Mpsi 1 - Corrigé du TP sur les polynômes lacunaires

Question 1) Rédigez vos fonctions concernant la somme de deux polynômes lacunaires.

Question 2) Même question avec le produit de deux polynômes lacunaires.

Corrigé

Question 1) Somme d'un monôme non nul m et d'un polynôme lacunaire p.

Le principe est simple : on parcourt tous les monômes m' du polynôme p (liste de monômes de degrés tous distincts). Si aucun monôme m' n'a le même degré que m, alors on ajoute m à la liste. Sinon, il existe un monôme m' de p de même degré que m : on les additionne, on obtient un monôme m''. Si $m'' = 0$, alors on supprime m' de la liste p, sinon on remplace m' par m''.

```
let rec somme_mono_poly m p =
    match p with
    | [] -> [m]
    | a :: q -> if a.deg = m.deg then
            ( let c = a.coef + m.coef in
              if c = 0 then q
              else {deg=a.deg;coef=c} :: q )
            else a :: somme_mono_poly m q;;
```

On prouve la justesse de l'algorithme par récurrence sur la longueur de la liste p. Si m est un monôme informatique {deg = d; coef = c} , je note $P(m)$ le monôme mathématique correspondant cX^d. De même, si p est un polynôme informatique : $p = [m_0;\ ...;\ m_{n-1}]$ (liste de monômes), je note $P(p)$ le poynôme mathématique qui lui correspond : $P(p) = \sum_{k=0}^{n-1} P(m_k)$. On veut donc montrer que si p' est la liste obtenue après application de la fonction aux monôme m et liste p, on obtient $P(p') = P(m) + P(p)$.

Si p est la liste vide (ce qui correspond au polynôme nul), alors on obtient la liste contenant le monôme m, ce qui correspond bien à la somme du polynôme nul et d'un monôme (non nul). Donc l'algorithme est correct pour les liste de longueur 0.

Si l'algorithme est correct pour les listes de longueur n, alors soit p une liste de longueur $n+1$, on considère le premier monôme a de la liste p et q la queue de cette liste. On a donc $P(p) = P(a) + P(q)$.

Si a et m ont le même degré d, alors l'algorithme calcule la somme c des coefficients de a et de m, c'est-à-dire le coefficient de $P(a) + P(m)$. Donc $P(a) + P(m) = cX^d$.
- Si $c = 0$, alors l'algorithme retourne la liste q, donc $P(p') = P(q) = P(p) - P(a)$. Or dans ce cas, on a $P(a) + P(m) = 0$, donc $P(m) = -P(a)$ donc $P(p') = P(p) + P(m)$.
- Si $c \neq 0$, alors l'algorithme retourne la liste q précédée du monôme μ ={deg=d; coeff=c}, donc $P(p') = P(\mu) + P(q) = cX^d + P(q) = P(m) + P(a) + P(q) = P(m) + P(p)$.

Si a et m n'ont pas le même degré, alors l'algorithme fait un appel récursif sur une liste de longueur n : par hypothèse de récurrence, le résultat de cet appel récursif est correct, on note π la liste ainsi obtenue, ce qui donne $P(\pi) = P(m) + P(q)$. On a donc $P(p') = P(a) + P(\pi) = P(a) + P(m) + P(q) = P(m) + P(p)$.

Dans tous les cas, on a bien $P(p') = P(m) + P(p)$, donc l'algorithme est correct pour les listes de longueur $n+1$.

D'après le principe de récurrence, l'algorithme est donc correct pour toutes les listes.

On notera que la liste obtenue en sortie de l'algorithme est aussi constituée de monômes de degré distincts, ce qui se démontre par récurrence de la même façon.

Somme de deux polynômes lacunaires.

L'algorithme de somme de deux polynômes en découle alors naturellement. On considère tous les monômes de p qu'on ajoute successivement à la liste q par l'algorithme précédent.

```
let rec somme p q =
    match p with
    | [] -> q
    | a :: r -> let s = somme_mono_poly a q in
            somme r s;;
```

La preuve est du même type. Si $p = []$, alors pour toute liste q, on retourne q, ce qui est bien la liste correspondant à la somme du polynôme nul et du polynôme $P(q)$.

