Ordre et induction

Guillaume Rousseau
MP2I Lycée Pierre de Fermat
guillaume.rousseau@ens-lyon.fr

28 janvier 2026

1 Relations, ordres

A Définitions

Définition 1

Soit X un ensemble. Une ***relation binaire*** sur X est un ensemble $\mathcal{R} \subseteq X^2$. Si $x, y \in X$ sont tels que $(x, y) \in \mathcal{R}$, on dit que x et y vérifient la relation $\mathcal{R}$. On note cela $x\mathcal{R}y$.

Exemple 1

- $\{(x, y) \in \mathbb{R}^2 \mid x = y\}$ est une relation binaire sur $\mathbb{R}$
- $\{(u, v) \in (\{a, b, ..., z\}^*)^2 \mid u.v = u.v\}$ est une relation binaire sur $\{a, b, ..., z\}^*$, elle met en relation tous les mots sur cet alphabet qui commutent.
- $\{(x, y) \in \mathbb{R}^2 \mid x < y\}$ est une relation binaire sur $\mathbb{R}$
- Pour X un ensemble et $f : X \to X$ une fonction, $\{(x, y) \in X^2 \mid y = f(x)\}$ est une relation binaire sur X.

Définition 2

Soit X un ensemble et $\mathcal{R}$ une relation binaire sur X. On dit que :

- $\mathcal{R}$ est réflexive si $\forall x \in X, x\mathcal{R}x$
- $\mathcal{R}$ est symétrique si $\forall x, y \in X, x\mathcal{R}y \Rightarrow y\mathcal{R}x$
- $\mathcal{R}$ est antisymétrique si $\forall x, y \in X, x\mathcal{R}y$ et $y\mathcal{R}x \Rightarrow x = y$
- $\mathcal{R}$ est transitive si $\forall x, y, z \in X, x\mathcal{R}y$ et $y\mathcal{R}z \Rightarrow x\mathcal{R}z$

Exercice 1

Pour chacune des 4 relations de l'exemple précédent, dire si elle est réflexive, symétrique, antisymétrique, transitive.

Représentation graphique Une relation $\mathcal{R}$ sur un ensemble X fini peut se représenter graphiquement par ce que l'on appelle un **graphe orienté** (voir chapitre 13). On représente chaque élément de X par un point dans l'espace, et on trace une flèche d'un point x_1 à un point x_2 pour indiquer $x_1\mathcal{R}x_2$. Par exemple, voici le graphe de la relation "est divisible par" sur l'ensemble $\{0, 1, 2, 3, 4, 5, 6\}$:

Exercice 2

Comment se traduit sur le graphe d'une relation $\mathcal{R}$ le fait qu'elle soit réflexive, symétrique, antisymétrique, ou transitive ?

Définition 3

Soit X un ensemble et $\mathcal{R}$ une relation binaire sur X.

— On dit que $\mathcal{R}$ est une relation ***d'équivalence*** si elle est réflexive, symétrique et transitive.

— On dit que $\mathcal{R}$ est une relation ***d'ordre*** si elle est réflexive, antisymétrique et transitive.

B Clôtures

Définition 4

Soit X un ensemble et $\mathcal{R}, \mathcal{R}'$ deux relations binaires sur X. On note $\mathcal{R}' \circ \mathcal{R}$ la relation suivante :

$$\forall x, z \in X, x\mathcal{R}' \circ \mathcal{R} z \Leftrightarrow \exists y \in X, x\mathcal{R}' y \text{ et } y\mathcal{R} z$$

On parle de ***composition*** de relations. La composition est associative, et pour $n \in \mathbb{N}$, on note $\mathcal{R}^n = \mathcal{R} \circ \cdots \circ \mathcal{R}$ la composée n fois de $\mathcal{R}$ avec elle-même.

Proposition 1

Pour $\mathcal{R}$ une relation binaire sur un ensemble X et $n \in \mathbb{N}$, $x\mathcal{R}^n y$ si, et seulement si, il existe $x_1, \dots x_{n-1} \in \mathbb{R}$ tels que $x\mathcal{R}x_1\mathcal{R}\dots\mathcal{R}x_{n-1}\mathcal{R}y$.

Remarque 1

En reprenant la représentation graphique présentée plus haut, la relation $\mathcal{R}^n$ représente donc les éléments x, y tels qu'il existe un chemin d'exactement n flèches de x à y dans le graphe de la relation $\mathcal{R}$.

Définition 5

Soit X un ensemble et $\mathcal{R}$ une relation binaire sur X. On note :

— $\mathcal{R}_r = \mathcal{R} \cup \{(x, x) \mid x \in X\}$ la clôture réflexive de $\mathcal{R}$

— $\mathcal{R}_s = \mathcal{R} \cup \{(y, x) \mid x\mathcal{R}y\}$ la clôture symétrique de $\mathcal{R}$

— $\mathcal{R}_t = \bigcup_{n \in \mathbb{N}^*} \mathcal{R}^n$ la clôture transitive de $\mathcal{R}$

Exercice 3

Donner une interprétation des trois relations de la définition précédente dans le graphe de la relation $\mathcal{R}$.

Proposition 2

Soit X un ensemble et $\mathcal{R}$ une relation binaire sur X.

- $\mathcal{R}_r$ est la plus petite relation réflexive contenant $\mathcal{R}$
- $\mathcal{R}_s$ est la plus petite relation symétrique contenant $\mathcal{R}$
- $\mathcal{R}_t$ est la plus petite relation transitive contenant $\mathcal{R}$

C Relations d'ordre

Définition 6

On dit qu'un ordre $(X, \leq)$ est ***total***, ou que X est ***totalement ordonné***, si $\forall x, y \in X$, $x \leq y$ ou $y \leq x$, autrement dit si deux éléments sont toujours comparables.

Définition 7

Soit $(X, \leq)$ un ensemble ordonné et $Y \subseteq X$.

- Soit $x \in X$. On dit que x est un minorant (resp. majorant) de Y si $\forall y \in Y, x \leq y$ (resp. $x \geq y$).
- Soit $x \in X$. x est un plus petit (resp. plus grand) élément de Y si x est un minorant (resp. majorant) de Y et $x \in Y$.
- Soit $y \in Y$. y est ***minimal*** dans Y s'il n'y a pas d'autre élément plus petit, i.e. si :

$$\forall y' \in Y, y' \leq y \Rightarrow y' = y$$

- Soit $y \in Y$. y est ***maximal*** dans Y s'il n'y a pas d'autre élément plus grand, i.e. si :

$$\forall y' \in Y, y' \geq y \Rightarrow y' = y$$

Attention à ne pas confondre un plus petit élément et un élément minimal. Un plus petit élément est **plus petit que tout le monde**, un élément minimal n'est **plus grand que personne**. Pour les ordres totaux les deux coïncident, mais ce n'est pas le cas en général.

