
**PROJET DE FIN D'ÉTUDES**

pour obtenir le diplôme de

l'**Université Galatasaray**
Spécialité : **Mathématiques**

# Formalisation du théorème de Desargues en Lean

Rapport 4

Préparé par **Abdullah Uyu**
Résponsable : **Can Ozan Oğuz**

*6 mai 2024*

# Table des matières

# Introduction

Lean est un assistant de preuve. La transcription d'une preuve en langage humain dans un assistant de preuve est appelée *formalisation*. Lean dispose d'une grande bibliothèque de preuves, y compris de nombreuses preuves du niveau de licence, appelée mathlib4. Dans cette bibliothèque, il y a d'importants théorèmes manquants, et le théorème de Desargues est l'un d'entre eux.

Les deux aspects principaux de ce projet sont l'apprentissage des rudiments de la théorie des géométries projectives, et de la programmation en Lean. Avec ces deux aspects, le but ultime est de formaliser le théorème de Desargues dans Lean.

# Résultats réquis

## 1 Première rencontre

Pour motiver les géométries projectives, on commence par considérer les droites passant par l'origine dans le plan. On peut représenter la plupart de ces lignes par des points sur l'axe $y = 1$. La seule droite que l'on n'a pas réussi à représenter est l'axe $x$. On notera également que l'on peut bien sûr choisir n'importe quel axe pour la représentation, à l'exception de ceux qui passent par l'origine. Cette impossibilité dans les cas d'exception est clairement visible sur la figure 1. Si l'on choisit un tel axe, la droite que l'on ne parviendra pas à représenter sera la droite (passant par l'origine) qui est parallèle à cet axe.

Effectuons la même procédure pour l'espace. On peut représenter la plupart des droites passant par l'origine par des points sur le plan $z = 1$. Maintenant, les seules droites que nous ne pouvons pas représenter sont exactement les droites du plan que nous avons représenté précédemment. La remarque sur le choix de l'axe de représentation s'étend ici comme tout plan ne passant pas par l'origine peut être utilisé. De plus, le plan irreprésentable sera celui (qui passe par l'origine) qui est parallèle au plan de représentation.

Ce processus s'appelle la *projectivisation d'un espace vectoriel*. Écrivons-le en langage d'algèbre linéaire.

## 2 Projectivisation d'un espace vectoriel

**Définition 1** ([FF00, p. 27])**.** Soit $V$ un espace vectoriel. Sur $V^\bullet := V \setminus \{0\}$, on définit une relation binaire comme suit : $x \sim y$ si et seulement si $x, y$ sont linéairement dépendants. Comme ceci est une relation d'équivalence, l'ensemble quotient $\mathscr{P}(V) := V^\bullet/\sim$ est bien défini. $\mathscr{P}(V)$ est appelée la *projectivisation de l'espace vectoriel* $V$.

Pour simplifier le langage, nous aurons également besoin de la définition de l'union disjointe.

**Définition 2.** Soit $A$ et $B$ deux ensembles. L'union $(A \times \{1\}) \cup (B \times \{0\})$, noté $A \dot\cup B$, est appelée *union disjointe* de $A$ et $B$.

La motivation ci-dessus peut donc être formulée comme suit :

$$\begin{aligned}\mathscr{P}(\mathbf{R}^3) &= \mathscr{P}(\mathbf{R}^2 \times \mathbf{R}) \\ &\cong \mathbf{R}^2 \dot\cup \mathscr{P}(\mathbf{R}^2) \\ &= \mathbf{R}^2 \dot\cup \mathscr{P}(\mathbf{R} \times \mathbf{R}) \\ &\cong \mathbf{R}^2 \dot\cup \mathbf{R} \dot\cup \mathscr{P}(\mathbf{R}).\end{aligned}$$

Cela résume ce que l'on fait mathématiquement lorsque l'on fait des dessins en perspective : On prend un plan ($\mathbf{R}^2$), on choisit un horizon ($\mathbf{R}$) et un point de fuite ($\mathscr{P}(\mathbf{R})$). La proposition suivante n'est qu'une généralisation de ce processus.

