Présentation du projet Lean

Ryan Lahfa

Plan de la présentation

Plan de la présentation

Cette présentation se focalisera sur les aspects avancés, et notamment sur la partie des espaces métriques et de la démonstration que $\mathbb{R}$ est complet.

- Tactiques avancées & *mathlib*

Plan de la présentation

Cette présentation se focalisera sur les aspects avancés, et notamment sur la partie des espaces métriques et de la démonstration que $\mathbb{R}$ est complet.

- Tactiques avancées & *mathlib*
- Quelques mots sur les théorèmes d'analyse et leurs versions séquentielles : skolémisation et tactique *choose*

Plan de la présentation

Cette présentation se focalisera sur les aspects avancés, et notamment sur la partie des espaces métriques et de la démonstration que $\mathbb{R}$ est complet.

- Tactiques avancées & *mathlib*
- Quelques mots sur les théorèmes d'analyse et leurs versions séquentielles : skolémisation et tactique *choose*
- $\mathbb{R}$ est complet, démonstration issue de ZF

Plan de la présentation

Cette présentation se focalisera sur les aspects avancés, et notamment sur la partie des espaces métriques et de la démonstration que $\mathbb{R}$ est complet.

- Tactiques avancées & *mathlib*
- Quelques mots sur les théorèmes d'analyse et leurs versions séquentielles : skolémisation et tactique *choose*
- $\mathbb{R}$ est complet, démonstration issue de ZF
- Bolzano-Weierstrass et la topologie de l'ordre

Plan de la présentation

Cette présentation se focalisera sur les aspects avancés, et notamment sur la partie des espaces métriques et de la démonstration que $\mathbb{R}$ est complet.

- Tactiques avancées & *mathlib*
- Quelques mots sur les théorèmes d'analyse et leurs versions séquentielles : skolémisation et tactique *choose*
- $\mathbb{R}$ est complet, démonstration issue de ZF
- Bolzano-Weierstrass et la topologie de l'ordre
- Espaces métriques

Plan de la présentation

Cette présentation se focalisera sur les aspects avancés, et notamment sur la partie des espaces métriques et de la démonstration que $\mathbb{R}$ est complet.

- Tactiques avancées & *mathlib*
- Quelques mots sur les théorèmes d'analyse et leurs versions séquentielles : skolémisation et tactique *choose*
- $\mathbb{R}$ est complet, démonstration issue de ZF
- Bolzano-Weierstrass et la topologie de l'ordre
- Espaces métriques
- Complétion des espaces métriques

Tactiques avancées & `mathlib`

Avant-propos

Le code contient beaucoup de constructions fondamentales issues de *mathlib* notamment en termes d'ensembles [1] ou d'ordres.

Mais aussi des tactiques spécifiquement issues ou enrichies par la *mathlib*.

Je rappelle aussi que l'esprit de *mathlib* est de donner des constructions à partir de leur « super-généralisation » autant que possible, ainsi que les constructions se font du haut vers le bas, tout d'abord en partant de la généralisation la plus grande puis en posant les constructions élémentaires en tant que corollaires de leur généralisation. C'est le cas par exemple avec l'intégration, où la première intégrale disponible dans *mathlib* était celle de Bochner.

[1] Construction au dessus de la théorie des types dépendants

Tactique `contrapose`

Cette tactique permet de contraposer un objectif qui possède une implication, elle ne fonctionne pas toujours lorsqu'il y a des types dépendants qui sont en jeu.

Tactique `contrapose`

Cette tactique permet de contraposer un objectif qui possède une implication, elle ne fonctionne pas toujours lorsqu'il y a des types dépendants qui sont en jeu.

Par défaut, Lean ne fait que rajouter des ¬, *`contrapose!`* permet de pousser la négation avec *`push_neg`*, une autre tactique dont le rôle est de pousser les négations.

Tactique `contrapose`

Cette tactique permet de contraposer un objectif qui possède une implication, elle ne fonctionne pas toujours lorsqu'il y a des types dépendants qui sont en jeu.

Par défaut, Lean ne fait que rajouter des $\neg$, *`contrapose!`* permet de pousser la négation avec *`push_neg`*, une autre tactique dont le rôle est de pousser les négations.

À savoir, que *`push_neg`* ne devine pas automatiquement comment transformer un $\neg P$ en Q qui lui semble équivalent, par exemple, $\neg \mathrm{set.infinite}$ n'est pas transformé en $\mathrm{set.finite}$, automatiquement. Une modification devrait être effectuée dans *`mathlib`* dans les prochaines semaines à venir pour supporter ce cas par exemple.

Tactique conv

Cette tactique permet de faire de la réécriture ciblé sur un objectif ou une hypothèse, on peut décomposer une expression en sous-expressions à l'aide de *congr* et opérer des modifications simples.

Tactique conv

Cette tactique permet de faire de la réécriture ciblé sur un objectif ou une hypothèse, on peut décomposer une expression en sous-expressions à l'aide de *congr* et opérer des modifications simples.

Cette tactique est utilisé pour la continuité séquentielle, par exemple. On peut souvent la remplacer par une autre stratégie plus rapide ou plus simple.

Tactique conv

Cette tactique permet de faire de la réécriture ciblé sur un objectif ou une hypothèse, on peut décomposer une expression en sous-expressions à l'aide de `congr` et opérer des modifications simples.

Cette tactique est utilisé pour la continuité séquentielle, par exemple. On peut souvent la remplacer par une autre stratégie plus rapide ou plus simple.

Cependant, parfois Lean ne parvient pas à reconnaître certaines expressions dans les cas complexes, elle devient donc très utile dans ces situations désespérées.

Tactique `ext`

Cette tactique permet de ramener une égalité formelle entre deux objets à une égalité relative à un $\forall$.

Elle supporte aussi des objets introduits par l'utilisateur avec le décorateur *`@[ext]`*.

Tactique `ext`

Cette tactique permet de ramener une égalité formelle entre deux objets à une égalité relative à un $\forall$.

Elle supporte aussi des objets introduits par l'utilisateur avec le décorateur *`@[ext]`*.

L'usage dont on en fait dans le projet consiste à prouver que l'égalité de fonctions se fait par l'égalité des images sur la source, par exemple.

Tactique `ext`

Cette tactique permet de ramener une égalité formelle entre deux objets à une égalité relative à un $\forall$.

Elle supporte aussi des objets introduits par l'utilisateur avec le décorateur *`@[ext]`*.

L'usage dont on en fait dans le projet consiste à prouver que l'égalité de fonctions se fait par l'égalité des images sur la source, par exemple.

Mais cela peut fonctionner pour les ordres ou les relations d'équivalences par exemple.

Tactique change

Cette tactique n'est pas évidente à comprendre, mais elle permet de remplacer une expression par une expression qui lui est égale définitionnellement [2], parfois la réécriture n'opère pas correctement sur la réécriture qu'on veut ou alors les définitions sont implicites (le cas des quotients) et on ne connaît pas leur nom mais seulement l'expression qu'on veut obtenir.

[2] Au sens où on a crée une définition qui est égale à cette expression.

Tactique change

Cette tactique n'est pas évidente à comprendre, mais elle permet de remplacer une expression par une expression qui lui est égale définitionnellement [2], parfois la réécriture n'opère pas correctement sur la réécriture qu'on veut ou alors les définitions sont implicites (le cas des quotients) et on ne connaît pas leur nom mais seulement l'expression qu'on veut obtenir.

On en fait un usage lors de la démonstration de la compatibilité de la distance de Cauchy pour la relation d'équivalence $x \equiv y \iff d_C(x, y) = 0$ où $d_C : (x, y) \mapsto \lim_n d(x_n, y_n)$ décrit la distance de Cauchy.

En effet, on peut changer une hypothèse $x \equiv y$ en $d_C(x, y) = 0$ ainsi.

[2] Au sens où on a crée une définition qui est égale à cette expression.

La théorie des ensembles

La théorie d'ensembles possède plusieurs fondations axiomatiques, on s'intéressera essentiellement à la version ZFC.

La théorie des ensembles

La théorie d'ensembles possède plusieurs fondations axiomatiques, on s'intéressera essentiellement à la version ZFC.

Avant de rentrer dans les détails, faisons un crochet sur la notion d'univers dans Lean.