Exercice 4

Pour les ensembles ordonnés suivants, donnez les éléments minimaux, et les plus petits éléments :

- $\mathbb{N}$ muni de l'ordre $\leq$ naturel
- $\mathbb{R}^{+*}$ muni de l'ordre $\leq$ naturel
- $(\mathbb{N} \setminus \{1\}, |)$
- $\mathcal{P}(X) \setminus \{\emptyset\}$ muni de l'inclusion $\subseteq$, où X est un ensemble quelconque.

Proposition 3

Soit $(X, \leq)$ un ensemble ordonné et $Y \subseteq X$ une partie admettant un plus petit élément. Alors, ce plus petit élément est unique, et on le note $\min(Y)$

Démonstration. Si y, y' sont deux plus petits éléments de Y, alors $y \in Y$ et $y' \in Y$. Donc, $y \leq y'$ et $y' \leq y$, et par antisymétrie, $y = y'$. D'où l'unicité. □

Proposition 4

Soit $(X, \leq)$ un ensemble **totalement** ordonné et $Y \subseteq X$, alors, tout élément minimal de Y est un plus petit élément de Y. En particulier, Y n'admet qu'au plus un élément minimal.

Démonstration. Soit y élément minimal de Y. Alors $y \in Y$, et si $z \in Y$
y, alors z n'est pas inférieur à y, et donc comme l'ordre est total, $y < z$. Donc, $y \leq z$ pour tout $z \in Y$ et $y \in Y$: y est un plus petit élément de Y. □

Les propriétés précédentes s'appliquent aussi aux éléments maximaux et aux plus grands éléments.

Définition 8

Soit $(X, \leq)$ un ensemble ordonné et $Y \subseteq X$. Si l'ensemble des minorants de Y admet un plus grand élément, on dit que c'est la borne inférieure de Y. De même, on dit qu'un élément de X est la borne supérieure de Y si c'est le plus petit des majorants de Y (sous réserve d'existence). Lorsque les bornes de Y existent, on les note $\inf(Y)$ et $\sup(Y)$, et elles vérifient donc :

$$\forall m \in X, (\forall y \in Y, m \leq y) \Rightarrow m \leq \inf(Y)$$
$$\forall M \in X, (\forall y \in Y, M \geq y) \Rightarrow M \geq \sup(Y)$$

Une propriété fondamentale de la borne inférieure est que tout autre minorant lui est inférieur. En particulier, si m est un minorant de Y et $m \geq \inf(Y)$ alors $m = \inf(Y)$. Idem pour la borne supérieure.

Exemple 2

On considère $Y = \{\frac{\lfloor 10^n \sqrt{2} \rfloor}{10^n} \mid n \in \mathbb{N}\}$, l'ensemble des écritures décimales tronquées de $\sqrt{2}$. Y admet une borne supérieure dans $\mathbb{R}$, $\sqrt{2}$, mais pas dans $\mathbb{Q}$. Notons que $\sqrt{2} \notin Y$: la borne supérieure d'un ensemble n'en fait pas forcément partie.

Exercice 5

Soit $(X, \leq)$ un ensemble ordonné et Y un ensemble admettant un plus petit élément a. Que dire de la borne inférieure de Y ?

D Ordre lexicographique, ordre produit

Un exemple fondamental d'ordre : l'ordre lexicographique. On rappelle que pour Σ un ensemble fini non vide, on note $\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n$ l'ensemble des mots finis sur Σ. On note ε l'unique élément de Σ^0.

Si $u \in \Sigma^n$, on le note $u = u_1u_2\dots u_n$ et on appelle $u_1, \dots, u_n$ les lettres de u. On note $|u| = n$ sa taille.

On note $u.v$ la concaténation de deux mots $u, v \in \Sigma^*$, c'est à dire le mot w de taille $|u|+|v|$ tel que $w_j = u_j$ pour $1 \leq j \leq |u|$ et $w_j = v_{j-|u|}$ si $j > |u|$.

Définition 9

Si $u, v \in \Sigma^*$ sont tels que $v = u.w$ pour un certain $w \in \Sigma^*$, alors on dit que u est un ***préfixe*** de v. On dit que c'est un préfixe strict si de plus $u \neq v$.

Définition 10

Si $(\Sigma, \leq_\Sigma)$ est un ensemble ordonné, alors on définit l'ordre lexicographique $\leq_{lex}$ sur Σ^* comme suit : soient $u, v \in \Sigma^*$. $u \leq_{lex} v$ si et seulement si l'une des deux conditions suivantes est satisfaite :

- u est un préfixe de v
- Il existe $j \in \mathbb{N}^*$ avec $j \leq |u|, j \leq |v|$ tel que $u_j < v_j$ et $u_k = v_k$ pour tout $k \in [\![1, j-1]\!]$.

L'ordre lexicographique correspond à l'ordre alphabétique : on compare d'abord la première lettre des deux mots, et si elles sont égales on passe à la deuxième, et ainsi de suite.

Exercice 6

Montrer que cette relation est bien un ordre sur Σ^*, que c'est un ordre total, et qu'elle admet le mot vide ε comme plus petit élément.

Exemple 3

Si Σ est l'alphabet latin minuscule, avec l'ordre $a < b < c... < z$ alors l'ordre lexicographique est simplement l'ordre alphabétique.

On a par exemple $a < aa < aaa < a^{1000} < ab < ba$.

Exercice 7

Q1. On considère l'alphabet $\Sigma = \{a, b\}$ avec l'ordre $a < b$. On munit Σ^* de l'ordre lexicogaphique, et on considère $A = \{a^n \mid s \in \mathbb{N}\} \subseteq \Sigma^*$ l'ensemble des mots constitués uniquement de a. A possède-t-il un plus grand élément ? Et une borne supérieure dans Σ^* ?

Définition 11

Soient $(X_1, \leq_1), \dots, (X_n, \leq_n)$ des ensembles ordonnés.

- On appelle également ***ordre lexicographique*** sur $X_1 \times \dots \times X_n$ l'ordre $\leq_{lex}$ suivant : Si $x = (x_1, \dots, x_n)$ et $y = (y_1, \dots, y_n)$ sont des éléments de $X_1 \times \dots \times X_n$, alors $x \leq y$ si $x = y$ ou bien si

$$\exists i \in [\![1, n]\!], x_i <_i y_i \text{ et } \forall j < i, x_j = y_j$$

Par exemple, sur $\mathbb{N} \times \mathbb{N}$, avec cet ordre, on a $(2, 30) < (3, 6)$ car $2 < 3$, et $(2, 30) < (2, 35)$ car $2 = 2$ et $30 < 35$.

