq platform* »
- Cette année, `jscoq` dans un navigateur.
- L'interface utilisateur est un sujet important... On en reparlera plus tard.

LogiqueIntuitionniste.v

Les sujets de travaux pratiques sont là :

`https://www.math.univ-paris13.fr/~rousselin/ipf.html`

Règles d'écriture des sujets

- Toujours commencer par un exemple commenté.
- Suivi d'au moins un exercice d'application directe.
- Un exercice doit toujours être faisable avec ce qui a été présenté précédemment (contrat).
- Idéalement, ne présenter qu'une notion à la fois et éviter d'ajouter du bruit. (maîtrise du flux d'informations)

Règles d'écriture des sujets

- Toujours commencer par un exemple commenté.
- Suivi d'au moins un exercice d'application directe.
- Un exercice doit toujours être faisable avec ce qui a été présenté précédemment (contrat).
- Idéalement, ne présenter qu'une notion à la fois et éviter d'ajouter du bruit. (maîtrise du flux d'informations)

Ce n'est pas toujours facile :

- On a parfois du mal à se retenir, par exemple l'introduction du premier fichier devrait s'abstenir de parler de logique intuitionniste et de logique classique. (TODO : changer ça avant l'année prochaine !)
- Coq nous force parfois la main : c'est le cas avec l'associativité à droite de `->`. Même avec `Set Printing Parentheses`, il n'y a pas de parenthèses affichées dans `P -> (Q -> P)` avant la version 8.19.

imp_refl

On peut faire des erreurs pédagogiques même avec un exemple aussi simple que `imp_refl`. Voici la version 2021 :

```
Theorem imp_refl : forall P : Prop, P -> P.
Proof.
  intros P.
  intros HP.
  assumption.
Qed.
```

- `assumption` un peu trop magique pour cette toute première preuve ;
- `HP` pour nommer l'hypothèse que `P` est vraie est peut-être une bonne pratique mais les étudiants pensaient que le nom de l'hypothèse influençait son type (pensée magique).

Tactiques et règles de déduction naturelle

connecteur	introduction (prouver)	élimination (utiliser)
$\rightarrow$	`intros`	`apply`
$\wedge$	`split`	`destruct`
$\vee$	`left` ou `right`	`destruct`
$\forall$	`intros`	`specialize`, instanciation [3]
$\bot$		`destruct`
$\exists$	`exists`	`destruct`

3. explicite ou implicite avec unification

Difficultés pour les étudiants

- Confusions entre les noms d'hypothèses et leurs types.
- Choix trop précoce :
  - de `left` ou `right` dans le cas d'une disjonction
  - du témoin dans le cas d'un quantificateur existentiel
  - ou variations plus subtiles sur ces thèmes
  - et pas assez de recul pour comprendre qu'il faut revenir en arrière.
- Exemple (examen 2023) :

```
Lemma or_induction : forall (A B C : Prop),
  (A -> C) -> (B -> C) -> A \/ B -> C.
Proof.
  intros A B C.
  intros H K L.
  apply H.
  destruct L as [L1 | L2].
  exact L1.
Admitted.
```

Des tactiques trop puissantes à ce stade

Le tableau de correspondance entre règles de déduction naturelle et tactiques montre un idéal pédagogique. En fait, la plupart des tactiques débordent de leurs cadres pédagogiques.
On commence avec `apply`.

```
Context (A B : Prop).
Goal A /\ B -> A.
Proof.
  intros H.
  apply H. (* apply est fort *)
Qed.
```

Des tactiques trop puissantes à ce stade

Le tableau de correspondance entre règles de déduction naturelle et tactiques montre un idéal pédagogique. En fait, la plupart des tactiques débordent de leurs cadres pédagogiques.
On commence avec `apply`.

```
Context (A B : Prop).
Goal A /\ B -> A.
Proof.
  intros H.
  apply H. (* apply est fort *)
Qed.

Goal (forall (X : Prop), X -> A /\ B) -> C -> A.
Proof.
  intros H.
  apply H. (* apply est très fort *)
Qed.
```

Des tactiques trop puissantes à ce stade

Même `split` s'y met :

```
Goal A -> B -> A /\ B.
Proof.
  split. (* intros *; split. *)
Admitted.
```

Des tactiques trop puissantes à ce stade

Même `split` s'y met :

```
Goal A -> B -> A /\ B.
Proof.
  split. (* intros *; split. *)
Admitted.
```