Les univers

Une notion classique en logique est celle des univers, elle provient de la volonté de résoudre et de lever les difficultés de paradoxes rencontrés par plusieurs théories des ensembles (par exemple, paradoxe de Russell).

[3] Ou comme une classe propre, selon ce qu'on accepte de faire.

Les univers

Une notion classique en logique est celle des univers, elle provient de la volonté de résoudre et de lever les difficultés de paradoxes rencontrés par plusieurs théories des ensembles (par exemple, paradoxe de Russell).

L'idée est de se donner un ensemble qui contient tout ce qu'on veut, et de définir tous les objets relativement à cet ensemble « univers », qu'on peut imaginer comme « un ensemble des ensembles » local. [3]

[3] Ou comme une classe propre, selon ce qu'on accepte de faire.

Les univers

Une notion classique en logique est celle des univers, elle provient de la volonté de résoudre et de lever les difficultés de paradoxes rencontrés par plusieurs théories des ensembles (par exemple, paradoxe de Russell).

L'idée est de se donner un ensemble qui contient tout ce qu'on veut, et de définir tous les objets relativement à cet ensemble « univers », qu'on peut imaginer comme « un ensemble des ensembles » local. [3]

Ce détail ne devient intéressant en réalité que pour les mathématiciens travaillent sur des théories requérant de parler d'ensembles « trop gros », donc on les rencontre en théorie des catégories, probabilités, modèles ou en théorie axiomatique des ensembles.

[3] Ou comme une classe propre, selon ce qu'on accepte de faire.

Les univers dans Lean

En Lean, chaque type a un niveau d'univers, donc est de la forme $Type\ \ u$ pour un certain niveau u potentiellement ordinal, et Lean fait les calculs des niveaux automatiquement mais on peut les spécifier.

Les univers dans Lean

En Lean, chaque type a un niveau d'univers, donc est de la forme `Type u` pour un certain niveau `u` potentiellement ordinal, et Lean fait les calculs des niveaux automatiquement mais on peut les spécifier.

Précisément, tout est un `Sort u` dans Lean, avec le fait que `Type u = Sort (u + 1)` et que `Prop` est un type qu'on dit proof-irrelevant, aucun calcul ne peut être effectué dessus, il agit comme une boîte noire. À ce titre, `Prop = Sort 0`.

Les univers dans Lean (suite)

Pourquoi est-ce important ? Prenons le cas de la relation d'équivalence sur « les ensembles » [4] où l'on met en relation deux ensembles s'ils sont en bijection, il est clair qu'on ne peut pas manipuler les classes d'équivalences dans la théorie des ensembles car elles sont presque toutes des classes propres.

Cependant, en théorie des types et avec cette notion d'univers, on peut juste proposer de mettre ces classes dans un niveau supérieur, i.e. `Type u → Type (u + 1)`, c'est ce qui se passe lorsqu'on prend un quotient dans Lean.

[4] Classe propre !

La théorie des ensembles dans Lean

Dans Lean, un ensemble est de type `set X := X → Prop`, à noter que les ensembles ne forment pas un type, mais chaque ensemble est un terme du type précédent. Pour recouvrir un ensemble, on peut procéder à une cœrcion vers `Type` où on construit le sous-type $\{x//x \in S\}$.

La théorie des ensembles dans Lean

Dans Lean, un ensemble est de type `set X := X → Prop`, à noter que les ensembles ne forment pas un type, mais chaque ensemble est un terme du type précédent. Pour recouvrir un ensemble, on peut procéder à une cœrcion vers `Type` où on construit le sous-type $\{x//x \in S\}$.

On peut comprendre la définition comme, étant donné un élément $x \in X$, on peut évaluer un ensemble en x pour déterminer (si `Prop` était calculable...), si oui ou non, x est dans l'ensemble.

La théorie des ensembles dans Lean

Dans Lean, un ensemble est de type `set X := X → Prop`, à noter que les ensembles ne forment pas un type, mais chaque ensemble est un terme du type précédent. Pour recouvrir un ensemble, on peut procéder à une cœrcion vers `Type` où on construit le sous-type $\{x//x \in S\}$.

On peut comprendre la définition comme, étant donné un élément $x \in X$, on peut évaluer un ensemble en x pour déterminer (si `Prop` était calculable...), si oui ou non, x est dans l'ensemble.

Après cela, vient des énormes pans de théories autour de cela, la notion d'ensemble vide, des fonctions `map` pour produire un ensemble de type `set β` à partir d'un ensemble `set α` avec `f : α → β` par exemple.

Notons que Lean implémente des morceaux de ZFC dans `set_theory.zfc`, où des notions de classes, d'axiome du choix sont définis sous le nom de `Set` (et non pas `set`).

La théorie des ordres dans `mathlib`

Lean propose toute une théorie autour des ordres, pour construire $\leq$ et $\geq$, afin de pouvoir généraliser un maximum et de ne pas dupliquer des théorèmes.

[5] Notez la majuscule, elle indique une opération sur des ensembles, tandis qu'en minuscule, elle est binaire.

La théorie des ordres dans `mathlib`

Lean propose toute une théorie autour des ordres, pour construire $\leq$ et $\geq$, afin de pouvoir généraliser un maximum et de ne pas dupliquer des théorèmes.

À noter que ce qui nous a intéressé dans ce projet est toute la théorie des sups/infs, qui utilise la théorie des réseaux, pour définir des sup/inf ponctuels, et peu à peu remonter jusqu'à des structures complètement ordonnées qui définissent *Inf* ou *Sup*[5], mais ces structures ne sont pas commodes, car elles sont l'équivalent de $\overline{\mathbb{R}}$, la droite achevée, ce qui n'est pas désirable pour une construction élémentaire.

[5] Notez la majuscule, elle indique une opération sur des ensembles, tandis qu'en minuscule, elle est binaire.

La théorie des ordres dans `mathlib` (suite)

Lean propose donc des structure conditionnellement complètement ordonnées, qui permettent d'obtenir des informations des bornes supérieures et inférieurs par des hypothèses de bornitudes par en haut ou par en bas : *`cSup`* ou *`cInf`*.

La théorie des ordres dans `mathlib` (suite)

Lean propose donc des structure conditionnellement complètement ordonnées, qui permettent d'obtenir des informations des bornes supérieures et inférieurs par des hypothèses de bornitudes par en haut ou par en bas : *cSup* ou *cInf*.

Dernier point, la définition technique du *Sup* et *Inf* est un peu spéciale, si en réalité, ceux-là n'existent pas, la valeur de retour est absolument arbitraire, une forme un peu plus pauvre du type option qu'on retrouve dans les langages fonctionnels.

« Équivalence » entre preuves ε/δ et preuves séquentielles

Exemple

Durant l'écriture des preuves de la complétude de $\mathbb{R}$, j'ai rencontré quelques problèmes désagréables sur le passage entre le monde séquentiel et le monde ε/δ.

Pour cela, prenons un cadre rapide, (X, d) est un espace métrique.

Précisément, prenons la définition d'une valeur d'adhérence dans une suite, la version séquentielle dit : a est valeur d'adhérence s'il existe une sous-suite (i.e. une extractrice strictement croissante[6]) ϕ tel que $\left(x_{\phi(n)}\right)_n$ converge vers a pour $x = (x_n)_n \in X^{\mathbb{N}}$ une suite donnée.

Si on écrit en quantificateurs cette définition, on obtient :

$$\exists \phi : \mathbb{N} \to \mathbb{N}, \left(\forall (p,q) \in \mathbb{N}^2, p < q \implies \phi(p) < \phi(q)\right)$$
$$\wedge \left(x_{\phi(n)} \underset{n \to +\infty}{\to} a\right)$$

Cependant, les mathématiciens ont tendance à librement changer entre cette version et la définition équivalente ε/δ suivante :

$$\forall \varepsilon > 0, \forall N \geq 0, \exists n \geq N, d(x_n, l) < \varepsilon$$

Problème-type

Cependant, les mathématiciens ont tendance à librement changer entre cette version et la définition équivalente ε/δ suivante :

$$\forall \varepsilon > 0, \forall N \geq 0, \exists n \geq N, d(x_n, l) < \varepsilon$$

En Lean, j'ai procédé à la seconde définition, plus simple à utiliser, sauf pour certaines preuves qui étaient plus simples (notamment Bolzano-Weierstrass version 1 dans le cas infini), où on a envie de dire qu'en réalité, ces définitions ne sont qu'un jeu de réécriture.

