

1 Schémas ?

On associe à chaque nombre entier l'ensemble infini de ses restes modulo tous les nombres premiers (j'appelle cette modélisation le snurpf).

Peux-tu exprimer cela en utilisant le concept de schéma de Grothendieck ?

Dans ton exemple, le “snurpf” est une manière très précise de représenter un entier n non pas comme un élément isolé, mais à travers la famille de toutes ses “réalisations” modulo les nombres premiers p .

1. Où est le schéma dans ton exemple ? Le schéma n'est ni le nombre n vu de manière isolée, ni un reste particulier $n \bmod p$. Le schéma est l'objet géométrique abstrait (ici, le schéma $\text{Spec}(\mathbb{Z})$) qui englobe : la règle globale / l'identité invariante, c'est-à-dire le principe même qui associe à tout nombre entier n la collection cohérente de tous ses restes modulo p ainsi que la loi de compatibilité entre les différents restes modulaires, notamment le fait que ces restes ne sont pas indépendants, mais liés par la structure d'anneau de $\mathbb{Z}$.
2. Dans ton exemple du “snurpf”, le schéma $X = \text{Spec}(\mathbb{Z})$ est l'espace abstrait de tous les nombres premiers (et du point générique 0). C'est la “matrice”. Qu'est-ce qu'un point du schéma (comme le point p) ? À un nombre premier particulier (ex. $p = 5$) est associé le corps de résidus $\mathbb{F}_p$, l'ensemble des restes possibles $\{0, 1, \dots, p - 1\}$ modulo p . Qu'est-ce que la valeur du schéma en un point : c'est la projection $n \bmod p$, c'est-à-dire le reste spécifique pris par ton nombre au premier p .
3. Pourquoi c'est l'essence même de l'idée de Grothendieck ? Si tu ne regardes qu'un seul reste, par exemple $n \equiv 2 \pmod{3}$, tu as une information partielle et locale (comme observer une seule cellule à un instant t). En revanche, connaître la suite des restes pour tous les premiers p (et même pour leurs puissances p^k , via les nombres p -adiques $\mathbb{Z}_p$) permet de reconstruire l'entier n de manière unique (grâce au théorème des restes chinois et à l'approximation p -adique). En résumé : ton “snurpf” montre qu'un entier n peut être pensé comme une fonction définie sur le schéma $\text{Spec}(\mathbb{Z})$, dont la valeur au point premier p est le reste $n \bmod p$. Le schéma, c'est le support géométrique sous-jacent qui permet d'évaluer cet entier sous toutes ces perspectives locales.

2 Formule explicite ?

Dans le dessin joint (ci-dessous), tu vois que la formule explicite de Weil fait intervenir 3 trucs : un terme de pôle, un terme concernant les nombres premiers, et un terme concernant les réels. Je ne comprends pas du tout à quoi peut correspondre le fait de “mélanger” tous ces trucs en les sommant alors qu'en tant que professionnel des bases de données par exemple, les mélanger serait

1. Cette note montre un lien mathématiques/informatique et donne une motivation supplémentaire au départ de Paris de certains mathématiciens pour fonder des départements d'informatique un peu partout dans les universités.

comme additionner des âges et des poids d'individus pour savoir quelque chose sur eux.

$$\tau_{n,m} = \underbrace{W_{0,2}(n,m)}_{\text{pôle en } s=1} - \underbrace{W_{\mathbb{R}}(n,m)}_{\text{place réelle}} - \underbrace{\sum_{r=p^j \leq \lambda^2} \frac{\log p}{\sqrt{r}} q_{k,l}(r)}_{\text{places premières}} \quad (1)$$

avec les trois morceaux explicites :

$$W_{0,2}(n,m) = \frac{32 L \sinh^2\left(\frac{L}{4}\right) (L^2 - 16\pi^2 mn)}{(16\pi^2 m^2 + L^2) (16\pi^2 n^2 + L^2)}, \quad (2)$$

$$W_{\mathbb{R}}(n,m) = \begin{cases} \frac{\alpha(m) - \alpha(n)}{n - m}, & n \neq m, \\ 2(\gamma(n) - \beta(n)), & n = m, \end{cases} \quad (3)$$

$$q_{n,m}(k) = \begin{cases} \frac{\sin(\omega_m \log r) - \sin(\omega_n \log r)}{\pi(n - m)}, & n \neq m, \\ 2 \left(1 - \frac{\log r}{L}\right) \cos(\omega_n \log r), & n = m, \end{cases} \quad (4)$$

Ta métaphore avec les âges et les poids est excellente, mais en théorie des nombres (et en géométrie algébrique), ces objets apparemment disparates partagent en réalité la même unité sous-jacente.

