

CHAPITRE (−1)

Ordres et induction

Hugo SALOU MPI*

Dernière mise à jour le 31 octobre 2022

1 Motivation

```
1 let rec fact n =
2   if n = 0 then 1
3   else n * (fact (n-1));;
CODE 1 – Calcul de factorielle
```

Ce programme calcule la factorielle d'un nombre? En développant, l'expression de $3!$, on a

$$\begin{aligned}(\text{fact } 3) &= 3 \times (\text{fact } 2) \\ &= 3 \times (2 \times (1 \times 1))\end{aligned}$$

On en déduit que ce programme calcule la factorielle car ce développement s'arrête à un certain point. Comment en être sûr? En effet, avec $n = -1$, on obtient une *Stack Overflow Error*; on n'a plus de mémoire. Pour en être sûr, il faut définir un *invariant*.

À faire : Ajouter 2^{ème} exemple

Un autre exemple :

```
1 let mystere2 n m =
2   let rec aux c b =
3     if c = 0 and b = m then 0
4     else if c = 0 then aux (b * m) (b + 1)
5     else 1 + aux (c - 1) b
6   in aux n 0;;
CODE 2 – Un programme mystère (2)
```

Ce programme a beaucoup plus de variables : les variables augmentent dans certains cas, puis diminuent... On peut représenter l'état des variables b et c dans une figure :

À faire : Figure à faire

FIGURE 1 – État des variables b et c

On en conclut que ce programme calcule la valeur de

$$n + m + 2m + \dots + m^2 = n + \frac{m^2 \times (m-1)}{2}.$$

Cherchons un variant. On peut penser à $b - m$ mais il ne diminue pas à chaque étape. Nous verrons quel est ce variant plus tard dans le chapitre.

Nouvel exemple : retourner une liste. Essayons de distinguer les différents cas possibles : si la liste est vide, on la renvoie; sinon, on extrait un élément x et on note le reste de la liste xs , on retourne xs puis on concatène à droite x . On peut donc écrire

```
1 let rec rev l =
2   match l with
3   | [] -> []
4   | x :: xs -> (rev xs) @ [x];;
CODE 3 – Inverser une liste
```

Cependant, le $@$ est une opération lente en OCaml. En effet ce programme a une complexité en $\mathcal{O}(n^2)$, où n est la taille de la liste. Cette complexité est douteuse pour une opération aussi simple. On peut également se demander si cette opération se termine. Cela paraît très simple : la taille de la liste diminue mais nous n'avons pas le côté mathématique d'une liste. En effet, qu'est ce qu'une liste et la taille de cette liste? On doit formaliser l'explication de pourquoi cet algorithme se termine.

Continuons avec un autre exemple :

```

1 let rec mystere3 m n =
2   if m = 0 then n + 1
3   else if n = 1 then mystere (n - 1) 1
4   else mystere (m - 1) (mystere m (n-1));;

```

CODE 4 – Un programme mystère (3)

Il s'agit de la fonction ACKERMANN. Sa complexité est très importante mais ce n'est pas le sujet de cette introduction. En effet, on a

$$\begin{aligned}
 A_{0,m} &= n + 1 \\
 A_{m,0} &= A_{n-1,1} \\
 A_{m,n} &= A_{m-1, A_{m,n-1}}
 \end{aligned}$$

Malgré ce que l'on peut penser, cette fonction se termine mais comment le prouver?

À faire : Exemple arbres binaires

On en conclut que, avec les outils de l'année passée, il est difficile de prouver que ces algorithmes se terminent rigoureusement.

2 Ordre

Définition (Éléments minimaux): Lorsque (E, \preccurlyeq) est un espace ordonné, et $A \subseteq E$ ("inclut ou égal") est une partie de E , on appelle *élément minimal* de A un élément $x \in A$ tel que

$$\forall y \in A, y \preccurlyeq x \implies y = x.$$

Définition (Ordre bien fondé): Un ordre est *bien fondé* s'il n'existe pas de suite infiniment strictement croissante.

Propriété: Une relation d'ordre \preccurlyeq sur un ensemble E bien fondé si et seulement si toute partie non vide de E admet un élément minimal.