Problème-type

Cependant, les mathématiciens ont tendance à librement changer entre cette version et la définition équivalente ε/δ suivante :

$$\forall \varepsilon > 0, \forall N \geq 0, \exists n \geq N, d(x_n, l) < \varepsilon$$

En Lean, j'ai procédé à la seconde définition, plus simple à utiliser, sauf pour certaines preuves qui étaient plus simples (notamment Bolzano-Weierstrass version 1 dans le cas infini), où on a envie de dire qu'en réalité, ces définitions ne sont qu'un jeu de réécriture.

D'où le besoin de pouvoir passer d'une définition à une autre et de donner des formes assez générales de ces types de propositions.

Skolémisation

Après quelques échanges avec des logiciens et M. Paugam, je suis parvenu à la conclusion que finalement, pousser au maximum aboutissait à la nécessité de réécrire tous les formalismes avec un objet qui généralise les suites, par exemple, les filtres. Or, l'objet de ce projet était de **reconstruire** les espaces métriques sans les filtres !

Skolémisation

Après quelques échanges avec des logiciens et M. Paugam, je suis parvenu à la conclusion que finalement, pousser au maximum aboutissait à la nécessité de réécrire tous les formalismes avec un objet qui généralise les suites, par exemple, les filtres. Or, l'objet de ce projet était de **reconstruire** les espaces métriques sans les filtres !

Cependant, on m'avait pointé vers la skolémisation comme solution **partielle**, qui consiste à prendre une formule logique de la forme : $\forall x, \exists y, P(x, y)$ et la transformer en $\exists f, \forall x, P(x, f(x))$.

Ce qui est exactement ce qui se passe plus ou moins lorsqu'on passe d'une forme ϵ/δ à une forme séquentielle, à la différence qu'ici, le $\forall x$ porte sur le même ensemble, autrement dit.

Skolémisation (suite)

$$\forall \varepsilon > 0, \exists N, P(\varepsilon, N) \longrightarrow \exists f : \mathbb{R} \to \mathbb{N}, \forall \varepsilon > 0, P(x, f(x))$$

Ce n'est pas tout à fait ce qu'on veut.

La solution est donc de procéder à une réécriture préliminaire où on affaiblit le quantificateur réel en entier.

Skolémisation (suite)

$$\forall \varepsilon > 0, \exists N, P(\varepsilon, N) \longrightarrow \exists f : \mathbb{R} \to \mathbb{N}, \forall \varepsilon > 0, P(x, f(x))$$

Ce n'est pas tout à fait ce qu'on veut.

La solution est donc de procéder à une réécriture préliminaire où on affaiblit le quantificateur réel en entier.

$$\begin{aligned}
&\forall \varepsilon > 0, \exists N, P(\varepsilon, N) \\
&\qquad \to \forall N \geq 0, \exists n, P\left(\frac{1}{N+1}, n\right) \\
&\qquad \to \exists f : \mathbb{N} \to \mathbb{N}, \forall N \geq 0, P\left(\frac{1}{N+1}, f\left[\frac{1}{N+1}\right]\right)
\end{aligned}$$

Cette fois-ci, on récupère bien ce qu'on veut.

Solution à travers la méta-programmation de Lean : la tactique `choose`

Même s'il existe bel et bien un lemme de skolémisation dans Lean : *`classical.skolem`*, il existe aussi la voie de la méta-programmation, la tactique *`choose`*.

[7] L'autre nom du doux axiome du choix.

Solution à travers la méta-programmation de Lean : la tactique `choose`

Même s'il existe bel et bien un lemme de skolémisation dans Lean : *`classical.skolem`*, il existe aussi la voie de la méta-programmation, la tactique *`choose`*.

C'est la voie choisie dans nos preuves, on se contente d'utiliser *`choose`* qui procède alors à des techniques de méta-programmation pour calculer la fonction (dans le cas où cela est possible), ou qui recourt à *`classical.some`*[7] pour produire l'élément requis.

[7] L'autre nom du doux axiome du choix.

$\mathbb{R}$ est complet

Dépendances axiomatiques de la preuve

On rassemble ici, ce qui me semble être, les dépendances « axiomatiques » sur lesquels se reposeront la preuve de la complétude de $\mathbb{R}$, à noter qu'on n'utilise pas la complétude de $\mathbb{R}$ issue de *mathlib* pour prouver ce point.

- Axiome du choix

Dépendances axiomatiques de la preuve

On rassemble ici, ce qui me semble être, les dépendances « axiomatiques » sur lesquels se reposeront la preuve de la complétude de $\mathbb{R}$, à noter qu'on n'utilise pas la complétude de $\mathbb{R}$ issue de *mathlib* pour prouver ce point.

- Axiome du choix
- Axiome de la borne supérieure

Dépendances axiomatiques de la preuve

On rassemble ici, ce qui me semble être, les dépendances « axiomatiques » sur lesquels se reposeront la preuve de la complétude de $\mathbb{R}$, à noter qu'on n'utilise pas la complétude de $\mathbb{R}$ issue de *mathlib* pour prouver ce point.

- Axiome du choix
- Axiome de la borne supérieure
- Caractère archimédien et valeur absolue usuelle de $\mathbb{R}$

Dépendances axiomatiques de la preuve

On rassemble ici, ce qui me semble être, les dépendances « axiomatiques » sur lesquels se reposeront la preuve de la complétude de $\mathbb{R}$, à noter qu'on n'utilise pas la complétude de $\mathbb{R}$ issue de *mathlib* pour prouver ce point.

- Axiome du choix
- Axiome de la borne supérieure
- Caractère archimédien et valeur absolue usuelle de $\mathbb{R}$
- Théorie d'ensembles de ZF[8]

[8] La distinction est faite, car certains morceaux ne requiert **que** ZF, d'autres **ZFC**.

Bolzano-Weierstrass : idée

On se propose de faire une preuve efficace et un petit peu originale de Bolzano-Weierstrass qui à prouver dans un premier temps :

Théorème
Pour toute partie S infinie de $\mathbb{R}$ bornée, alors il existe $l \in \mathbb{R}$ tel que l soit point limite de S.

Rappelons la définition de point limite afin de fixer les idées :

Définition
On dit que $x \in \mathbb{R}$ est un point limite de $A \subseteq \mathbb{R}$ si x est adhérent à $A\{x\}$.

Bolzano-Weierstrass : détails de la version 2

On se propose de poser comme définition d'ensemble fini, la suivante :

Bolzano-Weierstrass : détails de la version 2

On se propose de poser comme définition d'ensemble fini, la suivante :

Définition
E est un ensemble fini si pour toute partie de E non vide, il admet un plus petit et plus grand élément.

Cette définition est montrée comme équivalente à la finitude usuelle par *lemme_fondateur_de_bw*.

Notons qu'ici, nous n'utilisons **que ZF**.

Bolzano-Weierstrass : pourquoi que ZF ?

Pour prouver que cette définition de finitude est bien équivalente à l'usuelle, nous construisons une théorie des suites strictement croissantes qui à partir de :

- Un ensemble infini S

Bolzano-Weierstrass : pourquoi que ZF ?

Pour prouver que cette définition de finitude est bien équivalente à l'usuelle, nous construisons une théorie des suites strictement croissantes qui à partir de :

- Un ensemble infini S
- Le fait que $(S, \leq)$ est un ordre **bien fondée**

Bolzano-Weierstrass : pourquoi que ZF ?

Pour prouver que cette définition de finitude est bien équivalente à l'usuelle, nous construisons une théorie des suites strictement croissantes qui à partir de :

- Un ensemble infini S
- Le fait que $(S, \leq)$ est un ordre **bien fondée**

Bolzano-Weierstrass : pourquoi que ZF ?

Pour prouver que cette définition de finitude est bien équivalente à l'usuelle, nous construisons une théorie des suites strictement croissantes qui à partir de :

- Un ensemble infini S
- Le fait que $(S, \leq)$ est un ordre **bien fondée**

fournit une suite strictement croissante d'éléments de S.

