INFORMATION, CONCEPTS ET ENSEMBLES

A Les ensembles, les types et la logique

Comme mentionné dans l'introduction de l'informatique commune, l'informatique est la construction de l'information par le calcul. Mais comment peut-on calculer l'information, au-delà de l'information purement numérique ? Calculer sur les entiers est une chose, calculer une information en général en est une autre.

Le deux pierres angulaires aux fondements de l'informatique sont la théorie des ensembles et la logique. La fin du XIX^{e} siècle a marqué un tournant dans l'histoire des sciences et en particulier pour les mathématiques : la théorie des ensembles et ses paradoxes ont forcé les mathématiciens à mieux formaliser les raisonnements et les démonstrations, c'est-à-dire les types et la logique.

Lorsque l'être humain analyse l'information, il fait des regroupements : regrouper les informations d'un même type permet de trouver des propriétés à l'ensemble, de les manipuler par lot, de leur appliquer un même traitement : c'est une abstraction très efficace pour le calcul.

■ **Exemple 1 — Les ingrédients d'une recette de cuisine.** Les ingrédients d'une recette de cuisine forment un concept intéressant : on peut les classer (par texture, goût, couleur), les cuire, les mesurer, les combiner. Et même si un ingrédient est un concept qui n'a pas de réalité [a], les ingrédients forment un ensemble intéressant car on peut le construire : avec de la moutarde, un jaune d'œuf, du sel, du poivre et un mélange adapté, on construit une mayonnaise. La mayonnaise elle-même est un ingrédient : on peut l'utiliser pour élaborer d'autres ingérdients comme une salade de légumes par exemple. Cette salade pourra être incorporée dans un wrap... On en déduit qu'un ingrédient est lui-même un ingrédient, c'est un type récursif ! Ce type d'ensemble est nommé **ensemble inductif** et, si on prend la peine de réfléchir au monde qui nous entoure, il est très présent dans notre réalité.

a. un œuf n'existe pas, mais l'œuf qu'une poule a pondu sous vos yeux existe... De la même manière, une voiture est une abstraction : il n'existe que des modèles construits comme la 206 GTI ou la corolla break 2021 hybride.

a Ensembles inductifs

Les **ensembles inductifs** sont à la base du développement de l'informatique. C'est pourquoi ils irriguent toutes les parties du programme officiel de l'option informatique et donc tous les développement de ce cours. Ils ont été formalisés dans la théorie des types par Russel au début du XX^{e} siècle, ouvrant ainsi la voie aux systèmes formels. Pour ces ensembles inductifs, on dispose de **types de base** et de **constructeurs** qui permettent de créer des types à partir

d'autres types. Tout objet d'un ensemble inductif est d'un certain type et il existe un ordre lié à la construction de l'ensemble. C'est ce qui différencie cette approche de la théorie des ensembles. Le programme de l'option informatique aborde des structures de données qui sont des ensembles inductifs en première et deuxième année dont les listes chaînées, les arbres, formules logiques, les expressions régulières.

■ **Exemple 2 — Définir l'ensemble des ingrédients de manière inductive.** Pour nos ingrédients, les termes de base pourraient être le sel, le poivre, l'eau, la farine, le beurre, le sucre, le jaune et le blanc d'œuf. Les règles de construction : prendre, mélanger, étaler, broyer, fondre, cuire.

Formellement, on pourrait définir l'ensemble des ingrédients $\mathcal{I}$ de manière inductive de la façon suivante :

Termes de base $\mathcal{B} = \{\text{sel, poivre, eau, farine, beurre, sucre, jaune d'œuf, blanc d'œuf}\}$

Constructeur (FONDRE) $\forall i \in \mathcal{I}, \text{FONDRE}(i) \in \mathcal{I}$

Constructeur (MÉLANGER) $\forall i_1, i_2 \in \mathcal{I}, \text{MÉLANGER}(i_1, i_2) \in \mathcal{I}$

En appliquant le constructeur FONDRE à BEURRE, on obtient FONDRE(BEURRE), du beurre fondu. En appliquant les constructeurs :
MÉLANGER(SUCRE, FONDRE(BEURRE)),
on obtient un nouvel ingrédient qui nous permettra de créer un Kouign-amann.