Théorème (Induction bien fondée): Soit (E, \preccurlyeq) un ensemble ordonné et bien fondé. Soit P une propriété sur les éléments de E . Si $x \in E$, on note $E^{\preccurlyeq x} = \{y \in E \mid y \preccurlyeq x\}$. Si $\forall x \in E, (\forall y \in E^{\preccurlyeq x}, P(y)) \implies P(x)$, alors $\forall x \in E, P(x)$.

REMARQUE:

Si $(E, \preccurlyeq) = (\mathbb{N}, \leq)$, alors le théorème précédent se traduit par :

$$\text{si } \forall n \in \mathbb{N}, (\forall p < n, P(p)) \implies P(n), \text{ alors } \forall n \in \mathbb{N}, P(n).$$

Ce résultat correspond à la "récurrence forte." Décomposons ce "∀" : on extrait le cas où $n = 0$

$$\text{si } P(0) \text{ et } \forall n \in \mathbb{N}^*, (\forall p < n, P(p)) \implies P(n), \text{ alors } \forall n \in \mathbb{N}, P(n).$$

On peut donc utiliser le principe de la récurrence pour tout ensemble ordonné bien fondé. ■

2.1 Ordre produit

Définition: Soit (A, \preccurlyeq_A) et (B, \preccurlyeq_B) deux ensembles ordonnés on définit alors \preccurlyeq_\times sur $A \times B$ par

$$\forall (a, b), (a', b') \in A \times B, (a, b) \preccurlyeq_\times (a', b') \stackrel{\text{def}}{\iff} (a \preccurlyeq_A a' \text{ et } b \preccurlyeq_B b').$$

Propriété: \preccurlyeq_\times est une relation d'ordre. □

La proposition précédente est facilement vérifiée comme \preccurlyeq_A et \preccurlyeq_B sont, elles aussi, des relations d'ordre.

REMARQUE (\triangle):

Si \preccurlyeq_A et \preccurlyeq_B sont des ordres totaux, \preccurlyeq_\times ne l'est pas forcément.

Propriété: Soient (A, \preccurlyeq_A) et (B, \preccurlyeq_B) bien fondés, alors $(A \times B, \preccurlyeq_\times)$ l'est aussi. ■

REMARQUE:

On a défini une relation “produit,” on peut se demander si ces résultats s'appliquent aussi si la relation est “somme.” Ce n'est pas le cas : soit \preccurlyeq_+ définie comme

$$(a, b) \preccurlyeq_+ (a', b') \stackrel{\text{def}}{\iff} a \preccurlyeq_A a' \text{ ou } b \preccurlyeq_B b'.$$

On peut démontrer que ce n'est pas une relation d'ordre.

REMARQUE:

Si (A, \preccurlyeq_A) est une relation d'ordre alors $(A^\mathbb{N}, \preccurlyeq_\times)$ a les mêmes propriétés.

REMARQUE:

Sur $A^\mathbb{N}$, on définit $(u_n)_{n \in \mathbb{N}} \preccurlyeq_\times (v_n)_{n \in \mathbb{N}}$ si et seulement si

$$\forall i, u_i \preccurlyeq_A v_i.$$

L'ensemble ordonné $(A^\mathbb{N}, \preccurlyeq_\times)$ est-il bien fondé? La réponse est non.

Voici un contre-exemple : on pose $A = \{0, 1\}$.

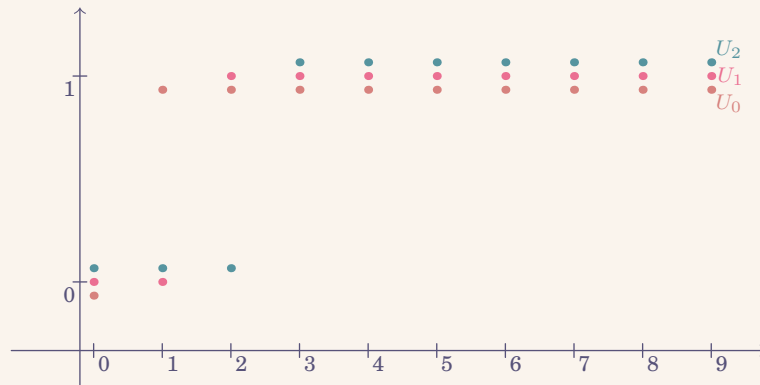


FIGURE 2 – Contre exemple : $(A^\mathbb{N}, \preccurlyeq_\times)$ est-il bien fondé?

