UNE PRÉSENTATION DE LEAN ET MATHLIB À BASE D'EXEMPLES MATHÉMATIQUES

SÉBASTIEN GOUËZEL

Résumé. Les assistants de preuve sont des langages dans lesquels on explique à un ordinateur des preuves mathématiques, l'ordinateur se chargeant de vérifier que le passage d'une ligne à la suivante respecte toujours les règles de la logique. Cela permet entre autres avantages de garantir quasiment à 100% la validité des démonstrations.

Ce texte présente l'assistant de preuve Lean et sa bibliothèque de mathématiques formalisées Mathlib, avec un point de vue de mathématicien praticien (et donc sans parler du tout du fonctionnement interne des assistants de preuve) : quelles différences de point de vue faut-il avoir quand on démontre un résultat sur papier ou quand on le formalise, quels sont les types d'arguments les plus difficiles de chaque côté, quels sont les écueils à éviter, etc.

Ce texte se base sur des exemples concrets. On démontre mathématiquement un théorème sur les vecteurs de translation asymptotique de fonctions 1-lipschitziennes, puis on discute sa formalisation. Ensuite, on se concentre sur la notion de fonction C^n, telle qu'exposée dans les livres ou dans les cours, puis telle que formalisée dans Mathlib – il y a des différences significatives entre les deux, qui illustrent bien les particularités des mathématiques formalisées.

1. Introduction

La pratique des mathématiques se fait généralement avec un cerveau, parfois un papier et un crayon, parfois des collègues, un tableau et des craies. L'ordinateur s'est ajouté récemment à ce paysage, dans des rôles très variés. La mise en forme des textes mathématiques se fait sur un ordinateur. L'ordinateur peut jouer un rôle avant les démonstrations : les expérimentations numériques permettent de se construire une intuition sur certains problèmes. L'ordinateur peut jouer un rôle après les démonstrations : des théorèmes, notamment en analyse numérique, justifient le comportement de certains algorithmes qu'on fait ensuite tourner sur des données complexes. L'ordinateur peut aussi jouer un rôle pendant les démonstrations : certains arguments nécessitent d'analyser un par un des milliers de cas similaires. C'est le cas de la démonstration du théorème des 4 couleurs affirmant que toute carte planaire peut être coloriée en utilisant 4 couleurs de telle sorte que deux pays adjacents ne soient jamais de la même couleur [AH77] ; ou encore de la résolution de la conjecture de Kepler sur les empilements les plus denses de sphères dans $\mathbb{R}^3$, montrant qu'on ne peut pas faire mieux asymptotiquement que l'empilement standard d'oranges [Hal05]. Dans la même direction, mentionnons la résolution par Helfgott de la conjecture de Goldbach faible, suivant laquelle tout nombre impair $n \geq 7$ est somme de trois nombres premiers : son argument

Date: 16 avril 2026.

analytique fonctionne pour $n \geq 10^{30}$, et les cas restants, en nombre fini raisonnable, sont vérifiés numériquement.

Ce texte est consacré à encore un autre type d'utilisation de l'ordinateur pour faire des mathématiques, les assistants de preuve, pour lesquels le terme de vérificateur de preuve serait d'ailleurs probablement plus adapté. Un assistant de preuve est un programme informatique auquel l'utilisateur explique des définitions, des théorèmes et des démonstrations. Le programme est chargé de s'assurer que les démonstrations respectent les règles de la logique, si bien qu'elles sont vérifiées avec un degré de certitude qui dépasse largement tout ce dont l'auteur ou le rapporteur le plus soigneux est capable. Les énoncés sont parfaitement précis, sans hypothèse implicite non écrite ou sans flou sur les définitions : l'utilisateur peut s'il le souhaite développer celles-ci pour voir les termes exacts employés.

Cette exigence de précision risque de se faire au détriment de la concision et de la lisibilité : les *Principia Mathematica* de Russell et Whitehead, publiés en 1913, se veulent parfaitement rigoureux mais ont besoin de 379 pages de formalisme illisible pour démontrer que $1 + 1 = 2$. L'enjeu des assistants de preuve est au contraire de les rendre aussi commodes à utiliser que possible, et proches des pratiques usuelles des mathématiciens. On verra toutefois dans ce texte que coller de trop près aux définitions données dans les livres de référence n'est pas toujours souhaitable.

Les assistants de preuve ont une longue histoire, qui commence en 1967 avec Automath, de de Bruijn. Une grande partie de cette histoire repose sur le fait qu'ils permettent également de démontrer les propriétés de programmes, ce qui est très important d'un point de vue plus appliqué. Nous nous consacrerons exclusivement à leur utilisation en mathématiques. Les trois assistants de preuve les plus utilisés actuellement pour cela sont Isabelle/HOL, Rocq (anciennement Coq) et Lean. Nous nous concentrerons dans ce texte sur Lean, et sa bibliothèque de mathématiques formalisées Mathlib.

Ce texte n'est *pas* une introduction formelle à Lean : à la fin, le lecteur ne saura pas installer le logiciel ni l'utiliser pour vérifier une preuve. Le format est en effet beaucoup trop court pour cela, et pas vraiment adapté car un tel apprentissage ne peut se faire que de manière interactive, en essayant des choses et en apprenant de ses erreurs. Notre objectif est plutôt de présenter un survol impressionniste. Dans une première partie, nous donnerons la démonstration, au format traditionnel, d'un énoncé sur l'itération des applications 1-lipschitziennes de $\mathbb{R}^d$, puis nous présenterons une formalisation en Lean de cette démonstration. Dans une deuxième partie, dans le but d'illustrer les subtilités de la formalisation, nous nous intéresserons à la notion de fonction de classe C^n, et nous verrons pourquoi sa formalisation dans Mathlib s'écarte sur plusieurs points de ce qu'on trouve dans les livres.

De nombreuses ressources sont disponibles pour le lecteur curieux qui voudrait aller plus loin. Le *Natural Number Game*[1] est une introduction à Lean sous forme de jeu dans lequel on démontre les propriétés des opérations sur les entiers en partant juste des axiomes de Peano, parfaitement accessible à des élèves de Maths Sup. Le livre *The Mechanics of Proof*[2] est également une introduction mathématique très accessible pour des étudiants de première

1. `https://adam.math.hhu.de/#/g/leanprover-community/nng4`
2. `https://hrmacbeth.github.io/math2001/`

année. Le livre *Mathematics in Lean*[3] couvre un peu plus de terrain que les deux références précédentes.

2. Comportement asymptotique des semi-contractions

Dans cette section, nous donnons la preuve, au sens traditionnel, d'un résultat sur le comportement asymptotique des semi-contractions de $\mathbb{R}^d$ muni de sa norme euclidienne. Le but est ensuite de décrire une formalisation de cette démonstration, mais indépendamment de cet objectif nous avons choisi d'exposer cette preuve pour son élégance.

Définition 2.1. *Une transformation f d'un espace métrique X est une semi-contraction si elle est* 1*-lipschitzienne, c'est-à-dire si $d(f(x), f(y)) \leqslant d(x, y)$ pour tous $x, y \in X$.*

Si f est une semi-contraction, ses itérations le sont également. Par conséquent, pour tout couple de points x et y, la distance entre $f^n(x)$ et $f^n(y)$ reste uniformément bornée par $d(x, y)$ (où l'on écrit $f^n = f \circ \cdots \circ f$). Ainsi, le comportement asymptotique de $f^n(x)$ (à une erreur bornée près) est indépendant de x.

Dans l'espace euclidien $\mathbb{R}^d$, les premiers exemples de semi-contractions sont donnés par les translations (où $f^n(x)$ tend vers l'infini comme $nv + O(1)$, où v est le vecteur de translation) et les homothéties de rapport $\leqslant 1$ (pour lesquelles $f^n(x)$ reste borné). Le théorème suivant, prouvé en 1981 dans [KN81] sous des hypothèses légèrement plus fortes, montre que ces exemples sont typiques puisqu'il existe toujours un vecteur de translation asymptotique. La preuve que nous donnons est due à Karlsson [Kar01].

Théorème 2.2. *Soit $f : X \to X$ une semi-contraction définie sur un sous-ensemble X de l'espace euclidien $\mathbb{R}^d$. Il existe alors un vecteur v tel que $f^n(x)/n$ converge vers v pour tout $x \in X$.*

Notons que le comportement asymptotique de $f^n(x)/n$ ne dépend pas de x ; il suffit donc de prouver le théorème pour un seul point x. En transposant le tout si nécessaire, on peut supposer que $0 \in X$ et prendre $x = 0$ pour simplifier les notations.

La preuve repose de manière cruciale sur les propriétés de sous-additivité de la suite $u_n = d(0, f^n(0))$.

Définition 2.3. *Une suite $(u_n)_{n\in\mathbb{N}}$ de nombres réels est* sous-additive *si $u_{k+\ell} \leqslant u_k + u_\ell$ pour tous k, ℓ.*

La propriété principale d'une telle suite est donnée dans le lemme suivant, dû à Fekete.

Lemme 2.4. *Soit u_n une suite sous-additive. Alors u_n/n converge vers* $\mathrm{Inf}\{u_n/n, n > 0\} \in \mathbb{R} \cup \{-\infty\}$.

Démonstration. Fixons un entier positif N. Il découle de la sous-additivité de u que $u_{kN+r} \leqslant ku_N + u_r$. En écrivant un entier arbitraire n sous la forme $kN + r$ avec $r < N$, en divisant par n et en prenant la limite supérieure, on obtient $\limsup u_n/n \leqslant u_N/N$. Par conséquent, $\limsup u_n/n \leqslant \mathrm{Inf}\{u_N/N\}$. Le résultat suit puisque $\liminf u_n/n \geqslant \mathrm{Inf}\{u_N/N\}$. □

3. `https://leanprover-community.github.io/mathematics_in_lean/index.html`

En rappelant la notation $u_n = d(0, f^n(0))$, vérifions que cette suite est sous-additive. On a

$$\begin{aligned} u_{k+\ell} = d(0, f^{k+\ell}(0)) &\leqslant d(0, f^k(0)) + d(f^k(0), f^k(f^\ell(0))) \\ &\leqslant d(0, f^k(0)) + d(0, f^\ell(0)) = u_k + u_\ell, \end{aligned} \tag{2.1}$$

où nous avons utilisé la semi-contraction de f^k. Par conséquent, le lemme de Fekete montre que u_n/n converge vers une limite $A \geqslant 0$. Au temps n, le point $f^n(0)$ est proche de la sphère de rayon An centrée en 0. Si $A = 0$, cela prouve le théorème 2.2. Cependant, si $A > 0$, nous devons également prouver la convergence directionnelle de $f^n(0)$. Pour cela, nous utiliserons des instants où la suite u est presque additive, donnés par le lemme suivant.

Lemme 2.5. *Soit $\varepsilon > 0$. Considérons une suite sous-additive u_n telle que $u_n/n \to A \in \mathbb{R}$. Il existe alors des entiers arbitrairement grands n tels que, pour tout $1 \leqslant i \leqslant n$,*

$$u_n \geqslant u_{n-i} + (A - \varepsilon)i. \tag{2.2}$$

Comme u_i est de l'ordre de grandeur Ai, cette inégalité peut être lue de manière informelle comme $u_n \geqslant u_{n-i} + u_i - \delta$, où δ est petit. Elle implique l'additivité de la suite à *tous* les instants intermédiaires entre 1 et n, à une erreur bien contrôlée près.

Démonstration. La suite $u_n - (A - \varepsilon)n$ est équivalente à εn et tend donc vers l'infini. En particulier, il existe des nombres n arbitrairement grands qui constituent des records pour cette suite, dépassant toutes les valeurs précédentes. Pour un tel n, on a pour $i \leqslant n$ l'inégalité $u_{n-i} - (A - \varepsilon)(n - i) \leqslant u_n - (A - \varepsilon)n$, ce qui équivaut au résultat à démontrer. □

Pour $\varepsilon_p = 2^{-p}$, considérons une suite correspondante de temps n_p donnée par le lemme 2.5, tendant vers l'infini. Soit h_p une forme linéaire de norme 1, égale à $-\|f^{n_p}(0)\|$ sur $f^{n_p}(0)$. Alors, pour tout $i \leqslant n_p$,

$$\begin{aligned} h_p(f^i(0)) = h_p(f^i(0) - f^{n_p}(0)) + h_p(f^{n_p}(0)) &\leqslant \|f^i(0) - f^{n_p}(0)\| - \|f^{n_p}(0)\| \\ &\leqslant \|f^{n_p - i}(0)\| - \|f^{n_p}(0)\| = u_{n_p - i} - u_{n_p} \leqslant -(A - \varepsilon_p)i, \end{aligned} \tag{2.3}$$

où la dernière inégalité découle de (2.2). Dans l'inégalité $h_p(f^i(0)) \leqslant -(A - \varepsilon_p)i$ que nous venons d'obtenir, il est remarquable que toute mention de n_p ait disparu.

Considérons maintenant h une valeur d'adhérence de la suite h_p ; c'est une forme linéaire de norme 1. Comme ε_p tend vers 0 lorsque p tend vers l'infini, nous déduisons de ce qui précède l'inégalité suivante :

$$\text{pour tout entier } i, \quad h(f^i(0)) \leqslant -Ai. \tag{2.4}$$

Cette inégalité implique que $f^i(0)$ appartient au demi-espace orienté par h, à une distance Ai de l'origine. Comme sa norme est également essentiellement égale à Ai, on en déduit qu'il est essentiellement orienté dans la direction de h (voir la figure 1). Cela démontre la convergence de $f^i(0)/i$. Si l'on souhaite un argument plus explicite, on peut par exemple considérer une valeur d'adhérence v de $f^i(0)/i$. C'est un vecteur de norme A, satisfaisant $h(v) = -A$. Comme h a norme 1, cela détermine de manière unique v grâce à la convexité stricte de la norme euclidienne. Par conséquent, $f^i(0)/i$ possède une unique valeur d'adhérence, et elle converge. Ceci conclut la preuve du théorème 2.2. □

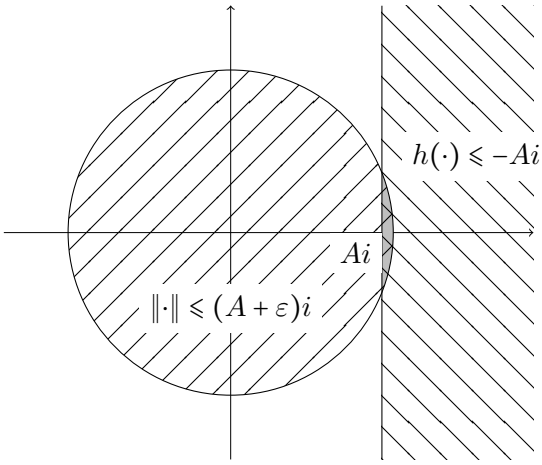

FIGURE 1. $f^i(0)$ appartient à l'intersection des zones hachurées

Remarque 2.6. La preuve n'a pas utilisé la dimension finie (si l'on remplace les limites fortes par des limites faibles). Par conséquent, le résultat reste vrai dans les espaces de Hilbert, ou plus généralement dans les espaces de Banach uniformément convexes.

Remarque 2.7. La majeure partie de la démonstration est valable dans un espace de Banach général : il existe toujours une forme linéaire h de norme au plus égale à 1 telle que $h(f^i(0)) \leqslant -Ai$ pour tout i (ce qui implique déjà des résultats non triviaux, par exemple la suite $(f^i(0))_{i\geqslant 0}$ est contenue dans un demi-espace si $A > 0$, car la forme linéaire h est nécessairement non nulle dans ce cas). Le seul point où la preuve échoue est le dernier argument, qui repose sur la convexité stricte de la norme.