R Les éléments d'un ensemble inductif sont appelés **termes**.

R Un constructeur possède une certaine **arité** : il permet de construire un terme à partir de un ou plusieurs termes. Dans l'exemple des ingrédients, FONDRE est un constructeur d'arité 1 car il ne prend qu'un seul paramètre. MÉLANGER en prend deux, il est d'arité 2. On ne peut donc pas mélanger du beurre, mais on peut mélanger du beurre et du sucre.

R On observe qu'il y a un **ordre dans les termes** que l'on peut construire à partir de la définition inductive des ingrédients : par exemple, la mayonnaise est un ingrédient nécessaire à l'élaboration d'un sandwich au crabe et le sel est nécessaire à l'élaboration de la mayonnaise. On dit que le sel est un sous-terme de la mayonnaise et que la mayonnaise est un sous-terme du sandwich au crabe. D'une manière générale, $t = \text{CONSTRUCTEUR}(t_1, t_2, \ldots, t_n)$ signifie que l'on construit le terme t à partir des sous-termes $t_1, t_2, \ldots, t_n$. **Cet ordre est appelé ordre structurel.**

b Propriétés des ensembles inductifs

Les ensembles inductifs ont des propriétés qu'il est possible de démontrer grâce à l'**induction structurelle**.

■ **Définition 1 — Principe d'induction structurelle.** Soit $\mathcal{I}$ un ensemble inductif de termes

de base $\mathcal{B}$ et de constructeurs $\mathcal{C}$. Soit $\mathcal{P}$ une propriété sur les termes de $\mathcal{I}$.

Si $\mathcal{P}$ est satisfaite pour chaque terme de base de $\mathcal{B}$ et pour chaque constructeur de $\mathcal{I}$, alors $\mathcal{P}$ est satisfaite pour tous les termes de $\mathcal{I}$.

Plus formellement,

$$\left.\begin{array}{ll} \forall b \in \mathcal{B}, & \mathcal{P}(b) \\ \forall c \in \mathcal{C}, \forall t_1, t_2, \dots t_n \in \mathcal{I}, & \mathcal{P}(c(t_1, t_2, \dots, t_n)) \end{array}\right\} \Longrightarrow \forall t \in \mathcal{I}, \mathcal{P}(t) \tag{1}$$

■ **Exemple 3 — Comestible.** Soit $\mathcal{I}$ l'ensemble inductif des ingrédients. Soit $\mathcal{P}$ la propriété *est comestible*. On cherche à montrer que tous les ingrédients sont comestibles par induction structurelle [a].

On sait que tous les ingrédients de base sont comestibles. Par ailleurs, faire fondre un ingrédient ne dégrade pas cette propriété, il est toujours comestible une fois fondu. Enfin, lorsqu'on mélange deux ingrédients comestibles, le résultat est comestible. C'est pourquoi, tous les ingrédients sont comestibles.

a. Attention, il ne s'agit que d'une modélisation des ingrédients limitée à des objets de base comestibles. Ne pas extrapoler le résultats ;-)

c Des fonctions pour calculer sur les termes d'un ensemble inductif

On peut facilement définir des fonctions sur les ensembles inductifs en s'appuyant sur leur définition. Cela permet de faire des calcul sur l'information qu'ils représentent.

■ **Exemple 4 — Masse d'un ingrédient.** On peut définir la masse d'un ingrédient en s'appuyant sur la définition inductive d'un ingrédient : la masse d'un ingrédient est la somme des masses des ingrédients qui la compose. Pour les ingrédients de base, on peut considérer que la masse d'un ingrédient $b \in \mathcal{B}$ est une constante connue m_b que l'on a mesurée.