C'est pareil pour `left` et `right`

```
Goal A -> A \/ B.
Proof.
  left. (* intros *; left. *)
Admitted.
```

destruct destruct

La tactique `destruct` s'applique aussi à une implication lorsque sa conséquence peut se détruire.

```
Goal (C -> A /\ B) -> C -> A.
Proof.
  intros H H'.
  destruct H. (* apparition du sous-but C *)
```

destruct destruct

La tactique `destruct` s'applique aussi à une implication lorsque sa conséquence peut se détruire.

```
Goal (C -> A /\ B) -> C -> A.
Proof.
  intros H H'.
  destruct H. (* apparition du sous-but C *)
```

On peut presque tout `destruct` :

```
Goal forall (n : nat),
  0 = 0 -> 0 <= 1 -> n < 3 -> False -> True -> 0 = 0.
Proof.
  intros n E I J H H'.
  destruct E.
  destruct H'.
  destruct J.
  destruct I.
  destruct n.
  all: destruct H.
Qed.
```

Tensions

- Coq n'est pas fait au départ pour l'enseignement en L1, on ne peut pas vraiment reprocher à `apply` et `destruct` d'en faire autant.
- Envie d'enseigner tout de même un Coq existant pour, par exemple, que les étudiant·e·s qui le souhaitent puissent suivre « Software Foundations » sans être perdu·e·s.
- Mais pas envie non plus d'expliquer que « en vrai, $\wedge$ et $\vee$ et $=$ sont des propositions inductives et donc `destruct` se comporte ainsi et aussi il y a des subtilités avec les paramètres et les indices et aussi $\bot$ n'a pas de constructeur et $\top$ a un constructeur qui s'appelle `I` ! ».
- Idéalement, il faudrait une option `Set Chick` (ou `Pebble`) ou des versions maison de ces tactiques.

Tensions

- Coq n'est pas fait au départ pour l'enseignement en L1, on ne peut pas vraiment reprocher à `apply` et `destruct` d'en faire autant.
- Envie d'enseigner tout de même un Coq existant pour, par exemple, que les étudiant·e·s qui le souhaitent puissent suivre « Software Foundations » sans être perdu·e·s.
- Mais pas envie non plus d'expliquer que « en vrai, $\wedge$ et $\vee$ et $=$ sont des propositions inductives et donc `destruct` se comporte ainsi et aussi il y a des subtilités avec les paramètres et les indices et aussi $\bot$ n'a pas de constructeur et $\top$ a un constructeur qui s'appelle `I` ! ».
- Idéalement, il faudrait une option `Set Chick` (ou `Pebble`) ou des versions maison de ces tactiques.
- Pour l'instant, on se contente de :

  Il est interdit d'utiliser de la magie pendant les cours.

Un problème de « *universe inconsistency* »

Avec Coq 8.17

```
Goal (A -> B -> C) -> (A -> B -> C).
Proof. Fail exact (imp_refl (_ -> _ -> _)). Admitted.
(*
In environment A, B, C : Prop
   The term "?T -> ?T0 -> ?T1"
   has type "Type" while it is expected to have type
   "Prop" (universe inconsistency: Cannot enforce
   max(coq_nous_embete.1, coq_nous_embete.2,
   coq_nous_embete.3) <= Prop).
*)
```

Un problème de « *universe inconsistency* »

Avec Coq 8.17

```
Goal (A -> B -> C) -> (A -> B -> C).
Proof. Fail exact (imp_refl (_ -> _ -> _)). Admitted.
(*
In environment A, B, C : Prop
   The term "?T -> ?T0 -> ?T1"
   has type "Type" while it is expected to have type
   "Prop" (universe inconsistency: Cannot enforce
   max(coq_nous_embete.1, coq_nous_embete.2,
   coq_nous_embete.3) <= Prop).
*)
```

Mais c'est réglé en Coq 8.18.

Contenus

- Logique propositionnelle
- Des énigmes :
  - Qui prend un dessert ?
  - Qui est ami avec qui ?
- Sujets d'ouverture :
  - logique classique
  - fonctions affines

Logique propositionnelle

- À peu près le même contenu qu'en L1, mais plus doucement
- On commence par $\to$ mais seulement l'introduction
- Avec des noms de variables « concrets » au début.
- Puis $\wedge$ et $\vee$ et seulement après l'élimination de $\to$.