On peut se demander s'il s'agit là d'une limite de la démonstration, ou si celle-ci rend compte de toutes les informations pertinentes. De fait, le théorème ci-dessus est faux sans hypothèses de convexité sur la norme. Décrivons rapidement un contre-exemple dû à [KN81], dans $\mathbb{R}^2$ avec la norme sup. Fixons les deux vecteurs $v_+ = (1,1)$ et $v_- = (1,-1)$, tous deux de norme 1. Nous définissons un chemin continu $\gamma : \mathbb{R}_+ \to \mathbb{R}^2$ partant de 0, de la forme $\gamma(t) = (t, \varphi(t))$, en suivant la direction v_+ pendant un temps S_0, puis la direction v_- pendant un temps $S_1 \gg S_0$, puis la direction v_+ pendant un temps $S_2 \gg S_1$, et ainsi de suite. L'angle entre $\gamma(t)$ et la droite horizontale varie entre $-\pi/4$ et $\pi/4$. Comme les pentes de v_+ et v_- sont égales à 1, le chemin γ est une isométrie de $\mathbb{R}_+$ vers son image. Soit $h(x_1, x_2) = x_1$ la première coordonnée. Alors l'application $f : x \mapsto \gamma(|h(x)| + 1)$ est une semi-contraction, en tant que composition de fonctions 1-lipschitziennes. On vérifie facilement que $f^n(0) = \gamma(n)$. Par conséquent, par construction, $f^n(0)/n$ ne converge pas.

3. Formalisation du théorème sur les semi-contractions

Comme premier contact avec Lean, nous allons décrire intégralement une formalisation du théorème 2.2. Pour simplifier un peu, on supposera que X est l'espace $\mathbb{R}^d$ entier. On va donner intégralement le texte de la formalisation, en différentes étapes.

Le langage ressemble à un langage de programmation, avec une syntaxe stricte, dans laquelle on peut donner des définitions, des énoncés, et des démonstrations. On commence

par définir ce que veut dire pour une application $f : X \to Y$ entre espaces métriques d'être une semicontraction. Une définition est introduite par le mot-clé `def`. On donne ensuite son nom, ses paramètres, puis après `:=` le contenu proprement dit de la définition. Comme dans tout langage informatique, il est important de donner suffisamment de commentaires, ici entre `/--` et `-/`.

```
variable {X Y : Type*} [MetricSpace X] [MetricSpace Y]

/-- Une semicontraction entre espaces métriques est une application qui n'augmente pas les
distances. -/
def semicontraction (f : X → Y) : Prop :=
  ∀ x y, dist (f x) (f y) ⩽ dist x y
```

Après cette première définition, donnons notre premier énoncé, et notre première démonstration. L'énoncé est introduit par le mot clé `lemma`. Suivent le nom de l'énoncé, toutes ses hypothèses, et la conclusion séparée des hypothèses par `:`. Ici, les hypothèses sont que les applications $f : X \to Y$ et $g : Y \to Z$ sont deux semicontractions entre espaces métriques, et la conclusion est que leur composition $g \circ f$ est également une semicontraction. Viennent ensuite `:=` puis la démonstration.

```
/-- La composition de deux semicontractions est une semicontraction. -/
lemma semicontraction.comp {Z : Type*} [MetricSpace Z] {g : Y → Z} {f : X → Y}
    (hg : semicontraction g) (hf : semicontraction f) :
    semicontraction (g ∘ f) :=
  fun x y ↦ (hg (f x) (f y)).trans (hf x y)
```

Cet énoncé est une évidence, qu'on ne se donnerait pas la peine d'énoncer dans un lemme spécifique dans un texte mathématique traditionnel, mais qu'on ne se priverait pas d'utiliser ensuite. En formalisation, de tels raccourcis ne sont pas possibles, et on est donc amené à écrire beaucoup plus de lemmes triviaux comme celui-ci. Je n'en dirais pas plus sur sa preuve, écrite ci-dessus de manière assez absconse. On pourrait donner d'autres preuves plus éclairantes, ou dire au système de se débrouiller tout seul avec `:= by grind [semicontraction]`. La *tactique* `grind` est un couteau suisse qui suffit souvent pour les énoncés où il suffit de dérouler les définitions.

Voici un lemme du même cachet, affirmant que les itérés f^n d'une semicontraction sont encore des semicontractions.

```
lemma iterate {f : X → X} (h : semicontraction f) (n : ℕ) :
    semicontraction (f ^[n]) := by
  induction n with
  | zero => simp [semicontraction]
  | succ n ih => simpa using comp ih h
```

Ce lemme est encore évident, mais il faut malgré tout détailler sa démonstration pour que le système l'accepte. Elle se fait par récurrence sur n, en traitant séparément le cas $n = 0$ d'une part, et le cas $n + 1$ en supposant le résultat démontré à l'étape n d'autre part. La syntaxe de la preuve correspond à cette décomposition. On découvre ici une autre tactique, le simplificateur `simp`. Notons que le cas $n + 1$ utilise le lemme précédent sur la composition : l'hypothèse de récurrence `ih` affirme que f^n est une semicontraction, l'hypothèse `h` affirme

que f est une semicontraction, donc le lemme précédent `comp` appliqué à ces deux énoncés affirme que $f^n \circ f$ est une semicontraction, et finalement le simplificateur est utilisé pour voir que cette fonction coïncide avec f^{n+1}, ce qui conclut la preuve.

La démonstration du théorème 2.2 utilise dans le lemme 2.5 le résultat suivant, qu'on y a énoncé comme une évidence. Dans le cadre formel, il faut cependant donner les détails. Ce lemme affirme que, étant donnée une suite (u_n) de nombre réels qui tend vers l'infini (hypothèse `hu`) et un entier N, alors il existe un entier $n \geqslant N$ tel que u_n surpasse toutes les valeurs précédentes de la suite. Ce lemme est facile, mais nécessite un argument. On peut par exemple le démontrer par l'absurde, en prouvant alors par récurrence que u_n reste toujours borné par le maximum M de u sur $[0, N]$ (en distinguant le cas $n \leq N$ et le cas $n > N$), ce qui contredit le fait que u tende vers l'infini. La preuve formalisée ci-dessous traduit exactement cet argument, la difficulté principale étant de trouver la bonne incantation pour le maximum de u sur $[0, N]$. Une fois cette architecture en place, il n'y a plus d'idée à avoir, et tous les objectifs intermédiaires sont traités par la tactique automatique `grind`.

```
lemma exists_high_score (u : ℕ → ℝ) (hu : Tendsto u atTop atTop) (N : ℕ) :
  ∃ n ⩾ N, ∀ m ⩽ n, u m ⩽ u n := by
by_contra!
let M := (Finset.image u (Finset.range (N+1))).max' (by simp)
have A n : u n ⩽ M := by
  induction n using Nat.strong_induction_on with | h n ih =>
  rcases le_total n N with hnN | hNn
  · apply Finset.le_max'
    grind
  · grind
obtain ⟨n, hn⟩ : ∃ n, M + 1 ⩽ u n := (tendsto_atTop.mp hu (M + 1)).exists
grind
```

Cet argument commence à être un peu plus long, et peut donc sembler impossible à écrire directement sans erreur de syntaxe ou d'argumentation. Effectivement, un point clé des assistants de preuve est qu'ils sont *interactifs* : quand on commence à écrire la preuve, Lean vérifie en temps réel s'il comprend les arguments, signale les erreurs de syntaxe, et donne l'état actuel des objets (où on en est, ce qu'on a déjà démontré, ce qu'il reste à faire).

La figure 2 montre ce qu'on voit dans une fenêtre annexe appelée Infoview si on place le curseur au début de la ligne `apply Finset.le_max'`. Cette fenêtre mentionne tous les objets en cours (par exemple u) et ce qu'on sait sur eux. Par exemple `this` est l'hypothèse du raisonnement par l'absurde, `ih` est l'hypothèse du raisonnement par récurrence, et `hnN` est le fait que $n \leq N$, qui correspond à la branche de la discussion qu'on est en train de traiter. Enfin, précédé du symbole $\vdash$, on voit ce qu'il faut démontrer.

On peut noter que l'affichage ne distingue pas explicitement ici les objets comme u ou N, et les propriétés qu'ils vérifient comme `ih` ou `hnN`. C'est un reflet de son fonctionnement interne, basé sur la théorie des types, qui rend les choses très confortables mais sur lequel nous n'insisterons pas dans ce texte

Une fois ces préliminaires réglés, commençons la preuve du théorème 2.2. On fixe le contexte : dans toute la suite, E est un espace vectoriel normé et f une semi-contraction de cet espace.

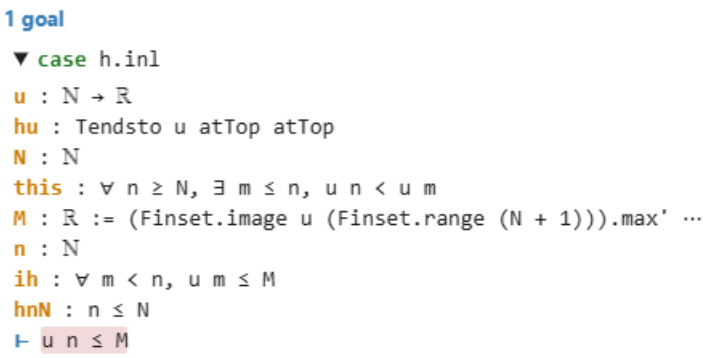

FIGURE 2. Fenêtre Infoview de Lean

```
variable {E : Type*} [NormedAddCommGroup E] {f : E → E} (h : semicontraction f)
include h
```

Comme dans la preuve informelle, on introduit une notation u_n pour la distance entre 0 et $f^n(0)$. On donne aussi la formalisation de la sous-additivité de cette suite.

```
/-- Une notation pratique pour la distance entre `0` et `f^n 0`. -/
local notation "u" => (fun n ↦ dist (f^[n] 0) 0)

lemma u_subadditive : Subadditive u := by
 intro m n
 calc u (m + n)
 _ = dist (f^[m + n] 0) 0 := rfl
 _ ⩽ dist (f^[m+n] 0) (f^[n] 0) + dist (f^[n] 0) 0 := dist_triangle _ _ _
 _ = dist (f^[n] (f^[m] 0)) (f^[n] 0) + dist (f^[n] 0) 0 := by rw [add_comm m n,
   iterate_add_apply]
 _ ⩽ dist (f^[m] 0) 0 + dist (f^[n] 0) 0 := add_le_add (h.iterate _ _ _) le_rfl
 _ = u m + u n := rfl
```

Cette preuve est présentée sous forme d'une suite d'inégalités, exactement comme dans l'équation (2.1). À chaque ligne, on doit cependant fournir la preuve de la validité de l'inégalité, sans la laisser au lecteur comme on l'avait implicitement fait dans la preuve informelle. Notons que la définition de `Subadditive` est déjà disponible dans la bibliothèque utilisée, appelée Mathlib, si bien que nous n'avons pas eu besoin de la redéfinir dans cet argument. De même, le lemme de Fekete est déjà disponible (sous le nom `Subadditive.tendsto_lim`), et nous pouvons donc l'utiliser directement pour obtenir la convergence de u_n vers sa limite, que nous notons `h.l` (qui correspond à A dans l'argument informel) :

```
/-- `h.l` est le nombre tel que `h.u n` croît comme `n * h.l`. -/
def l := h.u_subadditive.lim

lemma tendsto_lim : Tendsto (fun n ↦ u n / n) atTop (𝓝 h.l) := by
 have B : BddBelow (Set.range (fun n ↦ u n / n)) := by
```

```
  refine ⟨0, fun x hx ↦ ?_⟩
  obtain ⟨y, hy⟩ : ∃ y, ‖f^[y] 0‖ / y = x := by simpa using hx
  rw [← hy]
  positivity
 exact h.u_subadditive.tendsto_lim B

lemma l_nonneg : 0 ⩽ h.l :=
 ge_of_tendsto' h.tendsto_lim (fun n ↦ by positivity)
```

Notons une différence entre la version informelle et la version formalisée : le lemme 2.4 mentionne la convergence vers un élément de $\mathbb{R} \cup \{-\infty\}$. Il est possible de formaliser cet énoncé en Lean, mais il n'est pas pratique à utiliser puisque, si la limite est $-\infty$, on sort de l'ensemble des nombres réels ce qui introduit toutes sortes de complications. La variante affirmant que, si u_n est sous-additive et u_n/n est bornée inférieurement alors u_n/n converge dans $\mathbb{R}$, est beaucoup plus pratique à utiliser. C'est donc à celle-ci que nous faisons appel dans la preuve ci-dessus. C'est d'ailleurs ce que nous avions aussi fait implicitement dans la version informelle du texte, puisque nous n'avions jamais discuté la possibilité que la limite soit $-\infty$. Pour appliquer cette version dans la formalisation, il y a toutefois cette hypothèse de bornitude inférieure à vérifier – elle est triviale puisque la suite est positive ou nulle, mais il faut quand même le dire explicitement.

On déduit de la convergence de u_n/n vers `h.l` que $u_n - nw$ tend vers l'infini pour tout $w <$ `h.l`. C'est une manipulation de limites, qui peut se détailler comme suit : $u_n/n - w$ tend vers une limite strictement positive, donc $(u_n/n - w) * n$ tend vers l'infini, et cette suite coïncide avec $u_n - nw$ à partir d'un certain rang (précisément, pour $n \geqslant 1$), donc la suite initiale tend également vers l'infini. Voilà la traduction formalisée de cet argument.

```
lemma tendsto_sub_atTop {w : ℝ} (hw : w < h.l) :
   Tendsto (fun (n : ℕ) ↦ u n - n * w) atTop atTop := by
 have A : Tendsto (fun n ↦ u n / n - w) atTop (𝓝 (h.l - w)) :=
   h.tendsto_lim.sub tendsto_const_nhds
 have : Tendsto (fun n ↦ (u n / n - w) * n) atTop atTop := by
   have I : 0 < h.l - w := by linarith
   apply A.pos_mul_atTop I
   exact tendsto_natCast_atTop_atTop -- exact?
 apply Tendsto.congr' _ this
 filter_upwards [Ioi_mem_atTop 0] with n (hn : 0 < n)
 field_simp
```

On en vient au cœur de la preuve. À partir de maintenant, on expliquera moins les détails de la formalisation, même si on en donnera encore le texte complet. On se contentera d'expliquer le contenu des énoncés (qui correspondent à la preuve informelle donnée dans la section 2) et éventuellement de pointer certains points notables dans les preuves.

On commence par formaliser l'argument autour de l'équation (2.3) : étant donné $w <$ `h.l` et un entier N, il existe une forme linéaire v de norme au plus 1 telle que $v(f^i(0)) \leqslant -iw$ pour tout $i \leqslant N$. Cet argument utilise l'existence d'un temps record au-delà de N, qu'on a établi plus haut.

```
variable [InnerProductSpace ℝ E]

lemma exists_dual_up_to_of_lt {w : ℝ} (hw : w < h.l) (N : ℕ) :
  ∃ (v : StrongDual ℝ E), ‖v‖ ≤ 1 ∧ ∀ i ≤ N, v (f^[i] 0) ≤ - i * w := by
 obtain ⟨n, Nn, hn⟩ : ∃ n ≥ N, ∀ m ≤ n, u m - m * w ≤ u n - n * w :=
  exists_high_score _ (h.tendsto_sub_atTop hw) N
 obtain ⟨v, vnorm, hv⟩ :
  ∃ (v : StrongDual ℝ E), ‖v‖ ≤ 1 ∧ v (-(f^[n] 0)) = ‖-(f^[n] 0)‖ :=
   exists_dual_vector'' ℝ (-(f^[n] 0))
 refine ⟨v, vnorm, fun i hi ↦ ?_⟩
 have A : i ≤ n := hi.trans Nn
 calc
 v (f^[i] 0) = v (f^[i] 0 - (f^[n]) 0) - v (- (f^[n] 0)) := by
  simp only [map_sub, map_neg, sub_neg_eq_add, sub_add_cancel]
 _ ≤ 1 * ‖(f^[i]) 0 - (f^[n]) 0‖ - ‖-(f^[n]) 0‖ := by
   rw [hv]
   gcongr
   apply (le_abs_self _).trans
   exact v.le_of_opNorm_le vnorm _
 _ = dist (f^[i] 0) (f^[i] (f^[n-i] 0)) - dist 0 (f^[n] 0) := by
  rw [← iterate_add_apply, one_mul, dist_eq_norm, dist_eq_norm,
      zero_sub, ← Nat.add_sub_assoc A, Nat.add_sub_cancel_left]
 _ ≤ dist 0 (f^[n-i] 0) - dist 0 (f^[n] 0) := sub_le_sub (h.iterate i _ _) le_rfl
 _ = u (n - i) - u n := by simp only [dist_comm (0 : E)]
 _ ≤ - n * w + (n - i : ℕ) * w := by linarith [hn (n-i) (Nat.sub_le n i)]
 _ = - i * w := by rw [Nat.cast_sub A]; ring
```

Notons que dans l'argument informel on affirmait avoir une forme linéaire de norme exactement 1 satisfaisant la conclusion. C'était une imprécision délibérée de ma part, pour illustrer que des petites erreurs sont très difficiles à repérer sans formalisation (bravo à toi, lecteur, si tu ne t'es pas laissé tromper !). Le problème est que, sur l'espace vectoriel de dimension 0, il n'y a pas de forme linéaire de norme 1, toutes les formes linéaires sont nulles. Le meilleur énoncé possible donne donc une forme linéaire de norme au plus 1, ce qui suffit néanmoins pour la suite de l'argument.