— On appelle ***ordre produit*** sur $X_1 \times \cdots \times X_n$ l'ordre $\leq_{prod}$ suivant : Si $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ sont des éléments de $X_1 \times \cdots \times X_n$, alors

$$x \leq_{lex} y \Leftrightarrow \forall i \in [\![1, n]\!], x_i \leq_i y_i$$

Proposition 5

L'ordre produit et l'ordre lexicographique sont des relations d'ordres totales.

E Ordre bien fondé

Dans cette partie, on pose $(X, \leq_X)$ et $(Y, \leq_Y)$ deux ensembles ordonnés. On notera $<_X$ et $<_Y$ les ordres stricts correspondant. Ainsi, pour $x, x' \in X$, $x <_X x' \Leftrightarrow x \leq_X x'$ et $x \neq x'$.

Définition 12

— Une fonction $f : X \to Y$ est ***croissante*** ssi $\forall x_1, x_2 \in X$, $x_1 \leq_X x_2 \Rightarrow f(x_1) \leq_Y f(x_2)$.

— Une fonction $f : X \to Y$ est ***strictement croissante*** ssi $\forall x_1, x_2 \in X$, $x_1 <_X x_2 \Rightarrow f(x_1) <_Y f(x_2)$.

On étend ces définitions aux suites de $X^{\mathbb{N}}$ vues comme fonctions de $(\mathbb{N}, \leq_{\mathbb{N}}) \to (X, \leq_X)$, où $\leq_{\mathbb{N}}$ est l'ordre naturel sur N. Un suite $(x_n)_{n\in\mathbb{N}} \in X^{\mathbb{N}}$ est donc croissante si et seulement $\forall n \leq m \in \mathbb{N}$, $x_n \leq_X x_m$.

Exemple 4

On note $|$ la relation de divisibilité sur $\mathbb{N}$. C'est une relation d'ordre, mais l'ordre n'est pas total : 2 et 3 ne sont pas comparables. On considère les fonction suivante :

$$f : \begin{array}{ccc} (\mathbb{N}^*, \leq) & \longrightarrow & (\mathbb{N}^*, |) \\ x & \longmapsto & x \end{array}$$

$$g : \begin{array}{ccc} (\mathbb{N}^*, |) & \longrightarrow & (\mathbb{N}^*, \leq) \\ x & \longmapsto & x \end{array}$$

f n'est pas croissante, car $2 \leq 3$ mais 2 ne divise pas 3. En revanche, g est croissante : si $x, y \in \mathbb{N}^*$ et $x|y$, alors $x \leq y$.

Exercice 8

Soit $f : X \to Y$ strictement croissante.

Q1. On suppose que l'ordre sur X est total. Montrez que f est injective.

Q2. Trouver un contre-exemple lorsque X n'est pas total.

Définition 13

On dit que l'ordre $\leq_X$ est ***bien fondé*** ssi toute partie $A \subseteq X$ non vide admet un élément minimal.

Proposition 6

Un ordre $(X, \leq_X)$ est bien fondé ssi il n'existe aucune suite strictement décroissante de X^N

Démonstration. Montrons le résultat par double implication. On considère un ordre $(X, \leq_X)$

- $\Rightarrow$ On suppose que $(X, \leq_X)$ est bien fondé. Supposons par l'absurde qu'il existe une suite $(u_n)_{n\in\mathbb{N}} \in X^{\mathbb{N}}$ strictement décroissante. On pose $A = \{u_n | n \in \mathbb{N}\}$. $A \neq \emptyset$ donc A possède un élément minimal. Notons $n_0 \in \mathbb{N}$ tel que $u_{n_0} = \min(A)$. Par décroissance stricte, $u_{n_0+1} < u_{n_0}$, et par minimalité, $u_{n_0} \leq u_{n_0+1}$: d'où la contradiction.
- $\Leftarrow$ Par contraposée, on suppose que l'ordre n'est pas bien fondé. A n'est pas nul, on pose donc $a_0 \in A$ quelconque. a_0 n'est pas minimal dans A, donc il existe $a_1 \in A$ tel que $a_1 < a_0$. a_1 n'est pas minimal dans A, donc il existe $a_2 \in A$ tel que $a_2 < a_1$. Ainsi de suite, on peut construire une suite $(a_n)_{n\in\mathbb{N}}$ strictement décroissante dans A. [1]

□

Exercice 9

Pour chacun des ordres suivants, dire s'il est bien fondé :

- $(\mathbb{Z}, \leq)$
- $(\mathbb{N}, \leq)$
- $(\mathbb{R}^+, \leq)$
- $(\mathbb{N}, |)$
- $(\mathcal{P}(\mathbb{N}), \subseteq)$

1. Cette étape utilise sans le dire l'axiome du choix.

Proposition 7

Soit $(X_1, \leq_1), \ldots, (X_n, \leq_n)$ des ensembles bien fondés. Alors :

— L'ensemble $X_1 \times \cdots \times X_n$ muni de l'ordre produit est bien fondé.

— L'ensemble $X_1 \times \cdots \times X_n$ muni de l'ordre lexicographique est bien fondé

Démonstration. Nous avons déjà vu que ces relations sont des ordres.

— Notons $\leq_P$ l'ordre produit sur $X_1 \times \cdots \times X_n$. On suppose par l'absurde que ce n'est pas un ordre bien fondé. Alors, par propriété des ordres bien fondés, il existe une suite $(y_k)_{k\in\mathbb{N}} \in (X_1 \times \cdots \times X_n)^{\mathbb{N}}$ strictement décroissante. Pour $k \in \mathbb{N}$, notons $y_k = (x_k^1, \ldots x_k^n)$. Alors, la suite $(x_k^1)_{k\in\mathbb{N}}$ est strictement décroissante. En effet, pour $k \in \mathbb{N}$, $y_{k+1} <_P y_k$, donc $x_{k+1}^i <_i x_k^i$ pour $i \in [\![1, n]\!]$. Donc en particulier $x_{k+1}^1 <_1 x_k^1$. Or, l'ordre $(X_1, \leq_1)$ est bien fondé, ce qui fournit une contradiction.

Donc, $\leq_P$ est bien fondé.