Logique propositionnelle

- À peu près le même contenu qu'en L1, mais plus doucement
- On commence par $\to$ mais seulement l'introduction
- Avec des noms de variables « concrets » au début.
- Puis $\wedge$ et $\vee$ et seulement après l'élimination de $\to$.

```
Theorem hypotheses_faim_soif (Pjaifaim Pjaisoif : Prop) :
  Pjaifaim -> (Pjaisoif -> Pjaifaim /\ Pjaisoif).
Proof.
  (* Début Solution *)
  intros H1.
  intros H2.
  split.
  exact H1.
  exact H2.
Qed.
  (* Fin Solution *)
```

Le problème des desserts

Un classique dans lequel on a remplacé les prénoms `Albert`, `Bernard` et `Charles` (sic) par les prénoms de 3 stagiaires :

```
Variables Jihane_dessert Mehdi_dessert Orane_dessert : Prop.
Hypothesis H1 : Jihane_dessert -> Mehdi_dessert.
Hypothesis H2 : ~(Mehdi_dessert /\ Orane_dessert).
Hypothesis H3 : Jihane_dessert \/ Orane_dessert.
Hypothesis H4 : Orane_dessert -> Jihane_dessert.
```

Premiers pas avec relations et prédicats

```
Variable  (Eleve : Type).
Variables (Ylian Krystoffer Nakshatra Wael : Eleve).
Variable  (Amis : Eleve -> Eleve -> Prop).
Hypothesis CB : Amis Krystoffer Ylian.
Hypothesis nEB : ~(Amis Nakshatra Ylian).
Hypothesis ED : Amis Nakshatra Wael.
Hypothesis amis_amis :
  forall (e1 e2 : Eleve), (Amis e1 e2) -> (Amis e2 e1).
Variable (groupe1 : Eleve -> Prop).
Variable (groupe2 : Eleve -> Prop).
```

%Cpu: 100

Ces élèves de seconde, comme les étudiants de L1, sont actifs presque 100% du temps pendant ces TP.

- C'est vraiment frappant comparé à une séance classique d'exercices de maths
- surtout sur des exercices de logique.
- Question ouverte : est-ce que cette activité sert à quelque chose ?

EntiersNaturels.v

Les sujets de travaux pratiques sont là :

`https://www.math.univ-paris13.fr/~rousselin/ipf.html`

(Presque) rien sous le tapis ici

On a fait des choix clairement orientés « maths-info. »

- Présentation de `nat` comme type inductif.
- Utilisation de `discriminate H.` pour prouver `False` à partir de quelque chose comme `H : S n = 0`.
- La définition de l'addition est un programme.
- Coq sait calculer : commande `Compute`, tactique `simpl`.

Points faibles

Tout ceci se discute.

- `discriminate` est assez mal comprise par les étudiants qui y voient plutôt de la magie
- `simpl` est dangereuse et fait souvent autre chose que ce qu'on voudrait : utiliser `rewrite` est plus flexible
- les définitions avec `Fixpoint` sont en fait assez peu comprises

Autocritique : à ce stade, le cours manque de clarté sur ce sujet. Il faudrait

- soit mettre vraiment l'accent sur les aspects calculatoires et y passer beaucoup plus de temps ;
- soit les abandonner purement et simplement et présenter les entiers naturels sous forme axiomatique.

Les égalités : `rewrite`

On insiste sur `rewrite` car avec les réels il n'y aura plus que ça de toute façon.

- `rewrite <-`, `rewrite ->` et les variantes avec `in` ne posent pas de problème particulier
- Ça se complique avec `rewrite at`, c'est difficile d'expliquer

```
Goal n + m = m + n.
Proof.
  Fail rewrite Nat.add_comm at 2.
  (* Invalid occurrence number: 2. *)
```

Les égalités : `rewrite`

On insiste sur `rewrite` car avec les réels il n'y aura plus que ça de toute façon.

- `rewrite <-`, `rewrite ->` et les variantes avec `in` ne posent pas de problème particulier
- Ça se complique avec `rewrite at`, c'est difficile d'expliquer

```
Goal n + m = m + n.
Proof.
  Fail rewrite Nat.add_comm at 2.
  (* Invalid occurrence number: 2. *)
```

- Normalement il y a quelqu'un qui dit que je devrais utiliser le `rewrite` de `ssreflect`, ...

Les égalités : `rewrite`

On insiste sur `rewrite` car avec les réels il n'y aura plus que ça de toute façon.

- `rewrite <-`, `rewrite ->` et les variantes avec `in` ne posent pas de problème particulier
- Ça se complique avec `rewrite at`, c'est difficile d'expliquer

```
Goal n + m = m + n.
Proof.
  Fail rewrite Nat.add_comm at 2.
  (* Invalid occurrence number: 2. *)
```

- Normalement il y a quelqu'un qui dit que je devrais utiliser le `rewrite` de `ssreflect`, ...
- ... ce à quoi j'ai peu de choses à répondre à part « Software Foundations. » pour ssreflect en général, par contre je ne pense vraiment pas que ce soit adapté à ce niveau !
- `rewrite` unifie silencieusement
- Difficulté pour les étudiants : quels arguments fournir à `rewrite` ?
- *Wishlist :* `Set Printing Parentheses "+".`

Les égalités : `reflexivity`

Pour `reflexivity` on ne veut pas parler de convertibilité, donc on présente cette tactique en disant qu'elle « prouve une égalité lorsque ses deux membres sont identiques (syntaxiquement) ». Bien sûr, en vrai :

```
Goal 2 + 2 = 4.
Proof. reflexivity. Qed.
```

Donc on a encore le même phénomène de « débordement du cadre pédagogique » (et ce n'est pas fini...)