En prenant une limite des formes linéaires données par le lemme précédent, lorsque w tend vers `h.l` et N tend vers l'infini, on obtient une forme linéaire de norme au plus 1 qui vérifie $v(f^i(0)) \leqslant -i$`h.l` pour tout i.

```
variable [FiniteDimensional ℝ E]

lemma exists_dual : ∃ (v : StrongDual ℝ E), ‖v‖ ≤ 1 ∧ ∀ i, v (f^[i] 0) ≤ -i * h.l := by
 -- on part d'une suite `w_n` qui tend vers `h.l` par valeurs inférieures
 obtain ⟨w, -, w_lt, w_lim⟩ : ∃ (w : ℕ → ℝ), StrictMono w
  ∧ (∀ (n : ℕ), w n < h.l) ∧ Tendsto w atTop (𝓝 h.l) :=
   exists_seq_strictMono_tendsto _
 -- pour chaque `n`, on peut choisir un élément du dual de norme au plus `1`
```

```
-- tel que `y (f^[i] 0) ≤ - i w_ n` pour tout `i ≤ n`, d'après le lemme
-- précédent
have A n : ∃ (y : StrongDual ℝ E), ‖y‖ ≤ 1 ∧ ∀ i ≤ n, y (f^[i] 0) ≤ - i * w n :=
  h.exists_dual_up_to_of_lt (w_lt n) n
choose y hy using A -- oui, c'est l'axiome du choix !
-- on extrait une sous-suite `y_ {φ n}`, qui converge vers une limite `v`.
obtain ⟨v, v_mem, φ, φ_mono, φlim⟩ :
  ∃ v ∈ closedBall (0 : StrongDual ℝ E) 1, ∃ (φ : ℕ → ℕ),
    StrictMono φ ∧ Tendsto (y ∘ φ) atTop (𝓝 v) := by
  -- on utilise que `StrongDual ℝ E` est propre
  refine IsCompact.tendsto_subseq (ProperSpace.isCompact_closedBall _ _) ?_
  intro n
  simp [(hy n).1]
-- on va voir que cette limite convient.
refine ⟨v, by simpa using v_mem, fun i ↦ ?_⟩
-- on a fixé `i`, il faut voir que `v (f^[i] 0) ≤ -i h.l`.
-- Pour cela, on passe à la limite
-- dans les inégalités sur les `y_ n (f^[i] 0)`.
have A : Tendsto (fun n ↦ ((y ∘ φ) n) (f^[i] 0)) atTop (𝓝 (v (f^[i] 0))) :=
  ((isBoundedBilinearMap_apply.isBoundedLinearMap_left (f^[i] 0)).continuous.tendsto _)
    .comp φlim
have B : Tendsto (fun n ↦ -(i : ℝ) * w (φ n)) atTop (𝓝 (- i * h.l)) :=
  (tendsto_const_nhds.mul w_lim).comp φ_mono.tendsto_atTop
have C : ∀ᶠ n in atTop, ((y ∘ φ) n) (f^[i] 0) ≤ - i * w (φ n) := by
  apply eventually_atTop.2 ⟨i, fun n hn ↦ ?_⟩
  apply (hy (φ n)).2 i
  exact le_trans hn (φ_mono.id_le n)
exact le_of_tendsto_of_tendsto A B C
```

Dans cette preuve, pour chaque n le lemme précédent nous donne une forme linéaire y se comportant bien jusqu'au temps n (c'est l'affirmation `A n`). Ce n'est *pas* la même chose qu'avoir une suite $(y_n)_{n\in\mathbb{N}}$ où y_n se comporte bien jusqu'au temps n : c'est l'axiome du choix (dénombrable) qui permet de passer de la première formulation à la seconde. Cela ne pose pas de problème dans la formalisation : l'axiome du choix fait partie des règles autorisées par le système, si bien qu'on peut appliquer le lemme correspondant (caché ici derrière la tactique `choose` qui est plus pratique pour l'utilisateur).

Ensuite, on extrait de cette suite une sous-suite convergente, vers une limite v, et on vérifie que celle-ci convient. On utilise pour cela le fait qu'on considère un $\mathbb{R}$-espace vectoriel de dimension finie, dans lequel la boule unité est donc compacte. Ce fait a déjà été démontré ailleurs dans la bibliothèque.

L'utilisation de l'axiome du choix et l'extraction d'une sous-suite convergente à partir d'une suite bornée sont deux processus non explicites. Leur utilisation permet de dissiper une confusion sur les assistants de preuve. L'enjeu n'est pas de représenter des objets compliqués mais finis dans la mémoire de l'ordinateur pour en explorer les propriétés par des calculs pharaoniques. Au contraire, on travaille sur des preuves, auxquelles on peut penser comme

des suites finies de symboles respectant certaines règles, mais ces preuves capturent toute la complexité des mathématiques parlant d'objets souvent infinis.

Comme on est dans un espace euclidien, l'énoncé précédent sur l'existence d'une bonne forme linéaire peut se reformuler en l'existence d'un bon vecteur. Quitte à prendre l'opposé, ce bon vecteur v vérifie $\langle v, f^i(0)\rangle \geqslant i * \texttt{h.l}$ pour tout i.

```
open scoped RealInnerProductSpace

lemma exists_asymp_vector :
  ∃ (v : E), ‖v‖ ≤ 1 ∧ ∀ (i : ℕ), (i : ℝ) * h.l ≤ ⟪v, (f^[i] 0)⟫ := by
 obtain ⟨v', v'_norm, hv'⟩ :
  ∃ (v' : StrongDual ℝ E), ‖v'‖ ≤ 1 ∧ ∀ i, v' (f^[i] 0) ≤ -i * h.l :=
   h.exists_dual
 -- (marcherait sur un espace complet, pas besoin de dimension finie ici).
 let v := (InnerProductSpace.toDual ℝ E).symm (-v')
 refine ⟨v, by simpa [v] using v'_norm, fun i ↦ ?_⟩
 simp [v]
 linarith [hv' i]
```

On peut enfin conclure la démonstration du théorème 2.2 : il existe un vecteur de translation asymptotique. On le construit comme `h.l * v` où `h.l` est la vitesse asymptotique, et v est donné par le lemme précédent. Pour montrer la convergence, notre argument informel passait par l'unicité des valeurs d'adhérence. On donne ici un argument un peu différent et un peu plus calculatoire, à base d'estimation de normes. L'intérêt de cet argument est qu'il fonctionne aussi en dimension infinie, car il n'a pas besoin de la compacité locale de l'espace.

```
/-- Une semicontraction sur un espace euclidien de dimension finie admet un
vecteur de translation asymptotique. -/
theorem exists_tendsto_div :
  ∃ (v : E), Tendsto (fun (n : ℕ) ↦ (1 / (n : ℝ)) • (f^[n] 0)) atTop (𝓝 v) := by
 obtain ⟨v₀, v₀_norm, h₀⟩ :
  ∃ (v : E), ‖v‖ ≤ 1 ∧ ∀ (i : ℕ), (i : ℝ) * h.l ≤ ⟪v, (f^[i] 0)⟫ :=
   h.exists_asymp_vector
 let v := h.l • v₀
 use v
 have A : ∀ᶠ (n : ℕ) in atTop,
    ‖(1 / (n : ℝ)) • (f^[n] 0) - v‖^2 ≤ (u n / n)^2 - h.l^2 := by
  apply eventually_atTop.2 ⟨1, fun n hn ↦ ?_⟩
  have n_ne_zero : n ≠ 0 := (zero_lt_one.trans_le hn).ne'
  calc ‖(1 / (n : ℝ)) • (f^[n] 0) - v‖ ^ 2
  _ = ‖(1 / (n : ℝ)) • (f^[n] 0)‖^2 - 2 * ⟪(1 / (n : ℝ)) • (f^[n] 0), v⟫ + ‖v‖^2 :=
   norm_sub_sq_real _ _
  _ = (u n / n)^2 - 2 * h.l / n * ⟪v₀, (f^[n] 0)⟫ + h.l^2 * ‖v₀‖^2 := by
     congr 2
     · simp [norm_smul, div_eq_inv_mul, mul_pow]
     · simp only [one_div, real_inner_smul_right, real_inner_smul_left, v]
       rw [real_inner_comm]
```

```
      ring
    · simp [norm_smul, Real.norm_eq_abs, mul_pow, v]
  _ ≤ (u n / n) ^ 2 - 2 * h.l / n * (n * h.l) + h.l ^ 2 * 1 ^ 2 := by
    gcongr
    · have := h.l_nonneg
      positivity
    · exact h₀ n
  _ = (u n / n) ^ 2 - h.l ^ 2 := by field_simp [n_ne_zero]; ring
have B : Tendsto (fun (n : ℕ) ↦ (u n / n) ^ 2 - h.l^2) atTop (𝓝 (h.l^2 - h.l^2)) :=
  (h.tendsto_lim.pow 2).sub tendsto_const_nhds
have C : Tendsto (fun (n : ℕ) ↦ ‖(1 / (n : ℝ)) • (f^[n] 0) - v‖^2) atTop (𝓝 0) := by
  rw [sub_self] at B
  apply tendsto_of_tendsto_of_tendsto_of_le_of_le' tendsto_const_nhds B _ A
  exact Eventually.of_forall (fun n ↦ by simp)
have D : Tendsto (fun (n : ℕ) ↦ ‖(1 / (n : ℝ)) • (f^[n] 0) - v‖) atTop (𝓝 0) := by
  convert C.sqrt <;> simp
exact tendsto_iff_norm_sub_tendsto_zero.2 D
```

Comme la démonstration ci-dessus est intégralement vérifiée par Lean, on pourrait penser que l'humain n'a plus de rôle à jouer, et qu'une fois qu'un résultat est formalisé c'est une vérité absolue. Cette idée est trompeuse. Il est vrai qu'on n'a plus besoin de vérifier les preuves, mais il est très important de soigneusement vérifier que les énoncés correspondent bien à ce qu'on pense qu'ils disent. Voilà par exemple une preuve triviale d'un énoncé qui, bien que ressemblant à celui du théorème 2.2, en est subtilement différent.

```
lemma wrong_exists_tendsto_div :
  ∃ (v : E), Tendsto (fun (n : ℕ) ↦ (1 / n) • (f^[n] 0)) atTop (𝓝 v) :=
⟨(0 : E), tendsto_const_nhds.congr' <|
  eventually_atTop.2 ⟨2, fun n hn ↦ by simp [Nat.div_eq_of_lt hn]⟩⟩
```

Cet énoncé affirme apparemment l'existence d'un vecteur v tel que $(1/n) \cdot f^n(0)$ converge vers v. La preuve utilise le vecteur $v = 0$, et affirme que $(1/n) \cdot f^n(0)$ est égal à 0 à partir du rang 2, et converge donc vers 0. La subtilité est que la division $1/n$ est effectuée au sein des entiers naturels ! C'est une division tronquée puisque le résultat doit être entier, si bien que $1/n = 0$ (comme entiers naturels) dès que $n \geqslant 2$. L'écriture `(1 / n) • (f^[n] 0)` ne pose pas de problème d'interprétation à Lean puisqu'on a enregistré dans la bibliothèque le fait que les entiers naturels agissent par multiplication sur tout groupe additif.

Si l'on veut que la division soit faite dans les réels, il faut le spécifier par exemple en écrivant `(1 / (n : ℝ)) • (f^[n] 0)` comme on l'a fait dans l'énoncé correct plus haut. Cette formulation applique l'injection canonique de $\mathbb{N}$ à $\mathbb{R}$ à l'entier n. Ensuite, quand on écrit `(1 / (n : ℝ))`, Lean comprend tout seul que, puisque le dénominateur est un nombre réel, alors on est en train de considérer une division dans les nombres réels, et donc le 1 du numérateur doit être le 1 des nombres réels. Finalement quand on écrit `(1 / (n : ℝ)) • (f^[n] 0)` Lean sait que le scalaire est un nombre réel, et il cherche donc dans sa bibliothèque s'il sait que les nombres réels admettent une action sur l'espace auquel $f^n(0)$ appartient. C'est bien le cas puisqu'on a déclaré plus haut comme hypothèse que E était un espace vectoriel réel (en fait un espace euclidien). Il y a donc tout un travail invisible fait par Lean pour interpréter ce

que l'utilisateur écrit, et c'est à ce niveau que des erreurs subtiles peuvent se glisser dans les énoncés.

Notons que, dans l'énoncé du théorème `exists_tendsto_div` plus haut, tout un travail invisible est aussi fait au niveau topologique, quand on parle de la convergence vers v (écrite ici avec l'expression $\mathcal{N}$ `v` qui signifie "le filtre des voisinages de v", et dont le sens précis n'aura pas d'importance dans la suite). Pour donner un sens à cette expression, Lean a besoin de savoir que E a une structure d'espace topologique. Nous avons inclus dans nos hypothèses que E était un espace euclidien. La bibliothèque sur laquelle s'appuie la formalisation contient le fait qu'un espace euclidien est un espace vectoriel normé, et donc un espace topologique. Lean fait appels à ces faits de manière implicite, sans intervention de l'utilisateur.

Cette discussion met en évidence deux enjeux importants. Il y a d'une part, un enjeu d'utilisabilité : un assistant de preuve doit être capable de comprendre ce que veut dire l'utilisateur sans que celui-ci soit obligé de tout dire explicitement, sans quoi on serait noyé sous le formalisme. Par exemple, étant donné un nombre réel a, on doit pouvoir écrire $7 * a$ sans préciser que 7 est un nombre réel et que la multiplication est entre nombres réels. Le but est de coller aux notations et à la pratique des mathématiciens autant que faire se peut, tout en éliminant complètement les ambiguïtés (ce qui est parfois antinomique, et oblige à faire des choix pragmatiques).

D'autre part, tout ce travail d'interprétation invisible repose sur des faits qui doivent déjà avoir été démontrés et mis à disposition du système. La formalisation d'un résultat comme le théorème 2.2 ne peut pas se faire de manière isolée. Ne serait-ce que pour énoncer le théorème, on a par exemple besoin de définitions comme les nombres entiers, les itérées d'une fonction, les nombres réels, les notions de topologie et de limite, les espaces vectoriels, la notion de dimension finie, les produits scalaires. Pour démontrer le théorème, on a en plus besoin d'un certain nombre de résultats, par exemple le fait que la boule unité d'un espace vectoriel réel de dimension finie est compacte, ou l'identification entre vecteurs et formes linéaires dans un espace euclidien.

Un assistant de preuve a donc besoin de bibliothèques de mathématiques formalisées avec un large spectre pour être vraiment utilisable. La formalisation que j'ai exposée repose sur la bibliothèque Mathlib, pour l'assistant de preuve Lean. C'est une bibliothèque en accès ouvert, à laquelle n'importe qui peut contribuer en proposant des ajouts, avec une croissance rapide, qui reconstruit toutes les mathématiques sans rien admettre. Par exemple, les nombres réels y sont définis comme classes d'équivalences de suites de Cauchy de nombres rationnels. Cette bibliothèque couvre actuellement à peu près toutes les mathématiques jusqu'au niveau maîtrise, voire agrégation, même si elle serait pour l'instant loin d'être suffisante pour formaliser toutes les mathématiques de niveau recherche. C'est malgré tout l'objectif de long terme qu'ont en tête les gestionnaires de la bibliothèque. Cela implique certains choix sur la manière de formaliser les choses : comme le but est de pouvoir couvrir des applications très variées, les définitions et les théorèmes se doivent d'être énoncés dans une très grande généralité. On pourrait presque qualifier cette approche de bourbakiste. Un prix à payer est que la bibliothèque n'est pas toujours d'accès facile, et n'est pas forcément adaptée à de jeunes étudiants, par exemple de classe préparatoires. Toutefois, il est possible de s'appuyer sur cette bibliothèque pour faire des formalisations plus élémentaires. Par exemple, formaliser

tout ou partie d'un TIPE en s'appuyant sur Mathlib est complètement envisageable pour des étudiants très motivés.