Ceci ne repose **que** sur ZF et la théorie des ordres bien fondées, on effectue une construction par récurrence structurelle avec `well_founded.fix` `nat.lt_wf`.

Bolzano-Weierstrass : pourquoi que ZF ?

Pour prouver que cette définition de finitude est bien équivalente à l'usuelle, nous construisons une théorie des suites strictement croissantes qui à partir de :

- Un ensemble infini S
- Le fait que $(S, \leq)$ est un ordre **bien fondée**

fournit une suite strictement croissante d'éléments de S.

Ceci ne repose **que** sur ZF et la théorie des ordres bien fondées, on effectue une construction par récurrence structurelle avec *`well_founded.fix nat.lt_wf`*.

D'où, l'équivalence provient juste de la conjonction de l'absence d'un plus grand élément dans une suite strictement croissante et de l'existence de cette suite.

Bolzano-Weierstrass : détails de la version 2 (suite)

De là, on en tire Bolzano-Weierstrass presque immédiatement, en effet :

- Par l'absurde, supposons qu'il n'existe pas de points limites, montrer que l'ensemble est alors fini.

D'où, on obtient notre Bolzano-Weierstrass version 2.

Bolzano-Weierstrass : détails de la version 2 (suite)

De là, on en tire Bolzano-Weierstrass presque immédiatement, en effet :

- Par l'absurde, supposons qu'il n'existe pas de points limites, montrer que l'ensemble est alors fini.
- Il suffit pour cela de montrer que pour toute partie, la borne supérieure et la borne inférieure sont dans la partie.

D'où, on obtient notre Bolzano-Weierstrass version 2.

Bolzano-Weierstrass : détails de la version 2 (suite)

De là, on en tire Bolzano-Weierstrass presque immédiatement, en effet :

- Par l'absurde, supposons qu'il n'existe pas de points limites, montrer que l'ensemble est alors fini.
- Il suffit pour cela de montrer que pour toute partie, la borne supérieure et la borne inférieure sont dans la partie.
- Il suffit de dire que si ce n'était pas le cas, alors ce seraient des points limites de la partie.

D'où, on obtient notre Bolzano-Weierstrass version 2.

Bolzano-Weierstrass : détails de la version 2 (suite)

De là, on en tire Bolzano-Weierstrass presque immédiatement, en effet :

- Par l'absurde, supposons qu'il n'existe pas de points limites, montrer que l'ensemble est alors fini.
- Il suffit pour cela de montrer que pour toute partie, la borne supérieure et la borne inférieure sont dans la partie.
- Il suffit de dire que si ce n'était pas le cas, alors ce seraient des points limites de la partie.
- Donc, a fortiori, de l'ensemble.

D'où, on obtient notre Bolzano-Weierstrass version 2.

Bolzano-Weierstrass : détails de la version 2 (suite)

De là, on en tire Bolzano-Weierstrass presque immédiatement, en effet :

- Par l'absurde, supposons qu'il n'existe pas de points limites, montrer que l'ensemble est alors fini.
- Il suffit pour cela de montrer que pour toute partie, la borne supérieure et la borne inférieure sont dans la partie.
- Il suffit de dire que si ce n'était pas le cas, alors ce seraient des points limites de la partie.
- Donc, a fortiori, de l'ensemble.
- Donc, l'ensemble est fini, ce qui est absurde.

D'où, on obtient notre Bolzano-Weierstrass version 2.

Bolzano-Weierstrass : détails de la version 1

Ici, on revient à la définition classique.

Théorème
Pour toute suite réelle bornée, il existe une valeur d'adhérence de cette suite.

On discute sur la cardinalité des valeurs prises, le cas fini étant juste une application du principe des tiroirs, l'autre cas est une réécriture de la version 2 avec un coup de *choose* pour obtenir une sous-suite et terminer par adhérence séquentielle.

Bolzano-Weierstrass : généralisation ?

Quelque chose qui attire l'attention et qu'on ne peut vraiment voir que lorsqu'on travaille avec un assistant de preuve est la facilité d'affaiblir les hypothèses des théorèmes et de vérifier si ceux-là peuvent être adapté pour fonctionner à nouveau.

Bolzano-Weierstrass : généralisation ?

Quelque chose qui attire l'attention et qu'on ne peut vraiment voir que lorsqu'on travaille avec un assistant de preuve est la facilité d'affaiblir les hypothèses des théorèmes et de vérifier si ceux-là peuvent être adapté pour fonctionner à nouveau.

Ici, on a démontré un Bolzano-Weierstrass qui n'utilise vraiment qu'un point fondamental de $\mathbb{R}$.

Bolzano-Weierstrass : généralisation ? (suite)

Son ordre et plus précisément le fait que la topologie usuelle de $\mathbb{R}$ coïncide avec la topologie de l'ordre usuel.

Ainsi, je ne vois aucun inconvénient à généraliser le Bolzano-Weierstrass aux espaces métriques vérifiant que la topologie induite par la métrique est celle de la topologie de l'ordre !

Bolzano-Weierstrass : généralisation ? (suite)

Son ordre et plus précisément le fait que la topologie usuelle de $\mathbb{R}$ coïncide avec la topologie de l'ordre usuel.

Ainsi, je ne vois aucun inconvénient à généraliser le Bolzano-Weierstrass aux espaces métriques vérifiant que la topologie induite par la métrique est celle de la topologie de l'ordre !

Malheureusement, cela n'a pas été fait, faute de temps, puisqu'il faut construire un petit peu de théorie pour discuter de la topologie induite et de la topologie de l'ordre, et on finit par se demander s'il n'est pas plus simple de juste redescendre au niveau des espaces topologiques… et finalement d'utiliser des filtres !

Espaces métriques, angle d’attaque

Avant-propos

L'idée originale était de construire assez de lemmes et de théorèmes pour pouvoir compléter un espace métrique arbitraire en prenant le quotient des suites de Cauchy par la relation d'équivalence classique.

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts
- Limite d'une suite de Cauchy dans $\mathbb{R}$ avec l'axiome du choix (!)

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts
- Limite d'une suite de Cauchy dans $\mathbb{R}$ avec l'axiome du choix (!)
- Lemmes de passage à la limite dans les inégalités (utile pour l'inégalité triangulaire)

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts
- Limite d'une suite de Cauchy dans $\mathbb{R}$ avec l'axiome du choix (!)
- Lemmes de passage à la limite dans les inégalités (utile pour l'inégalité triangulaire)
- Suites constantes réelles : convergence, être de Cauchy, limite

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts
- Limite d'une suite de Cauchy dans $\mathbb{R}$ avec l'axiome du choix (!)
- Lemmes de passage à la limite dans les inégalités (utile pour l'inégalité triangulaire)
- Suites constantes réelles : convergence, être de Cauchy, limite
- Lemmes d'instance sur le pré-écart pour la structure pré-métrique : symétrie, inégalité triangulaire, positivité.

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts
- Limite d'une suite de Cauchy dans $\mathbb{R}$ avec l'axiome du choix (!)
- Lemmes de passage à la limite dans les inégalités (utile pour l'inégalité triangulaire)
- Suites constantes réelles : convergence, être de Cauchy, limite
- Lemmes d'instance sur le pré-écart pour la structure pré-métrique : symétrie, inégalité triangulaire, positivité.
- Distance de Cauchy

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts
- Limite d'une suite de Cauchy dans $\mathbb{R}$ avec l'axiome du choix (!)
- Lemmes de passage à la limite dans les inégalités (utile pour l'inégalité triangulaire)
- Suites constantes réelles : convergence, être de Cauchy, limite
- Lemmes d'instance sur le pré-écart pour la structure pré-métrique : symétrie, inégalité triangulaire, positivité.
- Distance de Cauchy
- Lemmes autour de la distance de Cauchy et la relation d'équivalence induite

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts
- Limite d'une suite de Cauchy dans $\mathbb{R}$ avec l'axiome du choix (!)
- Lemmes de passage à la limite dans les inégalités (utile pour l'inégalité triangulaire)
- Suites constantes réelles : convergence, être de Cauchy, limite
- Lemmes d'instance sur le pré-écart pour la structure pré-métrique : symétrie, inégalité triangulaire, positivité.
- Distance de Cauchy
- Lemmes autour de la distance de Cauchy et la relation d'équivalence induite
- Compatibilité de la distance de Cauchy sous le quotient

Décomposition du travail

Nous avons abordé ce sujet en décomposant le travail en plusieurs étapes :

- La complétude de $\mathbb{R}$, requise pour pouvoir parler de **la** limite d'une suite de Cauchy à valeurs dans $\mathbb{R}$
- Théorèmes fondamentaux autour des pré-écarts
- Limite d'une suite de Cauchy dans $\mathbb{R}$ avec l'axiome du choix (!)
- Lemmes de passage à la limite dans les inégalités (utile pour l'inégalité triangulaire)
- Suites constantes réelles : convergence, être de Cauchy, limite
- Lemmes d'instance sur le pré-écart pour la structure pré-métrique : symétrie, inégalité triangulaire, positivité.
- Distance de Cauchy
- Lemmes autour de la distance de Cauchy et la relation d'équivalence induite
- Compatibilité de la distance de Cauchy sous le quotient
- Le complété d'un espace métrique est un espace métrique !