**Proposition 1** ([FF00, p. 28])**.** *Pour un $K$-espace vectoriel $V$, il existe une bijection naturelle*

$$s : \mathscr{P}(V \times K) \to V \dot\cup \mathscr{P}(V)$$

*induite par la fonction $t : (V \times K)^\bullet \to V \dot\cup \mathscr{P}(V)$ définie par $t(x, \xi) = \xi^{-1}x$ si $\xi \neq 0$ et $t(x, 0) = [x]$ pour $x \neq 0$, où $[x]$ désigne le point de $\mathscr{P}(V)$ représenté par $x$.*

**Définition 3** ([FF00, p. 26])**.** Une *géométrie projective* est un ensemble $G$ accompagné d'une relation ternaire $\ell \subseteq G \times G \times G$ telle que les axiomes suivants sont satisfaits :

($L_1$) $\ell(a, b, a)$ pour tout $a, b \in G$.

($L_2$) $\ell(a, p, q), \ell(b, p, q)$ et $p \neq q \implies \ell(a, b, p)$.

($L_3$) $\ell(p, a, b)$ et $\ell(p, c, d) \implies \ell(q, a, c)$ et $\ell(q, b, d)$ pour un $q \in G$.

Les éléments de $G$ sont appelés les *points* de la géométrie. Et trois points $a, b, c$ sont dits *colinéaires* si $\ell(a, b, c)$.

Notons l'exemple le plus important pour une géométrie projective. Lors d'une discussion sur la chaîne de Zulip de Lean, Joseph Myers a conseillé d'énoncer le théorème sur cet exemple, mais pas sur la définition abstraite d'une géométrie projective. De plus, il est noté que la version du théorème en géométrie euclidienne, qui est probablement même connue de nombreux élèves du lycée, peut être déduite de l'énoncé du théorème sur cet important modèle de géométrie projective.

**Proposition 2** ([FF00, p. 27])**.** *Pour un espace vectoriel $V$, $\mathscr{P}(V)$ est une géométrie projective si pour tout élément $X, Y, Z \in \mathscr{P}(V)$ on définit : $\ell(X, Y, Z)$ si et seulement si $X, Y, Z$ ont des représentants $x, y, z$ linéairement dépendants.*

## 3 Isomorphismes

Les isomorphismes seront très utiles tout au long du voyage.

**Définition 4** ([FF00, p. 27])**.** Un *isomorphisme* de géométries projectives est une bijection $g : G_1 \to G_2$ satisfaisant $\ell_1(a, b, c)$ si et seulement si $\ell_2(ga, gb, gc)$. Si $G_1 = G_2$, alors on dit que $g$ est une *collinéation*.

Toutes les applications linéaires bijectives induisent une colinéation.

*Exemple* 1. Soit $T : \mathbf{R}^n \to \mathbf{R}^n$ une application linéaire bijective. L'application $g : \mathscr{P}(\mathbf{R}^n) \to \mathscr{P}(\mathbf{R}^n)$, $[x] \mapsto [T(x)]$ est une isomorphisme de géometries projectives.

L'application $g$ est bien définie : Soit $x, y \in \mathbf{R}^n$ tel que $[x] = [y]$, i.e. $x = ky$ pour un $k \in \mathbf{R}$. On a :

$$\begin{aligned} [T(x)] &= [T(ky)] && x \text{ définition} \\ &= [kT(y)] && T \text{ linéaire} \\ &= [T(y)] && \text{définition 1.} \end{aligned}$$

D'où, $g$ est bien définie.

Montrons que $g$ est injective. Soit $[x], [y] \in \mathscr{P}(\mathbf{R}^n)$ tel que $[T(x)] = [T(y)]$, i.e. $T(x) = kT(y)$ pour un $k \in \mathbf{R}$. Comme $T$ est linéaire, $T(x) = T(ky)$. De plus, $x = ky$ car $T$ est injective. Par la définition de la classe d'équivalence, on obtient $[x] = [y]$. D'où $g$ est injective. Pour la surjectivité, prenons $[x] \in \mathscr{P}(\mathbf{R}^n)$. Comme $T$ est bijective, $T^{-1}$ existe, et $[T^{-1}(x)] \in \mathscr{P}(\mathbf{R}^n)$. Ainsi $g([T^{-1}(x)]) = [T(T^{-1}(x))] = [x]$. D'où, $g$ est surjective. On a montré que $g$ est bijective.

Vérifions la condition d'isomorphisme. Soit $[x], [y], [z] \in \mathscr{P}(\mathbf{R}^n)$. Supposons que $\ell([x], [y], [z])$. On a des équivalences :

$$\begin{aligned} &\iff ax + by + cz = 0 && \text{proposition 2} \\ &\iff T(ax + by + cz) = 0 && T \text{ linéaire, injective} \\ &\iff aT(x) + bT(y) + cT(z) = 0 && T \text{ linéaire} \\ &\iff \ell([T(x)], [T(y)], [T(z)]) && \text{proposition 2} \end{aligned}$$

Mais les colinéations sont bien plus que des applications linéaires bijectives. En d'autres termes, il existe des collinéations qui ne sont pas induites par une application linéaire.