On obtient alors une fonction formulée récursivement :

Cas de base $\forall b \in \mathcal{B}, \text{MASSE}(b) = m_b$

Construction $\forall i \in \mathcal{I}, \text{MASSE}(\text{FONDRE}(i)) = \text{MASSE}(i)$

Construction $\forall i_1, i_2 \in \mathcal{I}, \text{MASSE}(\text{MÉLANGER}(i_1, i_2)) = \text{MASSE}(i_1) + \text{MASSE}(i_2)$

En utilisant cette définition inductive, on peut calculer la masse de tous les ingrédients.

B ℕ : paradigme d'un ensemble inductif dont l'ordre est bien fondé

Les entiers naturels, les éléments de l'ensemble ℕ, constituent le fondement de l'informatique. Or, on ne peut pas construire cet ensemble des nombres entiers naturels, ensemble pourtant constitué des objets mathématiques les plus évidents : tout le monde comprend les concepts 0, 1, 2, construire l'ensemble $\{0, 1, 2, \dots\}$ nécessite d'expliciter les «…», ce qui n'est pas simple a priori. C'est pourquoi on postule son existence.

Les premières définition axiomatiques de ℕ apparaissent à la fin du XIX$^{\text{e}}$ siècle, sous l'impulsion de Dedekind et Peano.

■ **Définition 2 — Définition axiomatique de l'ensemble des entiers naturels.** On postule qu'il existe un ensemble dit des «entiers naturels» noté $\mathbb{N}$ muni de :

1. l'élément zéro, noté 0 est un entier naturel.
2. une application «successeur» $s : \mathbb{N} \to \mathbb{N}$, vérifiant les axiomes de Peano suivants :
   (a) s est injective,
   (b) Zéro n'est le successeur d'aucun entier naturel,
   (c) (Axiome de récurrence) Soit A un sous-ensemble de $\mathbb{N}$ tel que :
   - $0 \in \mathrm{A}$,
   - $\forall n \in \mathrm{A}, s(n) \in A$ (stabilité de l'ensemble),

   alors $\mathrm{A} = \mathbb{N}$.

■ **Définition 3 — Addition.** L'opération addition sur les entiers naturels est définie par :

(B) $a + 0 = a$

(I) $a + s(b) = s(a + b)$

■ **Définition 4 — Multiplication.** L'opération multiplication sur les entiers naturels est définie par :

(B) $a \times 0 = 0$

(I) $a \times s(b) = a \times b + a$

R On peut vérifier que l'addition et la multiplication sont commutatives et que la multiplication est distributive par rapport à l'addition. On en déduit d'une relation d'ordre $\leqslant$ sur $\mathbb{N}$ compatible avec ces opérations :

$$\forall (m, n) \in \mathbb{N}^2, m \leq n \iff \exists k \in \mathbb{N}, n = m + k$$

R En notant $1 = s(0)$, on peut formellement se débarrasser de la fonction s en la remplaçant par $a \to a + 1$. On simplifie alors les démonstrations.

Théorème 1 Toute partie non vide de N admet un plus petit élément.

Démonstration. Par l'absurde. Soit A une partie non vide de $\mathbb{N}$ n'admettant pas de plus petit élément. Considérons l'ensemble B défini par $B = \{n \in \mathbb{N}, \forall k \leq n, k \notin A\}$.

- Zéro est dans B, car dans le cas contraire il serait dans A et constituerait un plus petit élément de A, ce qui contredit notre hypothèse.
- Soit n, un élément quelconque de B. On cherche à montrer que $n + 1$ est dans B. On procède par l'absurde. Supposons que $n + 1$ appartient à A. Comme $A \in \mathbb{N}$, A contient l'ensemble de ses successeurs et il existe un $k \in \mathbb{N}$ tel que $k \leqslant n$, c'est-à-dire $n + 1$ est le successeur de k dans A. Mais on a nécessairement $k = n + 1$, sinon k appartiendrait à B,

par définition de B. On a donc :

$$\forall k \leqslant n, k \notin A \text{ et } n+1 \in A$$

ce qui signifie $\forall i \in A, i \leqslant n+1$, c'est-à-dire tous les éléments de A sont plus grands que $n+1$. Donc A possède un plus petit élément, ce qui est absurde. Donc $n+1$ appartient à B.