Complétion des espaces métriques

Les quotients en théorie des types

Tout d'abord, on se donne un setoïde, c'est la donnée d'un $(E, \sim)$ où E est un ensemble et $\sim$ une relation d'équivalence. L'étude des setoïdes est assez riche en théorie des types [9] et fait jaillir le fait qu'on oublie que $\mathbb{R}$ est bien souvent un ensemble de classes d'équivalence de suites de Cauchy et non pas de réels.

Cela amène aussi le problème en Lean de devoir injecter les rationnels dans les réels pour pouvoir les mélanger librement dans une preuve, d'où l'intérêt de la cœrcion.

[9] intuitionnistes notamment.

Les quotients en théorie des types

Tout d'abord, on se donne un setoïde, c'est la donnée d'un $(E, \sim)$ où E est un ensemble et $\sim$ une relation d'équivalence. L'étude des setoïdes est assez riche en théorie des types [9] et fait jaillir le fait qu'on oublie que $\mathbb{R}$ est bien souvent un ensemble de classes d'équivalence de suites de Cauchy et non pas de réels.

Cela amène aussi le problème en Lean de devoir injecter les rationnels dans les réels pour pouvoir les mélanger librement dans une preuve, d'où l'intérêt de la cœrcion.

Une fois le quotient pris, ce qui se fait par `quotient (un setoïde)`, il faut transporter les théorèmes, structures, relations qu'on avait sur les ensembles originaux.

[9] intuitionnistes notamment.

Les quotients : usage pratique en Lean

En pratique, j'ai identifié trois lemmes dont j'ai fait un usage intensif :

- `quotient.sound` : $[\![X]\!] = [\![Y]\!]$ si et seulement si $x \sim y$.

Les quotients : usage pratique en Lean

En pratique, j'ai identifié trois lemmes dont j'ai fait un usage intensif :

- `quotient.sound` : $[\![X]\!] = [\![Y]\!]$ si et seulement si $x \sim y$.
- `quotient.lift₂` : à partir d'une fonction[10] agissant sur l'ensemble original et d'une preuve de compatibilité, on a une fonction qui agit sur les classes d'équivalence.

[10] À deux variables ici.

Les quotients : usage pratique en Lean

En pratique, j'ai identifié trois lemmes dont j'ai fait un usage intensif :

- `quotient.sound` : $[\![X]\!] = [\![Y]\!]$ si et seulement si $x \sim y$.
- `quotient.lift₂` : à partir d'une fonction[10] agissant sur l'ensemble original et d'une preuve de compatibilité, on a une fonction qui agit sur les classes d'équivalence.
- `quotient.induction_on₂` : à partir de classes d'équivalence[11], il suffit de prouver une proposition pour les classes d'équivalences des éléments originaux.

[10] À deux variables ici.

[11] Ici, pour deux arguments. Une version 3 existe aussi, utilisé pour l'inégalité triangulaire.

Les quotients : usage pratique en Lean

En pratique, j'ai identifié trois lemmes dont j'ai fait un usage intensif :

- `quotient.sound` : $[\![X]\!] = [\![Y]\!]$ si et seulement si $x \sim y$.
- `quotient.lift₂` : à partir d'une fonction[10] agissant sur l'ensemble original et d'une preuve de compatibilité, on a une fonction qui agit sur les classes d'équivalence.
- `quotient.induction_on₂` : à partir de classes d'équivalence[11], il suffit de prouver une proposition pour les classes d'équivalences des éléments originaux.

[10] À deux variables ici.

[11] Ici, pour deux arguments. Une version 3 existe aussi, utilisé pour l'inégalité triangulaire.

Les quotients : usage pratique en Lean

En pratique, j'ai identifié trois lemmes dont j'ai fait un usage intensif :

- `quotient.sound` : $[\![X]\!] = [\![Y]\!]$ si et seulement si $x \sim y$.
- `quotient.lift₂` : à partir d'une fonction[10] agissant sur l'ensemble original et d'une preuve de compatibilité, on a une fonction qui agit sur les classes d'équivalence.
- `quotient.induction_on₂` : à partir de classes d'équivalence[11], il suffit de prouver une proposition pour les classes d'équivalences des éléments originaux.

À noter, la définition de $x \sim y$ dans notre cas est **par définition** `cauchy.dist x y = 0`, donc pour prouver $x \sim y$, on prouve juste la seconde proposition.

[10] À deux variables ici.

[11] Ici, pour deux arguments. Une version 3 existe aussi, utilisé pour l'inégalité triangulaire.

Les quotients : quelques exemples tirés du code

```
presep := λ x y, quotient.induction_on₂ x y $ λ X Y H, show cauchy.dist T X Y = 0, from quotient.eq.1 H,
```

Par exemple, ici, on utilise le fait que `x = y` pour prouver que $X \sim Y$, de l'hypothèse `H: x = y` (où X, Y sont des représentants de x, y).

Les quotients : quelques exemples tirés du code

```
presep := λ x y, quotient.induction_on₂ x y $ λ X Y H, show cauchy.dist T X Y = 0, from quotient.eq.1 H,
```

Par exemple, ici, on utilise le fait que `x = y` pour prouver que $X \sim Y$, de l'hypothèse `H: x = y` (où X, Y sont des représentants de x, y).

```
sep := λ x y, quotient.induction_on₂ x y $ λ X Y H, show [[ X ]] = [[ Y ]], from quotient.sound H,
```

Ici, de façon intéressante, la séparation est une tautologie, puisque l'hypothèse est **exactement** la conclusion. Ce qui est rassurant, puisque c'est l'objectif de ce quotientage.

Les quotients : la preuve de compatibilité

```
def completion.dist_soundness (T: Type*) [espace_metrique T]:
 forall x₁ x₂: cauchy_seqs T, forall y₁ y₂: cauchy_seqs T, (x₁ ≈ y₁) → (x₂ ≈ y₂) → (espace_pre_metrique.d x₁ x₂ = espace_
  intros x y z t Hxz Hyt,
  change (espace_pre_metrique.d x z = 0) at Hxz,
  change (espace_pre_metrique.d y t = 0) at Hyt,
  apply le_antisymm,
  calc
   espace_pre_metrique.d x y ≤ espace_pre_metrique.d x z + espace_pre_metrique.d z y : espace_pre_metrique.triangle x z
   ... ≤ espace_pre_metrique.d x z + (espace_pre_metrique.d z t + espace_pre_metrique.d t y) : add_le_add_left (espace_
   ... = espace_pre_metrique.d z t + espace_pre_metrique.d y t : by rw [Hxz, zero_add, espace_pre_metrique.sym t y]
    ... = espace_pre_metrique.d z t : by rw Hyt; simp,
  calc
   espace_pre_metrique.d z t ≤ espace_pre_metrique.d z x + espace_pre_metrique.d x t : espace_pre_metrique.triangle z x
   ... ≤ espace_pre_metrique.d z x + (espace_pre_metrique.d x y + espace_pre_metrique.d y t) : add_le_add_left (espace_
   ... = espace_pre_metrique.d x z + espace_pre_metrique.d x y : by rw [Hyt, add_zero, espace_pre_metrique.sym z x]
    ... = espace_pre_metrique.d x y : by rw Hxz; simp,
end
```