*Exemple* 2. L'application $g : \mathscr{P}(\mathbf{R}^2) \to \mathscr{P}(\mathbf{R}^2)$ définie par $[(x_1, x_2)] \mapsto [((x_1/x_2)^3, 1)]$ si $x_2$ est non-nul, et $[(x_1, 0)] \mapsto [(x_1, 0)]$ sinon, est une collineation qui n'est pas induite par une application linéaire.

L'application $g$ est bien définie. Soit $x_1 > 0, x_2 > 0, x_1', x_2'$ des réels tel que $[(x_1, x_2)] = [(x_1', x_2')]$. Alors, $x_1' = kx_1$ et $x_2' = kx_2$ pour un $k$ non-nul. On a :

$$\begin{aligned}[((x_1'/x_2')^3, 1)] &= [((kx_1/kx_2)^3, 1)] \\ &= [((x_1/x_2)^3, 1)]\end{aligned}$$

D'où, $g$ est bien définie.

Montrons que $g$ est injective. Soit $x_1, x_2, x_1', x_2'$ des réels tel que

$$[((x_1/x_2)^3, 1)] = [((x_1'/x_2')^3), 1].$$

On va montrer que $[(x_1, x_2)] = [(x_1', x_2')]$. Par la définition de la classe d'équivalence, on a

$$((x_1/x_2)^3, 1) = k((x_1/x_2')^3, 1)$$

pour un $k$ non-nul. Alors, $k = 1$ et on a :

$$x_1/x_2 = x_1'/x_2'.$$

Si $x_1$ est nul, alors $x_1'$ l'est aussi, et donc on a :

$$\begin{aligned}[(0, x_2)] &= [0, \frac{x_2'}{x_2} x_2] \quad \text{classe d'équivalence définition} \\ &= [0, x_2]\end{aligned}$$

Sinon on a $x_1/x_1' = x_2/x_2'$ et donc :

$$\begin{aligned}[(x_1, x_2)] &= [(\frac{x_1' x_2}{x_2'}, \frac{x_1 x_2'}{x_1'})] \\ &= [(x_1', x_2')] \quad \text{classe d'équivalence définition}\end{aligned}$$

Enfin, on vérifie la reste de l'image. Soit $x_3$ un réel. Il est clair que $[((x_1/x_2)^3, 1)] \neq [(x_3, 0)]$. D'où, $g$ est injective. Pour la surjectivité, prenons des réels $x_1, x_2$. Si $x_2$ est nul, on a $g([(x_1, 0)]) = [(x_1, 0)]$. Sinon on a :

$$\begin{aligned}g([(x_1^{1/3}, x_2^{1/3})]) &= [((x_1^{1/3}/x_2^{1/3})^3, 1)] \quad g \text{ définition} \\ &= [(x_1/x_2, 1)] \\ &= [(x_1, x_2)] \quad \text{classe d'équivalence définition.}\end{aligned}$$

$g$ est automatiquement une collinéation car trois vecteurs quelconques sont toujours linéairement dépendants dans $\mathbf{R}^2$.

Montrons maintenant que $g$ n'est pas induite par une application linéaire. Supposons par l'absurde que $g$ est induite par l'application linéaire $T : \mathbf{R}^2 \to \mathbf{R}^2$.

Alors on a $T(1,0) = (k_1, 0)$ et $T(0,1) = (0, k_2)$ pour $k_1, k_2$ non-nuls. Soit $x_1$ un réel. D'une part, on a $T(x,1) = k_x(x^3, 1)$ pour un $k_x$ non-nul, et d'autre part, on a $T(x,1) = xT(1,0) + T(0,1)$. Donc, on a :

$$k_x(x^3, 1) = x(k_1, 0) + (0, k_2)$$

Par conséquent, on obtient l'égalité des polynômes $k_2x^3 = k_1x$ qui entraîne $k_1 = k_2 = 0$. Ceci oblige $T = 0$ qui est absurde.

