# 1 Les mots

1. `Étude d'un morphisme de monoïde, mots de Fibonacci`

   (a) Soit $m = c_1c_2...c_n \in A^*$ et $m' = c'_1c'_2...c'_n \in A^*$. Par définition, $\beta(m) = \beta(c_1c_2...c_n) = \alpha(c_1)\alpha(c_2)...\alpha(c_n)$ et $\beta(m') = \beta(c'_1c'_2...c'_n) = \alpha(c'_1)\alpha(c'_2)...\alpha(c'_n)$. De la même façon, $\beta(mm') = \alpha(c_1)\alpha(c_2)...\alpha(c_n)\alpha(c'_1)\alpha(c'_2)...\alpha(c'_n)$, donc
   $\beta(mm') = \alpha(c_1)\alpha(c_2)...\alpha(c_n)\alpha(c'_1)\alpha(c'_2)...\alpha(c'_n) = \beta(m)\beta(m')$.

   (b) Montrons par récurrence sur $n$ que pour tout $n \in \mathbb{N}$, $f_n$ est un préfixe de $f_{n+1}$.

   (I) pour $n = 0$ : $f_0 = d$ et $f_1 = \beta(f_0) = \alpha(d) = dv$. Le mot $f_0$ est donc bien un préfixe de $f_1$.

   (H) Soit $n \in \mathbb{N}^*$ tel que la propriété soit vraie pour $n-1$. Par construction, on a $f_{n+1} = \beta(f_n)$. Or, $f_{n-1}$ est un préfixe de $f_n$, donc il existe $w \in A^*$ tq $f_n = f_{n-1}w$. On obtient donc $f_{n+1} = \beta(f_{n-1}w) = \beta(f_{n-1})\beta(w) = f_n\beta(w)$. $f_n$ est donc bien un préfixe de $f_{n+1}$.

   (C) La propriété est donc initialisée et héréditaire, elle est donc vraie pour tout $n \in \mathbb{N}$.

   (c) $f_1 = 01$, $f_2 = 010$, $f_3 = 01001$, $f_4 = 01001010$
   Montrons par récurrence sur $n$ que pour tout $n \in \mathbb{N}$, $f_{n+2} = f_{n+1}f_n$.

   (I) pour $n = 0$ : on a bien $f_2 = 010 = f_1f_0$.

   (H) Soit $n \in \mathbb{N}$ tel que la propriété soit vraie pour $n$. Par construction, on a $f_{n+3} = \beta(f_{n+2})$, donc $f_{n+3} = \beta(f_{n+1}f_n) = \beta(f_{n+1})\beta(f_n) = f_{n+2}f_{n+1}$. La propriété est donc vraie au rang $n+1$.

   (C) La propriété est donc initialisée et héréditaire, elle est donc vraie pour tout $n \in \mathbb{N}$.

   (d) Montrons par récurrence sur $n$ que pour tout $n \in \mathbb{N}^*$, $f_n$ se termine par 01 si $n$ est impair et par 10 si $n$ est pair.

   (I) pour $n = 1$ ou $n = 2$ : on a $f_1 = 01$ et $f_2 = 010$. La propriété est bien initialisée.

   (H) Soit $n \in \mathbb{N}^*$ tel que la propriété soit vraie pour $n$ et $n+1$. Par construction, on a $f_{n+2} = f_{n+1}f_n$. La fin de $f_{n+2}$ est bien la même que la fin de $f_n$. Or $n+2$ et $n$ ont la même parité. La propriété est donc vraie au rang $n+1$.

   (C) La propriété est donc initialisée et héréditaire, elle est donc vraie pour tout $n \in \mathbb{N}$.

   (e) Montrons par récurrence sur $n$ que pour tout $n \in \mathbb{N}^*$, que la propriété est vraie.

   (I) pour $n \in [\![1;4]\!]$ : on a $g_1 = \varepsilon$, $g_2 = 0$, $g_3 = 010$ et $g_4 = 010010$. La propriété est bien initialisée.

   (H) Soit $n \geq 2$ tel que la propriété soit vraie pour $n-1$, $n$ et $n+1$. Par construction, on a $f_{n+2} = f_{n+1}f_n = f_nf_{n-1}f_n$.
   - si $n$ est impair, $f_{n+2} = g_n01g_{n-1}10g_n01$
   - si $n$ est pair, $f_{n+2} = g_n10g_{n-1}01g_n10$

   Dans les deux cas, $g_n$ et $g_{n-1}$ étant des palindromes, la propriété est bien vraie au rang $n+1$.

(C) La propriété est donc initialisée et héréditaire, elle est donc vraie pour tout $n \in \mathbb{N}$.

2. Étude de la fermeture réflexive, symétrie et transitive d'une relation

(a) Soit $m \in A^*$. Soit $\mathcal{E} = \{m' \in A^*; m\mathscr{S}^*m' \text{ et } |m| > |m'|\}$. Cet ensemble est fini, donc on peut fixer un $m_0 \in \mathcal{E}$ tel que $m_0$ soit de longueur minimale.

Supposons par l'absurde que $m_0$ ne soit pas de la forme $b^ia^j$. Dans ce cas, $m_0$ contient un facteur $ab$, c'est-à-dire qu'il existe $u \in A^*$ et $v \in A^*$ tels que $m_0 = uabv$. Ceci implique que $m_0\mathscr{S}uv$ et donc que $m\mathscr{S}^*uv$. Or $|uv| < |m_0|$, ce qui est absurde. Ainsi, on a prouvé que $m_0$ est bien de la forme attendue.