Les quotients : la preuve de compatibilité

```
def completion.dist_soundness (T: Type*) [espace_metrique T]:
 forall x₁ x₂: cauchy_seqs T, forall y₁ y₂: cauchy_seqs T, (x₁ ≈ y₁) → (x₂ ≈ y₂) → (espace_pre_metrique.d x₁ x₂ = espace_
  intros x y z t Hxz Hyt,
  change (espace_pre_metrique.d x z = 0) at Hxz,
  change (espace_pre_metrique.d y t = 0) at Hyt,
  apply le_antisymm,
  calc
   espace_pre_metrique.d x y ≤ espace_pre_metrique.d x z + espace_pre_metrique.d z y : espace_pre_metrique.triangle x z
   ... ≤ espace_pre_metrique.d x z + (espace_pre_metrique.d z t + espace_pre_metrique.d t y) : add_le_add_left (espace_
   ... = espace_pre_metrique.d z t + espace_pre_metrique.d y t : by rw [Hxz, zero_add, espace_pre_metrique.sym t y]
    ... = espace_pre_metrique.d z t : by rw Hyt; simp,
  calc
   espace_pre_metrique.d z t ≤ espace_pre_metrique.d z x + espace_pre_metrique.d x t : espace_pre_metrique.triangle z x
   ... ≤ espace_pre_metrique.d z x + (espace_pre_metrique.d x y + espace_pre_metrique.d y t) : add_le_add_left (espace_
   ... = espace_pre_metrique.d x z + espace_pre_metrique.d x y : by rw [Hyt, add_zero, espace_pre_metrique.sym z x]
    ... = espace_pre_metrique.d x y : by rw Hxz; simp,
end
```

Il faut noter que la définition de $x \sim z$ et $y \sim t$, est exactement celle à laquelle on s'attend, mais il faut demander à Lean de la changer, pour qu'on puisse travailler avec.

Conclusion

Sur Lean

Lean est un assistant de preuve, intéressant, développé industriellement par une équipe de Microsoft Research, qui servent les besoins d'avant tout, une communauté d'utilisateurs de Microsoft [12] mais aussi la communauté des mathématiciens.

[12] Preuves en interne, certification, etc.

Sur Lean

Lean est un assistant de preuve, intéressant, développé industriellement par une équipe de Microsoft Research, qui servent les besoins d'avant tout, une communauté d'utilisateurs de Microsoft [12] mais aussi la communauté des mathématiciens.

Lean étant plus jeune que ses camarades, a le potentiel d'apprendre des erreurs de Coq et les autres assistants, aussi un avantage non négligeable est l'usage du C++ par opposition à OCaml pour Coq.

[12] Preuves en interne, certification, etc.

Sur Lean

Lean est un assistant de preuve, intéressant, développé industriellement par une équipe de Microsoft Research, qui servent les besoins d'avant tout, une communauté d'utilisateurs de Microsoft [12] mais aussi la communauté des mathématiciens.

Lean étant plus jeune que ses camarades, a le potentiel d'apprendre des erreurs de Coq et les autres assistants, aussi un avantage non négligeable est l'usage du C++ par opposition à OCaml pour Coq.

C'est un point de discorde et tout à fait discutable, à mon sens, le C++ de Lean me paraît plus accessible et plus lisible que le OCaml de Coq, ajoutons aussi que l'écosystème C++ a bien plus d'outils pour la gestion de projet que OCaml, ce qui rend le développement de Coq plus difficile pour une personne complètement extérieure.

[12] Preuves en interne, certification, etc.

Sur les mathématiques

Lean a renforcé mon opinion sur la nécessité de passer à des preuves intégralement formalisées en mathématiques, j'ai découvert beaucoup de littérature prolongeant cette idée et une communauté entière qui s'attarde à pousser ce mouvement, notamment auprès des collègues mathématiciens avec les efforts de Kevin Buzzard.

Sur les mathématiques

Lean a renforcé mon opinion sur la nécessité de passer à des preuves intégralement formalisées en mathématiques, j'ai découvert beaucoup de littérature prolongeant cette idée et une communauté entière qui s'attarde à pousser ce mouvement, notamment auprès des collègues mathématiciens avec les efforts de Kevin Buzzard.

Venant d'un cursus mathématiques-informatique, cette passerelle forte entre les mathématiques et l'informatique représente tout à fait l'endroit où j'aimerais continuer mes études, la logique, la théorie des types, la calculabilité, la théorie des catégories. Elle nourrit une forte intuition en moi guidé par ma compréhension fine de certains concepts en programmation fonctionnelle et leurs applications directes en mathématiques (théorie des types homotopiques, catégories cartésiennes closes, etc.) que je ne connais que de surface et qui m'intéresse énormément.

Au delà de l'aspect, « comment faire », je pense que ces travaux ont tout à gagner à être nettoyé puis packagé dans un jeu-défi type « Natural Number Games » [13], d'après Patrick Massot, l'un des cours enseignés à Orsay propose déjà du Lean et des démonstrations orientés vers les espaces métriques.

[13] Assez facilement faisable

[14] À noter que le complété d'un espace métrique n'est pas au programme de L3 DM-IM en tout cas, peut-être qu'il est au programme de L3 mono mathématiques.

Sur les travaux réalisés

Au delà de l'aspect, « comment faire », je pense que ces travaux ont tout à gagner à être nettoyé puis packagé dans un jeu-défi type « Natural Number Games » [13], d'après Patrick Massot, l'un des cours enseignés à Orsay propose déjà du Lean et des démonstrations orientés vers les espaces métriques.

Ces travaux pourraient être prolongés dans un jeu-défi maintenu par Sorbonne Université autour des espaces métriques et intégrer comme des compléments de cours. Ainsi, cela permettrait à la fois de découvrir Lean aux étudiants mais aussi d'offrir un enseignement **systématique** de ces notions. [14]

[13] Assez facilement faisable

[14] À noter que le complété d'un espace métrique n'est pas au programme de L3 DM-IM en tout cas, peut-être qu'il est au programme de L3 mono mathématiques.

Questions

Pourquoi Lean ?

Plusieurs raisons :

- Les mathématiques sont-elles rigoureuses ?

Pourquoi Lean ?

Plusieurs raisons :

- Les mathématiques sont-elles rigoureuses ?
- Les mathématiques sont-elles reproductibles ?

Pourquoi Lean ?

Plusieurs raisons :

- Les mathématiques sont-elles rigoureuses ?
- Les mathématiques sont-elles reproductibles ?
- Peut-on simplifier le travail des mathématiciens dans leur recherche ?

Pourquoi Lean ?

Plusieurs raisons :

- Les mathématiques sont-elles rigoureuses ?
- Les mathématiques sont-elles reproductibles ?
- Peut-on simplifier le travail des mathématiciens dans leur recherche ?
- Pour l'éducation et l'archivage

Les mathématiques sont-elles rigoureuses ?

Les mathématiques ont une belle histoire autour de la rigueur de leurs travaux, on le voit par exemple avec Euler et sa manipulation des séries ou des produits infinis complètement informels, mais de façon assez stupéfiante, Euler trouvait des choses justes en général.

Les mathématiques sont-elles rigoureuses ?

Les mathématiques ont une belle histoire autour de la rigueur de leurs travaux, on le voit par exemple avec Euler et sa manipulation des séries ou des produits infinis complètement informels, mais de façon assez stupéfiante, Euler trouvait des choses justes en général.

Peu à peu, les mathématiques se sont dotés en fondations, en rigueur de plus en plus grande.

Les mathématiques sont-elles rigoureuses ?

Les mathématiques ont une belle histoire autour de la rigueur de leurs travaux, on le voit par exemple avec Euler et sa manipulation des séries ou des produits infinis complètement informels, mais de façon assez stupéfiante, Euler trouvait des choses justes en général.

Peu à peu, les mathématiques se sont dotés en fondations, en rigueur de plus en plus grande.

Mais sont-elles toujours aussi rigoureuses qu'on le pense ?

Les mathématiques sont-elles reproductibles ?

On dit que les mathématiques sont souvent reproductibles, car pour une publication donnée, on peut reproduire le raisonnement et le vérifier, théorème par théorème.

Les mathématiques sont-elles reproductibles ?

On dit que les mathématiques sont souvent reproductibles, car pour une publication donnée, on peut reproduire le raisonnement et le vérifier, théorème par théorème.