# Formalisation

Commençons par transcrire la définition de la géométrie projective (définition 3). Il faut d'abord représenter d'une certaine manière les points, c'est-à-dire les éléments d'une géométrie projective, et la relation de colinéarité. Les points de la géométrie projective seront abstraits, c'est-à-dire un *type*. La relation de colinéarité peut être représentée par une fonction qui prend trois points et renvoie vrai ou faux ; après tout, lorsque l'on dit qu'une relation $\ell$ est valable, ce que l'on fait, c'est prendre trois points et se demander s'ils sont colinéaires ou non.

```
class HasCollinear (P : Type) where
ell : P → P → P → Prop

export HasCollinear (ell)

variable {Point : Type} [HasCollinear Point]
```

Ici, la flèche entre les types `P` peut être considérée comme la flèche entre le domaine et le codomaine dans les définitions de fonctions dans le langage habituel des mathématiques, c'est-à-dire le symbole "→".

Comme le point reste abstrait, il a fallu dire à Lean que l'existence d'une interprétation de la colinéarité est assurée, c'est-à-dire que la colinéarité des points est quelque chose que l'on peut décider. Dans le code, cela est satisfait par la dernière ligne : `Point` est un type dont la colinéarité existe.

Maintenant, on peut énoncer les axiomes $L_1$, $L_2$ et $L_3$ dans la définition 3.

```
axiom l1 (a b : Point) : ell a b a
axiom l2 (a b p q : Point) : ell a p q → ell b p q → p ≠ q → ell a b p
axiom l3 (a b c d p: Point) : ell p a b → ell p c d →
                              ∃ q : Point , ell q a c ∧ ell q b d
```

Il y a quelques points à noter ici. Tout d'abord, les flèches utilisées ici sont différentes de celles que nous avons utilisées ci-dessus, même si elles sont représentées par le même caractère typographique : Il s'agit de flèches d'implication, pour lesquelles nous utilisons normalement le symbole "$\Longrightarrow$". Deuxièmement, les conditions des axiomes qui contiennent des conjonctions logiques sont converties en implications sérielles. Expliquons cette équivalence logique. Supposons qu'on a une condition de la forme $(p \wedge q) \implies r$. On a des équivalences suivantes :

$$\begin{aligned}
&\iff \neg(p \wedge q) \vee r \\
&\iff (\neg p \vee \neg q) \vee r \\
&\iff \neg p \vee (\neg q \vee r) \\
&\iff p \implies (\neg q \vee r) \\
&\iff p \implies (q \implies r)
\end{aligned}$$

Grâce à ces axiomes, nous pouvons prouver la proposition suivante.

**Proposition 3.** *Toute relation ternaire $\ell$ qui satisfait les deux axiomes $L_1$ et $L_2$ est symétrique.*

Dans la preuve, on va dériver la colinéarité de toutes les permutations possibles de trois points $a, b, c$ à partir de $\ell(a, b, c)$. Ici, on ne donnera que $\ell(a, c, b)$ et les cinq autres versions seront omises.

```
theorem acb
  (a b c : Point) :
    ell a b c → ell a c b := by
  intro acb_col -- Supposons que $\ell(a,b,c)$.
  obtain rfl | bc_neq := eq_or_ne b c -- Si $b = c$,
  exact acb_col -- le résultat est trivial,
  apply l2 a c b c -- sinon on applique $\text{L}_2$;
  exact acb_col -- première condition de $\text{L}_2$
  apply l1 c b -- deuxième condition de $\text{L}_2$
  exact bc_neq -- troisième condition de $\text{L}_2$.
```

Ce code est la transcription exacte de la phrase "Si $b = c$, le résultat est trivial, et sinon on applique $\mathrm{L}_2$ à $\ell(a, b, c)$ et $\ell(c, b, c)$".

## 1 Regrouper le point et les axiomes

Cette proposition la plus simple peut être considérée comme un premier théorème prouvé avec cette configuration. Des théorèmes plus ambitieux peuvent

également être prouvés, mais décrire une géométrie projective en déclarant son point et axiomes les uns après les autres n'est pas la manière la plus élégante. Pour l'instant, ils n'ont qu'une relation sérielle dans le fichier. Ce qui est plus utile, c'est une structure qui contiendrait toutes les informations nécessaires à une géométrie projective. La structure suivante répond à ce besoin :

```
class ProjectiveGeometry
  (point : Type u) where
  ell : point → point → point → Prop
  l1  : ∀ a b, ell a b a
  l2  : ∀ a b p q, ell a p q → ell b p q → p ≠ q → ell a b p
  l3  : ∀ a b c d p, ell p a b → ell p c d →
                  ∃ q : point, ell q a c ∧ ell q b d
```

De cette manière, on a regroupé la relation de colinéarité et les axiomes ("bundling"), et le point est resté dégroupé ("unbundled") parce qu'il s'agit d'un paramètre et que l'on veut que différents types de points correspondent à différentes géométries projectives. Pour mieux expliquer le contraste, on ne veut pas que des relations de colinéarité différentes correspondent à des géométries projectives différentes ; on pense à une relation de colinéarité canonique pour obtenir une géométrie projective. Ceci est expliqué en détail ici.