Preuves par récurrence

- Les étudiants jouent le *Natural Number Game*[4] jusqu'à la commutativité de la multiplication.
- On s'exerce à la démonstration par récurrence avec `induction`.
- On insiste sur la différence entre une preuve par récurrence et une preuve par cas sur la nullité d'un entier (`destruct`).
- On évite toujours les inductions multiples, en général :
  - induction sur une variable (laquelle ?)
  - cas sur l'autre.
- On évite d'utiliser `simpl`, qui est susceptible soit de cacher les arguments utilisés, soit de rendre la preuve beaucoup trop difficile.
- Difficulté : la généralité de l'hypothèse de récurrence dépend de la position de la variable dans la formule.

4. `https://adam.math.hhu.de`

Création du module « Initiation aux preuves formelles »

Logique propositionnelle

MathC2+ avec des élèves de seconde

Entiers naturels

Calcul des prédicats et logique classique

Déboires à la maison

Real numbers game

Devoir à la maison final et partiel

Interface utilisateur

Bilan et perspectives

On rentre dans le dur.

- C'est toujours à ce moment que les difficultés commencent pour presque tous les étudiants.
- Je ne sais pas pourquoi !
- Cette année, j'ai abandonné les « ensembles » (`A -> Prop`) qui n'apportaient rien, à ce stade, de plus que le calcul des prédicats (avec des énoncés plus ou moins abstraits).

Injectivité, raisonnement vers l'arrière

```
Lemma inj_comp (f : X -> Y) (g : Y -> Z) :
  injective f -> injective g -> injective (fun x => g (f x)).
Proof.
  intros Hf Hg x x' H.
  apply Hf.
  apply Hg.
  exact H.
Qed.
```

- Les étudiants (qui avaient pourtant écrit des preuves bien plus complexes) sont restés bloqués sur cette preuve.
- Avec un raisonnement vers l'arrière, la preuve ci-dessus paraît pourtant totalement évidente.
- Peut-être que la difficulté est liée à l'ordre supérieur (l'énoncé est quantifié sur des fonctions).

Logique classique

- Nouvel essai cette année avec un sujet spécifique sur la logique classique
- Sauf erreur de ma part, le Coq standard propose bien peu de choses (en fait l'axiome `classic` et `NNPP` et... pas tellement plus).
- TODO : pour un cours de mathématiques on voudrait des tactiques pour la logique classique. La Mathlib de Lean a `push_neg`, `contrapose`, `by_contra`, ...
- De façon surprenante, les étudiants ont été très lents sur cette partie :
  - le tiers exclu peut être utilisé tout le temps avec n'importe quelle proposition
  - cela contraste avec la logique intuitionniste qui donne un sens assez naturel à la preuve

Logique classique

```
Theorem de_morgan_not_and : forall P Q : Prop, ~ (P /\ Q) -> (~
Proof.
  (* Début Solution *)
  intros P Q. intros H.
  destruct (classic P) as [HP | HnP].
  - destruct (classic Q) as [HQ | HnQ].
    + exfalso. apply H. split.
      * exact HP.
      * exact HQ.
    + right. exact HnQ.
  - left. exact HnP.
Qed.
  (* Fin Solution *)
```

Création du module « Initiation aux preuves formelles »

Logique propositionnelle

MathC2+ avec des élèves de seconde

Entiers naturels

Calcul des prédicats et logique classique

Déboires à la maison

Real numbers game

Devoir à la maison final et partiel

Interface utilisateur

Bilan et perspectives

Un devoir sur $\leq$ et $|$

- En 2023, le devoir à la maison a porté sur $\leq$ et $|$.
- Contenu assez exigeant avec réinvestissement de tout ce qui précède.
- L'ordre $\leq$ n'est pas défini, on part de :

```
le_0_l : forall n : nat, 0 <= n
nle_succ_0 : forall n : nat, ~ S n <= 0
succ_le_mono : forall n m : nat, n <= m <-> S n <= S m
```

- À cause de l'équivalence dans `succ_le_mono`, on doit présenter `apply ->` et `apply <-`, ainsi que des versions avec `in`.
- Et on découvre à ce moment que `apply ->` est très différent de `apply` et unifie différemment...

