Corrigé : génération d'objets combinatoires

Partie I. Génération des parties d'un ensemble

Question 1.

a) À toute application $f \in \mathcal{F}(\llbracket 1,n\rrbracket,\{0,1\})$ associons l'ensemble $\mathrm{A}_f = \left\{k \in \llbracket 1,n\rrbracket \;\middle|\; f(k)=1\right\}$. Alors l'application $f \mapsto \mathrm{A}_f$ est l'application réciproque de $\mathrm{A} \mapsto \chi_{\mathrm{A}}$.

b) Par unicité de la décomposition d'un entier en base 2, l'application $(b_1,\dots,b_n) \longmapsto \sum_{k=1}^{n} b_k 2^{k-1}$ est une bijection de $\{0,1\}^n$ vers $\llbracket 0, 2^n-1\rrbracket$.
Par ailleurs, il est clair que l'application $f \longmapsto \big(f(1),\dots,f(n)\big)$ est une bijection de $\mathcal{F}(\llbracket 1,n\rrbracket,\{0,1\})$ vers $\{0,1\}^n$. On en déduit, car la composée de plusieurs bijections est encore une bijection, que l'application $\phi : \mathrm{A} \longmapsto \sum_{k=1}^{n} \chi_{\mathrm{A}}(k) 2^{k-1}$ réalise une bijection entre $\mathscr{P}_n$ et $\llbracket 0, 2^n-1\rrbracket$.

Question 2.

a) On définit la fonction :

```
let rec ajoute k = function
  |  []  -> []
  | t::q -> (k::t)::(ajoute k q) ;;
```

ou si on préfère utiliser une fonctionnelle :

```
let ajoute k = map (function l -> k::l) ;;
```

b) On dispose de la relation : $\mathscr{P}_n = \mathscr{P}_{n-1} \cup (n \oplus \mathscr{P}_{n-1})$ (cette union étant disjointe). En effet, $\mathscr{P}_{n-1}$ est l'ensemble des parties de $\llbracket 1,n\rrbracket$ qui ne contiennent pas n, et $n \oplus \mathscr{P}_{n-1}$ celles qui le contiennent. Par ailleurs, $\mathscr{P}_0 = \{\emptyset\}$; on en déduit la fonction :

```
let rec parties1 = function
  | 0 -> [[]]
  | n -> let q = parties1 (n-1) in q @ (ajoute n q) ;;
```

Montrons par récurrence sur n que **`parties1 n`** retourne la liste des éléments de $\mathscr{P}_n$ rangés par ordre croissant.

- C'est clair si $n = 0$.
- Si $n \geqslant 1$, supposons le résultat acquis au rang $n-1$. Par hypothèse de récurrence, **`q`** est rangé par ordre croissant. Soit $\mathrm{A} \in \mathscr{P}_{n-1}$. Puisque $n \notin \mathrm{A}$ on a $\phi(n \oplus \mathrm{A}) = 2^{n-1} + \phi(\mathrm{A})$. De ceci il résulte que **`ajoute n q`** retourne une liste d'ensembles rangés par ordre croissant. Or pour tout $(\mathrm{A},\mathrm{B}) \in \mathscr{P}_{n-1} \times (n \oplus \mathscr{P}_{n-1})$,

$$\phi(\mathrm{A}) \leqslant \sum_{k=1}^{n-1} 2^{k-1} = 2^n - 1 < 2^n \leqslant \phi(\mathrm{B}).$$

Ceci montre que **`q @ (ajoute n q)`** est encore rangé par ordre croissant.

Question 3.

a) On obtient successivement :

$$\mathrm{A}_1 = \{4\},\ \mathrm{A}_2 = \{3,4\},\ \mathrm{A}_3 = \{2,3,4\},\ \mathrm{A}_4 = \{1,2,3,4\},\ \mathrm{A}_5 = \{1,3,4\},\ \mathrm{A}_6 = \{2,4\},\ \mathrm{A}_7 = \{1,2,4\},$$
$$\mathrm{A}_8 = \{1,4\},\ \mathrm{A}_9 = \{3\},\ \mathrm{A}_{10} = \{2,3\},\ \mathrm{A}_{11} = \{1,2,3\},\ \mathrm{A}_{12} = \{1,3\},\ \mathrm{A}_{13} = \{2\},\ \mathrm{A}_{14} = \{1,2\},$$
$$\mathrm{A}_{15} = \{1\},\ \mathrm{A}_{16} = \emptyset.$$

b) Montrons par récurrence sur $n \in \mathbb{N}^*$ que $\mathrm{A}_1,\dots,\mathrm{A}_{2^n}$ sont définis, que $\mathrm{A}_{2^n} = \emptyset$, et que $\mathscr{P}_n = \left\{\mathrm{A}_p \;\middle|\; p \in \llbracket 1, 2^n\rrbracket\right\}$.

– C'est clair si $n = 1$ car alors $A_1 = \{1\}$, $A_2 = \emptyset$.