## 2 Instances

Il est maintenant possible de déclarer que la projectivisation d'un espace vectoriel est une géométrie projective.

```
instance : ProjectiveGeometry (ℙ K V) :=
  ⟨
  fun X Y Z => ¬ Independent ![X, Y, Z],
  sorry,
  sorry,
  sorry
  ⟩
```

Expliquons comment cela fonctionne : Tout d'abord, une relation de colinéarité sera fournie. Celle-ci est déjà établie mathématiquement dans la proposition 2. La fonction `Independent` provient de mathlib4. Elle prend trois points de la projectivisation et affirme que leurs représentants sont linéairement dépendants. Sur le plan technique, il prend une famille de points de projectivisation. Une famille est une fonction d'indices dans des points. Heureusement, Lean dispose

d'une syntaxe pour convertir une liste en une telle fonction : "!". Si elle n'existait pas, on devrait l'écrire manuellement :

```
def indexer_generator
  (X Y Z : ℙ K V) :
    Fin 3 → ℙ K V :=
  fun i : Fin 3 => match i with
  | 0 => X
  | 1 => Y
  | 2 => Z
```

Ensuite, les axiomes seront vérifiés. Lean remplacera la relation de colinéarité dans les axiomes pour nous. Les "sorry" servent de substitut à des preuves qui n'ont pas encore été écrites.

Ainsi, nous pouvons continuer à étudier le côté purement synthétique et nos résultats s'appliqueront automatiquement à la projectivisation d'un espace vectoriel : Le côté concret. Pour les choses qui sont spécifiques aux projectivisations, nous pouvons bien sûr passer du côté concret et continuer à développer des propriétés pour elles.

## Références

[FF00] C.A. Faure et A. Frölicher. *Modern Projective Geometry*. T. 521. Dordrecht : Kluwer Academic Publishers, 2000.

# Table des figures

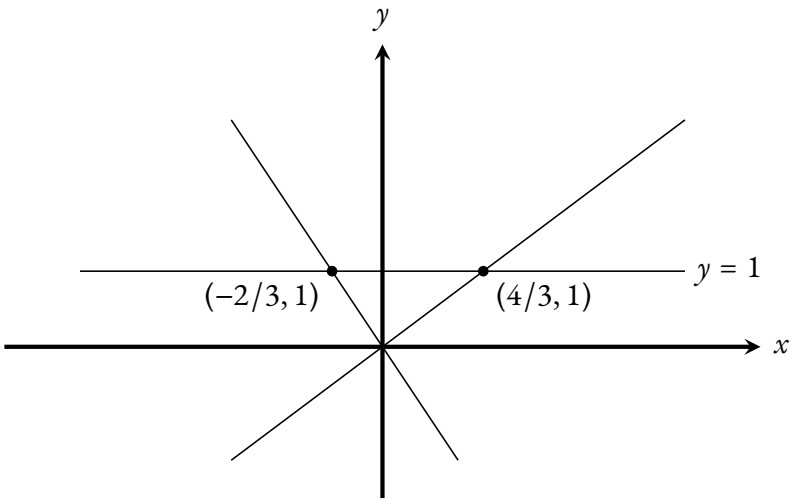


FIGURE 1 – La représentation des droites passant par l'origine dans le plan

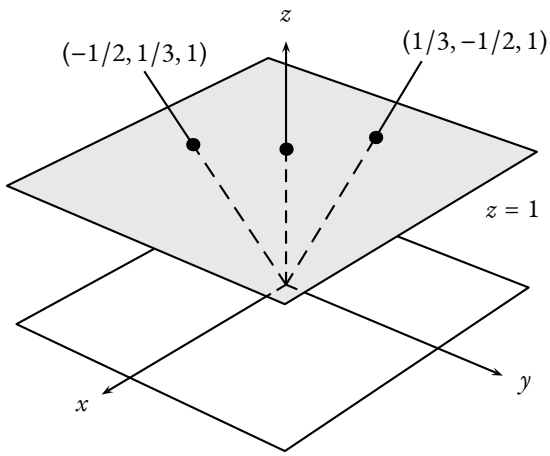


FIGURE 2 – La représentation des droites passant par l'origine dans l'espace