— Notons $\leq_L$ l'ordre lexicographique sur $X_1 \times \cdots \times X_n$. On suppose par l'absurde que ce n'est pas un ordre bien fondé. Alors, par propriété des ordres bien fondés, il existe une suite $(y_k)_{k\in\mathbb{N}} \in (X_1 \times \cdots \times X_n)^{\mathbb{N}}$ strictement décroissante. Pour $k \in \mathbb{N}$, notons $y_k = (x_k^1, \ldots x_k^n)$. L'ensemble $\{x_k^1 | k \in \mathbb{N}\}$ est non vide, et donc admet un plus petit élément a^1, car $(X_1, \leq_1)$ est bien fondé. Notons $k_1 \in \mathbb{N}$ tel que $x_{k_1}^1 = a^1$. Par décroissance, $x_k^1 = a^1$ pour $k \geq k_1$.

L'ensemble $\{x_k^2 | k \geq k_1\}$ est non vide, et admet donc un plus petit élément a^2. Notons $k_2 \geq k_1 \in \mathbb{N}$ tel que $x_{k_2}^2 = a^2$. Par décroissance, $x_k^2 = a^2$ pour $k \geq k_2$. Donc, pour $k \geq k_2$, $x_k^1 = a^1$ et $x_k^2 = a^2$.

On définit ainsi de suite $k_1, k_2, \ldots k_n \in \mathbb{N}$ et $a^1 \in X_1 \ldots a_n \in X_n$ tels que pour $j \in [\![1, n]\!]$, pour $k \geq k_j$, on a $x_k^1 = a^1, \ldots, x_k^j = a^j$.

En particulier, pour $k \geq k_n$, on a $y_k = (a^1, \ldots, a^n)$, la suite $(y_k)_k$ n'est donc pas strictement décroissante : on aboutit à une contradiction.

□

En revanche, si $(\Sigma, \leq)$ est un ordre bien fondé, Σ^* muni de l'ordre lexicographique n'est pas nécessairement bien fondé (Cf TD).

Une première utilité des ordre bien fondés est de montrer la terminaison d'une fonction récursive, en généralisant le principe de descente infinie de Fermat. En effet :

Proposition 8

Si f est une fonction récursive, définie sur un ensemble X muni d'un ordre $\leq$ bien fondé, et que pour tout $x \in X$, l'appel à $f(x)$ cause des appels récursifs $f(x_1), \ldots, f(x_k)$ avec $x_1 < x, \ldots, x_k < x$, alors la fonction termine.

En effet, si la fonction ne terminait pas, on pourrait trouver une suite d'appels récursifs infinie, et donc trouver une suite strictement décroissante dans X. On verra dans la prochaine section comment formaliser cette idée.

Exemple 5

On considère les fonctions suivantes :

```
1  (* Nombre de 1 dans l'écriture en base 3 de x
2     pour x entier positif ou nul *)
3  let rec uns_base_3 x = assert (x >=0);
4      if x = 0 then 0 else
5    match x mod 3 with
6    | 1 -> uns_base_3 (x/3) + 1
7    | _ -> uns_base_3 (x/3) ;;
8
9  let rec pgcd x y =
10   assert (x>=0 && y >= 0);
11   assert (x>0 || y > 0) ;
12   match (x, y) with
13   | (0, a) | (a, 0) -> a
14   | _ -> if x > y then pgcd (x-y) x
15        else pgcd x (y-x);;
```

Montrons que la fonction `uns_base_3` termine. On remarque que si $x \geq 0$, alors l'appel récursif causé par `uns_base_3 x` se fait sur une entrée $x' \geq 0$ avec $x' < x$. Donc, $\mathbb{N}$ étant bien fondé, la fonction termine sur toutes les entrées positives. De plus, l'assertion garanti la terminaison pour les entrées strictement négatives.

Appliquons le même schéma à la fonction `pgcd`. En prenant en compte les assertions, on se restreint aux entrées dans $E = (\mathbb{N} \times \mathbb{N}) \backslash \{(0,0)\}$. On munit cet ensemble de l'ordre lexicographique. On remarque ainsi que chaque appel à `pgcd` causant un appel récursif le fait sur une entrée strictement inférieure selon cet ordre. En effet, pour $(x, y) \in \mathbb{N}^*$:

— Si $x > y$ alors $x - y > 0$, donc $(x - y, x) \in E$, et $(x - y, x) < (x, y)$ car $y > 0$

— Sinon, alors $(x \leq y)$, donc $y - x \geq 0$, et $(x, y - x) < (x, y)$ car $x > 0$.

Donc, la fonction termine.

Proposition 9

Soit X un ensemble quelconque et $(Y, \leq_Y)$ un ensemble bien fondé. Soit $f : X \to Y$ On définit la relation $\mathcal{R}$ sur X par :

$$\forall x, x' \in X, x \mathcal{R} x' \Leftrightarrow f(x) < f(x')$$

Alors $(X, \mathcal{R}_r)$ est un ordre, et il est bien fondé (Rappel : $\mathcal{R}_r$ est la clôture réflexive).

Démonstration. Montrons que $\mathcal{R}_r$ est une relation d'ordre : Exercice

Montrons que c'est un ordre bien fondé. Supposons par l'absurde qu'il existe une suite $(x_n)_{n\in\mathbb{N}} \in X^{\mathbb{N}}$ strictement décroissante. Donc, pour tout $n \in \mathbb{N}, x_n \mathcal{R} x_{n+1}$. Donc par définition de $\mathcal{R}$, pour tout $n \in \mathbb{N}, f(x_n) < f(x_{n+1})$. Alors, la suite $(f(x_n))_{n\in\mathbb{N}} \in Y^{\mathbb{N}}$ est strictement décroissante : c'est absurde car Y est bien fondé. □

Cette dernière propriété permettra de généraliser la notion de ***variant de boucle***. Lorsque l'on étudie une fonction récursive, définie sur un ensemble X, un schéma classique de preuve de terminaison sera :

1. Exhiber un sous-ensemble $X_0 \subseteq X$ d'entrées valides (typiquement, cet ensemble sera donné par les préconditions, les hypothèses, les assertions)
2. Exhiber une fonction $f : X_0 \to Y$ où Y est muni d'un ordre bien fondé
3. Montrer que tout appel sur une entrée $x \in X_0$ cause uniquement des appels récursifs sur des entrées $x' \in X_0$ avec $f(x') < f(x)$

Exemple 6

On considère les fonctions suivante :

```
let rec f l = match l with
  | [] -> failwith "liste vide"
  | [x] -> x
  | x::y::q -> f ((max x y)::q)


let rec g (l1, l2) =
  match (l1, l2) with
  | [], _ -> somme l2
  | x1::q1, [] -> g ([],g (q1, (x1+1)::q1)::q1)
  | x1::q1, x2::q2 -> g ((x1+x2)::q1, q2)
;;
```

— Pour f : On considère l'ensemble des listes non vides X_0. Les appels récursifs causés par l'appel de f sur une liste non vide se font également sur des listes non vides. On considère la fonction $L : l \mapsto |l|$ qui associe à une liste sa longueur. Cette fonction est à valeurs dans $\mathbb{N}$, qui est bien fondé, et les appels récursifs font décroître $L(l)$ strictement. Donc, la fonction termine.