4. Calcul différentiel

Dans la suite, on va illustrer les considérations architecturales de Mathlib par une étude de cas sur un exemple particulier, la définition des fonctions de classe C^n. La définition actuelle est le résultat d'un processus de refonte qui est en cours depuis plusieurs années : chaque fois qu'une application ne pouvait pas être mise en œuvre, la définition a dû être modifiée.

Le calcul différentiel est un sujet fondamental et central des mathématiques, souvent abordé d'abord pour les fonctions de $\mathbb{R}$ vers $\mathbb{R}$, puis de $\mathbb{R}^k$ vers $\mathbb{R}^\ell$, et enfin pour les fonctions entre espaces vectoriels normés — où l'abstraction et la généralité supplémentaires sont nécessaires pour de nombreuses applications importantes. L'idée fondamentale du calcul différentiel est d'approximer localement une fonction par une application linéaire, sa différentielle. Souvent, cela ne suffit pas, et il faut approcher une fonction par un polynôme ou, de manière équivalente, étudier les différentielles itérés, c'est-à-dire la différentielle de la différentielle, et ainsi de suite. C'est ce qu'on appelle le calcul différentiel d'ordre supérieur.

Le calcul différentiel du premier ordre est l'étude des dérivées des applications. La définition classique donnée dans les manuels est la suivante :

Définition 4.1. *Soient E et F deux espaces vectoriels normés réels. Une application $f : E \to F$ est dérivable en un point $x \in E$ s'il existe une application linéaire continue L de E vers F telle que, lorsque y tend vers x,*

$$f(y) = f(x) + L(y - x) + o(y - x).$$

La notation $o()$ signifie ici que $\|f(y) - f(x) - L(y - x)\|/\|y - x\|$ tend vers zéro lorsque y tend vers x tout en étant différent de x.

Lorsque f est dérivable en un point x, l'application linéaire L de la définition est unique. Il s'agit de la *différentielle* (ou *dérivée*, ou *dérivée de Fréchet* si l'on veut souligner la différence avec la dimension 1) de f en x, souvent notée $Df(x)$, $D_x f$ ou $f'(x)$.

Pour les notions de différentiabilité d'ordre supérieur, le livre de Cartan [Car67, Section 5.3] est une excellente référence. Notons par $\mathcal{L}(E, F)$ l'espace des applications linéaires continues de E vers F, lorsque E et F sont des espaces vectoriels normés réels. Avec la norme d'opérateur, $\mathcal{L}(E, F)$ est également un espace vectoriel normé réel. De manière informelle, une fonction a n degrés de régularité si elle possède n dérivées successives qui sont toutes continues. On dira dans ce cas que la fonction est C^n. Plus formellement, la définition standard de la différentiabilité d'ordre supérieur est la définition par récurrence suivante.

Définition 4.2. *Soient E et F deux espaces vectoriels normés réels, et soit $f : E \to F$. On dit que f est C^0 si elle est continue. Pour un nombre naturel $n > 0$, on dit que f est C^n si elle est dérivable et si sa dérivée $Df : E \to \mathcal{L}(E, F)$ est C^{n-1}. On dit que f est C^∞ si elle est C^n pour tout $n \in \mathbb{N}$.*

Lorsque f est une fonction C^2, sa dérivée seconde $D^2 f(x)$ est bien définie, en tant qu'élément de $\mathcal{L}(E, \mathcal{L}(E, F))$. Il existe une identification canonique entre $\mathcal{L}(E, \mathcal{L}(E, F))$

et l'espace des applications bilinéaires continues de $E \times E$ vers F, de sorte que $D^2 f(x)$ est généralement plutôt considérée comme une application bilinéaire continue. De la même manière, $\mathcal{L}(E, \mathcal{L}(E, \cdots \mathcal{L}(E, F) \cdots)$ est identifié avec l'espace des applications multilinéaires continues de E^n vers F, de sorte que la dérivée itérée d'ordre n de f, notée $D^n f(x)$, est généralement considérée comme une application multilinéaire continue.

Mathlib est une bibliothèque mathématique intégrée, dont l'objectif ultime est de permettre la formalisation de l'ensemble de la recherche mathématique actuelle. La formalisation des notions dans Mathlib doit donc être conçue de manière à permettre toutes les applications importantes à venir. Une application centrale du calcul différentiel d'ordre supérieur est la théorie des variétés. Parmi les différents types de variétés rencontrés dans la littérature, mentionnons les variétés à bord (elles constituent les blocs de base du programme de Thurston visant à classifier les variétés de dimension 3, résolu par Perelman et impliquant la conjecture de Poincaré en dimension 3), les variétés sur les nombres p-adiques (particulièrement importantes pour définir les groupes de Lie p-adiques), les variétés de Banach (essentielles pour étudier les groupes de difféomorphismes des variétés réelles compactes). Pour ces applications, la définition 4.2 n'est pas suffisante. Soulignons les limites de cette définition.

(1) Elle n'est donnée que sur les nombres réels. Pour les groupes de Lie p-adiques, on a besoin d'un calcul différentiel d'ordre supérieur sur un corps arbitraire.

(2) Elle n'est écrite que dans l'espace entier (ou, par une généralisation triviale, sur les ensembles ouverts de l'espace vectoriel). Pour étudier les variétés à bord, nous avons besoin d'une notion de fonctions régulières dans un demi-espace fermé (ou dans un quadrant fermé pour les variétés avec coins). Plus généralement, nous devrions définir les fonctions C^n sur des sous-ensembles d'espaces vectoriels.

(3) Le calcul sur les corps normés autres que $\mathbb{R}$ ou $\mathbb{C}$ ne se comporte généralement bien que pour les fonctions analytiques (voir la section 8 pour plus de détails à ce sujet). C'est pourquoi Bourbaki [Bou71] étudie les variétés de classe C^n pour $n \in \mathbb{N} \cup \{\infty, \omega\}$ (où C^ω désigne les fonctions analytiques), et suppose, lorsque le corps n'est ni $\mathbb{R}$ ni $\mathbb{C}$, que $n = \omega$, c'est-à-dire que les changements de coordonnées sont analytiques. Pour obtenir une théorie unifiée des variétés sur divers corps, sans duplication de code, il est donc nécessaire d'avoir une définition unique des fonctions C^n pour $n \in \mathbb{N} \cup \{\infty, \omega\}$.

Une définition appropriée des fonctions C^n dans Mathlib devrait résoudre simultanément ces trois difficultés. Dans la suite du texte, nous allons décrire les choix qui ont été faits pour cela. Un point intéressant est que les difficultés qu'on va rencontrer, et les solutions apportées, seront de nature mathématique : la formalisation est ici juste un contexte qui met en évidence des difficultés qui existent déjà mais qu'on a tendance à ignorer dans une pratique plus traditionnelle. La suite du texte contiendra d'ailleurs très peu d'énoncés formalisés.

Remarque 4.3. Il existe une autre définition standard des fonctions C^n en termes de dérivées partielles $\partial^n f / \partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}$ avec $\alpha_1 + \cdots + \alpha_d = n$, pour $f : \mathbb{R}^d \to F$. Cette définition ne fonctionne pas en dimension infinie et repose sur le fait que l'espace source est précisément de la forme $\mathbb{R}^d$ (ce qui pourrait être évité en spécifiant plutôt une base finie de l'espace vectoriel source, ou en considérant les dérivées partielles dans toutes les directions). Ces lacunes la

rendent moins satisfaisante, tant d'un point de vue mathématique que du point de vue de la formalisation, que la définition 4.2, bien qu'elle puisse sembler séduisante à première vue car elle nécessite moins d'algèbre (notamment, aucune discussion sur les espaces de fonctions multilinéaires).

Les dérivées partielles peuvent être obtenues à partir de la dérivée itérée complète. En désignant par e_i le i-ième vecteur de base, alors par définition

$$\frac{\partial^n f(x)}{\partial x_1^{\alpha_1}\cdots\partial x_d^{\alpha_d}} = D^n f(x)(\underbrace{e_1,\dots,e_1}_{\alpha_1 \text{ termes}},\underbrace{e_2,\dots,e_2}_{\alpha_2 \text{ termes}},\dots,\underbrace{e_d,\dots,e_d}_{\alpha_d \text{ termes}}).$$

Nous n'utiliserons pas ce point de vue dans cet article, car il est plus lourd sur le plan des notations et de la combinatoire que l'approche conceptuelle de la définition 4.2, tout en ne s'appliquant qu'à des situations plus restreintes.

Remarque 4.4. Il existe une définition encore plus simple des fonctions C^n lorsque l'espace source est le corps $\mathbb{k}$ lui-même. Dans ce cas, la différentielle en un point est une application linéaire continue de $\mathbb{k}$ vers F. Il existe une identification canonique entre $\mathcal{L}(\mathbb{k}, F)$ et F, qui envoie L sur $L(1)$. Grâce à cette identification, on retrouve la notion élémentaire de dérivée unidimensionnelle d'une fonction $f : \mathbb{k} \to F$, en tant que fonction $f' : \mathbb{k} \to F$. Ce processus peut être itéré puisque les types ne changent pas, ce qui donne la n-ième dérivée unidimensionnelle $f^{(n)} : \mathbb{k} \to F$. Une fonction est C^n dans ce contexte si toutes les fonctions $f^{(m)}$ pour $m \le n$ sont bien définies et continues. Dans les approches les plus pédagogiques, on étudie d'abord soigneusement les dérivées en dimension 1 avant de passer au cas général. Dans Mathlib, au contraire, on a bien une notion de dérivée unidimensionnelle, mais elle est construite à partir de la version générale, ce qui évite de redémontrer plusieurs fois des énoncés similaires, dans un cas particulier et dans le cas général.

5. Calcul différentiel du premier ordre dans Mathlib

Les dérivées des fonctions dans Mathlib sont définies pour tout corps normé, dans des domaines. Le prédicat principal est

```
def HasFDerivWithinAt (f : E → F) (f' : E → L[𝕜] F) (s : Set E) (x : E) :=
  (fun y ↦ f y − f x − f' (y − x)) =o[𝕜; 𝓝[s] x] (fun y ↦ y − x)
```

(La lettre `F` dans `HasFDerivWithinAt` signifie *Fréchet*, afin de la distinguer de la dérivée unidimensionnelle abordée dans la remarque 4.4.) Dans cette définition, $\mathbb{k}$ est un corps normé, E et F sont deux espaces vectoriels normés sur $\mathbb{k}$, et $E \to L[\mathbb{k}] F$ désigne l'espace des applications linéaires continues de E vers F, et $\mathcal{N}[s]x$ est le filtre des voisinages de x dans l'ensemble s (nul besoin de savoir ce que cela signifie pour lire la suite !). Il s'agit de l'analogue direct de la définition 4.2, sauf que le corps des scalaires ne doit pas nécessairement être $\mathbb{R}$, et de plus cette définition s'applique à l'intérieur d'un domaine, c'est-à-dire que nous exigeons seulement que $f(y) = f(x) + f'(y-x) + o(y-x)$ lorsque y appartient à l'ensemble s. L'utilisation de domaines est importante pour l'application aux variétés à bord, mais aussi pour des énoncés plus élémentaires tels que le théorème fondamental de l'analyse, où l'on considère les dérivées à l'intérieur d'intervalles.

Nous avons ensuite le prédicat `DifferentiableWithinAt 𝕜 f s x` qui indique qu'il existe une application linéaire continue f' telle que `HasFDerivWithinAt f f' s x`. Une telle valeur

de la dérivée est enregistrée dans `fderivWithin 𝕜 f s x`. Cette fonction est totale, c'est-à-dire qu'elle est également définie lorsque la fonction n'est pas dérivable, pour faciliter son utilisation. Nous attribuons la valeur 0 à la dérivée dans ce cas par convention, et aussi parce que ce choix de valeur par défaut garantit que de nombreux lemmes restent vrais même pour des fonctions non dérivables. Voici la définition formelle :

```
def fderivWithin (𝕜) (f : E → F) (s : Set E) (x : E) : E →L[𝕜] F :=
 if HasFDerivWithinAt f (0 : E →L[𝕜] F) s x
  then 0
 else if h : DifferentiableWithinAt 𝕜 f s x
  then Classical.choose h
 else 0
```

Dans les domaines généraux (contrairement aux ensembles ouverts), la dérivée n'est pas unique, c'est pourquoi la définition recourt à `Classical.choose` pour choisir une valeur de la dérivée. Utiliser la valeur 0 comme dérivée chaque fois que c'est possible, comme ci-dessus, est pratique dans plusieurs contextes — par exemple, cela garantit que la dérivée d'une fonction constante sur un domaine est nulle, alors que sinon, la dérivée d'une fonction constante sur l'ensemble vide pourrait être arbitraire, par exemple.

Mathlib contient également des spécialisations de ces notions lorsque s est l'espace entier, appelées respectivement `HasFDerivAt f f' x`, `DifferentiableAt 𝕜 f x` et `fderiv 𝕜 f x`. Il contient également les prédicats `DifferentiableOn 𝕜 f s` indiquant que f est dérivable dans s en chacun de ses points, et `Differentiable 𝕜 f` indiquant que f est dérivable partout. Une interface complète est ensuite fournie concernant ces notions, incluant des faits standard tels que la dérivabilité de la somme ou de la composition de fonctions dérivables.

Il existe une subtilité concernant les domaines. Il est vrai que, si f et g ont respectivement des dérivées f' et g' en x dans s, alors $f+g$ a une dérivée $f'+g'$ en x dans s. De la même manière, si f et g sont dérivables en x dans s, alors $f+g$ l'est également. Cependant, même si f et g sont dérivables en x dans s, il n'est *pas* vrai en général que

```
fderivWithin 𝕜 (f + g) s x = fderivWithin 𝕜 f s x + fderivWithin 𝕜 g s x
```

en raison de l'absence d'unicité des dérivées à l'intérieur des ensembles. Par exemple, supposons que $E = \mathbb{R}^2$ et $F = \mathbb{R}$, soit L l'application linéaire envoyant (x_1, x_2) sur x_1 et M l'application linéaire envoyant (x_1, x_2) sur $x_1 + x_2$. Soit $s = \mathbb{R} \times \{0\}$. La fonction L a pour dérivée elle-même partout, et en particulier le long de s, et il en va de même pour M. Cependant, comme L et M coïncident sur s, l'application L admet également M comme dérivée le long de s. Il est donc possible que les fonctions $f = g = L$ aient une `fderivWithin` le long de s égale à L, tandis que $f + g = 2L$ pourrait avoir une `fderivWithin` le long de s égale à $2M$, ce qui viole l'égalité ci-dessus.

Il est parfois utile de s'assurer que l'égalité ci-dessus est vérifiée. Pour cela, le domaine doit être suffisamment grand pour garantir l'unicité des dérivées. C'est le cas lorsque les directions tangentes de s en x engendrent un sous-ensemble dense de l'espace entier. Nous définissons un prédicat `UniqueDiffWithinAt 𝕜 s x` enregistrant cette propriété, ainsi que `UniqueDiffOn 𝕜 s` enregistrant que s est un ensemble avec cette propriété autour de chacun de ses points. Avec ces définitions à disposition, le fait que la dérivée de la somme soit la somme des dérivées s'écrit par exemple

```
theorem fderivWithin_add {f g : E → F} {x : E} {s : Set E}
  (hxs : UniqueDiffWithinAt 𝕜 s x)
  (hf : DifferentiableWithinAt 𝕜 f s x)
  (hg : DifferentiableWithinAt 𝕜 g s x) :
  fderivWithin 𝕜 (f + g) s x = fderivWithin 𝕜 f s x + fderivWithin 𝕜 g s x
```

Pour les dérivées totales (c'est-à-dire lorsque s est l'espace entier), aucune autre condition n'est nécessaire, car l'espace entier (ou les ensembles ouverts) sont des ensembles avec unicité de la différentielle. Cela correspond à l'unicité des dérivées mentionnée après la définition 4.1. Plus généralement, les ensembles convexes dont l'intérieur n'est pas vide sont également des ensembles avec unicité de la différentielle. Cela s'applique par exemple au demi-espace ou au quadrant, ce qui s'avère important en théorie des variétés.