Prouver A avec A

- On a importé `PeanoNat.Nat` pour avoir ces lemmes sur $\leq$ et tous les lemmes de base sur $+$ et $\times$.
- Certains étudiants n'ont pas compris qu'ils ne devaient pas, par exemple, utiliser `le_antisymm` pour prouver `le_antisymm`.

Exemple de rendu :

```
Theorem le_antisymm : forall n m : nat, n <= m -> m <= n -> n =
Proof.
  induction n as [| n' IH].
  -intros n m. intros. apply le_antisymm. exact m. exact H.
  -intros. apply le_antisymm. exact H. exact H0.
Qed.
```

Les `Import` sélectifs

Dans le devoir suivant on a utilisé :

```
Import Nat(add, add_0_l, add_0_r, add_1_l, add_comm).
```

- les lemmes non importés doivent être qualifiés (par exemple avec `Nat.mul_comm`)
- avantage pour l'enseignant·e : permet de vérifier que tous les lemmes nécessaires à la correction ont bien été rappelés
- avantage pour l'étudiant·e : le prévient quand il prouve A avec A (certains ont tout de même prouvé `A` avec `Nat.A` mais ça devient de la mauvaise foi)
- mais ajouté en 8.17 (idem pour `Disable Notation` et Debian

`reflexivity`

Ce premier devoir de 2023 fait apparaître un nouveau débordement du cadre pédagogique, cette fois avec `reflexivity` et `rewrite`.

reflexivity

Ce premier devoir de 2023 fait apparaître un nouveau débordement du cadre pédagogique, cette fois avec `reflexivity` et `rewrite`.

```
Theorem le_refl : forall n, n <= n.
Proof.
induction n as [|n']. reflexivity. reflexivity.
Qed.
Theorem le_trans : forall n m p, n <= m -> m <= p -> n <= p.
Proof.
  induction n as [| n' IH].
  intros n m p.
  -intros H. apply le_0_l.
  -intros m p. intros A B. rewrite A. exact B.
Qed.
```

reflexivity

```
(* En fait, on peut directement écrire : *)
Theorem le_refl' : forall n, n <= n.
Proof. reflexivity. Qed. (* remarquer aussi le intros *; *)
(* En fait, on peut directement écrire : *)
Theorem le_trans' : forall n m p, n <= m -> m <= p -> n <= p.
Proof. intros n m p H H'. rewrite H. exact H'. Qed.
```

Parce que dans `Coq.Numbers.NatInt.NZOrder` :

```
#[global]
Instance le_preorder : PreOrder le.
Proof. split. - exact le_refl. - exact le_trans. Qed.
```

Les réels de Coq

- Historiquement formalisés sous forme axiomatique (thèse de Micaela Mayero)
- avec de nombreux empilements de sous-théories (`Ranalysis[1-5].v`, intégrale de Riemann, trigonométrie, ...) pas toujours développées de façon cohérente.
- utilisés dans l'industrie et notamment `Coquelicot`, `Flocq`, `Intervals`
- 82 fichiers, 50000 lignes, un point connu de ralentissement dans la compilation de Coq
- Récemment (Vincent Semeria, 2019), ajout d'une partie constructive et construction des réels classiques via les coupure de Dedekind (avec l'extensionnalité fonctionnelle, le tiers exclu pour les négations et le « principe d'omniscience limité »)

Corps puis corps ordonné

On présente les réels sous forme axiomatique.

1. Groupe additif (4 axiomes)
2. Corps (+ 6 axiomes)
3. Corps ordonné (+5 axiomes)
4. (borne supérieure magiquement constructive, on n'en a pas eu besoin jusqu'ici) (+1 axiome)

Algèbre

- Les étudiants reconstruisent les propriétés usuelles en partant des axiomes.
- On est sur $\mathbb{R}$, mais les preuves sont des preuves générales d'algèbre.

$$r \times 0 = r \times (0 + 0) = r \times 0 + r \times 0$$

  donc $r \times 0 = 0$.
- Plus de calcul, plus de `simpl`, uniquement `rewrite` et `apply`.
- Est-ce qu'on voudrait le `calc` de Lean ?

Inégalités

- Les étudiants ont peu de bon réflexes avec les inégalités.
- On repasse sur les propriétés du lycée : ajouter un même nombre aux deux membres d'une inégalité, multiplier par un nombre (positif, négatif), ...
- Règles des signes, les carrés sont positifs.
- $0 \leq 1$.