L'ensemble B vérifie donc le principe de récurrence et on a $B = \mathbb{N}$. On en déduit que $A = \emptyset$, ce qui est absurde puisqu'on a supposé que A n'était pas vide. ■

Le premier principe d'induction est un principe fondamental en informatique : au-delà de son utilisation en mathématique, ce principe permet de calculer une information à partir d'une information initiale et d'une règle de construction.

Théorème 2 — Principe d'induction. Soit $\mathcal{P}_n$ une propriété fonction d'un entier naturel n. Lorsque les deux conditions suivantes sont vérifiées :

(B) $\mathcal{P}_0$ est vraie,

(I) $\mathcal{P}_n$ implique $\mathcal{P}_{n+1}$ pour tout n,

alors pour tout entier naturel n, $\mathcal{P}_n$ est vraie.

Démonstration. Par l'absurde. Soit $\mathcal{P}_n$ une propriété fonction d'un entier naturel n vérifiant les conditions (B) et (I). Supposons qu'il existe des entiers k pour lesquels la proposition $\mathcal{P}_k$ soit fausse et notons l'ensemble de ces entiers X : $X = \{k \in \mathbb{N}, \mathcal{P}_k \text{ est faux}\}$.

Si X est vide, alors Pc_n est vraie pour pour tout entier naturel.

Si X est non vide, il admet un plus petit élément v d'après le théorème 1.

- soit $v = 0$ et alors $\mathcal{P}_0$ est fausse, ce qui contredit la propriété (B) et notre hypothèse. C'est absurde.
- soit $v > 0$, $\mathcal{P}_v$ est fausse et $\mathcal{P}_{v-1}$ est vraie. Donc (I) n'est pas vérifiée par $\mathcal{P}_n$, ce qui contredit notre hypothèse. C'est absurde.

On en conclut que $\mathcal{P}_n$, à partir du moment où elle vérifie les propriétés (B) et (I), est vraie pour tout entier naturel n. ■

(R) L'élément qui cristallise l'intérêt des informaticiens dans le principe d'induction, c'est qu'il est constructif : il permet, à partir d'une information simple (cas de base B) et de règles d'induction simples (I), de calculer un ensemble d'information.

■ **Définition 5 — Ordre bien fondé.** Soit $(E, \leqslant)$ un ensemble muni d'une relation d'ordre. On dit que l'ordre sur E est bien fondé s'il n'existe pas de suite infinie strictement décroissante d'éléments de E.

■ **Définition 6 — Ensemble bien ordonné.** Soit $(E, \leqslant)$ un ensemble muni d'une relation d'ordre, c'est-à-dire un ensemble ordonné. On dit que l'ordre sur E est bien ordonné s'il est également bien fondé.

R Un ensemble bien ordonné est nécessairement doté d'une relation d'ordre totale.

Théorème 3 — Caractérisation des ordres bien fondés. Soit Soit $(E, \leqslant)$ un ensemble. L'ordre de E est bien fondé si et seulement si toute partie non vide F de E admet un élément minimal.

R On déduit des théorèmes 3 et 1 que l'ordre $\leqslant$ associé à l'ensemble $\mathbb{N}$ des entiers naturels est bien fondé.

R $(\mathbb{N}, \leqslant)$ est un ensemble bien ordonné mais $(\mathbb{Z}, \leqslant)$ ne l'est pas.

C De l'induction à l'induction structurelle

Cette section étend le principe d'induction à tout ensemble construit de manière inductive et bien ordonné.

■ **Définition 7 — Définition inductive d'un ensemble.** Soit E un ensemble. Une définition inductive d'une partie X de E consiste à se donner :

(B) un sous ensemble non vide $\mathcal{B}$ de E appelé ensemble de base,

(I) et d'un ensemble de règles $\mathcal{R}$: chaque règle $r_i \in \mathcal{R}$ est une fonction r_i de $E^{n_i} \to E$ telle que $\forall x_1, \cdots x_{n_i} \in X \Rightarrow r_i\left(x_1, \cdots, x_{n_i}\right) \in X$ pour $n_i > 1$.