On considère la suite U_0 qui a pour tout $n \in \mathbb{N}$ la valeur de 1. Puis, on considère la suite U_1 qui a, pour $n = 0$, la valeur de 0 puis pour les autres valeurs de n , la valeur de 1. Ensuite, on considère la suite U_2 qui, pour $n = 0, 1$, la valeur de 0 puis, pour les autres valeurs de n , la

valeur de 1. En itérant ce procédé, on crée une suite de suite $(U_n)_{n \in \mathbb{N}}$ infiniment strictement décroissante :

$$U_0 \succ_{\times} U_1 \succ_{\times} U_2 \succ_{\times} \dots$$

On considère le programme suivant :

```
1 let rec pgcd a b =
2   if a = b then a
3   else if a > b then pgcd (a-b) b
4   else pgcd a (b-a);;
```

CODE 5 – Calcul du PGCD

Étudions ce programme. Ce programme se termine si et seulement si $a = 0$ et $b = 0$ où si $a > 0$ et $b > 0$. Prouvons-le rigoureusement. On choisit comme variant (a, b) vivant dans l'ensemble ordonné $(\mathbb{N}^* \times \mathbb{N}^*, \prec_{\times})$ où \prec_{\times} est la relation d'ordre produit. **À faire : Recopier une partie du cours ici.** On a donc bien une décroissance stricte de la valeur de l'expression (a, b) valeurs dans un espace bien fondé. D'où terminaison.

Démontrons maintenant la correction, c'est-à-dire, démontrons que

$$\forall (a, b) \in \mathbb{N}^* \times \mathbb{N}^*, (\text{pgcd } a \ b) = a \wedge b.$$

Pour cela, on procède par induction sur $((\mathbb{N}^*)^2, \prec_{\times})$ pour démontrer la proposition

$$P(a, b) = (\text{pgcd } a \ b) = a \wedge b.$$

— Soit $(a, b) = (1, 1)$. On a

$$(\text{pgcd } a \ b) \underset{(\text{code})}{=} a = a \wedge b.$$

— Soit $(a, b) \neq (1, 1) \in (\mathbb{N}^*)^2$ tel que pour tout $(c, d) \in (\mathbb{N}^*)^2$ tel que $(c, d) \prec_{\times} (a, b)$ on ait $P(c, d)$. Montrons donc $P(a, b)$.

— Si $a = b$:

$$(\text{pgcd } a \ b) \underset{(\text{code})}{=} a = a \wedge b.$$

— Si $a > b$:

$$\begin{aligned} (\text{pgcd } a \ b) \underset{(\text{code})}{=} & (\text{pgcd } (a - b) \ b) \\ & \underset{(\text{hypothèse})}{=} (a - b) \wedge b \\ & \underset{(\text{maths})}{=} a \wedge b. \end{aligned}$$

2.2 Ordre lexicographique

Définition: Soit (A, \prec_A) et (B, \prec_B) deux ensembles ordonnés, on définit alors sur $A \times B$ l'ordre

$$(a, b) \preceq_{\ell} (a', b') \stackrel{\text{def}}{\iff} (a \prec_A a') \text{ ou } (a = a' \text{ et } b \preceq_B b').$$

Propriété: \preceq_{ℓ} est une relation d'équivalence. ■

Propriété: Si \prec_A est totale et \prec_B est totale alors \preceq_{ℓ} est totale. ■

Propriété: Si (A, \preccurlyeq_A) et (B, \preccurlyeq_B) sont bien fondés, alors $(A \times B, \preccurlyeq_\ell)$ l'est aussi. ■

REMARQUE:

On peut généraliser à un ensemble (A^n, \preccurlyeq_ℓ) .

Par exemple, avec $n = 3$, on a

$$(a, b, c) \preccurlyeq_\ell (a', b', c') \stackrel{\text{def}}{\iff} a \prec_A a' \text{ ou } (a = a' \text{ et } b \prec_B b') \text{ ou } (a = a' \text{ et } b = b' \text{ et } c \prec_C c').$$

Même question qu'avec l'ordre produit, l'ensemble $(A^\mathbb{N}, \preccurlyeq_\ell)$ est-il bien fondé? De même, la réponse est non, la même suite de suite est un contre-exemple.