— Pour g : On considère la fonction $L : (l_1, l_2) \mapsto (|l_1|, |l_2|)$. L est à valeur dans $\mathbb{N}^2$, que l'on munit de l'ordre lexicographique. Cet ordre est bien fondé, et les appels récursifs causés par f sont tels que pour (l_1, l_2) deux listes, tout appel récursif `f(l1', l2')` causé par `f (l1, l2)` vérifie $L(l_1', l_2') < L(l_1, l_2)$.
On a donc trouvé une fonction sur les entrées dont la valeur décroît strictement à chaque appel récursif, et qui est à valeur dans un ensemble ordonné bien fondé : la fonction termine.

2 Induction

Le raisonnement par induction va permettre de généraliser le principe de récurrence. On rappelle le principe de récurrence faible :

Proposition 10

Soit P une propriété sur $\mathbb{N}$, telle que :

— $P(0)$

— $\forall n \in \mathbb{N}, P(n) \Rightarrow P(n+1)$

Alors, $\forall n \in \mathbb{N}, P(n)$.

Démonstration. On considère $A = \{n \in \mathbb{N} \mid P(n) \text{ est fausse}\}$. On suppose par l'absurde que A est non-vide. $A \subseteq \mathbb{N}$ donc A admet un élément minimal n_0. $n_0 \neq 0$ car $P(0)$ est vraie. Donc, $n_0 - 1 \in \mathbb{N}$ et par minimalité, $P(n_0 - 1)$ est vraie. Donc, $P(n_0)$ est vraie : absurde. □

On a plus généralement le principe de récurrence forte :

Proposition 11

Soit P une propriété sur $\mathbb{N}$, telle que : $\forall n \in \mathbb{N}, (\forall k < n, P(k)) \Rightarrow P(n)$. Alors, $\forall n \in \mathbb{N}, P(n)$.

Ce principe signifie que si une propriété est telle qu'en la supposant vraie pour tous les entiers inférieurs à n, alors elle est vraie pour n également, alors cette propriété est vraie pour tous les entiers.

A Induction bien fondée

Proposition 12: Principe d'induction bien fondée

Soit $(X, \leq)$ un ensemble ordonné non vide et $\mathcal{P}$ une propriété sur X. Alors, si $\leq$ est bien fondé, on a :

$$(\forall x \in x, (\forall y \in X, y < x \Rightarrow \mathcal{P}(y)) \Rightarrow \mathcal{P}(x)) \Rightarrow \forall x \in X, \mathcal{P}(x)$$

Démonstration. Notons $\mathcal{I}_\mathcal{P}(x) : (\forall y \in X, y < x \Rightarrow \mathcal{P}(y)) \Rightarrow \mathcal{P}(x))$. $\mathcal{I}_\mathcal{P}(x)$ signifie : "si $\mathcal{P}$ est vraie sur tous les éléments inférieurs à x, alors $\mathcal{P}$ est vraie sur x".
Il faut montrer que si $\mathcal{I}_\mathcal{P}(x)$ est vraie pour tout $x \in X$, alors $\mathcal{P}(x)$ est vraie pour tout $x \in X$.

Supposons $\forall x \in X, \mathcal{I}_\mathcal{P}(x)$. On considère $A = \{x \in X \mid \mathcal{P}(x) \text{ est fausse}\}$. On suppose par l'absurde que A est non-vide. $A \subseteq X$ et l'ordre est bien fondé, donc A admet un élément minimal x_0. Par contraposée de $\mathcal{I}_\mathcal{P}(x_0)$, il existe $y \in X$ avec $y < x$ et $\mathcal{P}(y)$ fausse. Alors, $y \in A$, et par minimalité de x, on n'a pas $y \leq x$, ce qui est absurde. □

Cette propriété généralise le principe de récurrence à tous les ensembles bien fondés.

Le principe d'induction dit donc que pour montrer que $\mathcal{P}$ est vraie sur tout X, il suffit de montrer que $\mathcal{I}_\mathcal{P}$ est vraie sur tout X.

Par exemple, pour l'ordre naturel sur $\mathbb{N}$, le principe d'induction dit que pour prouver une propriété $\mathcal{P}$ sur $\mathbb{N}$, il faut prouver que pour $n \in \mathbb{N}$, si $\mathcal{P}(k)$ est vraie pour $k < n$, alors $\mathcal{P}(n)$ est vraie : c'est bien le principe de récurrence forte ! On remarque que pour montrer $\mathcal{I}_\mathcal{P}(0)$, il faut directement montrer $\mathcal{P}(0)$, car 0 n'a aucun prédécesseur.

Utilisons ce principe pour étudier quelques fonctions. Reprenons la fonction de pgcd vue plus haut :

```
let rec pgcd x y =
  assert (x>=0 && y >= 0);
  assert (x>0 || y > 0) ;
  match (x, y) with
  | (0, a) | (a, 0) -> a
  | _ -> if x > y then pgcd (x-y) x
      else pgcd x (y-x);;
```

Montrons que cette fonction réalise bien le pgcd. Pour $n, m \in \mathbb{N}$ deux entiers, avec $(n, m) \neq (0, 0)$, on note $n \wedge m$ le pgcd de n et m.

On induit $\mathbb{N} \times \mathbb{N}$ de l'ordre lexicographique. Montrons par induction sur (n, m) :

$$\forall (n, m) \in (\mathbb{N} \times \mathbb{N}) \backslash \{(0, 0)\}, P(n, m) : \textbf{pgcd}(n, m) = n \wedge m$$

On considère donc $(n, m) \in (\mathbb{N} \times \mathbb{N}) \backslash \{(0, 0)\}$, et on suppose que pour tout couple $(n', m') \in (\mathbb{N} \times \mathbb{N}) \backslash \{(0, 0)\}$ avec $(n', m') < (n, m)$, la propriété $P(n', m')$ est vraie.