Si l'algorithme est correct pour les listes p de longueur n et pour toutes les listes de longueur q, alors on écrit $P(p) = P(a) + P(r)$ et on effectue un appel récursif que la liste r, qui est de longueur n : par hypothèse de récurrence, la liste obtenue p' obtenue vérifie $P(p') = P(r) + P(s)$. Or $P(s) = P(a) + P(q)$ donc $P(p') = P(r) + P(a) + P(q) = P(p) + P(q)$, ce qui prouve la correction de l'algorithme pour les listes p de longueur $n+1$.

D'après le principe de récurrence, l'algorithme est donc correct pour toutes les listes p (et toutes les listes q).

Dans le premier algorithme, on voit que dans le pire cas, il faudra parcourir toute la liste p et que toutes les opérations autres que l'appel récursif sont de complexité $O(1)$, donc on en déduit qu'il est de complexité $O(n)$ (où n est la longueur de la liste p).

Dans le second algorithme, je note n, n' les longueurs des listes p et q respectivement et $C(n, n')$ le coût de calcul de la somme de p et q. On constate que $C(0, n') = 1$ et $C(n, n') \leqslant O(n') + C(n-1, n'+1)$, car s est de longueur au plus $n'+1$. Donc par une récurrence simple, on en déduit que $C(n, n') = O(n^2 + nn') = O((n+n')^2)$.

Question 2) Produit d'un monôme m et d'un polynôme lacunaire p.

L'algorithme est encore plus simple que pour la somme, puisqu'il n'y a plus à supprimer de monômes... La preuve se fait de la même façon. Et encore une fois, la liste obtenue est constituée de monômes de degrés distincts.

```
let rec produit_mono_poly m p =
    match p with
    | [] -> []
    | a :: q -> let c = a.coef and d = a.deg in
        {deg=(d + m.deg); coef=(c * m.coef)} :: produit_mono_poly m q;;
```

Produit de deux polynômes lacunaires.

```
let rec produit p q =
    match p with
    | [] -> []
    | a :: r -> let s = produit_mono_poly a q in
            somme s (produit r q );;
```

L'algorithme est correct pour la liste vide, car cela revient à multiplier par le polynôme nul.

Si l'algorithme est correct pour les listes p de longueur n, alors soit p une liste de longueur $n+1$. On écrit $P(p) = P(a) + P(r)$, puis l'algorithme calcule s telle que $P(s) = P(a) \times P(q)$, puis enfin calcule la liste finale p' telle que $P(p') = P(s) + P(\text{produit } r\ q)$. Par hypothèse de récurrence, comme r est de longueur n, on a $P(\text{produit } r\ q) = P(r) \times P(q)$, donc $P(p') = P(s) + P(r)P(q) = P(a)P(q) + P(r)P(q) = (P(a) + P(r)) \times P(q) = P(p)P(q)$, ce qui prouve que l'algorithme est correct pour les listes p de longueur $n+1$.

D'après le principe de récurrence, l'algorithme est toujours correct.

On notera que la propriété des degrés tous distincts est toujours vérifiée et que la longueur de la liste obtenue est au plus $n + n'$.

La complexité du premier algorithme produit d'un monôme et d'un polynôme est manifestement linéaire en la longueur de q.

Si on note $D(n, n')$ le coût de calcul du produit de p et q, alors on a $D(0, n') = 1$ et $D(n, n') = O(n') + D(n-1, n') + C(n', (n-1) + n')$ (avec les notations précédentes : dans l'ordre, coût du calcul de s, de celui du produit de r et q, de la somme de s et de ce produit), ce qui donne $D(n, n') = O(n') + D(n-1, n') + O(n'^2 + n'n) = O(n'^2 + nn') + D(n-1, n')$, donc on obtient $D(n, n') = O(n^2 n' + nn'^2)$.