– Si $n \geqslant 2$, supposons le résultat acquis au rang $n-1$. On a $A_1 = \{n\}$ et $A_2 = \{n-1, n\} = A'_1 \cup \{n\}$, en notant $A'_1 = \{n-1\}$. Par hypothèse de récurrence, $A'_1, \ldots, A'_{2^{n-1}-1}$ sont définis, constituent les éléments de $\mathscr{P}_{n-1} \setminus \{\emptyset\}$, et $A'_{2^{n-1}-1} = \{1\}$. On en déduit que $A_1, \ldots, A_{2^{n-1}}$ sont définis, constituent les éléments de $n \oplus \mathscr{P}_{n-1}$, et $A_{2^{n-1}} = \{1, n\}$. Mais alors $A_{2^{n-1}+1} = \{n-1\}$, et en appliquant de nouveau l'hypothèse de récurrence, on en déduit que $A_{2^{n-1}+1}, \ldots, A_{2^{n-1}+2^{n-1}}$ sont définis, constituent les éléments de $\mathscr{P}_{n-1}$, et $A_{2^n} = \emptyset$.
Sachant que $\mathscr{P}_{n-1} \cup (n \oplus \mathscr{P}_{n-1}) = \mathscr{P}_n$, on peut conclure quant au résultat au rang n.

Question 4.

a) Il est naturel d'opérer par filtrage :

```
let suivant = function
 |        [] -> failwith "suivant"
 |       [1] -> []
 |       [n] -> [n-1;n]
 |   1::t::q -> (t-1)::q
 | t::q as l -> (t-1)::l ;;
```

b) La fonction qui suit engendre la liste $(A_1 = \{n\}, A_2, A_3, \ldots, A_{2^n} = \emptyset)$

```
let parties2 n =
  let rec aux = function
    | [] -> [[]]
    | a -> a::(aux (suivant a))
  in aux [n] ;;
```

Partie II. Génération des permutations

Question 5.

a) $c_i(\sigma)$ est le nombre d'entiers j strictement supérieurs à i qui apparaissent avant i dans la liste $\langle \sigma(1), \ldots, \sigma(n) \rangle$. Ainsi :

$$\sigma = \langle 2, 1, 4, 5, 3 \rangle \Rightarrow c(\sigma) = (1, 0, 2, 0, 0).$$

b) La somme $\displaystyle\sum_{i=1}^{n} c_i(\sigma)$ représente le nombre de couples $(i, j) \in [\![1, n]\!]^2$ tel que $i < j$ et $\sigma^{-1}(j) < \sigma^{-1}(i)$. Mais σ est une bijection, donc l'application $(i, j) \mapsto \big(\sigma(i), \sigma(j)\big)$ est une bijection de $[\![1, n]\!]^2$ dans lui-même, et en posant $(i', j') = \big(\sigma(i), \sigma(j)\big)$, on a :

$$\Big(i < j \quad \text{et} \quad \sigma^{-1}(j) < \sigma^{-1}(i)\Big) \iff \Big(j' < i' \quad \text{et} \quad \sigma(i') < \sigma(j')\Big).$$

Ainsi, $\displaystyle\sum_{i=1}^{n} c_i(\sigma)$ est le nombre d'inversions de σ.

Question 6.

a) On obtient successivement : $\ell_5 = \langle 5 \rangle$, $\ell_4 = \langle 5, 4 \rangle$, $\ell_3 = \langle 5, 3, 4 \rangle$, $\ell_2 = \langle 5, 3, 2, 4 \rangle$, $\ell_1 = \langle 5, 3, 2, 4, 1 \rangle$.

b) Il est clair que ℓ_1 est une liste de longueur n formée des n premiers entiers naturels non nuls, donc représente une permutation. Montrons pour tout $k \in [\![1, n]\!]$ que $c_k(\ell_1) = \gamma_k$.

– Si $k = 1$, on sait que ℓ_1 est de la forme $\langle a_1, \ldots, a_{\gamma_1}, 1, a_{\gamma_1+1}, \ldots, a_{n-1} \rangle$, chaque a_i étant supérieur ou égal à 2 (1 vient d'être inséré après l'élément d'ordre γ_1). On a donc bien $c_1(\ell_1) = \gamma_1$.

– Si $k > 1$, la définition de c_k montre que l'on peut supprimer de la liste ℓ_1 les entiers inférieurs strictement à k sans modifier la valeur de $c_k(\ell_1)$. Autrement dit, $c_k(\ell_1) = c_k(\ell_k)$, et cette dernière valeur, comme pour le cas $k = 1$, vaut clairement γ_k.

Ainsi, $c(\ell_1) = (\gamma_1, \ldots, \gamma_n)$.

c) Nous venons de montrer que l'application $d : \gamma \mapsto \ell_1$ vérifie : $c \circ d = \mathrm{Id}_{K_n}$; autrement dit, elle est injective. Sachant que $\operatorname{card} \mathfrak{S}_n = n! = \operatorname{card} K_n$, il s'agit en fait d'une bijection, et $c = d^{-1}$.