Le fait d'utiliser des valeurs par défaut lorsque l'objet n'est pas bien défini, comme on l'a fait ci-dessus pour la dérivée, est extrêmement pratique même si cela heurte parfois notre sensibilité mathématique. Par exemple, la division des nombres réels dans Lean n'est pas tout à fait la même que la division habituelle. En général, on déclare que a/b n'est pas défini lorsque $b = 0$, et que la division est donc une fonction de $\mathbb{R} \times \mathbb{R}^*$ dans $\mathbb{R}$. En pratique, toutefois, quand on écrit les choses on ne justifie jamais que le dénominateur est non nul, et on laisse ce soin au lecteur. On n'a pas cette latitude dans une formalisation. Il y a donc deux options : soit utiliser une opération de division qui prend trois arguments, a, b, et une preuve du fait que b est non nul ; soit autoriser $b = 0$, et renvoyer alors une valeur par défaut (en général 0). La première branche de l'alternative correspond à ce qu'on fait en théorie, mais pas à ce qu'on fait en pratique. Son utilisation en formalisation est extrêmement pénible. Avec l'expérience, tous les assistants de preuve modernes ont fait le choix de la deuxième branche de l'alternative, dans laquelle la division est une fonction totale. Cela ne crée bien sûr pas de paradoxes, puisque l'opération qu'on utilise est une opération bien définie même si ce n'est pas exactement la fonction de division classique. L'hypothèse de non-nullité du dénominateur n'apparaît alors plus dans la définition, mais dans certains énoncés. Par exemple, le lemme affirmant que $x/x = 1$ ne s'applique que lorsque x est non nul. Certains énoncés, comme le fait que $(a + b)/c = a/c + b/c$, n'ont en revanche pas besoin de cette hypothèse : lorsque $c = 0$, cette égalité se réduit en $0 = 0 + 0$, qui est bien vraie. Cela permet à l'utilisateur de ce lemme de l'appliquer sans avoir besoin de se donner du mal pour démontrer que c est non nul, ce qui est autant de temps de gagné.

Revenons aux valeurs par défaut pour la dérivée. Lorsque la fonction n'est pas dérivable, on a décrété que la dérivée serait nulle – cela nous permet d'écrire la dérivée d'une fonction sans avoir à démontrer à chaque fois que la fonction est dérivable, ce qui est beaucoup plus léger que l'alternative. Cela permet des énoncés pour le moins surprenants – voilà encore un exemple où il faut faire attention à ce qu'un théorème formalisé dit vraiment. Voilà par exemple une des version du théorème de Rolle dans Mathlib :

```
theorem exists_deriv_eq_zero' (hab : a < b) (hfa : Tendsto f (𝓝[>] a) (𝓝 l))
  (hfb : Tendsto f (𝓝[<] b) (𝓝 l)) : ∃ c ∈ Ioo a b, deriv f c = 0 := by
```

Cet énoncé se lit comme suit : étant donnés $a < b$ et une fonction $f : \mathbb{R} \to \mathbb{R}$ qui tend vers une même limite l à droite de a et à gauche de b, alors il existe $c \in]a, b[$ avec `deriv f c` $= 0$.

Présenté ainsi, cet énoncé a l'air évidemment faux puisqu'il y manque l'hypothèse de différentiabilité à la base du théorème de Rolle. Cependant, il est bien juste, et validé par Lean. La démonstration est la suivante : s'il existe un point de $]a, b[$ en lequel f n'est pas dérivable, on prend pour c ce point, et on y a bien `deriv f c = 0` grâce à notre choix de valeur par défaut. Sinon, f est dérivable sur l'intervalle, et on est ramené à la formulation habituelle du théorème de Rolle.

Cette version ne prétend bien sûr pas avoir un intérêt mathématique profond : à chaque fois qu'on va utiliser le théorème de Rolle pour démontrer des choses intéressantes, ce sera pour des fonctions dérivables. L'intérêt de ce type de formulation est que retirer des hypothèses rend le théorème plus pratique à appliquer pour l'utilisateur : même si sa fonction est évidemment dérivable, cela fait du travail en moins s'il n'a pas à le justifier auprès du système.

6. Fonctions analytiques

Une fonction unidimensionnelle f est analytique (ou développable en série entière) autour de 0 si elle peut s'écrire sous la forme d'une série convergente $f(z) = \sum_n c_n z^n$. Cela se généralise facilement : une fonction $f : \mathbb{R}^d \to \mathbb{R}$ est analytique en 0 si elle peut s'écrire autour de 0 sous la forme

$$f(x_1, \dots, x_d) = \sum_\alpha c_\alpha x_1^{\alpha_1} \cdots x_d^{\alpha_d},$$

où la somme porte sur tous les multi-indices $\alpha = (\alpha_1, \dots, \alpha_d)$ et doit converger en norme dans un voisinage de 0 (de manière équivalente, les coefficients c_α doivent satisfaire une borne $|c_\alpha| \le CR^{\alpha_1 + \cdots + \alpha_d}$). Cependant, cette définition est restreinte à la dimension finie. La définition générale correcte est la suivante (voir par exemple [Muj86]) :

Définition 6.1. *Soient E et F deux espaces vectoriels normés sur un corps normé $\mathbb{k}$. Une fonction $f : E \to F$ est analytique en x s'il existe une suite $(p_n)_{n\in\mathbb{N}}$, où $p_n : E^n \to F$ est n-multilinéaire et satisfait $\|p_n\| \leqslant CR^n$ pour certains $C > 0$ et $R < \infty$, telle que pour tout y suffisamment proche de 0, on a*

$$f(x + y) = \sum_n p_n(y, \dots, y). \tag{6.1}$$

En une dimension, $p_n(y^{(1)}, \dots, y^{(n)})$ est simplement $c_n y^{(1)} \cdots y^{(n)}$. Dans le cas de la dimension finie, le terme $p_n(y, \dots, y)$ est la somme des termes $c_\alpha y_1^{\alpha_1} \cdots y_d^{\alpha_d}$ pour tous les multi-indices tels que $\alpha_1 + \cdots + \alpha_d = n$.

Pour formaliser cette définition, Mathlib définit d'abord le type `E [×n]→L[𝕜] F` des applications n-multilinéaires continues de E^n vers F, doté de sa structure canonique d'espace normé sur $\mathbb{k}$ (où E et F sont des espaces vectoriels normés sur $\mathbb{k}$). Soit `FormalMultilinearSeries 𝕜 E F` l'espace des suites $(p_n)_{n\in\mathbb{N}}$ avec p_n une application n-multilinéaire continue. On introduit une structure indiquant qu'une fonction admet un développement en série entière comme dans la définition 6.1, sur une boule de rayon r intersectée avec un ensemble s, car nous voulons des définitions dans des domaines :

```
structure HasFPowerSeriesWithinOnBall (f : E → F)
  (p : FormalMultilinearSeries 𝕜 E F) (s : Set E)
  (x : E) (r : ℝ≥0∞) : Prop where
```

```
r_le : r ⩽ p.radius
r_pos : 0 < r
hasSum : ∀ y, x + y ∈ insert x s → y ∈ ball (0 : E) r
  → HasSum (fun n ↦ p n (fun _ : Fin n ↦ y)) (f (x + y))
```

Dans cette définition, $\mathbb{R}\geqslant 0\infty$ est le type des réels non négatifs étendus, c'est-à-dire $[0, +\infty]$, et `p.radius` est le rayon sur lequel la série entière définie par p converge. La condition `hasSum` exige la condition (6.1) pour $x + y \in s \cup \{x\}$ plutôt que simplement s, car sinon la théorie est très mauvaise, et dans les applications intéressantes x appartient de toute façon à s.

Une fois ce prédicat disponible, on dit que f est analytique dans s en x s'il existe p et $r > 0$ tels que f possède une série entière donnée par p sur la boule de rayon r. Formellement,

```
def AnalyticWithinAt (𝕜) (f : E → F) (s : Set E) (x : E) :=
  ∃ p : FormalMultilinearSeries 𝕜 E F, ∃ r, HasFPowerSeriesWithinOnBall f p s x r
```

Ce prédicat est similaire à `DifferentiableWithinAt 𝕜 f s x`, bien qu'il exige une propriété de régularité bien plus forte. Parallèlement au cas de la différentiabilité, on introduit d'autres prédicats : `AnalyticOn`, qui indique qu'une fonction est analytique dans s en chacun de ses points, et `AnalyticAt`, qui indique qu'une fonction est analytique en un point (ce qui correspond à s étant l'espace entier). Mathlib fournit également une interface complète pour ces notions.

La suite p_n n'est pas déterminée de manière unique par f : toute suite p'_n telle que $p_n(y, \dots, y) = p'_n(y, \dots, y)$ convient également, à condition que sa norme ne croisse pas de manière superexponentielle.

Exemple 6.2. Soit $\mathbb{k}$ un corps normé et soit $f : \mathbb{k}^2 \to \mathbb{k}$ définie par $f(x_1, x_2) = x_1 x_2$. Soit $p_n = 0$ pour $n \neq 2$ et $p_2((x_1, x_2), (x'_1, x'_2)) = x_1 x'_2$. Alors $f(y) = \sum_n p_n(y, \dots, y)$, ce qui montre que f est analytique autour de 0. La suite p'_n donnée par $p'_n = 0$ pour $n \neq 2$ et $p'_2((x_1, x_2), (x'_1, x'_2)) = x'_1 x_2$ satisfait également $f(y) = \sum_n p'_n(y, \dots, y)$.

Si l'on souhaitait un choix canonique pour p_n, on pourrait essayer d'imposer une symétrie : dans cet exemple, on poserait

$$p_2((x_1, x_2), (x'_1, x'_2)) = (x_1 x'_2 + x'_1 x_2)/2.$$

Cependant, cela n'est pas possible si $\mathbb{k}$ est de caractéristique 2, par exemple $\mathbb{k} = \mathbb{F}_2((t))$. Dans ce cas, il n'y a pas de représentant symétrique, contrairement à la caractéristique 0 où un tel choix canonique est toujours possible.

Si f est analytique autour d'un point x, et si y est suffisamment petit, alors f est également analytique autour de $x + y$ si l'espace cible est complet. En particulier, l'ensemble des points d'analyticité de f est ouvert. Cela découle du calcul suivant :

$$\begin{aligned} f(x + y + z) &= \sum_n p_n(y + z, \dots, y + z) \\ &= \sum_n \sum_{I \subseteq \{0, \dots, n-1\}} \sum p_n(w_1^I, \dots, w_n^I) \end{aligned}$$

par multilinéarité, où w_i^I est égal à z si $i \in I$, et à y sinon. En ne faisant varier que les coordonnées à l'intérieur de I, on peut considérer cette dernière expression comme une application $\#I$-multilinéaire. En regroupant les termes ayant le même nombre de variables,

on obtient $f(x+y+z) = \sum q_k(z,\dots,z)$ où q_k est k-multilinéaire. Pour en faire une preuve rigoureuse, il faut vérifier la convergence de la série définissant q_k, qui découle des estimations de norme et de la complétude de l'espace cible.

La principale difficulté ici consiste à formuler les choses avec précision, comme c'est souvent le cas lorsque la démonstration sur papier comporte de nombreux points de suspension. Les estimées sur les normes sont alors directes, bien que fastidieuses.

Si f est analytique en x (donnée par la série des p_n), alors f est également dérivable en x, avec une dérivée égale à p_1 (modulo l'identification entre les applications continues 1-multilinéaires et les applications linéaires continues). Cela découle de l'expression en série et du fait que, pour $n \geq 2$, alors $p_n(y,\dots,y) = O(\|y\|^n) = o(\|y\|)$ (conjointement avec les contrôles de norme uniformes). Ceci est formalisé comme suit :

```
theorem HasFPowerSeriesAt.hasFDerivAt (h : HasFPowerSeriesAt f p x) :
  HasFDerivAt f (continuousMultilinearCurryFin1 𝕜 E F (p 1)) x

theorem AnalyticAt.differentiableAt (h : AnalyticAt 𝕜 f x) :
  DifferentiableAt 𝕜 f x
```

Les versions dans un ensemble sont également fournies dans la bibliothèque.

Exemple 6.3. Voici un exemple montrant que la complétude de l'espace cible est nécessaire pour que l'ensemble des points d'analyticité soit ouvert. Soit $\ell^\infty(\mathbb{R})$ l'espace des suites réelles bornées, muni de la norme du sup. Définissons $f : \mathbb{R} \to \ell^\infty(\mathbb{R})$ comme suit : pour $|x| < 1$, posons $f(x) = (1, x, x^2, \dots)$. Sinon, posons $f(x) = 0$.

La fonction f est analytique en 0. En effet, pour $|x| < 1$, on a $f(x) = \sum c_n x^n$, où $c_n \in \ell^\infty(\mathbb{R})$ est la suite qui prend la valeur 1 à l'indice n et zéro partout ailleurs. Comme $\ell^\infty(\mathbb{R})$ est complet, il s'ensuit que f est analytique en tout x tel que $|x| < 1$, et donc dérivable. Sa dérivée est $\sum n c_n x^{n-1}$.

Soit F le sous-espace vectoriel de $\ell^\infty(\mathbb{R})$ engendré par les c_n et les valeurs de $f(x)$ pour $x \in \mathbb{R}$. Cet espace n'est plus complet. Comme f prend ses valeurs dans F, on peut définir une nouvelle fonction $g : \mathbb{R} \to F$ égale à f. Le développement $g(x) = \sum c_n x^n$ est toujours vrai dans F, donc g est analytique en 0. Cependant, nous affirmons que g n'est analytique en aucun autre point x tel que $|x| < 1$, ce qui montre que l'ensemble des points d'analyticité de g n'est pas ouvert.

Supposons par l'absurde que g soit analytique en x. Alors g est dérivable en x. Par conséquent, la dérivée $\sum_n n c_n x^{n-1}$ appartient à F. Par définition de F, c'est la somme d'une suite à support fini et d'une somme finie $\sum_i a_i(1, y_i, y_i^2, \dots)$. En considérant le n-ième coefficient de la suite, on obtient $nx^{n-1} = \sum_i a_i y_i^n$ pour n grand. En considérant les valeurs paires de n, on peut même supposer que tous les y_i sont positifs ou nuls, en remplaçant y_i par $-y_i$ si nécessaire. Alors le taux de croissance du terme de droite est cz^n, où z est le plus grand des y_i ayant un coefficient non nul. C'est une contradiction, car il est censé croître comme nx^{n-1}.

Dans cet exemple, la fonction g n'est dérivable qu'en 0. Sa dérivée en ce point est $x \mapsto c_1 x$. En d'autres points, elle n'est pas dérivable. Selon notre convention pour la dérivée, celle-ci sera alors nulle en ces autres points. Cela implique que la dérivée de g n'est pas continue en 0, bien que g y soit analytique !

Parmi les résultats classiques sur les fonctions analytiques formalisés dans Mathlib, on a que la composition de fonctions analytiques est analytique, et que l'inverse local d'une fonction analytique est analytique, c'est-à-dire la version analytique du théorème d'inversion locale. Définir une suite q_n pour l'inverse d'une application f correspondant à une suite p_n est facile d'un point de vue formel, mais vérifier que la croissance de $\|q_n\|$ n'est pas plus rapide que n'importe quelle exponentielle nécessite un argument supplémentaire. Les livres classiques utilisent l'analyse complexe et la formule intégrale de Cauchy pour contrôler cette croissance. Cependant, le développement des fonctions analytiques dans Mathlib précède l'analyse complexe (qui utilise beaucoup de matériel sur les fonctions analytiques). L'utiliser pour la preuve du théorème de la fonction inverse analytique nécessiterait donc de nombreux croisements entre les deux théories, ce qui nuirait à la cohérence interne de la bibliothèque. À la place, Mathlib utilise une preuve directe, de nature plus combinatoire et s'appuyant sur une estimation inductive astucieuse provenant de [Pös21].