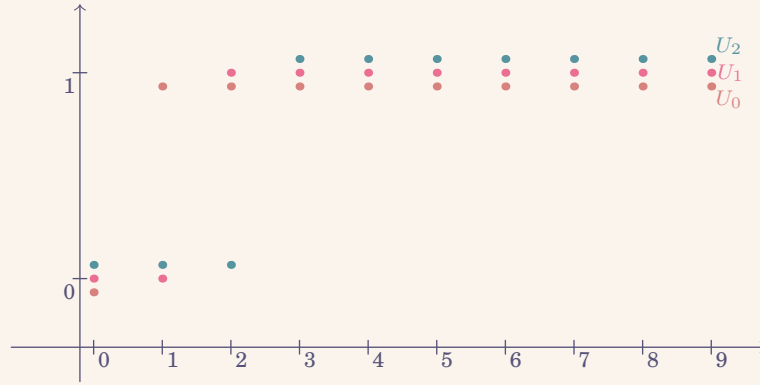


FIGURE 3 – Contre exemple : $(A^\mathbb{N}, \preccurlyeq_\ell)$ est-il bien fondé?

L'ordre lexicographique est, comme son nom l'indique, l'ordre utilisé dans le dictionnaire. La seule différence est que l'on peut comparer des mots de longueurs différentes.

RAPPEL:

Si A est un ensemble, alors $A^* = \bigcup_{n \in \mathbb{N}} A^n$. L'ensemble A^* contient toutes les suites finies d'éléments de A .

Par exemple, avec $A = \{0, 1\}$, on a

$$A^* \supseteq \{(), (1), (0), (0, 1), (1, 0), (1, 1), (0, 0), (1, 1, 0), (0, 0, 0), \dots\}.$$

Définition: Si (A, \preccurlyeq_A) est un ensemble ordonné, on définit sur A^* :

$$(u_p)_{p \in \llbracket 1, n \rrbracket} \prec_\ell (v_p)_{p \in \llbracket 1, m \rrbracket} \stackrel{\text{def}}{\iff} \begin{array}{l} \exists i \in \llbracket 1, \min(n, m) + 1 \rrbracket, \\ (\forall j \in \llbracket 1, i - 1 \rrbracket, u_j = v_j) \\ \text{et } (i = n + 1 \text{ ou } u_i \prec_A v_i). \end{array}$$

Propriété: C'est une relation d'ordre. Elle est totale si \preccurlyeq_A est totale. □

Même question avec cette nouvelle définition de l'ordre lexicographique, l'ensemble $((A^*)^\mathbb{N}, \preccurlyeq_\ell)$ est-il bien fondé? De même, la réponse est non. Voici un contre-exemple : on considère la suite de suite $U_0 = (1)$, $U_1 = (0, 1)$, $U_2 = (0, 0, 1)$. On crée une suite infiniment strictement décroissante.

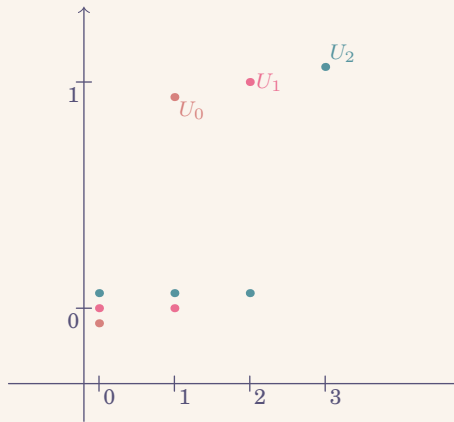


FIGURE 4 – Contre exemple : $((A^*)^{\mathbb{N}}, \preceq_\ell)$ est-il bien fondé? (2)

```

1 let rec mystere3 m n =
2   if m = 0 then n + 1
3   else if n = 1 then mystere (n - 1) 1
4   else mystere (m - 1) (mystere m (n-1));;

```

CODE 6 – La fonction ACKERMANN

La fonction ACKERMANN utilise l'ordre lexicographique; dans ce cas ci, l'ensemble ordonné est bien fondé. C'est comme cela que l'on a la terminaison de cette fonction.