Or, ce n'est pas totalement vrai. Beaucoup de publications reposent sur des travaux précédents, publiés (ou non), et parfois même retirés ou faux.

Pièce à conviction n°1

Le problème de Busemann-Petty indique que si on se donne, K, T des objets symétriques convexes sur $\mathbb{R}^n$ tel que :

$\forall \mathcal{H}$ hyperplan passant par l'origine, $\mathrm{Vol}_{n-1}(K \cap \mathcal{H}) \leq \mathrm{Vol}_{n-1}(T \cap \mathcal{H})$

Alors : $\mathrm{Vol}_{n-1}(K) \leq \mathrm{Vol}_{n-1}(T)$.

Pièce à conviction n°1

Le problème de Busemann-Petty indique que si on se donne, K, T des objets symétriques convexes sur $\mathbb{R}^n$ tel que :

$\forall \mathcal{H}$ hyperplan passant par l'origine, $\mathrm{Vol}_{n-1}(K \cap \mathcal{H}) \leq \mathrm{Vol}_{n-1}(T \cap \mathcal{H})$

Alors : $\mathrm{Vol}_{n-1}(K) \leq \mathrm{Vol}_{n-1}(T)$.

Professeur Gaoyong Zhang prouve en 1994 que le cas $n = 4$ ne vérifie pas la conjecture et le publie dans Annals of Mathematics : Volume 140 (1994), 331-346, donc vérifié et accepté.

Pièce à conviction n°1

Le problème de Busemann-Petty indique que si on se donne, K, T des objets symétriques convexes sur $\mathbb{R}^n$ tel que :

$\forall \mathcal{H}$ hyperplan passant par l'origine, $\mathrm{Vol}_{n-1}(K \cap \mathcal{H}) \leq \mathrm{Vol}_{n-1}(T \cap \mathcal{H})$

Alors : $\mathrm{Vol}_{n-1}(K) \leq \mathrm{Vol}_{n-1}(T)$.

Professeur Gaoyong Zhang prouve en 1994 que le cas $n = 4$ ne vérifie pas la conjecture et le publie dans Annals of Mathematics : Volume 140 (1994), 331-346, donc vérifié et accepté.

Mais, professeur Gaoyong Zhang prouve en 1999 que le cas $n = 4$ vérifie la conjecture et le publie dans Annals of Mathematics : Volume 149 (1999), 535-543.

Pièce à conviction n°2

En 2019, Balakrishnan, Dogra, Mueller, Tuitman and Vonk trouvent toutes les solutions rationnelles à une courbe quadratique de deux variables importantes[15].

Ce calcul a des conséquences importantes en arithmétiques[16]

[15] La courbe modulaire $X_s(13)$.

[16] Problème de classification des corps quadratiques imaginaires $\mathbb{Q}[\sqrt{d}]$ qui ont un nombre de classe 1.

Pièce à conviction n°2

En 2019, Balakrishnan, Dogra, Mueller, Tuitman and Vonk trouvent toutes les solutions rationnelles à une courbe quadratique de deux variables importantes[15].

Ce calcul a des conséquences importantes en arithmétiques[16]

La preuve utilise le système de calcul formel : *magma*, non vérifié et à code source fermé qui utilise des algorithmes rapides qui n'ont pas été publiés.

[15] La courbe modulaire $X_s(13)$.

[16] Problème de classification des corps quadratiques imaginaires $\mathbb{Q}[\sqrt{d}]$ qui ont un nombre de classe 1.

Pièce à conviction n°2

En 2019, Balakrishnan, Dogra, Mueller, Tuitman and Vonk trouvent toutes les solutions rationnelles à une courbe quadratique de deux variables importantes[15].

Ce calcul a des conséquences importantes en arithmétiques[16]

La preuve utilise le système de calcul formel : *magma*, non vérifié et à code source fermé qui utilise des algorithmes rapides qui n'ont pas été publiés.

Même si on le portait à SAGE, lui aussi est non vérifié techniquement.

---

[15] La courbe modulaire $X_s(13)$.

[16] Problème de classification des corps quadratiques imaginaires $\mathbb{Q}[\sqrt{d}]$ qui ont un nombre de classe 1.

Pièce à conviction n°3

Les travaux sur les $(\infty, 1)$-catégories commencent à prendre de l'ampleur, Lurie a écrit plus de 1000 pages sur le sujet, les nouveaux travaux de Peter Scholze se repose sur les catégories infinies.

Pièce à conviction n°3

Les travaux sur les $(\infty, 1)$-catégories commencent à prendre de l'ampleur, Lurie a écrit plus de 1000 pages sur le sujet, les nouveaux travaux de Peter Scholze se repose sur les catégories infinies.

Gaitsgory-Rozenblyum ont eu besoin de résultats analogues sur $(\infty, 2)$-catégories, mais pour gagner du temps, ont indiqué : « Les preuves manquantes seront donnés ultérieurement. »

Pièce à conviction n°3

Les travaux sur les $(\infty, 1)$-catégories commencent à prendre de l'ampleur, Lurie a écrit plus de 1000 pages sur le sujet, les nouveaux travaux de Peter Scholze se repose sur les catégories infinies.

Gaitsgory-Rozenblyum ont eu besoin de résultats analogues sur $(\infty, 2)$-catégories, mais pour gagner du temps, ont indiqué : « Les preuves manquantes seront donnés ultérieurement. »

D'après Kevin Buzzard, Gaitsgory a besoin de 100 pages supplémentaires pour démontrer ce qui reste.

Pièce à conviction n°4

Le lemme de Morse est un résultat fondamental dans la théorie des espaces de hyperboliques au sens de Gromov, publié en 2013 dans Journal of Functional Analysis.

Pièce à conviction n°4

Le lemme de Morse est un résultat fondamental dans la théorie des espaces de hyperboliques au sens de Gromov, publié en 2013 dans Journal of Functional Analysis.

Il contient une erreur basique, une inégalité dans le mauvais sens.

Pièce à conviction n°4

Le lemme de Morse est un résultat fondamental dans la théorie des espaces de hyperboliques au sens de Gromov, publié en 2013 dans Journal of Functional Analysis.

Il contient une erreur basique, une inégalité dans le mauvais sens.

L'erreur a été découverte par Sébastien Gouezel, lors de sa formalisation du lemme dans l'assistant Isabelle. Un nouvel argument par ce dernier et son collègue Vladimir Shchur a été trouvé.

Pièce à conviction n°4

Le lemme de Morse est un résultat fondamental dans la théorie des espaces de hyperboliques au sens de Gromov, publié en 2013 dans Journal of Functional Analysis.

Il contient une erreur basique, une inégalité dans le mauvais sens.

L'erreur a été découverte par Sébastien Gouezel, lors de sa formalisation du lemme dans l'assistant Isabelle. Un nouvel argument par ce dernier et son collègue Vladimir Shchur a été trouvé.

Cette publication nécessite-t-il une évaluation par les pairs ?

Pièces à convictions

Des tas d’erreurs existent dans les mathématiques, certaines ne sont pas trouvés avant formalisation, d’autres sont connues mais non-rétractées ou corrigées.

Pièces à convictions

Des tas d'erreurs existent dans les mathématiques, certaines ne sont pas trouvés avant formalisation, d'autres sont connues mais non-rétractées ou corrigées.

Et ça n'arrive pas qu'en théorie des nombres !

En logique, le théorème fondamental de la logique linéaire a été démontré 20 ans après sa première preuve (fausse) de Jean-Yves Girard par Michel Pagani et Lorenzo Tortora de Falcob dans « Strong normalization property for second order linear logic » en 2010.

Ou alors, la proposition 1 de « Expressiveness of streaming string transducers » de Rajeev Alur et Pavol Černý en 2010.

Comment un assistant de preuve peut aider les mathématiciens

Une preuve complètement formalisé est vérifié à 100 % par un ordinateur, sous hypothèse que l'assistant de preuve repose sur des fondations vérifiables, e.g. le calcul des constructions.

Comment un assistant de preuve peut aider les mathématiciens

Une preuve complètement formalisé est vérifié à 100 % par un ordinateur, sous hypothèse que l'assistant de preuve repose sur des fondations vérifiables, e.g. le calcul des constructions.