Toutes ces constructions s'appuient fortement sur les implémentations des sommes finies et des sommes infinies dans Mathlib, ainsi que sur des résultats concernant la complétude. Une bibliothèque cohérente de mathématiques formalisées est comme une cathédrale, dans laquelle chaque pierre repose sur d'autres pierres et dans laquelle il est impossible de poser la flèche si l'on n'a pas, patiemment, construit toutes les fondations sur lesquelles elle s'appuiera.

7. Fonctions C^n

7.1. **La définition formelle.** Dans ce paragraphe, nous donnons la définition des fonctions C^n dans Mathlib. Comme cette définition est plus compliquée qu'on pourrait le croire à première vue, nous commencerons par évoquer plusieurs tentatives infructueuses afin d'illustrer les difficultés. Nous ne parlerons dans un premier temps que des fonctions C^n pour $n \leq \infty$, puis nous ajouterons les fonctions analytiques par la suite.

Tentative 7.1. Dans un premier temps, nous pourrions suivre à la lettre la définition 4.2 : définir par récurrence la dérivée itérée d'ordre n de f sur l'ensemble s comme la dérivée dans s de la dérivée itérée d'ordre $(n-1)$, et dire que f est C^n dans s si toutes ces dérivées itérées sont bien définies et continues jusqu'au rang n.

La non-unicité des dérivées dans les domaines implique que cette stratégie échoue : avec cette définition, on ne peut pas prouver que les applications linéaires continues sont C^n sur les domaines, alors que cela devrait être vrai pour toute définition raisonnable.

Par exemple, considérons la fonction $f : (x_1, x_2) \mapsto x_1$ définie sur $\mathbb{R}^2$, et le domaine $s = \mathbb{R} \times \{0\}$. Tout application linéaire ℓ_α de la forme $(u, v) \mapsto u + \alpha v$ est une dérivée de f le long de s, car seule la direction horizontale importe. Il s'ensuit que `fderivWithin ℝ f s x`, telle que définie ci-dessus en choisissant une dérivée arbitraire qui convient, pourrait être n'importe quelle fonction $\ell_{\alpha(x)}$. Si α n'est pas continue sur s, alors f ne serait même pas C^1 sur s.

Cette tentative infructueuse montre qu'une définition raisonnable ne doit pas s'appuyer sur la dérivée spécifique `fderivWithin ℝ f s x` (qui pourrait présenter un comportement indésirable), mais plutôt sur l'existence d'une suite appropriée de dérivées successives. En utilisant le type `FormalMultilinearSeries 𝕜 E F` des suites p_n d'applications n-multilinéaires, on définit une assertion `HasFTaylorSeriesUpToOn n f p s` qui enregistre le fait que p_0 coïncide avec f sur s, que p_{m+1} y est une dérivée de p_m pour tout $m < n$, et que p_n est continue. Cela

permettrait de dire qu'une fonction est C^n si elle admet une série de Taylor d'ordre n (sauf qu'en réalité, nous devrons à nouveau adapter cette définition).

La définition formalisée est la suivante :

```
structure HasFTaylorSeriesUpToOn
  (n : WithTop ℕ∞) (f : E → F)
  (p : E → FormalMultilinearSeries 𝕜 E F) (s : Set E) : Prop where
 zero_eq : ∀ x ∈ s, (p x 0).curry0 = f x
 fderivWithin : ∀ m : ℕ, m < n → ∀ x ∈ s,
  HasFDerivWithinAt (p · m) (p x m.succ).curryLeft s x
 cont : ∀ m : ℕ, m ≤ n → ContinuousOn (p · m) s
```

Dans cette définition, `WithTop ℕ∞` désigne l'ensemble $\mathbb{N}$ auquel ont été ajoutés les deux éléments ∞ et ω. Nous incluons ces deux éléments supplémentaires afin d'anticiper la définition de C^n pour $n \in \mathbb{N} \cup \{\infty, \omega\}$. Le `F` dans le nom signifie *Fréchet*, pour souligner qu'il ne s'agit pas d'une notion unidimensionnelle. La propriété `zero_eq` exige essentiellement que f et p_0 coïncident sur s. Toutefois, $f(x)$ appartient à F tandis que $p_0(x)$ est une fonction de E^0 vers F, de sorte qu'elles ne peuvent pas coïncider formellement : pour obtenir un énoncé sensé, il faut ajouter l'identification canonique entre F et $E^0 \to F$, une identification invisible que l'on passe en général sous silence quand on fait des mathématiques non formalisées (comme l'inclusion de $\mathbb{N}$ dans $\mathbb{R}$ qu'on a croisée page 13). De la même manière, la propriété `fderivWithin` exige essentiellement que $p_{m+1}(x)$ soit une dérivée de p_m en x, sauf que p_{m+1} est une application multilinéaire $E^{m+1} \to F$ tandis que la dérivée de p_m est une application linéaire de E vers les applications multilinéaires de E^m vers F. Encore une fois, ces deux ensembles de fonctions sont identifiés de manière canonique par une fonction notée `curryLeft` dans la définition. Enfin, nous devrions également exiger la continuité de p_n, sauf que cela n'a pas de sens pour $n = \infty$ ou ω. Par conséquent, `cont` exige plutôt la continuité de tous les p_m pour $m \leq n$, bien que la nouvelle information ne concerne que le dernier terme s'il y en a un : grâce à `fderivWithin`, nous savons déjà que les autres termes sont dérivables, et donc continus.

Remarque 7.2. Les séries multilinéaires formelles, c'est-à-dire les suites $(p_n)_{n\in\mathbb{N}}$ d'applications n-multilinéaires continues, apparaissent à la fois dans la définition des fonctions analytiques sous la forme $f(x+y) = \sum p_n(y,\dots,y)$ dans la définition 6.1, et dans les suites de dérivées itérées des séries de Taylor. Ces deux utilisations du même objet formel ne correspondent *pas* l'une à l'autre. Par exemple, considérons la fonction réelle $f(x) = 1/(1-x)$ autour de 0. Elle est analytique en 0, avec le développement $f(x) = \sum x^n$, donc $p_n(y,\dots,y) = y^n$. La n-ième dérivée de f en 0, cependant, est $D^n f(0)(v_1,\dots,v_n) = n! v_1 \cdots v_n$ (ce qui correspond à l'énoncé en termes de dérivées unidimensionnelles selon lequel $f^{(n)}(0) = n!$). Il existe un écart de $n!$ entre le point de vue des fonctions analytiques et celui des dérivées. Cela n'est pas surprenant compte tenu du développement standard d'une fonction analytique réelle unidimensionnelle autour de 0,

$$f(y) = \sum \frac{f^{(n)}(0)}{n!} y^n.$$

Le coefficient $n!$ a ici une jolie interprétation combinatoire en tant que cardinal du groupe des permutations de $\{1,\dots,n\}$, voir (8.2).

Tentative 7.3. Dans un deuxième temps, on pourrait dire qu'une fonction est C^n (pour $n \in \mathbb{N} \cup \{\infty\}$) sur un ensemble s s'il existe une suite (p_m) qui est une série de Taylor pour f jusqu'à l'ordre n dans s au sens indiqué ci-dessus.

Le problème avec cette définition est la localité, tant en espace qu'en régularité. Une bonne notion de fonction C^n devrait satisfaire ce qui suit : supposons que, pour tout $x \in s$, il existe un voisinage u de x dans s sur lequel f est C^n. Alors f est C^n dans s. De même, si une fonction est C^n dans s pour tout entier naturel n, alors elle devrait être C^∞ dans s. La définition ci-dessus ne satisfait a priori pas ces deux propriétés, comme nous allons l'expliquer maintenant.

Si f est C^n dans le sens précédent sur un voisinage u de x dans s, on obtient une suite correspondante d'applications multilinéaires $p_m^{(u)}$, indexées par u, qui donne une série de Taylor pour f sur u. Si celles-ci étaient compatibles au sens où $p_m^{(u)} = p_m^{(v)}$ sur $u \cap v$, on pourrait les coller ensemble pour f sur l'ensemble s entier. Cependant, en raison de l'absence d'unicité des dérivées, il n'y a aucune garantie de compatibilité. Dans un espace vectoriel réel de dimension finie, on pourrait plutôt extraire un recouvrement localement fini des u, puis utiliser une partition lisse de l'unité ρ_u subordonnée à ce recouvrement localement fini, et former la somme (localement finie) $\sum_u \rho_u p_m^{(u)}$. Cela constituerait une série de Taylor pour f sur l'ensemble entier s. Cependant, l'existence de partitions lisses de l'unité n'est pas toujours vérifiée en dimension infinie, ni sur d'autres corps. Par exemple, il n'existe pas de partition de l'unité différentiable au sens complexe sur $\mathbb{C}$, car une fonction différentiable au sens complexe à support compact est uniformément nulle (puisqu'elle est analytique).

Une solution à ce problème consiste à intégrer la notion de localité à la définition : nous dirons que f est de classe C^n en s au point x (pour $n \in \mathbb{N} \cup \{\infty\}$) si, pour tout nombre naturel $n' \leq n$, il existe un voisinage u de x dans s et une série multilinéaire formelle (p_m) telle que f admette (p_m) comme série de Taylor en x sur u jusqu'à l'ordre n'. Nous dirons que f est C^n dans s si elle est y C^n en chacun de ses points. Par construction, cette définition est locale tant en espace qu'en régularité.

Intégrons maintenant les fonctions analytiques dans notre définition des fonctions C^n.

Tentative 7.4. Dans une troisième tentative, disons que f est C^n dans s en x (pour $n \in \mathbb{N} \cup \{\infty\}$) en utilisant la définition locale ci-dessus, et qu'elle est C^ω dans s en x si elle est analytique dans s en x.

Avec cette définition, cependant, il n'est pas vrai que si f est C^n et $m \leq n$, alors f est C^m, alors que c'est une propriété que l'on attend clairement d'une échelle de fonctions régulières. Ceci est dû au mauvais comportement des fonctions analytiques sur des espaces non complets. Par exemple, la fonction g de l'exemple 6.3 est analytique en 0 mais elle n'est pas C^1 en 0, comme expliqué à la fin de l'exemple.

Une solution consisterait à exiger que tous nos espaces cibles soient complets, comme c'est par exemple le cas dans [Bou71]. Cependant, cette approche est inutilement restrictive : la majeure partie de la théorie peut être construite sans recourir à la complétude, et celle-ci ne devrait être introduite comme hypothèse que dans les théorèmes qui en ont réellement besoin. Au lieu de cela, la stratégie dans Mathlib consiste à exiger que les fonctions C^ω soient analytiques, ainsi que toutes leurs dérivées. De cette manière, nous récupérons la monotonie de l'échelle des fonctions régulières jusqu'à ω.

Nous pouvons maintenant donner la définition formalisée des fonctions C^n utilisées dans Mathlib. Nous introduisons un prédicat `ContDiffWithinAt 𝕜 n f s x` indiquant que f est C^n (sur le corps $\mathbb{k}$) dans s en x, défini comme suit :

```
def ContDiffWithinAt (𝕜) (n : WithTop ℕ∞) (f : E → F) (s : Set E) (x : E) : Prop :=
 match n with
 | (n : ℕ∞) => ∀ m : ℕ, m ≤ n → ∃ u ∈ 𝓝[insert x s] x,
    ∃ p : E → FormalMultilinearSeries 𝕜 E F, HasFTaylorSeriesUpToOn m f p u
 | ω => ∃ u ∈ 𝓝[insert x s] x,
    ∃ p : E → FormalMultilinearSeries 𝕜 E F,
    HasFTaylorSeriesUpToOn ω f p u ∧
    ∀ i, AnalyticOn 𝕜 (fun x ↦ p x i) u
```

Si l'exposant de régularité n appartient à $\mathbb{N} \cup \{\infty\}$, alors pour tout entier $m \leq n$, nous exigeons l'existence d'un voisinage u de x dans $s \cup \{x\}$ sur lequel f admet une série de Taylor jusqu'au rang m. Cela correspond à la discussion ci-dessus, sauf que nous utilisons $s \cup \{x\}$ au lieu de s : cela évite les pathologies où la série de Taylor serait discontinue en x. Pour $n = \omega$, nous exigeons l'existence d'un voisinage u de x dans s sur lequel f admet une série de Taylor d'ordre infini, composée de fonctions analytiques.

Une fois ce prédicat établi, nous définissons une version non localisée à un ensemble, appelée `ContDiffAt` (obtenue en prenant pour s l'espace entier dans la définition ci-dessus), une version exigeant que f soit C^n dans s en chacun de ses points, appelée `ContDiffOn`, et une version dans l'espace entier appelée `ContDiff`. Ce sont les analogues parfaits des définitions correspondantes pour les fonctions différentiables et pour les fonctions analytiques : une telle cohérence est très utile aux utilisateurs de la bibliothèque. Les énoncés pour ces trois notions suivent également un schéma de nommage totalement identique.

Énonçons maintenant quelques propositions élémentaires démontrant que la définition ci-dessus tient ses promesses.

L'échelle des fonctions C^n est monotone par rapport à n :

```
theorem ContDiffWithinAt.of_le (h : ContDiffWithinAt 𝕜 n f s x) (hmn : m ≤ n) :
   ContDiffWithinAt 𝕜 m f s x
```

En particulier, une fonction C^ω est C^∞.

Dans un espace complet, C^ω et analytique coïncident :

```
theorem contDiffWithinAt_omega_iff_analyticWithinAt [CompleteSpace F] :
   ContDiffWithinAt 𝕜 ω f s x ↔ AnalyticWithinAt 𝕜 f s x
```

Pour $n \neq \infty$, une fonction est C^n dans un ensemble en un point si et seulement si il existe un voisinage sur lequel elle possède une série de Taylor d'ordre n, et de plus, si $n = \omega$, tous les termes de cette série de Taylor doivent être analytiques :

```
lemma contDiffWithinAt_iff_of_ne_infty (hn : n ≠ ∞) :
   ContDiffWithinAt 𝕜 n f s x ↔ ∃ u ∈ 𝓝[insert x s] x,
    ∃ p : E → FormalMultilinearSeries 𝕜 E F, HasFTaylorSeriesUpToOn n f p u ∧
      (n = ω → ∀ i, AnalyticOn 𝕜 (fun x ↦ p x i) u)
```

Ce lemme n'est pas vrai pour $n = \infty$: considérons $f : \mathbb{R} \to \mathbb{R}$ qui est C^n et pas mieux sur $(-1/n, 1/n)$. Elle est alors C^∞ en 0, mais pas C^∞ dans un voisinage de 0.

Pour $n \neq \infty$, une fonction est C^{n+1} dans un domaine si et seulement si, localement, elle possède une dérivée qui est C^n (et de plus, la fonction est analytique lorsque $n = \omega$).

```
theorem contDiffWithinAt_succ_iff_hasFDerivWithinAt
  (hn : n ≠ ∞) :
  ContDiffWithinAt 𝕜 (n + 1) f s x ↔ ∃ u ∈ 𝓝[insert x s] x,
    (n = ω → AnalyticOn 𝕜 f u) ∧ ∃ f' : E → E →L[𝕜] F,
    (∀ x ∈ u, HasFDerivWithinAt f (f' x) u x) ∧ ContDiffWithinAt 𝕜 n f' u x
```

7.2. **Les opérations sur les fonctions préservent la régularité.** Il y a une tension entre notre définition et les démonstrations classiques des propriétés des fonctions C^n. Voici la démonstration classique du fait que la somme de deux fonctions C^n est C^n, se basant sur la définition 4.2.

Proposition 7.5. *Soient E, F deux espaces vectoriels normés réels et $f, g : E \to F$ deux fonctions C^n pour $n \in \mathbb{N}$. Alors $f + g$ est C^n.*

Démonstration. Nous raisonnons par récurrence sur n. Pour $n = 0$, le résultat est trivial. Pour $n > 0$, d'après la définition 4.2, nous devons vérifier que $f + g$ est dérivable, ce qui est trivial, et que $D(f + g)$ est de classe C^{n-1}. Comme $D(f + g) = Df + Dg$, cela découle de l'hypothèse de récurrence appliquée aux fonctions Df et Dg de E vers $\mathcal{L}(E, F)$. □

Notre définition des fonctions C^n dans Mathlib n'est pas récursive, donc la preuve ci-dessus ne peut pas être adaptée directement. La preuve formalisée ne devrait probablement pas suivre la preuve inductive classique ci-dessus, mais plutôt s'appuyer davantage sur le choix de définition que nous avons fait. Voici une autre preuve sur papier de la proposition 7.5, mais plus proche de notre définition formelle.