— Si $n = 0$, $\textbf{pgcd}(n, m) = m = 0 \wedge m$, la propriété est vraie

— Si $m = 0$, $\textbf{pgcd}(n, m) = n = n \wedge 0$, la propriété est vraie

— Sinon, si $n > m$ alors $n - m > 0$, et comme $m > 0$, $n - m < n$. Donc, par hypothèse d'induction, $P(n - m, n)$ est vraie. Donc $\textbf{pgcd}(n, m) = \textbf{pgcd}(n - m, n) = (n - m) \wedge n = n \wedge m$

— Sinon, alors $n \leq m$, et comme $n > 0$, $m - n < m$. Donc, par HI, $P(n, m - n)$ est vraie. On conclut de manière analogue au cas précédent.

Au dernier chapitre, nous avons montré formellement la correction de fonctions sur les listes, en raisonnant par récurrence sur la taille des listes. On peut aussi raisonner directement par induction sur les listes, on considérant l'ordre induit par la longueur des listes. On rappelle le code du tri fusion :

```
let rec separer l = match l with
  | [] | [_] -> (l, [])
  | x::y::q -> let (l1, l2) = separer q in (x::l1, y::l2)

let rec fusionner l1 l2 = match l1, l2 with
  | [], l | l, [] -> l
  | x1::q1, x2::q2 -> if x1 < x2 then x1 :: fusionner q1 l2
                  else          x2 :: fusionner l1 q2

let rec tri_fusion l = match l with
  | [] | [_] -> l
  | _ -> let l1, l2 = separer l in
      let l1, l2 = tri_fusion l1, tri_fusion l2 in
      fusionner l1 l2
```

Montrons formellement la correction de `fusionner` :

On note L l'ensemble des listes. On munit L de l'ordre induit par la fonction **taille** : $L \to \mathbb{N}$. Cet ordre est bien fondé. On munit ensuite L^2 de l'ordre lexicographique, lui aussi bien fondé.

Terminaison : On remarque que les entrées décroissent strictement à chaque appel récursif : la fonction termine.

Correction : On pose $P(l_1, l_2)$: Si l_1 et l_2 sont triées, alors **fusionner**(l_1, l_2) est triée et contient les éléments de l_1 et l_2.

Montrons par induction sur L^2 que

$$\forall (l_1, l_2) \in L^2, P(l_1, l_2)$$

On considère donc l_1, l_2 deux listes triées.

- Si l_1 est vide, alors **fusionner**$(l_1, l_2) = l_2$, donc la propriété est vérifiée
- Si l_2 est vide, alors **fusionner**$(l_1, l_2) = l_2$, donc la propriété est vérifiée
- Si aucune des deux listes est vide, alors $l_1 = x_1 :: q_1$ et $l_2 = x_2 :: q_2$.
  - Si $x_1 < x_2$: par hypothèse d'induction, $P(q_1, l_2)$ est vraie car $(q_1, l_2) < (l_1, l_2)$. De plus, q_1 est triée. Donc **fusionner**(q_1, l_2) contient les éléments de q_1 et l_2, et est triée. De plus, x_1 est l'élément minimal de l_1, et est inférieur à x_2 qui est l'élément minimal de l_2. Donc, x_1 est inférieur à tous les éléments de **fusionner**(q_1, l_2), et donc **fusionner**$(l_1, l_2) = x_1 ::$ **fusionner**$(q1, l_2)$ est bien triée, et contient les éléments de l_1 et l_2. La propriété est bien vérifiée.
  - Sinon, le raisonnement est analogue

Ainsi, la propriété est vraie pour toute paire de listes, et donc la fonction est correcte.

Exercice 10

Montrer de la même manière la correction du tri fusion. On pourra admettre la correction de la fonction de séparation de liste.

B Construction d'un ensemble par induction

Considérons le code suivant :

```
type couleur =
  | Rouge
  | Jaune
  | Bleu
  | Melange of couleur*couleur;;

(* Renvoie la fraction de la couleur primaire cp présente dans c *)
let rec frac_primaire cp c =
  match c with
  | Rouge | Jaune | Bleu -> if c = cp then 1. else 0.
  | Melange (c1, c2) -> 0.5 *. frac_primaire cp c1 +. 0.5 *. frac_primaire cp c2 ;;
```

De quel ordre bien fondé munir l'ensemble de définition de cette fonction pour y appliquer les résultats précédents ? Plus généralement, quel est l'ensemble mathématique qui représente le type `couleur` ?

Les ensembles définis par induction vont permettre de représenter les types somme en OCaml : les listes, le type `couleur`, etc...

Définition 14

Une ***règle de construction*** est un triplet (C, r, P) où :

- C est un symbole, appelé ***constructeur***
- r est un entier, appelé ***arité***
- P est un ensemble

Si $r = 0$ on parle de règle de base, sinon de règle d'induction.

Une règle de construction correspondra à un constructeur d'un type OCaml, et donc un type, qui est une liste finie de constructeurs, correspondra à une liste finies de règles.

Par exemple, on considère le type OCaml suivant :

```
type boisson =
  | Eau
  | Jus of string
  | Melange of boisson * boisson * float ;;
```

Il est représenté par trois règles de constructions :

- $(\textbf{Eau}, 0, \{\bullet\})$, où $\{\bullet\}$ est un ensemble singleton
- $(\textbf{Jus}, 0, \Sigma^*)$ où Σ est l'ensemble des caractères ASCII
- $(\textbf{Melange}, 2, \mathbb{R})$

L'arité correspond donc au nombre de paramètres récursifs que nécessite le constructeur, et l'ensemble P d'une règle correspond aux autres paramètres, ceux qui ne sont pas récursifs.

Exercice 11

Soit A un ensemble. Donner deux règles de constructions permettant de représenter les listes d'éléments de A.

Définition 15

Soit $\mathcal{C}$ un ensemble de règles de constructions, dont tous les symboles de constructeurs sont distincts. On note $\mathcal{B}$ l'ensemble des règles de base, et $\mathcal{I}$ l'ensemble des règles d'induction.

On définit :

— $X_0 = \{(C, p) | (C, 0, P) \in \mathcal{B}, p \in P\}$

— Pour $n \in \mathbb{N}$, $X_{n+1} = X_n \cup \{(C, p, x_1, \ldots, x_r) | (C, r, P) \in \mathcal{I}, p \in P, (x_1, \ldots, x_r) \in X_n^r\}$

On note ensuite X la limite des ensembles X_n :

$$X = \bigcup_{n \in \mathbb{N}} X_n$$

On dit que X est ***l'ensemble construit par induction à partir de*** $\mathcal{C}$.