L'affaire du raisonnement vers l'avant

- En 2021, on a introduit le raisonnement vers l'avant (`apply in`) dès le premier sujet
- et on s'est rendu compte que cela créait beaucoup de confusions.
- On l'a introduit deux semaines plus tard en 2022.
- En 2023, j'ai pensé rendre le cours plus simple en ne l'utilisant presque pas, et j'ai enlevé, par exemple

```
Lemma Rplus_eq_compat_l :
  forall r r1 r2, r1 = r2 -> r + r1 = r + r2.
```

qui n'a aucun contenu mathématique et ne sert qu'en raisonnement vers l'avant

L'affaire du raisonnement vers l'avant

- En 2021, on a introduit le raisonnement vers l'avant (`apply in`) dès le premier sujet
- et on s'est rendu compte que cela créait beaucoup de confusions.
- On l'a introduit deux semaines plus tard en 2022.
- En 2023, j'ai pensé rendre le cours plus simple en ne l'utilisant presque pas, et j'ai enlevé, par exemple

```
Lemma Rplus_eq_compat_l :
  forall r r1 r2, r1 = r2 -> r + r1 = r + r2.
```

  qui n'a aucun contenu mathématique et ne sert qu'en raisonnement vers l'avant
- en pensant bien faire.
- Ça ne s'est pas bien passé !
- Malédiction de la connaissance

`rewrite` et `apply`

- L'autre grande difficulté qui survient est qu'on a des égalités en conclusions des lemmes.

```
Lemma Rplus_eq_reg_l
  forall r r1 r2, r + r1 = r + r2 -> r1 = r2.
```

- et les étudiants (conjecture : tout le monde) s'emmêlent les pinceaux entre `rewrite` et `apply` (et variantes avec `in`).
- C'est donc un point à travailler soigneusement (avec plus de temps).

Problèmes sur les réels

- Il manque beaucoup de lemmes élémentaires (par exemple, changer de membre des opérandes).
- La convention de nommage n'est pas cohérente avec le reste de la `stdlib`. `Nat.add_0_r` contre `Rplus_0_r`.

```
Lemma foo (x y : R) : x + y = y + x.
Proof.
  rewrite Radd_comm.
(*
ring_theory ?Goal0 ?Goal1 Rplus ?Goal2 ?Goal3 ?Goal4 eq
*)
```

Suite(s)

- On autorise à ce moment les tactiques automatiques puissantes : `lia`, `lra`, `field`, `field_simplify`
- Il faut bien préciser dans les énoncés ce qui est autorisé ou non. Pour `Lra` et `Lia`, il suffit de les `Require` au bon moment. Pour `field` et `ring`, on n'a pas cette possibilité.

Objectif

```
Theorem CV_plus (An Bn : nat -> R) (l1 l2 : R) :
  Un_cv An l1 -> Un_cv Bn l2 -> Un_cv (fun n => An n + Bn n) (l1 + l2).
Proof.
  intros HA HB eps Heps.
  destruct (HA (eps / 2)) as [n1 Hn1]. lra.
  destruct (HB (eps / 2)) as [n2 Hn2]. lra.
  remember (max n1 n2) as n3 eqn:def_n3.
  exists n3.
  intros n Hn.
  replace eps with (eps/2 + eps/2) by lra.
  apply (Rle_lt_trans _ ((R_dist (An n) l1) + (R_dist (Bn n) l2))).
    apply R_dist_plus.
  apply Rplus_lt_compat.
  - apply Hn1. lia.
  - apply Hn2. lia.
Qed.
```

Objectif

```
Theorem CV_plus (An Bn : nat -> R) (l1 l2 : R) :
  Un_cv An l1 -> Un_cv Bn l2 -> Un_cv (fun n => An n + Bn n) (l1 + l2).
Proof.
  intros HA HB eps Heps.
  destruct (HA (eps / 2)) as [n1 Hn1]. lra.
  destruct (HB (eps / 2)) as [n2 Hn2]. lra.
  remember (max n1 n2) as n3 eqn:def_n3.
  exists n3.
  intros n Hn.
  replace eps with (eps/2 + eps/2) by lra.
  apply (Rle_lt_trans _ ((R_dist (An n) l1) + (R_dist (Bn n) l2))).
    apply R_dist_plus.
  apply Rplus_lt_compat.
  - apply Hn1. lia.
  - apply Hn2. lia.
Qed.
```

- Seulement atteint en 2022 par ≈ 5 étudiants en dernière séance.
- Défi pédagogique, mais probablement accessible avec plus de temps et des sujets encore améliorés

Création du module « Initiation aux preuves formelles »