Prenons un autre exemple :

```

1 let rec mystere n m p =
2   if m > 0 then 1 + mystere n (m - 1) p
3   else if m = 0 && n > 0 then 1 + mystere (n
4     -1) p (p+1)
5   else 0

```

CODE 7 – Une fonction mystère (5)

À faire :

- s'assurer de la terminaison (comme celle du PGCD)
- démontrer (par induction) que

$$\forall (m, n, p) \in \mathbb{N}^3, (\text{mystere } n \ m \ p) = \frac{n(n+1)}{2} + pn + m.$$

Les preuves de correction et de terminaison sont basées sur la supposition qu'un entier en OCaml est identique à un entier mathématique.

3 Induction nommée

Définition (Règle de Construction nommée): On appelle *Règle de Construction nommée* la donnée de

- un symbole S ,
- un entier $r \in \mathbb{N}$,
- un ensemble non vide C .

On écrira alors cette règle

$$“S \Big|_C^r” \quad \text{ou encore} \quad “S(y, \underbrace{\square, \square, \dots, \square}_r) \text{ pour } y \in C.”$$

REMARQUE:

On a parfois besoin d'un ensemble C trivial (de taille 1 et contenant un objet inutile), on note alors la règle $S \Big|_C^r$.

Définition: — On appelle *règle de base* une règle de la forme $S|_C^0$.
— On appelle *règle d'induction* une règle de la forme $S|_C^n$.

Définition: Étant donné un ensemble fini de règles $R = \underbrace{B}_{\text{règle de base}} \cup \underbrace{I}_{\text{règle d'induction}}$ avec $B \neq \emptyset$, on définit alors

$$X_0 = \{(S, a) \mid S|_C^r \text{ et } a \in C\}$$

puis, pour tout $n \in \mathbb{N}$,

$$X_{n+1} = X_n \cup \{(S, a, t_1, t_2, \dots, t_r) \mid S|_C^r \in \underbrace{R}_C, a \in C, t_1 \in X_n, t_2 \in X_n, \dots, t_r \in X_n\}.$$

On appelle alors $\bigcup_{n \in \mathbb{N}} X_n$ l'ensemble défini par induction nommée à partir des règles de R .

REMARQUE (Notation):

On note un n -uplet ayant pour premier élément un symbole S puis $n-1$ éléments (a_1, \dots, a_{n-1}) .
Au lieu de (S, a_1, \dots, a_{n-1}) , on note $S(a_1, \dots, a_{n-1})$.

REMARQUE:

Pour l'ensemble A défini par induction à partir de $R = \{0|, S|_C^1\}$, on dira plutôt

“Soit A l'ensemble défini par induction tel que $0 \in A$ et $\forall a \in A, S(a) \in A$.”

Définition: Soit R un ensemble fini de règles et A l'ensemble défini par induction à partir de ces règles. Sur A , on définit la relation binaire \diamond par

$$x \diamond y \stackrel{\text{def}}{\iff} y = S(\dots, x, \dots) \text{ avec } S|_C^r \in R.$$

On définit alors

$$x \preceq y \stackrel{\text{def}}{\iff} \exists p \in \mathbb{N}, \exists (a_1, \dots, a_p) \in A^p, x \diamond a_1 \text{ et } a_1 \diamond a_2 \text{ et } \dots \text{ et } a_p \diamond y \text{ ou } x = y.$$

Définition (hauteur): Soit R un ensemble fini de règles d'induction nommée définissant un ensemble $A = \bigcup_{n \in \mathbb{N}} X_n$. On définit alors

$$h : A \longrightarrow \mathbb{N}$$

$$x \longmapsto \min\{n \in \mathbb{N} \mid x \in X_n\}.$$

REMARQUE:

Si $x \in A$, il existe alors $n_0 \in \mathbb{N}$ tel que $x \in X_{n_0}$ donc $\{n \in \mathbb{N} \mid x \in X_n\} \neq \emptyset$ donc le minimum existe.