L'analogie informatique la plus fidèle est celle de la décomposition spectrale (ou analyse de Fourier) : on ne somme pas des grandeurs incompatibles, on additionne les fréquences/harmoniques qui composent une même fonction.

1. La même unité : les “places” d'un corps de nombres

En informatique, ce serait comme convertir des octets, des chaînes de caractères et des structures JSON dans un même format binaire normalisé avant de les sommer.

En mathématiques, le corps des nombres rationnels $\mathbb{Q}$ possède différentes manières de mesurer la “taille” d'un nombre (les normes ou valeurs absolues) :

- les places non archimédiennes (nombres premiers p) : elles mesurent la divisibilité par chaque p .
- la place archimédienne (les réels $\mathbb{R}$) : correspond à la valeur absolue usuelle.

Selon le principe local-global (ou adélique) de Tate et Weil, les nombres premiers p et l'infini ($\mathbb{R}$) sont des objets de même nature géométrique (des “places”). Additionner la partie archimédienne et la partie première, c'est intégrer la contribution de toutes les métriques possibles d'un objet.

2. Pourquoi les additionner ? L'analogie du traitement du signal

Considère la formule explicite de Weil comme une formule des traces (type Selberg ou Poisson) :

$$\sum \text{zéros} / \text{spectre} = \text{terme principal (pôle)} + \sum \text{contributions locales (premiers + réels)}$$

- côté spectral (à gauche) : la distribution des zéros de la fonction ζ .
- côté géométrique / géodésique (à droite) :
 - le pôle ($s = 1$) : la composante continue / continue à 0 Hz (le DC offset ou la moyenne globale du signal) ;
 - les nombres premiers : Les “fréquences discrètes” ou les longueurs des orbites périodiques du système ;
 - la partie archimédienne ($\mathbb{R}$) : La régularisation à l'origine / l'effet de bord continu dû à la métrique usuelle.

Additionner ces termes ne revient pas à faire âge + poids, mais à appliquer un théorème de Parseval / Plancherel : l'énergie totale dans le domaine temporel (premiers + réel) est égale à l'énergie totale dans le domaine fréquentiel (zéros + pôle).

En résumé

Vision intuitive	Réalité mathématique / informatique
pôle + réels + premiers	même type de variable (mesures adéliques sur $\mathbb{Q}$)
Addition de métriques	Reconstruction d'un signal à partir de toutes ses composantes locales
Mélange arbitraire	Dualité de Fourier (Côté géométrique $\rightleftharpoons$ Côté spectral)

Ce “mélange” est la contrepartie exacte du fait que pour reconstituer un signal propre, il faut additionner sa composante continue (pôle), ses harmoniques discrètes (premiers) et son bruit/lissage continu aux bords (archimédien).

3 Exact dans un seul sens ?

Lors d'une conférence récente d'Alain Connes, que pouvait bien signifier “morphisme exact dans un seul sens” ? Utiliser les bases de données pour comprendre le vocabulaire de Grothendieck

Dans la théorie des topos de Grothendieck, un morphisme géométrique $f : \mathcal{E} \rightarrow \mathcal{F}$ entre deux topos n'est pas un simple foncteur. C'est un couple de foncteurs adjoints :

$$f^* \dashv f_*$$

où $f^* : \mathcal{F} \rightarrow \mathcal{E}$ est l'image réciproque et $f_* : \mathcal{E} \rightarrow \mathcal{F}$ est l'image directe.

Quand Alain Connes mentionne que c'est exact “dans un seul sens”, il fait référence à la dissymétrie dans la définition quant aux propriétés d'exactitude :

1. Le foncteur image réciproque f^* est exact à gauche.

Par définition, f^* conserve les limites finies (on dit qu'il est exact à gauche). De plus, étant un adjoint à gauche, il conserve automatiquement toutes les colimites.

Il préserve donc à la fois les limites finies et toutes les colimites, ce qui en fait un foncteur très régulier qui transporte toute la structure logique élémentaire du topos.

2. Le foncteur image directe f_* n'est pas exact à gauche en général

Étant un adjoint à droite, f_* conserve toutes les limites. En revanche, il ne conserve pas les colimites (ni même les colimites finies) en général. Il n'est donc pas exact à droite.

Pourquoi cette dissymétrie ?

Cette définition généralise l'image réciproque et l'image directe des faisceaux induites par une application continue $f : X \rightarrow Y$ entre deux espaces topologiques :

- prendre l'image réciproque d'un faisceau (f^*) préserve les fibres (les stalks), et donc les opérations locales comme les intersections finies ou les réunions.
- prendre la poussée d'un faisceau (f_*) applique une section globale, ce qui ne préserve pas les recouvrements ni les sommes en général (par exemple, les sections d'une somme ne sont pas la somme des sections).