Logique propositionnelle

MathC2+ avec des élèves de seconde

Entiers naturels

Calcul des prédicats et logique classique

Déboires à la maison

Real numbers game

Devoir à la maison final et partiel

Interface utilisateur

Bilan et perspectives

Devoir à la maison 2021

En 2021, pas le temps, on met tout ce qu'on n'a pas eu le temps de traiter dans un gros devoir à la maison « à la carte », beaucoup trop dur.

```
(*
Petit message :D :

Votre DM, est d'une difficulté !!! Parmis tout ceux qui sont dans le groupe
Discord qu'on a créé pour se coordonner, on est, si je ne me trompe pas que ~5
à avoir à peine réussi à faire la moitié des exos seulement :P !

Et perso j'ai commencé le Mardi pendant les vacances ^_^' !

C'était juste pour vous dire : Courage, vous allez voir des horreurs.

Sérieusement, je trouve Coq très intéressant, et y a de très fortes chances
que j'y retourne après que tout ça soit finit. Mais, je trouve que ça aurait
mérité plus d'heures ou plus d'explications, parce qu'avec les injections et
sur les suites dans la Partie 4, on se retrouve un peu perdu...

La pente de difficulté était beaucoup trop raide, et la plupart se sont
retrouvé perdu (sans même parler de ce DM, où là, tout le monde est perdu !).

C'est aussi un peu dommage de faire tout ça pour ne plus jamais en faire.
*)
```

Tentatives en 2023

- Des `push_neg` à la main (diversement appréciés) :

```
(** **** Exercice [***] : *)
(** Dans l'énoncé suivant, remplacer [True] par une formule sans négation,
    équivalente à la première (les non-égalités ([x <> y]) sont autorisées)
    Ensuite, prouver cette formule. *)
Lemma ex3 : ~(exists (x : R), forall (y : R), f x <= f y) <->
  True.
```

- Un exercice de maths non guidé mais *open bar* :

```
(** Exercice [****] : signe d'une fonction polynôme de degré 2 *)
(** Pour s'amuser, un dernier boss.
    Ici, vous avez droit à tout.
    Il faut déjà savoir le faire sur papier. *)
Lemma polynome2_positif (a b c : R) : a > 0 -> (b² - 4 * a * c < 0) ->
  forall x, a * x² + b * x + c > 0.
```

Un rendu :

```
Proof.
  unfold Rdiv,Rsqr.
  intros H H1 x.
  replace (a*(x * x) + b * x + c) with
    (a*((x+(b*/(2*a)))² +(- (b²-4*a*c))*/(4*a²))).
  - apply (Ropp_gt_lt_0_contravar ((b²-4*a*c))) in H1.
    apply Rmult_gt_0_compat.
      + exact H.
      + apply Rplus_le_lt_0_compat.
        * apply Rle_0_sqr.
        * apply Rmult_gt_0_compat.
          -- exact H1.
          -- apply Rinv_0_lt_compat. apply Rmult_gt_0_compat.
            --- lra.
            --- apply Rlt_0_sqr. apply Rgt_0_not_0. exact H.
  -unfold Rsqr. field.
  apply Rgt_0_not_0. exact H.
Qed.
```

Examen et évaluation

- Des examens toujours beaucoup trop longs.
- Un vrai raté cette année (2023) : j'ai considéré les `Fixpoint` comme acquis et très peu d'étudiants ont compris cette définition :

```
Fixpoint triangle (n : nat) :=
  match n with
    0 => 0
  | S n' => S n' + triangle n'
  end.
```

- Note finale toujours généreuse (ne pas pénaliser les étudiants de DL).
- Bons résultats en 2022

CoqIDE et coqdoc

Pour la première version du cours, on a fait comme *Software Foundations*.

- La source est un fichier `.v` avec des commentaires `coqdoc`
- et quelques balises maison pour générer le sujet et le corrigé avec `sed`.
- Ensuite on génère une page web (un peu moche) avec `coqdoc`, mais il n'y a pas d'interactions dans cette fenêtre.

CoqIDE et coqdoc

Pour la première version du cours, on a fait comme *Software Foundations*.

- La source est un fichier `.v` avec des commentaires `coqdoc`
- et quelques balises maison pour générer le sujet et le corrigé avec `sed`.
- Ensuite on génère une page web (un peu moche) avec `coqdoc`, mais il n'y a pas d'interactions dans cette fenêtre.
- En largeur, ça donne 1 demi écran pour le code, 1 demi écran pour l'état de la preuve et 1 demi écran pour le document web.
- De plus, le document web n'est pas synchronisé avec le code.

CoqIDE et coqdoc

Pour la première version du cours, on a fait comme *Software Foundations*.