Exemple 7

Reprenons le type `boisson`. On a donc :

$$\begin{array}{rcl} \mathcal{C} & = & \{(\textbf{Eau}, 0, \{\bullet\}), (\textbf{Jus}, 0, \Sigma^*), (\textbf{Melange}, 2, \mathbb{R})\} \\ \mathcal{B} & = & \{(\textbf{Eau}, 0, \{\bullet\}), (\textbf{Jus}, 0, \Sigma^*)\} \\ \mathcal{I} & = & \{(\textbf{Melange}, 2, [0, 1])\} \end{array}$$

Que contient X ? On commence par regarder le contenu des différents X_n pour $n \in \mathbb{N}$.

— X_0 contient $(\textbf{Eau}, \bullet)$, $(\textbf{Jus}, \text{"orange"})$, $(\textbf{Jus}, \text{"fraise"})$, $(\textbf{Jus}, \text{"goyave"}), \ldots$. On les notera simplement $\textbf{Eau}, \textbf{Jus}(\text{"orange"}), \textbf{Jus}(\text{"fraise"}), \textbf{Jus}(\text{"goyave"})$.

— X_1 contient tous les éléments de X_0 ainsi que :

  — $\textbf{Melange}(\textbf{Eau}, \textbf{Eau}, 0.5)$

  — $\textbf{Melange}(\textbf{Eau}, \textbf{Jus}(\text{"fraise"}), 0.87)$

  — $\textbf{Melange}(\textbf{Jus}(\text{"framboise"}), \textbf{Jus}(\text{"pomme"}), 0.97)$

  — $\ldots$

— X_2 contient tous les éléments de X_1 ainsi que :

  — $\textbf{Melange}(\textbf{Melange}(\textbf{Eau}, \textbf{Jus}(\text{"fraise"}), 0.87), \textbf{Jus}(\text{"pomme"}), 0.14)$

  — ...

Il apparaît que X_n permet de représenter l'ensemble des valeurs OCaml du type `boisson` ayant une "profondeur" de récursivité d'au plus n, et X représente les valeurs OCaml du type `boisson` de profondeur finie quelconque.

C Induction structurelle

Dans la suite, on considère X un ensemble construit par induction à partir de $\mathcal{C} = \mathcal{B} \cup \mathcal{I}$. On identifiera une règle $C = (\mathbf{S}, r, P)$ avec son symbole $\mathbf{S}$, et on supposera donc que les règles ont des symboles distincts. On note comme dans la définition :

— $X_0 = \{(C, p) \mid (C, 0, P) \in \mathcal{B}, p \in P\}$

— Pour $n \in \mathbb{N}$, $X_{n+1} = X_n \cup \{(C, p, x_1, \ldots, x_r) \mid (C, r, P) \in \mathcal{I}, p \in P, (x_1, \ldots, x_r) \in X_n^r\}$.

Il reste maintenant à définir un ordre bien fondé sur X.

Définition 16

Soit $x \in X$. On appelle ***hauteur*** de x l'indice n_0 minimal tel que $x \in X_n$. On définit donc la fonction suivante :

$$h : \begin{array}{ccc} X & \longrightarrow & \mathbb{N} \\ x & \longmapsto & \min\{n \in \mathbb{N} \mid x \in X_n\} \end{array}$$

Proposition 13

On a :

— $X_0 = \{x \in X \mid h(x) = 0\}$

— $\forall n \in \mathbb{N}^*, \{x \in X \mid h(x) = n\} = X_n \backslash X_{n-1}$

Définition 17

On définit la relation $\mathcal{R}$ sur X par :

$$\forall x, y \in X, a\mathcal{R}b \Leftrightarrow \exists (\mathbf{S}, r, P) \in \mathcal{I}, \exists p \in P, \exists x_1, \ldots, x_r \in X, \exists i \in [\![1, r]\!], y = \mathbf{S}(p, x_1, \ldots, x_r) \text{ et } x_i = x$$

Autrement dit, $x\mathcal{R}y$ si y est construit à partir de x.

Remarque 2

On a $\forall x, y \in X, x\mathcal{R}y \Rightarrow h(y) \geq h(x) + 1$.

Proposition 14

On note $\leq_X$ la clôture réflexive transitive de $\mathcal{R}$:

$$\leq_X = \bigcup_{n \in \mathbb{N}} \mathcal{R}^n$$

Alors $\leq_X$ est une relation d'ordre, et c'est un ordre bien fondé.

Démonstration. $\leq_X$ est transitive et réflexive. Il reste à montrer qu'elle est antisymétrique. Soient $x, y \in X$ tels que $x \leq_X y$ et $y \leq_X x$. Il existe donc n_1, n_2 tels que $x\mathcal{R}^{n_1}y$ et $y\mathcal{R}^{n_1}x$. On suppose par l'absurde que $x \neq y$. Donc, n_1, n_2 sont non nuls. En itérant la remarque précédente, on a donc $h(y) \geq h(x) + n_2 > h(x)$ et $h(x) \geq h(y) + n_1 > h(y)$. Donc, $h(y) > h(x) > h(y)$: absurde.

$\leq_X$ est bien un ordre. De plus, la fonction $h : (X, \leq_X) \to (\mathbb{N}, \leq)$ est strictement croissante et $(\mathbb{N}, \leq)$ est bien fondé : $(X, \leq_X)$ est donc également bien fondé. □

On applique le principe d'induction bien fondé à cet ordre pour faire des preuves sur les ensembles construits par induction. Ce principe, que l'on utilisera très souvent pour étudier de nombreux objets inductifs, s'appelle le **principe d'induction structurelle**.

En pratique, on n'utilise souvent que les prédécesseurs directs. Plus précisément, on utilisera la propriété suivante, qui est l'analogue de la récurrence faible sur $\mathbb{N}$:

Proposition 15

Soit $\mathcal{P}$ une propriété sur X telle que :

— $\forall (S, 0, P) \in \mathcal{B}, \forall p \in P,\ \mathcal{P}(\mathbf{S}(p))$ est vraie

— $\forall (S, r, P) \in \mathcal{I}, \forall p \in P, \forall x_1, \ldots, x_r \in X, (\forall i \in [\![1, r]\!], \mathcal{P}(x_i)) \Rightarrow \mathcal{P}(\mathbf{S}(p, x_1, \ldots, x_r))$

Alors, $\mathcal{P}(x)$ est vraie pour tout $x \in X$.

Autrement dit, si une propriété est vraie sur tous les éléments construits avec les constructeurs de base, et qu'elle se transmet bien via les constructeurs inductifs, alors elle est vraie sur tout X.

Pour voir comment l'utiliser dans une preuve, il nous faut pouvoir définir des fonctions par induction, afin d'exprimer des propriétés dessus.