Une autre démonstration de la proposition 7.5. Comme f et g sont des fonctions de classe C^n, elles admettent respectivement des séries de Taylor p et q jusqu'au rang n. On vérifie alors aisément que $p + q$ est une série de Taylor jusqu'au rang n pour $f + g$, ce qui prouve que $f + g$ est une fonction de classe C^n. □

Cette preuve ne présente aucun caractère inductif, ce qui signifie qu'il est facile de la formaliser pour obtenir ce qui suit :

```
theorem ContDiffWithinAt.add
  (hf : ContDiffWithinAt 𝕜 n f s x) (hg : ContDiffWithinAt 𝕜 n g s x) :
  ContDiffWithinAt 𝕜 n (fun x ↦ f x + g x) s x
```

Une leçon à retenir ici est qu'au lieu de prouver par récurrence que les fonctions sont C^n, il est plus efficace, avec notre définition, de *donner une formule* pour les dérivées successives, en tant qu'opération sur la série de Taylor des fonctions originales. Pour la somme de fonctions, c'est très simple (la série de Taylor de la somme est la somme des séries de Taylor), mais cela pose une difficulté pour la composition.

Théorème 7.6. *Soient E, F, G des espaces vectoriels normés réels. Considérons $f : E \to F$ et $g : F \to G$, deux fonctions de classe C^n. Alors la composition $g \circ f$ est de classe C^n.*

Voici la preuve par récurrence classique qu'on trouve dans les livres.

Démonstration. L'énoncé est facile à vérifier pour $n = 0$. Pour $n > 0$, la fonction $g \circ f$ est dérivable en tant que composition de fonctions dérivables, et sa dérivée est donnée par $D(g \circ f)(x)(v) = Dg(f(x))(Df(x)(v))$. Nous devons montrer qu'il s'agit d'une fonction C^{n-1} de x.

Soit $M : \mathcal{L}(E, F) \times \mathcal{L}(F, G) \to \mathcal{L}(E, G)$ la composition des applications linéaires. Il s'agit d'une application bilinéaire continue, donc C^∞. Soit $\Phi : E \to \mathcal{L}(E, F) \times \mathcal{L}(F, G)$ l'application $(Df, (Dg) \circ f)$. Comme f est C^n, alors Df est C^{n-1}. De plus, $(Dg) \circ f$ est C^{n-1} d'après notre hypothèse de récurrence, en tant que composition de deux fonctions C^{n-1}. Il s'ensuit que Φ est C^{n-1}. Par construction, $D(g \circ f) = M \circ \Phi$. C'est la composition de deux fonctions C^{n-1}, donc C^{n-1} à nouveau d'après notre hypothèse de récurrence. □

Cette démonstration ne permet pas de déterminer d'emblée comment donner une formule pour la dérivée de $g \circ f$, et cette formule serait probablement très compliquée. Par exemple, la dérivée seconde est

$$\begin{aligned} D^2(g \circ f)(x)(v_0, v_1) = {} & D^2 g(f(x))(Df(x)v_0, Df(x)v_1) \\ & + Dg(f(x))(D^2 f(x)(v_0, v_1)). \end{aligned}$$

Dans une implémentation précédente de Mathlib des fonctions C^n, dans laquelle les fonctions analytiques n'étaient pas encore intégrées dans la hiérarchie lisse, le fait que la composition respecte la classe des fonctions C^n était prouvé en suivant l'argument inductif ci-dessus. Cela permettait d'éviter la formule compliquée pour la dérivée d'ordre n d'une composition.

Dans la refonte visant à intégrer les fonctions analytiques dans la hiérarchie des fonctions régulières, la régularité de la composition des fonctions a constitué le point le plus difficile : on a dû changer d'approche dans les démonstrations, car l'argument récursif n'a pas de sens pour les fonctions C^ω. À la place, Mathlib contient et utilise maintenant la formule pour la série de Taylor d'une composition, appelée formule de Faà di Bruno. Cette formule est généralement donnée pour les fonctions en une variable, ou en utilisant les dérivées partielles pour les fonctions à un nombre fini de variables, mais il existe une formule générale en termes de dérivées de Fréchet que nous allons maintenant décrire. Elle s'exprime en termes de partitions I de $\{0, ..., n-1\}$. Pour une telle partition, notons k son nombre de parties, écrivons les parties sous la forme $I_0, \dots, I_{k-1}$ ordonnées de telle sorte que $\max I_0 < ... < \max I_{k-1}$, et notons i_m le nombre d'éléments de I_m. Alors

$$D^n(g \circ f)(x)(v_0, \dots, v_{n-1}) = \sum_I D^k g(f(x))(D^{i_0} f(x)(v_{I_0}), ..., D^{i_{k-1}} f(x)(v_{I_{k-1}}))$$

où la somme porte sur toutes les partitions de $\{0, \dots, n-1\}$ et par v_{I_m} on entend la famille de vecteurs (v_i) dont les indices i appartiennent à I_m.

La formule est facile à démontrer par récurrence. En dérivant

$$D^k g(f(x))(D^{i_0} f(x)(v_{I_0}), \dots, D^{i_{k-1}} f(x)(v_{I_{k-1}})),$$

on obtient une somme de $k+1$ termes où l'on dérive soit $D^k g(f(x))$, soit l'un des $D^{i_m} f(x)$, ce qui revient à ajouter à la partition I soit un nouvel atome $\{-1\}$ à sa gauche, soit à étendre I_m en y ajoutant -1. De cette manière, on obtient de façon bijective toutes les partitions de $\{-1, \dots, n-1\}$, et la preuve peut se poursuivre (à réindiçage près).

La formalisation de cette formule est assez redoutable. Le plus difficile est d'écrire les choses avec précision, comme l'indique le nombre de points de suspension dans la formule ci-dessus. La partie la plus technique consiste à montrer qu'étendre les partitions de $\{0, \dots, n-1\}$ par les processus décrits ci-dessus permet d'obtenir de manière bijective toutes les partitions de $\{-1, \dots, n-1\}$. Au total, la formalisation de la formule de Faà di Bruno nécessite un peu plus de 1 000 lignes.

Une fois cette formule disponible, la preuve que la composition de fonctions C^n est C^n est facile, car la formule indique comment construire une série de Taylor pour $g \circ f$ en termes de séries de Taylor pour g et f. Pour les fonctions C^ω, il faut en outre utiliser le fait que la composition de fonctions analytiques est analytique, un fait qui a déjà été abordé dans la section 6.

L'inversion est étroitement liée à la composition. Le théorème d'inversion locale assure que, si une fonction entre espaces complets est C^n en un point x_0 et possède une dérivée inversible, alors elle admet un inverse local qui est également C^n. Ce théorème comporte deux parties. Premièrement, si f est C^1 en x_0 et possède une dérivée inversible, alors elle admet un inverse local g. Deuxièmement, si en fait f est C^n, alors g est automatiquement C^n elle aussi. Ces deux parties sont essentiellement indépendantes.

La plus profonde est la première, qui construit l'inverse par une application appropriée du principe de contraction de Banach. L'hypothèse C^1 n'est pas la bonne sur un corps général : il faudrait plutôt exiger que f soit strictement dérivable en x_0, c'est-à-dire que $f(y) - f(z) - Df(x_0)(y-z) = o(y-z)$ lorsque y et z tendent vers x_0. Cela découle d'une hypothèse C^1 sur $\mathbb{R}$ ou $\mathbb{C}$ (par l'inégalité des accroissements finis), mais pas sur les corps généraux. Cependant, la différentiabilité stricte découle de l'analyticité, qui est de toute façon la bonne condition de régularité à considérer sur les corps généraux, comme nous le verrons plus en détail dans la section 8.

Pour la deuxième partie, on peut vérifier la régularité par récurrence : comme $f \circ g = \mathrm{id}$, on obtient $Df(g(x))Dg(x) = \mathrm{id}$, donc $Dg(x) = Df(g(x))^{-1}$. Cette formule permet de transférer une hypothèse C^n sur g à une hypothèse C^n sur Dg, ce qui permet de déduire que g est C^{n+1} et de procéder par récurrence. Notons que cet argument permet de déduire que g est C^{n+1} à partir du fait qu'il est C^n. Les espaces ou les applications ne changent pas au cours de l'induction, il n'y a donc ici aucune difficulté de formalisation. Le cas analytique doit être traité séparément car il nécessite des contrôles de croissance supplémentaires, mais ce point a déjà été abordé dans la section 6. Finalement, nous obtenons un énoncé générique du théorème d'inversion locale, valable à la fois pour $n \neq \omega$ et $n = \omega$ de manière unifiée.

La première étape est formalisée en construisant un inverse local à partir d'une application strictement dérivable avec une dérivée inversible. Elle est formulée dans Mathlib avec `PartialHomeomorph`, qui contient les données d'une application (ici f) et d'une autre application, qui sont localement inverses l'une de l'autre.

```
def HasStrictFDerivAt.toPartialHomeomorph
  (f : E → F) {f' : E ≃L[𝕜] F} [CompleteSpace E]
  (hf : HasStrictFDerivAt f f' a) :
  PartialHomeomorph E F
```

Ici, `E ≃L[𝕜] F` désigne l'espace des bijections linéaires continues entre E et F. Par conséquent, notre hypothèse `hf` revient à dire que la dérivée de f est inversible.

La deuxième étape est formalisée en disant que si deux fonctions sont localement inverses l'une de l'autre, alors la seconde hérite des propriétés de régularité de la première :

```
theorem PartialHomeomorph.contDiffAt_symm
  [CompleteSpace E] (f : PartialHomeomorph E F)
  {f₀' : E ≃L[𝕜] F} {a : F} (ha : a ∈ f.target)
  (hf₀' : HasFDerivAt f f₀' (f.symm a))
  (hf : ContDiffAt 𝕜 n f (f.symm a)) :
  ContDiffAt 𝕜 n f.symm a
```

Cela peut s'appliquer à l'homéomorphisme partiel `HasStrictFDerivAt.toPartialHomeomorph f hf` qui a été construit lors de la première étape.

7.3. **Dérivées itérées.** Pour contourner les difficultés liées au caractère non unique des dérivées, nous avons défini les fonctions C^n par l'existence d'une bonne série de Taylor. Cependant, comme pour `fderivWithin` et `fderiv`, il est souvent utile de disposer d'un choix arbitraire de dérivée itérée – bien qu'en général, on ne puisse prouver qu'elle se comporte bien que sous l'hypothèse d'unicité des différentielles dans le domaine. Mathlib définit la dérivée d'ordre $(n+1)$ comme la dérivée de la dérivée d'ordre n (avec des choix à chaque étape). La définition formelle est la suivante :

```
def iteratedFDerivWithin (𝕜) (n : ℕ) (f : E → F) (s : Set E) : E → E[×n]→L[𝕜] F :=
 Nat.recOn n
   (fun x ↦ ContinuousMultilinearMap.uncurry0 𝕜 E (f x))
   (fun _ rec x ↦ ContinuousLinearMap.uncurryLeft (fderivWithin 𝕜 rec s x))
```

Pour $n = 0$, on utilise la fonction vue comme une application 0-multilinéaire, et pour $n+1$, on utilise la dérivée de la n-ième dérivée, vue comme une application $(n+1)$-multilinéaire. On définit également `iteratedFDeriv`, correspondant à prendre pour s l'espace entier.

Dans les domaines avec unicité de la différentielle (par exemple l'espace entier, les ensembles ouverts, ou les ensembles convexes à intérieur non vide), si une fonction possède une série de Taylor jusqu'à l'ordre n, alors cette série de Taylor doit coïncider avec `iteratedFDerivWithin`, par unicité. Par conséquent, nos preuves selon lesquelles la somme ou la composition de fonctions C^n sont des fonctions C^n, donnant des formules pour la série de Taylor, fournissent également des formules pour les dérivées itérées. Par exemple, pour l'addition, on obtient

```
theorem iteratedFDerivWithin_add_apply
  (hf : ContDiffWithinAt 𝕜 i f s x) (hg : ContDiffWithinAt 𝕜 i g s x)
  (hu : UniqueDiffOn 𝕜 s) (hx : x ∈ s) :
  iteratedFDerivWithin 𝕜 i (f + g) s x =
    iteratedFDerivWithin 𝕜 i f s x + iteratedFDerivWithin 𝕜 i g s x
```

Lorsque nous avons défini les fonctions C^n, nous avons dû localiser la définition, pour éviter les problèmes liés la non-unicité des dérivées, comme nous l'avons expliqué dans la tentative 7.3. Dans les domaines où les dérivées sont uniques, le problème disparaît : les différentes séries de Taylor autour de différents points doivent coïncider localement et peuvent donc être assemblées. En d'autres termes, elles coïncident avec un objet global, qui

est `iteratedFDerivWithin`. Soit `ftaylorSeriesWithin 𝕜 f s` la série de Taylor formée à l'aide de la suite des dérivées itérées. On obtient

```
theorem ContDiffOn.ftaylorSeriesWithin
  (h : ContDiffOn 𝕜 n f s) (hs : UniqueDiffOn 𝕜 s) :
  HasFTaylorSeriesUpToOn n f (ftaylorSeriesWithin 𝕜 f s) s
```

Remarque 7.7. La formalisation des propriétés des fonctions C^n comporte de nombreuses isomorphismes canoniques afin d'identifier, par exemple, $\mathcal{L}(E, \mathcal{L}(E, F))$ avec les applications bilinéaires continues de $E \times E$ vers F. Dans les livres de référence tels que [Car67], ces identifications sont abordées au début, puis utilisées implicitement partout. Il y a cependant une subtilité ici. Soit $D(Df)$ la dérivée de la dérivée de f, en tant qu'élément de $\mathcal{L}(E, \mathcal{L}(E, F))$, et $D^2 f$ sa version en application bilinéaire continue. Lorsque nous disons qu'une fonction est C^3, signifie-t-on que $D(Df)$ possède une dérivée qui est continue, ou que $D^2 f$ possède une dérivée qui est continue ? Et obtient-on les mêmes résultats en dérivant ces deux notions de dérivée seconde puis en prenant les isomorphismes canoniques ? La réponse à ces deux questions est oui, heureusement. Cela vient du fait que l'identification canonique est elle-même une bijection linéaire continue entre deux espaces vectoriels de fonctions, et que les bijections linéaires continues respectent la différentiabilité et commutent avec la dérivation. Grâce à notre choix de valeur par défaut suivant lequel les fonctions non dérivables ont des dérivées nulles, la composition avec une bijection linéaire continue et la dérivation commutent même pour les fonctions non dérivables, ce qui est extrêmement pratique dans le processus de formalisation. Notons toutefois que cette discussion n'est valable que pour les fonctions sur des domaines avec unicité de la différentielle : sinon, la non-unicité des dérivées casse a priori cette commutation, ce qui doit être pris en compte dans la formalisation.

8. Symétrie de la dérivée seconde

Une propriété importante de la dérivée seconde dans un espace vectoriel réel est sa symétrie $D^2 f(x)(v, w) = D^2 f(x)(w, v)$. En voici une version précise avec des hypothèses faibles, voir [Car67, Théorème 5.1.1] :

Théorème 8.1. *Soient E, F deux espaces vectoriels normés réels et soit $f : E \to F$. Supposons que f est dérivable au voisinage d'un point x, et que Df est dérivable en x. Alors $D^2 f(x)(v, w) = D^2 f(x)(w, v)$ pour tous vecteurs v, w.*

Ce résultat découle du fait que $(f(x+tv+tw) - f(x+tv) - f(x+tw) + f(x))/t^2$ converge vers $D^2 f(x)(v, w)$ lorsque t tend vers 0. Cette assertion n'est pas tout à fait évidente à démontrer avec les hypothèses faibles qu'on fait, mais elle est correcte. Comme cette expression de gauche est symétrique en v et w, il s'ensuit que la limite l'est également.