- La source est un fichier `.v` avec des commentaires `coqdoc`
- et quelques balises maison pour générer le sujet et le corrigé avec `sed`.
- Ensuite on génère une page web (un peu moche) avec `coqdoc`, mais il n'y a pas d'interactions dans cette fenêtre.
- En largeur, ça donne 1 demi écran pour le code, 1 demi écran pour l'état de la preuve et 1 demi écran pour le document web.
- De plus, le document web n'est pas synchronisé avec le code.
- En pratique les étudiants ont peu à peu abandonné le document web.

Besoin d'autre chose que le code en texte ?

- On pourrait se dire que ce n'est pas grave, le code source contient déjà le cours (les commentaires coqdoc).
- Mais en pratique, les étudiants les lisent peu.
- Et on les comprend, c'est un petit pâté de caractères mono-châsse, si c'était agréable à lire, Knuth n'aurait pas inventé `tex`.
- Et idéalement on voudrait aller plus loin que des paragraphes et des listes avec, par exemple

Besoin d'autre chose que le code en texte ?

- On pourrait se dire que ce n'est pas grave, le code source contient déjà le cours (les commentaires coqdoc).
- Mais en pratique, les étudiants les lisent peu.
- Et on les comprend, c'est un petit pâté de caractères mono-châsse, si c'était agréable à lire, Knuth n'aurait pas inventé `tex`.
- Et idéalement on voudrait aller plus loin que des paragraphes et des listes avec, par exemple
  - des formules mathématiques,
  - des courbes, des dessins,

Besoin d'autre chose que le code en texte ?

- On pourrait se dire que ce n'est pas grave, le code source contient déjà le cours (les commentaires coqdoc).
- Mais en pratique, les étudiants les lisent peu.
- Et on les comprend, c'est un petit pâté de caractères mono-châsse, si c'était agréable à lire, Knuth n'aurait pas inventé `tex`.
- Et idéalement on voudrait aller plus loin que des paragraphes et des listes avec, par exemple
  - des formules mathématiques,
  - des courbes, des dessins,
  - des parties exécutées mais cachées,
  - des exemples en lecture seule,
  - des vidéos de chat quand la preuve est finie

Choix de `jscoq`

- Document unique pour le cours et les preuves à remplir
- Avantage non négligeable : tout le monde travaille avec la même version.
- Inconvénient : loin d'être sobre, plusieurs centaines de Mo de bibliothèques `javascript` à télécharger contre quelques Ko.

Demandes des étudiants

- pouvoir charger un travail enregistré
- avoir une troisième colonne avec les lemmes utilisables (ceux donnés en début d'énoncé et ceux prouvés précédemment dans le même sujet)
- des messages d'erreur qui les aident davantage.

Demandes des étudiants

- pouvoir charger un travail enregistré
- avoir une troisième colonne avec les lemmes utilisables (ceux donnés en début d'énoncé et ceux prouvés précédemment dans le même sujet)
- des messages d'erreur qui les aident davantage.
- TODO : voir ce qui est fait dans `https://adam.math.hhu.de/`

Liste de lemmes

- Irréalisable à plus grande échelle de donner la liste de tous les lemmes utiles.
- On revient à `Search`, mais
  - `Search` expose (presque) toutes les constantes, il y en a des milliers
  - Risque de voir revenir « prouver `A` avec `Nat.A` »
- Pour l'enseignement, besoin en général de contrôle fin
  - sur la sortie de `Search`,
  - en général sur l'environnement, ou à défaut sur ce que l'étudiant·e peut y voir.
- *Wishlist :* un `Search` dans le fichier courant jusqu'à la ligne courante
- (bonus) avec possibilité d'ajouter des lemmes à cette liste.

Création du module « Initiation aux preuves formelles »

Logique propositionnelle

MathC2+ avec des élèves de seconde

Entiers naturels

Calcul des prédicats et logique classique

Déboires à la maison

Real numbers game

Devoir à la maison final et partiel

Interface utilisateur

Bilan et perspectives

Une expérience qui n'est pas isolée

- `https://www.inria.fr/fr/liberabaci`
- `https://smf.emath.fr/publications/la-gazette-de-la-societe-mathematique-de-france-174-octobre`
- *Proof Assistants for Teaching*
- `https://pat2023.icube.unistra.fr/`
- `https://www.uc.pt/en/congressos/thedu/ThEdu24`
- *Stream Teaching [with] Coq* sur `https://coq.zulipchat.com/`

Tentative de conclusion sur l'existant

Frame intentionally left blank

Perspectives

- Travail en cours sur `Reals` : uniformiser les noms avec le reste de la bibliothèque standard
- A priori l'année prochaine, pour le module « Initiation aux preuves formelles » passe à $36h$, avec des parties TD aussi pour faire le lien avec l'écriture de démonstrations en mathématiques.
- À Villetaneuse, Nouveau cours en L2 informatique en 2025-2026 avec Coq