Quand un morphisme géométrique vérifie la condition plus forte où f^* possède lui-même un adjoint à gauche $f_!$ (ce qui rend f^* exact à droite et donc pleinement exact), on parle alors de morphisme géométrique essentiel.

En mathématiques, on a deux topos $\mathcal{E}$ et $\mathcal{F}$. Le morphisme $f : \mathcal{E} \rightarrow \mathcal{F}$ va de $\mathcal{E}$ vers $\mathcal{F}$:

- $\mathcal{F}$ est l'espace d'arrivée.
- $\mathcal{E}$ est l'espace de départ.

Il faut retenir que :

- l'image réciproque f^* va dans le sens inverse ($\mathcal{F} \rightarrow \mathcal{E}$). Elle tire les données en arrière sans les déformer $\rightarrow$ exacte.
- l'image directe f_* va dans le sens direct ($\mathcal{E} \rightarrow \mathcal{F}$). Elle pousse les données en avant en les compressant/regroupant $\rightarrow$ pas exacte.

En informatique, $\mathcal{E}$ et $\mathcal{F}$ sont deux bases de données.

Reprenons sans métaphore, uniquement avec nos deux bases de données $\mathcal{E}$ et $\mathcal{F}$.

Soit une application f de la base $\mathcal{E}$ vers la base $\mathcal{F}$:

Base E		Base F
Alice	$\xrightarrow{f}$	Projet A
Bob	$\xrightarrow{f}$	Projet A
Charlie	$\xrightarrow{f}$	Projet B

Un morphisme géométrique est le couple de deux opérations de transfert de données : f^* (l'image réciproque) et f_* (l'image directe).

1. l'image réciproque $f^* : \mathcal{F} \rightarrow \mathcal{E}$ (exacte)

f^* prend une condition ou un ensemble dans $\mathcal{F}$ et regarde tous les éléments de $\mathcal{E}$ qui y sont envoyés par f .

Exemple avec une réunion (union) :

Prenons deux ensembles dans $\mathcal{F}$: {Projet A} et {Projet B}.

- la réunion dans $\mathcal{F}$ est {Projet A, Projet B}.
- l'image réciproque par f^* donne les personnes dans $\mathcal{E}$:

$$f^*({\text{Projet A}} \cup {\text{Projet B}}) = \{\text{Alice}, \text{Bob}, \text{Charlie}\}$$

Si on calcule séparément puis qu'on réunit dans $\mathcal{E}$:

- $f^*({\text{Projet A}}) = \{\text{Alice}, \text{Bob}\}$
- $f^*({\text{Projet B}}) = \{\text{Charlie}\}$
- la réunion des deux donne {Alice, Bob, Charlie}.

f^* préserve les réunions et les intersections : il ne déforme pas les données. On dit qu'il est exact.

2. l'image directe $f_* : \mathcal{E} \rightarrow \mathcal{F}$ (non exacte)

f_* prend un ensemble dans $\mathcal{E}$ et calcule son "image directe" dans $\mathcal{F}$ (ce qui équivaut à un regroupement/agrégation).

Pour les faisceaux, $f_*(X)$ représente les éléments de $\mathcal{F}$ dont toute la fibre est contenue dans X .

Exemple avec une réunion dans $\mathcal{E}$:

Prenons deux groupes de personnes dans $\mathcal{E}$:

- $X = \{\text{Alice}\}$
- $Y = \{\text{Bob}\}$

- (a) si on réunit d'abord dans $\mathcal{E}$ puis qu'on applique $f_* : X \cup Y = \{\text{Alice}, \text{Bob}\}$. Le projet A est entièrement couvert par cette réunion, donc :

$$f_*(X \cup Y) = \{\text{Projet A}\}$$

(b) si on applique f_* séparément puis qu'on réunit dans $\mathcal{F}$:

- $f_*(X)$: Alice seule ne couvre pas tout le Projet A (Bob manque), donc $f_*(X) = \emptyset$.
- $f_*(Y)$: Bob seul ne couvre pas tout le Projet A, donc $f_*(Y) = \emptyset$.
- la réunion dans $\mathcal{F}$ donne $\emptyset \cup \emptyset = \emptyset$.

$f_*(X \cup Y) \neq f_*(X) \cup f_*(Y)$. L'opération f_* ne préserve pas les réunions : elle n'est pas exacte.

4 S'initier à la théorie des catégories par les bases de données relationnelles

4.1 Introduction et démarche

La théorie des catégories est souvent qualifiée d'abstraite ("la métamathématique de la métamathématique"). Cependant, une **base de données relationnelle** fournit une concrétisation visuelle et très intuitive de ses concepts fondamentaux.

En reliant les deux domaines, on s'aperçoit qu'un schéma de base de données est une représentation directe d'une catégorie, et que les données réelles en constituent une instance fonctionnelle.