Ce résultat est fondamental pour de nombreuses constructions en géométrie. Par exemple, définissons le crochet de Lie de deux champs de vecteurs $V, W : E \to E$ par

$$[V, W](x) = DW(x)(V(x)) - DV(x)(W(x)).$$

Le crochet de Lie encode le défaut de commutation entre les flots de V et W. Cette définition est invariante par difféomorphisme local, et s'étend donc aux variétés. Pour vérifier

l'invariance, on obtient à la fin du calcul une erreur donnée par un terme d'ordre 2, de la forme

$$D^2f(x)(V(x),W(x)) - D^2f(x)(W(x),V(x)). \tag{8.1}$$

Il s'annule grâce à la symétrie de la dérivée seconde, ce qui donne bien l'invariance annoncée.

Mathlib contient la symétrie de la dérivée seconde, dans une version légèrement plus générale puisque les domaines convexes sont autorisés :

```
theorem Convex.second_derivative_within_at_symmetric
  (s_conv : Convex ℝ s) (hne : (interior s).Nonempty)
  (hf : ∀ x ∈ interior s, HasFDerivAt f (f' x) x) (xs : x ∈ s)
  (hx : HasFDerivWithinAt f' f'' (interior s) x) (v w : E) :
  f'' v w = f'' w v
```

La preuve est essentiellement la même que plus haut, en faisant attention à ne prendre que des points à l'intérieur de s.

Cela s'étend facilement des espaces vectoriels réels aux espaces vectoriels complexes, puisque l'on peut considérer un espace vectoriel complexe comme un espace vectoriel réel. Pour couvrir également les variétés p-adiques, nous aimerions étendre cette propriété à d'autres corps normés, mais malheureusement, la symétrie de la dérivée seconde n'est plus vraie en général : la preuve sur les réels utilise l'inégalité des accroissements finis, qui ne s'applique pas dans des contextes discontinus. Nous allons décrire un contre-exemple, qui est essentiellement [Sch84, Corollaire 84.2]. La proposition suivante montre que la dérivée d'une fonction sur $\mathbb{Q}_p^2$ peut être essentiellement arbitraire, ce qui permet d'obtenir des dérivées secondes non symétriques. Pour simplifier, nous travaillons dans le sous-ensemble ouvert $\mathbb{Z}_p$ de $\mathbb{Q}_p$, où p est un nombre premier quelconque. Le lecteur qui n'est pas familier avec les nombres p-adiques peut sans problème sauter cet argument.

Proposition 8.2. *Soit p un nombre premier. Pour $x, y \in \mathbb{Z}_p^2$, soit $M(x,y)$ une matrice de $\mathbb{Q}_p^2$ vers $\mathbb{Q}_p$ dépendant de manière continue de (x,y). Il existe alors une fonction $f : \mathbb{Z}_p^2 \to \mathbb{Q}_p$ de classe C^1 dont la dérivée est M.*

Notons que le résultat analogue est complètement faux sur $\mathbb{R}$, puisque l'intégrale de la dérivée le long d'une boucle de 0 à 0 doit être nulle, de sorte que tout est beaucoup plus rigide.

La proposition ci-dessus nous permet de construire un contre-exemple à la symétrie des dérivées secondes. Soit $M(x,y) = (0,x)$. Alors la fonction correspondante f a pour dérivée M, qui est C^∞, et elle est donc elle-même C^∞. De plus, $\partial f/\partial x = 0$, donc $\partial^2 f/\partial y \partial x = 0$, tandis que $\partial f/\partial y = x$, donc $\partial^2 f/\partial x \partial y = 1$.

Démonstration. Un point dans $\mathbb{Z}_p$ est déterminé par ses réductions successives dans $\mathbb{Z}/p^n\mathbb{Z}$: à chaque étape, on affine en ajoutant les données du $n+1$-ième chiffre. De manière équivalente, $\mathbb{Z}_p$ est l'union de p^n boules de rayon p^{-n}, chaque boule étant l'union de p boules de rayon $p^{-(n+1)}$. Il en va de même dans $\mathbb{Z}_p^2$, sauf qu'il y a p^{2n} boules de rayon p^{-n}. Choisissons un centre pour chaque boule (le choix canonique consiste à prendre le point de la boule dont les chiffres suivants sont tous nuls), et soit C_n l'ensemble des centres des boules de profondeur n, avec $C_n \subseteq C_{n+1}$.

Nous définissons f de manière récursive sur C_n, en partant de $f(0) = 0$. Supposons que f soit définie sur C_n. Pour tout $b \in C_{n+1}$, prenons $a \in C_n$ appartenant à la même boule de rayon p^{-n}. Nous posons alors $f(b) = f(a) + M(a) \cdot (b-a)$. Notons que pour $a = b$, les deux côtés donnent la même formule, de sorte que cette définition par récurrence est cohérente. De plus, $\|f(b) - f(a)\| \leqslant K\|b-a\| \leqslant Kp^{-n}$. La sommabilité de cette série garantit que la fonction f ainsi définie sur l'union croissante des C_n est uniformément continue. Par conséquent, elle s'étend de manière unique à une fonction sur $\mathbb{Z}_p^2$, que nous désignons à nouveau par f.

Il reste à montrer que f est dérivable en chaque point a, avec pour dérivée $M(a)$. Soit $a_n \in C_n$ la suite des centres des boules contenant a, et de même pour un point b. Alors

$$f(b) - f(a) = \sum M(b_n) \cdot (b_{n+1} - b_n) - \sum M(a_n)(a_{n+1} - a_n).$$

Supposons que $\|b-a\| = p^{-(N+1)}$. Alors $a_n = b_n$ pour $n \leq N$, de sorte que nous avons

$$f(b) - f(a) = \sum_{n \geq N} M(b_n) \cdot (b_{n+1} - b_n) - \sum_{n \geq N} M(a_n)(a_{n+1} - a_n).$$

Soit $\varepsilon > 0$. Si N est suffisamment grand, les deux matrices $M(b_n)$ et $M(a_n)$ sont à une distance au plus ε de $M(a)$, en vertu de la continuité de M. Par conséquent, $f(b) - f(a)$ est égal à

$$\begin{aligned} &M(a) \sum_{n \geq N} (b_{n+1} - b_n) - M(a) \sum_{n \geq N} (a_{n+1} - a_n) \pm 2\varepsilon \sum_{n \geq N} p^{-n} \\ &= M(a)(b - b_N) - M(a)(a - a_N) \pm 2\varepsilon p^{-N}/(1 - p^{-1}) \\ &= M(a)(b-a) \pm 2\varepsilon \|b-a\| p/(1-p^{-1}). \end{aligned}$$

Nous avons montré que, dès que b est suffisamment proche de a, nous avons pour $C = 2p/(1-p^{-1})$ l'estimation

$$\|f(b) - f(a) - M(a)(b-a)\| \leqslant C\varepsilon\|b-a\|.$$

C'est la définition de la différentiabilité en a avec une dérivée $M(a)$. □

La construction est suffisamment explicite pour que l'on puisse tout écrire. Pour le contre-exemple à la symétrie des dérivées secondes, avec $M(x,y) = (0,x)$, la fonction f s'écrit

$$f\big(\sum x_n p^n, \sum y_n p^n\big) = \sum_{k<l} x_k y_l p^{k+l}.$$

Le contre-exemple ci-dessus peut s'adapter à un cadre moins naturel mais plus élémentaire. Notons $K \subseteq [0,1]$ le Cantor triadique standard, formé des points dont un développement en base 3 n'admet que des 0 et des 2. Alors il existe une fonction $f : \mathbb{R}^2 \to \mathbb{R}$ qui est C^∞ sur K^2 (dans le sens développé plus haut, i.e., en ne calculant les dérivées successives qu'en prenant des suites dans K^2) mais dont la dérivée seconde n'est pas symétrique. La construction est exactement la même que celle décrite ci-dessus, avec $p = 2$ et C_n par exemple l'ensemble des points en bas à gauche des produits d'intervalles construits à la n-ème génération.

La pathologie ci-dessus ne peut pas se produire pour les applications analytiques : si f est localement donnée par une série $f(x+y) = \sum_n p_n(y, \dots, y)$ comme dans la définition 6.1,

alors on peut calculer les dérivées itérées de f comme suit : on a

$$D^n f(x)(v_1, \dots, v_n) = \sum_{\sigma \in \mathfrak{S}_n} p_n(v_{\sigma(1)}, \dots, v_{\sigma(n)}) \tag{8.2}$$

où $\mathfrak{S}_n$ désigne l'ensemble des permutations de $\{1, \dots, n\}$. Cette expression est manifestement symétrique par rapport aux v_i. En particulier,

$$D^2 f(x)(v_1, v_2) = p_2(v_1, v_2) + p_2(v_2, v_1).$$

Il découle de cette discussion que la symétrie des dérivées secondes des fonctions C^n est vérifiée lorsque $n \geq 2$ si le corps est $\mathbb{R}$ ou $\mathbb{C}$, et pour $n = \omega$ dans le cas contraire. Tout calcul avancé sur les champs de vecteurs n'est possible que sous de telles conditions de régularité. C'est la raison pour laquelle, sur des corps différents de $\mathbb{R}$ ou $\mathbb{C}$, Bourbaki ne considère que des variétés analytiques dans [Bou71].

Voici la formulation dans Mathlib de la formule ci-dessus pour la dérivée itérée d'une fonction analytique :

```
theorem HasFPowerSeriesWithinOnBall.iteratedFDerivWithin_eq_sum
  (h : HasFPowerSeriesWithinOnBall f p s x r)
  (h' : AnalyticOn 𝕜 f s) (hs : UniqueDiffOn 𝕜 s) (hx : x ∈ s) (v : Fin n → E) :
  iteratedFDerivWithin 𝕜 n f s x v = ∑ σ : Perm (Fin n), p n (fun i ↦ v (σ i))
```

On voudrait maintenant exprimer que la dérivée seconde est symétrique sous une condition de régularité qui dépend du corps. Soit `minSmoothness 𝕜 n` égal à n si le corps est isomorphe en tant que corps normé à $\mathbb{R}$ ou $\mathbb{C}$, et égal à ω dans le cas contraire. La version formalisée générale de la symétrie de la dérivée seconde est la suivante :

```
theorem ContDiffAt.isSymmSndFDerivAt (v w : E)
  (hf : ContDiffAt 𝕜 n f x) (hn : minSmoothness 𝕜 2 ≤ n) :
  fderiv 𝕜 (fderiv 𝕜 f) x v w = fderiv 𝕜 (fderiv 𝕜 f) x w v
```

Ce choix de conception avec une fonction `minSmoothness` permet d'exprimer des propositions de manière unifiée. Par exemple, le fait que tirer en arrière des champs de vecteurs commute avec le crochet de Lie s'écrit comme suit :

```
lemma pullback_lieBracket
  (hn : minSmoothness 𝕜 2 ≤ n) (h'f : ContDiffAt 𝕜 n f x)
  (hV : DifferentiableAt 𝕜 V (f x)) (hW : DifferentiableAt 𝕜 W (f x)) :
  pullback 𝕜 f (lieBracket 𝕜 V W) x = lieBracket 𝕜 (pullback 𝕜 f V) (pullback 𝕜 f W) x
```

Cette affirmation est d'abord démontrée sans l'hypothèse `hn`, en exigeant à la place que la dérivée seconde de f en x soit symétrique afin d'annuler les termes (8.1). Le théorème `ContDiffAt.isSymmSndFDerivAt` est ensuite appliqué pour montrer que la symétrie est satisfaite sous `hn`.

Donnons un exemple d'application qui a été une des motivations-clé pour le développement de cette théorie dans Mathlib. Cet exemple est un peu trop avancé pour le cadre de ce texte, mais je l'inclus quand même juste pour l'aspect culturel.

On va construire l'algèbre de Lie des groupes de Lie sur des corps arbitraires. Un groupe de Lie G est une variété dotée d'une multiplication et d'une inversion qui sont de classe C^n pour un certain n. Étant donnés deux éléments v et w de l'espace tangent $\mathfrak{g}$ en l'identité,

on peut former les champs de vecteurs invariants à gauche V et W correspondants en transportant les vecteurs partout dans le groupe par multiplication à gauche. La valeur en l'identité du crochet de Lie $[V, W]$ de ces champs de vecteurs est un nouveau vecteur dans $\mathfrak{g}$, appelé le crochet de Lie (dans le groupe) de v et w. Cela dote $\mathfrak{g}$ d'une structure d'algèbre de Lie, car le crochet de Lie des champs de vecteurs satisfait les propriétés requises (linéarité, antisymétrie, identité de Jacobi). Toutes ces propriétés sont vérifiées par des calculs directs dans les espaces vectoriels, et transportées à la variété puisque le tiré en arrière du crochet de Lie des champs de vecteurs par difféomorphismes locaux est bien défini, comme expliqué ci-dessus. Cela nécessite un certain degré de régularité, qui dépend du corps de base : sur $\mathbb{R}$ ou $\mathbb{C}$, on a besoin de C^3 pour cette construction, tandis que sur les autres corps C^ω est nécessaire. La version formalisée est la suivante :

```
instance instLieAlgebraGroupLieAlgebra
  {I : ModelWithCorners 𝕜 E H} [TopologicalSpace G]
  [ChartedSpace H G] [Group G]
  [LieGroup I (minSmoothness 𝕜 3) G] [CompleteSpace E] :
  LieAlgebra 𝕜 (GroupLieAlgebra I G)
```

Le point essentiel dans cet énoncé est l'hypothèse `LieGroup I (minSmoothness 𝕜 3) G` qui exige que la multiplication et l'inversion soient de classe C^3 si le corps est $\mathbb{R}$ ou $\mathbb{C}$ et analytiques dans le cas contraire, et ce de manière unifiée.

Dans un livre, on pourrait très bien développer toute la théorie juste dans le cas de $\mathbb{R}$ ou $\mathbb{C}$, et rajouter un appendice expliquant que tout se passe pareil dans le cas p-adique à condition de supposer que les groupes de Lie sont analytiques. Dans une bibliothèque formalisée, ce n'est pas possible. Il faut soit tout refaire une deuxième fois (avec les problèmes de maintenance que cela pose, par exemple il faudra penser à dupliquer tout résultat qu'on rajoutera dans le cas réel ou complexe), soit trouver un cadre unifié. La deuxième option est évidemment largement préférable, mais elle demande de bien réfléchir à l'architecture des définitions comme ce texte l'a illustré.

Références

[AH77] K. Appel & W. Haken – « Every planar map is four colorable. I. Discharging », *Illinois J. Math.* **21** (1977), no. 3, p. 429–490. Cité page 1.

[Bou71] N. Bourbaki – *éléments de mathématique. Fasc. XXXVI. Variétés différentielles et analytiques. Fascicule de résultats (Paragraphes 8 à 15)*, Actualités Scientifiques et Industrielles [Current Scientific and Industrial Topics], vol. No. 1347, Hermann, Paris, 1971. Cité pages 16, 25 et 34.

[Car67] H. Cartan – *Calcul différentiel*, Hermann, Paris, 1967. Cité pages 15 et 31.

[Hal05] T. C. Hales – « A proof of the Kepler conjecture », *Ann. of Math. (2)* **162** (2005), no. 3, p. 1065–1185. Cité page 1.

[Kar01] A. Karlsson – « Non-expanding maps and Busemann functions », *Ergodic Theory Dynam. Systems* **21** (2001), p. 1447–1457. Cité page 3.

[KN81] E. Kohlberg & A. Neyman – « Asymptotic behavior of nonexpansive mappings in normed linear spaces », *Israel J. Math.* **38** (1981), p. 269–275. Cité pages 3 et 5.

[Muj86] J. Mujica – *Complex analysis in Banach spaces*, North-Holland Mathematics Studies, vol. 120, North-Holland Publishing Co., Amsterdam, 1986. Cité page 20.

[Pös21] J. Pöschel – « On the Siegel-Sternberg linearization theorem », *J. Dynam. Differential Equations* **33** (2021), no. 3, p. 1399–1425. Cité page 23.

[Sch84] W. H. Schikhof – *Ultrametric calculus*, Cambridge Studies in Advanced Mathematics, vol. 4, Cambridge University Press, Cambridge, 1984, An introduction to p-adic analysis. Cité page 32.

IRMAR, CNRS UMR 6625, Université de Rennes, 35042 Rennes, France
Courriel: sebastien.gouezel@univ-rennes1.fr