On dit que X est alors défini inductivement. On appelle les r_i les constructeurs de X.

Théorème 4 — Théorème du point fixe. À une définition inductive d'un ensemble correspond un plus petit ensemble qui vérifie les propriétés suivantes :

1. il contient $\mathcal{B}$, c'est-à-dire $\mathcal{B} \subset X$,
2. il est stable pour les règles de $\mathcal{R}$: pour chaque règle $r_i \in \mathcal{R}$, pour tout $x_1, \cdots, x_{n_i} \in X$, on a $r_i\left(x_1, \cdots, x_{n_i}\right) \in X$.

Démonstration. Soit E un ensemble défini inductivement comme en 7 à l'aide d'un ensemble de base $\mathcal{B}$ et des règles $\mathcal{R}$.

Soit $\mathcal{P}$ l'ensemble des parties de E vérifiant (B) et (I).

$\mathcal{P}$ est non vide car il contient au moins l'ensemble E qui vérifie les conditions (B) et (I) : $E \in \mathcal{P}$.

Considérons alors X l'ensemble défini comme l'intersection de tous les éléments de $\mathcal{P}$:

$$X = \bigcap_{Y \in \mathcal{P}} Y$$

Par définition, $\mathcal{B}$ est inclus dans chaque $Y \in \mathcal{P}$. $\mathcal{B}$ est donc inclus dans X et X vérifie la condition (B).

On peut montrer que X vérifie également la condition (I). Soit une règle $r_i \in \mathcal{R}$ et des $x_1, \cdots, x_{n_i} \in X$. On a $x_1, \cdots, x_{n_i} \in Y$ pour chaque $Y \in \mathcal{P}$. Pour chaque tel Y, puisque Y est stable par la règle r_i, on doit avoir $r\left(x_1, \cdots, x_{n_i}\right) \in Y$. Puisque cela est vrai pour tout $Y \in \mathcal{P}$, on a aussi $r\left(x_1, \cdots, x_{n_i}\right) \in X$. Donc X est stable par la règle r_i.

Enfin, X est le plus petit ensemble qui vérifie les conditions (B) et (I), car, de par sa définition, il est inclus dans tout autre ensemble vérifiant les conditions (B) et (I). ■

R Puisqu'on sait qu'il existe, on considère alors **le plus petit de ces ensembles** défini inductivement qui contient $\mathcal{B}$ et qui est stable par les règles de construction $\mathcal{R}$.

En effet, si on définit l'ensemble des entiers pairs P par 0 et la règle $\forall n \in P, n+2 \in P$, alors on constate que $\mathbb{N}$ vérifie bien ces deux propriétés. Néanmoins, ce n'est pas le plus petit des ensembles caractérisés par cette définition. L'ensemble des nombres pairs est donc le plus petit de ces ensembles définis inductivement.

■ **Définition 8 — Principe d'induction structurelle.** Soit E un ensemble **muni d'un ordre bien fondé** défini **inductivement**, de termes de base $\mathcal{B}$ et de règles de construction $\mathcal{R}$

Soit $\mathcal{P}$ une propriété sur les termes de E.

Si $\mathcal{P}$ est satisfaite pour chaque terme de base de $\mathcal{B}$ (cas de base) et pour chaque constructeur de $\mathcal{R}$ (pas d'induction), alors $\mathcal{P}$ est satisfaite pour tous les termes de E.

Plus formellement,

$$\left.\begin{array}{ll} \forall b \in \mathcal{B}, & \mathcal{P}(b) \\ \forall r \in \mathcal{R}, \forall x_1, x_2, \ldots x_n \in E, & \mathcal{P}(r(x_1, x_2, \ldots, x_n)) \end{array}\right\} \Longrightarrow \forall x \in E, \mathcal{P}(x) \qquad (2)$$

Théorème 5 — Fonction définie inductivement. Soit $X \subset E$ un ensemble défini inductivement de façon non ambigüe à partir de l'ensemble de base $\mathcal{B}$ et des règles $\mathcal{R}$. Soit Y un ensemble.