4.2 Exemple concret : gestion de commandes

Considérons une structure relationnelle classique comportant trois tables :

- *client* : les informations sur les clients.
- *commande* : les commandes passées par un client.
- *lignedunecommande* : le détail des produits associés à une commande.

4.2.1 Code SQL (définition du schéma)

En SQL, la structure du schéma est établie par les déclarations de **clés étrangères** (*Foreign Keys*) :

```
- 1. Table des Clients
CREATE TABLE Client (
    client_id INT PRIMARY KEY,
    nom VARCHAR(100)
);

- 2. Table des commandes (reliee au client)
CREATE TABLE commande (
    commande_id INT PRIMARY KEY,
    client_id INT NOT NULL,
    date_commande DATE,
    FOREIGN KEY (client_id) REFERENCES Client(client_id)
);

- 3. Table des lignes de commande (reliee a la commande)
CREATE TABLE linedunecommande (
    ligne_id INT PRIMARY KEY,
    commande_id INT NOT NULL,
    produit VARCHAR(100),
    quantite INT,
```

```

    FOREIGN KEY (commande_id) REFERENCES commande(commande_id)
);

```

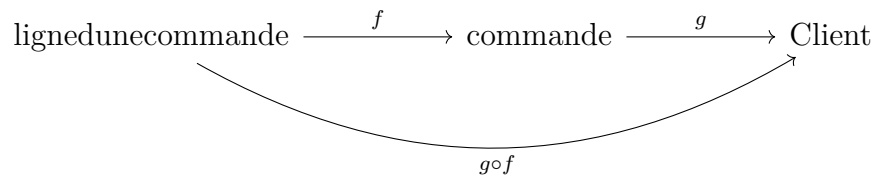
4.3 La catégorie schéma $\mathcal{C}$

Dans le langage des catégories, ce schéma définit une catégorie $\mathcal{C}$ (ou un graphe orienté muni de lois de composition) :

- **les objets** : les tables (lignedunecommande, commande, Client).
- **les flèches (morphisms)** : les **clés étrangères**, orientées de la table fille vers la table mère (chaque ligne fille pointant de façon unique vers une ligne mère).

4.3.1 Diagramme commutatif

Le diagramme suivant illustre les objets, les flèches primitives (clés étrangères) et leur composition :



- $f : \text{lignedunecommande} \rightarrow \text{commande}$ représente la clé étrangère `commande_id`.
- $g : \text{commande} \rightarrow \text{Client}$ représente la clé étrangère `client_id`.
- $g \circ f : \text{lignedunecommande} \rightarrow \text{Client}$ représente la composition des relations (savoir quel client est associé à quelle ligne de commande).

4.3.2 Correspondances systématiques

Base de données	Théorie des catégories	Signification pratique
Table	Objet	Type d'entité / Ensemble sous-jacent.
Clé étrangère	Flèche (Morphism)	Fonction $A \rightarrow B$ associant à chaque élément de A exactement un élément de B .
Jointure	Composition ($g \circ f$)	Traversée de relations successives (JOIN).
Clé Primaire	Identité (id_A)	Flèche $\text{id}_A : A \rightarrow A$ associant chaque élément à lui-même.

Dictionnaire entre le modèle relationnel et la théorie des catégories.

4.4 L'instance de données : le foncteur $D : \mathcal{C} \rightarrow \mathbf{Set}$

Le schéma abstrait $\mathcal{C}$ décrit la structure. L'instanciation de données réelles correspond à un *foncteur* D allant de la catégorie schéma $\mathcal{C}$ vers la catégorie des ensembles **Set** :

4.4.1 Application sur les objets (tables $\rightarrow$ ensembles)

- $D(\text{dclient}) = \{\text{Alice}, \text{Bob}\}$
- $D(\text{commande}) = \{C_1, C_2\}$
- $D(\text{lignedunecommande}) = \{L_1, L_2, L_3\}$

4.4.2 Application sur les flèches (clés étrangères $\rightarrow$ fonctions)

- $D(g) : D(\text{commande}) \rightarrow D(\text{Client})$

$C_1 \mapsto \text{Alice}$

$C_2 \mapsto \text{Bob}$

- $D(f) : D(\text{lignedunecommande}) \rightarrow D(\text{commande})$

$L_1 \mapsto C_1$

$L_2 \mapsto C_1$

$L_3 \mapsto C_2$

5 Équivalent de la composition en SQL

La composition $g \circ f$ calculée au niveau fonctionnel par $(D(g) \circ D(f))(L_1) = D(g)(C_1) = \text{Alice}$ correspond exactement à la requête SQL de jointure suivante :

```
SELECT
    linedunecommande.ligne_id ,
    client.nom
FROM linedunecommande
JOIN commande ON linedunecommande.commande_id = commande.commande_id
JOIN client ON commande.client_id = client.client_id;
```