D Fonctions définies par induction

Pour définir sur X une fonction par induction, il faudra définir d'une part son comportement sur les constructeurs de bases, et d'autre part son comportement sur les constructeurs inductifs. On pourra définir les fonctions inductives de manière similaire à ce que l'on fait en OCaml avec les let rec et les match with.

Par exemple, on considère le type OCaml des listes d'entiers :

```
1 type liste = ListeVide | ListeNonVide of int * liste ;;
```

On considère donc l'ensemble **Liste** construit par induction à partir des règles suivantes :

— $(\mathbf{ListeVide}, 0, \bullet)$. On notera $[] = \mathbf{ListeVide}(\bullet)$.

— $(\mathbf{ListeNonVide}, 1, \mathbb{N})$. On notera $n :: q = \mathbf{ListeNonVide(n,q)}$.

On considère ensuite la fonction **len** définie comme suit :

— $\mathbf{len}([]) = 0$

— Pour $n \in \mathbb{N}$, $q \in \mathbf{Liste}$, $\mathbf{len}(\mathbf{ListeNonVide}(n, q)) = 1 + \mathbf{len}(q)$

Cette fonction correspond à la longueur d'une liste. On l'a ***définie*** par induction en définissant son comportement sur chacun des deux constructeurs de l'ensemble **Liste**.

Montrons maintenant par induction sur la structure des listes la propriété suivante : $\forall l \in \mathbf{Liste}, len(l) \geq 0$.

Il y a deux cas :

— $l = []$. Alors, $len(l) = 0$: la propriété est vérifiée

— $l = n :: q$ avec $n \in \mathbb{N}$, $q \in \mathbf{Liste}$. On suppose par induction que $len(q) \geq 0$. Alors, $len(l) = 1 + len(q) \geq 0$: la propriété est vérifiée.

Ainsi, par induction structurelle, $\forall l \in \mathbf{Liste}, len(l) \geq 0$.

E Méthode de preuve par induction structurelle

Lorsqu'on a un ensemble construit X par induction à partir de n constructeurs $C_1, \dots, C_n$, et que l'on doit faire une preuve par induction sur cet ensemble, on suivra le schéma suivant :

1. On annonce la preuve : "Montrons par induction structurelle sur X la propriété suivante : $\forall x \in X, P(x)$:"
2. On annonce le nombre de cas, i.e. le nombre de constructeurs : "Il y a n cas :"
3. Pour chaque cas, on fait l'étape d'induction correspondante : "$x = C_1(p, x_1, \dots, x_r)$, supposons par induction que $P(x_1), \dots, P(x_r)$ sont vraies. Alors ..."
4. Une fois chaque cas traité, on conclut : "Ainsi, par induction structurelle, $\forall x, P(x)$".

Exercice 12

On considère l'ensemble $\mathcal{E}$ des expressions arithmétiques, défini par induction avec les constructeurs suivants :

— $(\mathbf{Const}, 0, \mathbb{N})$: représente les constantes entières

— $(\mathbf{Add}, 2, \{\bullet\})$: représente la somme de deux expressions

— $(\mathbf{Mult}, 2, \{\bullet\})$: représente le produit de deux expressions

— $(\mathbf{Puiss}, 1, \mathbb{N})$: représente une expressions mise à une certaine puissance.

On définit la fonction $\mathbf{eval} : \mathcal{E} \to \mathbb{N}$ par induction structurelle sur $\mathcal{E}$:

— $\forall n \in \mathbb{N}, \mathbf{eval}(\mathbf{Const}(n)) = n$

— $\forall e_1, e_2 \in \mathcal{E}, \mathbf{eval}(\mathbf{Add}(e_1, e_2)) = \mathbf{eval}(e_1) + \mathbf{eval}(e_2)$

— $\forall e_1, e_2 \in \mathcal{E}, \mathbf{eval}(\mathbf{Mult}(e_1, e_2)) = \mathbf{eval}(e_1) \times \mathbf{eval}(e_2)$

— $\forall e_1 \in \mathcal{E}, \forall a \in \mathbb{N}, \mathbf{eval}(\mathbf{Puiss}(e_1, a)) = \mathbf{eval}(e_1)^a$

Cette fonction évalue une expression en une valeur, donnant aux constructeurs le sens attendu. On définit également la fonction $\mathbf{estNul} : \mathcal{E} \to \mathbb{B}$ par induction :

— $\forall n \in \mathbb{N}, \mathbf{eval}(\mathbf{Const}(n)) = (0 \text{ si } n = 0, 1 \text{ sinon})$

— $\forall e_1, e_2 \in \mathcal{E}, \mathbf{estNul}(\mathbf{Add}(e_1, e_2)) = (\mathbf{estNul}(e_1) \text{ et } \mathbf{estNul}(e_2))$

— $\forall e_1, e_2 \in \mathcal{E}, \mathbf{estNul}(\mathbf{Mult}(e_1, e_2)) = (\mathbf{estNul}(e_1) \text{ ou } \mathbf{estNul}(e_2))$

— $\forall e_1 \in \mathcal{E}, \forall n \in \mathbb{N}, \mathbf{estNul}(\mathbf{Puiss}(e_1, a)) = (\mathbf{estNul}(e_1) \text{ et } a \neq 0)$.

Montrer par induction structurelle sur $e \in \mathcal{E}$ que $\mathbf{eval}(e) = 0 \iff estNul(e)$.

F Étude des fonctions récursives OCaml

On considère deux fonctions calculant la somme d'une liste :

```
let rec somme l = match l with
  | [] -> 0
  | x::q -> x + somme q

(* version récursive terminale *)
let rec somme_plus l (r: int) : int
  | [] -> r
  | x::q -> somme q (x+r)
```

On note L l'ensemble des listes d'entiers. On montre par induction structurelle sur L :

$$\forall l \in L, P(l) : \forall r \in \mathbb{Z}, \textbf{somme\_plus}(l, r) = \textbf{somme}(l) + r$$

Il y a 2 cas :

— Pour la liste vide $[]$, $\textbf{somme\_plus}([], r) = r = 0 + r = \textbf{somme}([]) + r$: $P([])$ est vraie.

— Pour $l = x :: q$ avec x entier et q une liste. On suppose par induction $P(q)$. Alors par hypothèse d'induction $\textbf{somme\_plus}(q, x + r) = \textbf{somme}(q) + (x + r) = \textbf{somme}(l) + r$. Donc, $\textbf{somme\_plus}(l, r) = \textbf{somme}(l) + r$: $P(l)$ est vraie.

Ainsi, par induction structurelle sur les listes, $\forall l \in L, P(l)$.

Exercice 13

Reprendre la preuve de correction du tri fusion, en utilisant le formalisme de l'induction structurelle.