Question 7.

a) On définit :

```
let rec insere a j = function
  | l when j = 0 -> a::l
  | []           -> failwith "insere"
  | t::q         -> t::(insere a (j-1) q) ;;
```

b) Nous allons utiliser une fonction auxiliaire **`aux k`** qui calcule la liste ℓ_k lorsque $k \in [\![1, n]\!]$.

```
let genere c =
  let n = vect_length c in
  let rec aux = function
    | k when k = n -> [n]
    | k            -> let l = aux (k+1) in insere k c.(k-1) l
  in aux 1 ;;
```

On notera que c_k est stocké dans la case **`c.(k-1)`** car les vecteurs sont indexés à partir de 0.

Question 8.

a) On peut bien entendu procéder par récurrence, ou bien faire un calcul direct par télescopage :

$$\sum_{k=0}^{j} k.k! = \sum_{k=0}^{j} (k+1-1).k! = \sum_{k=0}^{j} ((k+1)! - k!) = (j+1)! - 1.$$

b) Si $p \in [\![1, n!-1]\!]$, il existe un unique entier $j \in [\![1, n-1]\!]$ tel que $j! \leqslant p \leqslant (j+1)!-1$. Une condition nécessaire d'existence de la décomposition est que $d_{j+1} = \cdots = d_n = 0$ car si l'une de ces valeurs est non nulle la somme dépasse $(j+1)!$.

Par ailleurs, si une telle décomposition existe on a : $p = d_j j! + \sum_{k=1}^{j-1} d_k k!$, et la question précédente montre que $0 \leqslant \sum_{k=1}^{j-1} d_k k! \leqslant j! - 1$. Ceci prouve que d_j est le quotient de la division euclidienne de p par $j!$, soit $d_j = \left\lfloor \frac{p}{j!} \right\rfloor$.

c) Ce qui précède assure l'unicité d'une telle décomposition, à supposer qu'elle existe. Nous allons prouver son existence en raisonnant par récurrence sur p.

- Si $p = 0$, on a clairement $d_1 = \cdots = d_{n-1} = 0$.
- Si $p > 0$, supposons le résultat acquis jusqu'au rang $p-1$. Par hypothèse de récurrence, $p - \left\lfloor \frac{p}{j!} \right\rfloor j!$ (l'entier j étant celui défini à la question précédente) admet une unique décomposition $\sum_{k=0}^{j-1} d_k k!$, et en posant $d_j = \left\lfloor \frac{p}{j!} \right\rfloor$, on obtient bien $p = \sum_{k=1}^{j} d_k k!$ avec $d_j < \frac{(j+1)!}{j!} = j+1$.

Question 9.

a) On définit la fonction :

```
let rec factorielle = function
  | 0 -> 1
  | n -> n * factorielle (n-1) ;;
```

b) Pour tout $j \in [\![1, n]\!]$, posons $u_{j-1} = j u_j + \tilde{d}_{j-1}$ avec $\tilde{d}_{j-1} \in [\![0, j-1]\!]$ (autrement dit, $\tilde{d}_{j-1}$ est le reste de la division euclidienne de u_{j-1} par j). Il s'agit de prouver que $\tilde{d} = d$.

On a $(j-1)!u_{j-1} = j!u_j + \tilde{d}_{j-1}(j-1)!$ donc par télescopage $u_0 - n!u_n = \sum_{j=1}^{n} \tilde{d}_{j-1}(j-1)! = \sum_{k=0}^{n-1} \tilde{d}_k k!$.

Sachant que $u_0 = p$, il suffit de prouver que $u_n = 0$ et d'invoquer l'unicité de la décomposition établie à la question 8 pour en déduire que $\tilde{d} = d$.

Or il est aisé d'établir par récurrence que $0 \leqslant u_j \leqslant \frac{n!-1}{j!}$, donc $0 \leqslant u_n \leqslant 1 - \frac{1}{n!}$ et s'agissant d'un entier, $u_n = 0$.

On en déduit la fonction :

```
let lehmer n p =
  let d = make_vect n 0 in
  let u = ref p in
  for j = 1 to n - 1 do
    u := !u / j ;
    d.(n-1-j) <- !u mod (j+1)
  done ;
  d ;;
```

Question 10. La question 8 établit une bijection entre un entier $p \in [\![0, n!-1]\!]$ et un n-uplet $(d_{n-1}, d_{n-2}, \ldots, d_1, d_0) \in \mathrm{K}_n$. Compte tenu de la question 6, l'application $f : p \longmapsto c^{-1}(d_{n-1}, \ldots, d_1, d_0)$ réalise une bijection entre $[\![0, n!-1]\!]$ et $\mathfrak{S}_n$. Il s'agit donc d'énumérer les permutations $f(0), f(1), \ldots, f(n!-1)$.

```
let permutations n =
  let fact = factorielle n in
   let rec aux = function
     | p when p = fact -> []
     | p               -> let q = aux (p+1)
                          and t = genere (lehmer n p)
                          in t::q
   in aux 0 ;;
```