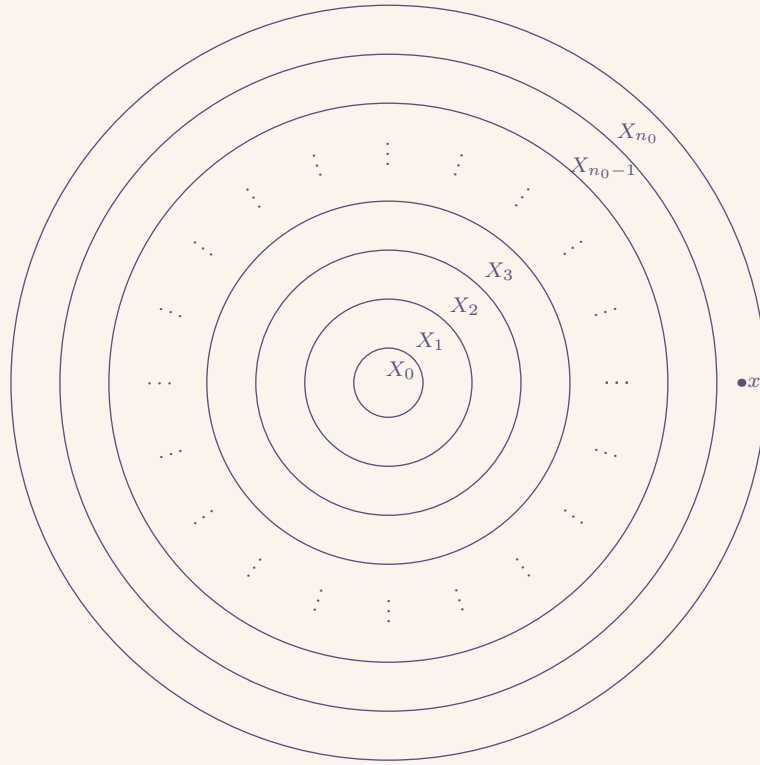


FIGURE 5 – Structure des ensembles X_1, \dots, X_n

Propriété: Si $x \diamond y$, alors $h(x) < h(y)$.

■

Corollaire: Si $n \leq y$, alors $h(x) = h(y) \iff x = y$ et $h(x) < h(y) \iff x \prec y$.

Corollaire: La relation \preccurlyeq est antisymétrique.

REMARQUE:

La relation \preccurlyeq est trivialement transitive et réflexive. Elle est donc d'ordre.

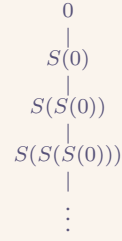


FIGURE 6 – Ensemble obtenu avec les règles S et 0

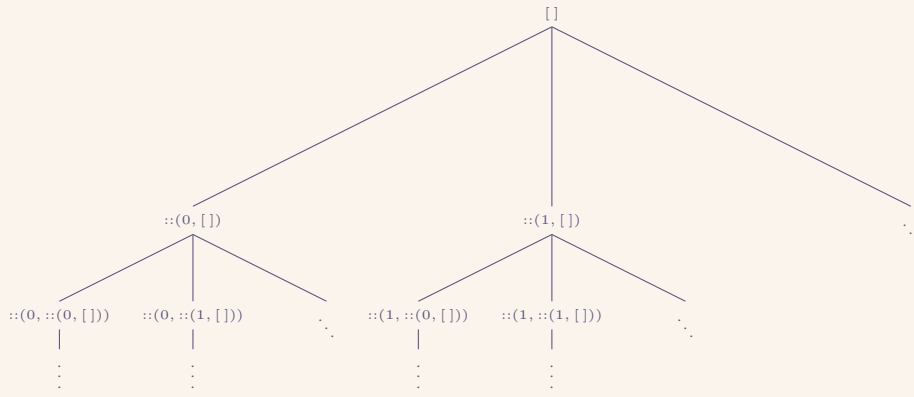


FIGURE 7 – Ensemble obtenu avec les règles $::$ et $[]$

Propriété: La relation \preccurlyeq est bien fondée.

■

REMARQUE:

On peut faire des preuves par induction bien fondée.

Propriété: Étant donné,

- un ensemble A défini par induction à partir d'un ensemble de règles nommé R ,
- un ensemble \mathbb{I} ,

— une fonction $f_T : C \times \mathbb{I}^r \rightarrow \mathbb{I}$ pour chaque règle $T = S|_C^r \in R$,
on définit de manière unique une fonction $f : A \rightarrow \mathbb{I}$ telle que, pour tout $x = S(a, t_1, \dots, t_r) \in A$, soit $T = S|_C^r \in R$ alors $f(x) = f_T(a, f(t_1), \dots, f(t_r))$.