Pour qu'une application f de X dans Y soit parfaitement définie, il suffit de se donner :

(B) la valeur de $f(x)$ pour chacun des éléments $x \in B$,

(I) pour chaque règle $r_i \in \mathcal{R}$, la valeur de $f(x)$ pour $x = r_i\left(x_1, \cdots, x_{n_i}\right)$ en fonction de la valeur $x_1, \ldots, x_{n_i}, f(x_1), \ldots$ et $f\left(x_{n_i}\right)$.

Démonstration. On cherche à montrer que la fonction f ainsi définie associe bien une unique valeur dans Y à chaque $x \in X$. On procède par induction.

- Cas de base : la valeur associée à chaque $x \in \mathcal{B}$ est bien unique d'après (B).

- Pas d'induction : supposons que cela est vrai pour $x_1, \cdots, x_{n_i}$. On cherche à montrer que cela est vrai en $x = r_i\left(x_1, \cdots, x_{n_i}\right)$, c'est-à-dire que les $f(x_1), \ldots, f(x_{n_i})$ sont définis de manière unique. Comme x ne peut être obtenu que par la règle r_i à partir de $x_1, \ldots, x_{n_i}$, $f(x)$ est donc parfaitement définie par la contrainte pour la règle (I).

■

R Ce théorème est fondamental car il affirme que si l'on sait programmer récursivement, c'est-à-dire en utilisant la fonction elle-même dans la définition de la fonction, alors une fonction peut être parfaitement définie par 5 sur un ensemble X défini inductivement.

■ **Définition 9 — Factorielle, définie inductivement.** Comme on a définit l'ensemble des entiers naturels $\mathbb{N}$ inductivement (cf. définition 2), la fonction factorielle peut être définie inductivement par :

Fact : $\mathbb{N} \to \mathbb{N}$

(B) $\text{Fact}(0) = 1$,

(I) $\text{Fact}(s(n)) = s(n) \times \text{Fact}(n)$, ce qui peut également s'écrire : $\text{Fact}(n+1) = (n+1) \times \text{Fact}(n)$.

D Pourquoi OCaml ?

Comme on peut le voir ci-dessous et comme on le verra dans la section suivante, le langage OCaml est particulièrement adapté aux ensembles inductifs pour plusieurs raisons :

1. les types en OCaml sont nativement récursifs,
2. les types OCaml peuvent être des types somme `|` ou produit `*`. On les appelle des types algébriques.
3. OCaml procure la syntaxe dite du filtrage de motifs,
4. OCaml permet d'écrire des fonctions récursives.

Combiner ces quatre fonctionnalités permet de traduire directement la plupart des concepts mathématiques exprimés sous la forme d'un ensemble inductif. Les **fonctions** dont les paramètres peuvent être des types algébriques concrétisent les calculs sur les termes des ensembles inductifs.

Ces fonctionnalités de OCaml [1] expliquent en grande partie le choix du langage OCaml en CPGE pour l'option informatique.

Vocabulary 1 — Pattern matching ⇜⇝ Filtrage de motifs

```
type ingredient =
        | Sel | Poivre | Beurre | Sucre | Farine
        | Fondre of ingredient
```

1. Tous les langages ne dispose pas de ces fonctionnalités : Python ne dispose pas de types récursifs et algébriques et son paradigme est impératif, pas fonctionnel.

```
        | Melanger of ingredient*ingredient;;

let rec masse = function
        | Sel -> 30
        | Poivre -> 10
        | Beurre -> 250
        | Sucre -> 250
        | Farine  -> 10
        | Fondre i ->  masse i
        | Melanger (i1,i2) -> (masse i1) + (masse i2);;

let kouignamann = Melanger(Beurre, Sucre);;
let m = masse(kouignamann);;
```