La preuve de Gouezel et Shchur est vérifiable dans Isabelle/HOL, elle ne requiert pas vraiment d'évaluation par les pairs.

Comment un assistant de preuve peut aider les mathématiciens

Une preuve complètement formalisé est vérifié à 100 % par un ordinateur, sous hypothèse que l'assistant de preuve repose sur des fondations vérifiables, e.g. le calcul des constructions.

La preuve de Gouezel et Shchur est vérifiable dans Isabelle/HOL, elle ne requiert pas vraiment d'évaluation par les pairs.

Un assistant de preuve peut aussi permettre d'explorer sa théorie de façon efficace, en modifiant les hypothèses, en les affaiblissant, en poussant à la super-généralisation afin d'éviter de reformaliser de façon fastidieuses les mêmes choses.

Une base de données des mathématiques

La bibliothèque *mathlib* a finalement aussi ce rôle d'archiver les preuves et les énoncés des mathématiques connues[17].

[17] et formalisables !

Une base de données des mathématiques

La bibliothèque *mathlib* a finalement aussi ce rôle d'archiver les preuves et les énoncés des mathématiques connues[17].

Même si le code super-généralisé est difficile d'accès lorsqu'on connaît pas bien les concepts, cela permet d'enseigner ou de s'auto-enseigner des constructions hors des sentiers battus des livres et des cours, ce qui améliore l'accès à l'éducation et produit donc plus de mathématiciens.

[17] et formalisables !

Une base de données des mathématiques

La bibliothèque *mathlib* a finalement aussi ce rôle d'archiver les preuves et les énoncés des mathématiques connues[17].

Même si le code super-généralisé est difficile d'accès lorsqu'on connaît pas bien les concepts, cela permet d'enseigner ou de s'auto-enseigner des constructions hors des sentiers battus des livres et des cours, ce qui améliore l'accès à l'éducation et produit donc plus de mathématiciens.

Ajoutons qu'il est tout à fait possible de procéder à une extraction automatisé et un traitement de fond sur le code de Lean pour en tirer des méta-données pour fabriquer des jeux-défis (CodeWars Lean) ou des explications étapes par étapes de théorème automatique, c'est toute la puissance offerte de ce langage.

[17] et formalisables !

Chercher n’importe quel théorème

Aujourd’hui, chercher un théorème n’est pas simple, le langage naturel décrit mal et ne capture pas **juste** la réalité mathématiques, or, un théorème typé permet de générer un moteur de recherche comme https://hoogle.haskell.org/ où on tape la signature pour trouver de la documentation.

Chercher n’importe quel théorème

Aujourd’hui, chercher un théorème n’est pas simple, le langage naturel décrit mal et ne capture pas **juste** la réalité mathématiques, or, un théorème typé permet de générer un moteur de recherche comme https://hoogle.haskell.org/ où on tape la signature pour trouver de la documentation.

C’est d’ailleurs l’objectif d’un de mes projets personnels d’obtenir une recherche dans Lean basé sur les types afin d’avoir un moteur d’auto-complétion utilisable ou une documentation *`mathlib`* plus interactive.

Où en est le théorème d'Ostrowski ?

Mis en pause malheureusement afin de faire avancer les travaux (qui sont avant tout le premier objectif) du projet présenté, cependant.

Le théorème d'Ostrowski

Se faire la main sur des travaux plus simples a été bénéfique, mais pour avoir une meilleure maîtrise, il faudrait :

- Contribuer la prise en charge de $\neg \mathrm{set.infinite}$ pour *push_neg* à *mathlib*

Le théorème d'Ostrowski

Se faire la main sur des travaux plus simples a été bénéfique, mais pour avoir une meilleure maîtrise, il faudrait :

- Contribuer la prise en charge de $\neg \mathrm{set.infinite}$ pour *push_neg* à *mathlib*
- Contribuer le paradoxe de Banach-Tarski-Hausdorff à *mathlib*

Le théorème d'Ostrowski

Se faire la main sur des travaux plus simples a été bénéfique, mais pour avoir une meilleure maîtrise, il faudrait :

- Contribuer la prise en charge de $\neg\mathrm{set.infinite}$ pour *push_neg* à *mathlib*
- Contribuer le paradoxe de Banach-Tarski-Hausdorff à *mathlib*
- Contribuer un correctif pour différencier $\leq$ et $<$ sur la documentation de *mathlib*

Le théorème d'Ostrowski

Se faire la main sur des travaux plus simples a été bénéfique, mais pour avoir une meilleure maîtrise, il faudrait :

- Contribuer la prise en charge de $\neg\mathrm{set.infinite}$ pour *push_neg* à *mathlib*
- Contribuer le paradoxe de Banach-Tarski-Hausdorff à *mathlib*
- Contribuer un correctif pour différencier $\leq$ et $<$ sur la documentation de *mathlib*
- Retravailler tout le premier jet de la preuve pour intégrer directement des éléments dans *mathlib*

Le théorème d'Ostrowski

Se faire la main sur des travaux plus simples a été bénéfique, mais pour avoir une meilleure maîtrise, il faudrait :

- Contribuer la prise en charge de $\neg\mathrm{set.infinite}$ pour *push_neg* à *mathlib*
- Contribuer le paradoxe de Banach-Tarski-Hausdorff à *mathlib*
- Contribuer un correctif pour différencier $\leq$ et $<$ sur la documentation de *mathlib*
- Retravailler tout le premier jet de la preuve pour intégrer directement des éléments dans *mathlib*
- Intégrer la notion de décomposition en base 10 pour les entiers relatifs dans *mathlib*

Le théorème d'Ostrowski

Se faire la main sur des travaux plus simples a été bénéfique, mais pour avoir une meilleure maîtrise, il faudrait :

- Contribuer la prise en charge de $\neg\mathrm{set.infinite}$ pour *push_neg* à *mathlib*
- Contribuer le paradoxe de Banach-Tarski-Hausdorff à *mathlib*
- Contribuer un correctif pour différencier $\leq$ et $<$ sur la documentation de *mathlib*
- Retravailler tout le premier jet de la preuve pour intégrer directement des éléments dans *mathlib*
- Intégrer la notion de décomposition en base 10 pour les entiers relatifs dans *mathlib*
- Retravailler la théorie des valeurs absolues pour y intégrer les valuations au sens d'Artin

Le théorème d'Ostrowski

Se faire la main sur des travaux plus simples a été bénéfique, mais pour avoir une meilleure maîtrise, il faudrait :

- Contribuer la prise en charge de $\neg\mathrm{set.infinite}$ pour *push_neg* à *mathlib*
- Contribuer le paradoxe de Banach-Tarski-Hausdorff à *mathlib*
- Contribuer un correctif pour différencier $\leq$ et $<$ sur la documentation de *mathlib*
- Retravailler tout le premier jet de la preuve pour intégrer directement des éléments dans *mathlib*
- Intégrer la notion de décomposition en base 10 pour les entiers relatifs dans *mathlib*
- Retravailler la théorie des valeurs absolues pour y intégrer les valuations au sens d'Artin
- Généraliser autant que possible la preuve pour en tirer assez facilement le résultat sur $F(T)$

Le théorème d'Ostrowski (suite)

Voilà la feuille de route, à noter qu'une preuve semi-fonctionnel existe toujours mais qu'elle requiert un raffinement au sens précédent.

Le théorème d'Ostrowski (suite)

Voilà la feuille de route, à noter qu'une preuve semi-fonctionnel existe toujours mais qu'elle requiert un raffinement au sens précédent.

J'ai aussi acquis une bien meilleure compréhension de comment structurer l'API et la preuve afin de la rendre lisible et intégrable à *mathlib*.

Le théorème d'Ostrowski (suite)

Voilà la feuille de route, à noter qu'une preuve semi-fonctionnel existe toujours mais qu'elle requiert un raffinement au sens précédent.

J'ai aussi acquis une bien meilleure compréhension de comment structurer l'API et la preuve afin de la rendre lisible et intégrable à *mathlib*.

Ce qui idéalement pourrait conduire à une publication à CPP2021, avec potentiellement Alex J. Best : https://alexjbest.github.io/ qui m'a partagé un de ses débuts de preuve sur le théorème et son intérêt pour la terminer intégralement, notamment pour $F(T)$.