(b) Soit $m \in A^*$ et $w \in A^*$ tels que $m\mathscr{R}w$. Alors il existe $u \in A^*$ et $v \in A^*$ tq $m = uabv$ et $w = uv$.

Ainsi :

- $\forall k \leq |u|, |m_k|_b - |m_k|_a = |w_k|_b - |w_k|_a$ ;
- $|m_{|u|+1}|_b - |m_{|u|+1}|_a \leq |w_{|u|}|_b - |w_{|u|}|_a$ ;
- $|m_{|u|+2}|_b - |m_{|u|+2}|_a = |w_{|u|}|_b - |w_{|u|}|_a$ ;
- $\forall |u| \leq k \leq |v|, |m_k|_b - |m_k|_a = |w_{k+2}|_b - |w_{k+2}|_a$.

On a donc $\max_k\{|m_k|_b - |m_k|_a\} = \max_k\{|w_k|_b - |w_k|_a\}$. Cela est vrai pour tout $m \in A^*$ et $w \in A^*$ tels que $m\mathscr{R}w$. Par définition de $\mathscr{S}$ et de $\mathscr{S}^*$, l'égalité est vraie également pour tout $m \in A^*$ et $w \in A^*$ tels que $m\mathscr{S}w$ ou $m\mathscr{S}^*w$.

Soit $m \in A^*$, d'après la question précédente, il existe $(i,j) \in \mathbb{N}^2$ tq $m\mathscr{S}^*b^ia^j$. On sait à présent que $\max_k\{|m_k|_b - |m_k|_a\} = \max_k\{|(b^ia^j)_k|_b - |(b^ia^j)_k|_a\}$. Or $\max_k\{|(b^ia^j)_k|_b - |(b^ia^j)_k|_a\} = i$. Donc $i$ est unique.

De la même façon, si on pose $d_m = |m|_a - |m|_b$, la quantité est préservée par les relations $\mathscr{R}$, $\mathscr{S}$ et $\mathscr{S}^*$. Ainsi, on a forcément $j = d_m + i$. Le couple $(i,j)$ est donc bien unique.

3. `Relations d'ordre dans` $A^*$

(a) i. Soit $(u,v,w) \in (A^*)^3$.

- réflexivité : $u$ est bien un préfixe de $u$, donc on a $u \leq_p u$.
- antisymétrie : si $u \leq_p v$, $u$ est un préfixe de $v$ ; si en plus $v \leq_p u$, alors en particulier, $|v| \leq |u|$. Ainsi, $u$ est un préfixe de $v$ qui fait la même longueur que $v$. On a donc bien $u = v$.
- transitivité : tout préfixe d'un préfixe d'un mot est un préfixe de ce mot ; ainsi, si $u \leq_p v$ et $v \leq_p w$, alors $u \leq_p w$.

Dès que $A$ a au moins deux lettres l'ordre n'est pas total. Par exemple $a$ et $b$ ne sont pas préfixes de l'un ou de l'autre.

ii.
```
let plusPetit u v =
  let nu = String.length u in
  let nv = String.length v in
  let rec parcours i =
    if i = nu
    then 1
    else (if i = nv
          then 0
          else (if u.[i] <> v.[i]
                then 0
                else parcours (i+1)))
  in
  parcours 0;;
```

(b) i. Soit $(u, v, w) \in (A^*)^3$.

- réflexivité : $u$ est bien un préfixe de $u$, donc on a $u \leq_l u$.
- antisymétrie : supposons que $u \leq_l v$ et $v \leq_l u$, et supposons par l'absurde que $\neg(u \leq_l v)$. Soit $k$ le rang de la première lettre qui diffère entre $u$ et $v$.
  - Si $k$ n'existe pas, alors $u$ est un préfixe de $v$ car $u \leq_l v$ et $v$ est un préfixe de $u$ car $v \leq_l u$, donc on a bien $u = v$.
  - Si $u_k < v_k$, alors on ne peut pas avoir $v \leq_l u$ ; si $v_k < u_k$, alors on ne peut pas avoir $v \leq_l u$. Ainsi, comme $\leq$ est totale, on a $u_k = v_k$ ce qui est absurde. Les mots $u$ et $v$ sont donc égaux.
- transitivité : Soit $u, v$ et $w$ dans $A^\star$ tels que $u \leq_l v$ et $v \leq_l w$. On veut montrer que $u \leq_l w$. Remarquons pour commencer que si $u = v$ ou $v = w$ le résultat est vrai. On peut donc supposer que $u$ est strictement inférieur à $v$ et que $v$ est strictement inférieur à $w$. Il existe donc $i$ tel que $u_i < v_i$ et $u_k = v_k$ pour $k < i$ (avec la convention que $\emptyset \leq a$ pour toute lettre $a$ pour gérer le cas où $u$ est un préfixe de $v$ ou la cas où $v$ est un préfixe de $w$). De même il existe $j$ tel que $v_j < w_j$ et $v_k = w_k$ pour $k < j$. On en déduit qu'en posant $l$ le minimum de $i$ et de $j$, $u_k = w_k$ pour $k < l$ et $u_l < w_l$. Cela implique bien que $u \leq_l w$.

Prouvons par l'absurde que cet ordre est total. Soit $(u, v) \in (A^*)^2$, supposons par l'absurde que l'on a ni $u \leq_l v$, ni $v \leq_l u$. Dans ce cas, $u$ n'est pas un préfixe de $v$ et $v$ n'est pas un préfixe de $u$. On peut donc décomposer $u$ et $v$ comme dans la définition et l'ordre que les lettres étant total, on obtient $u \leq_l v$ ou $v \leq_l u$.

ii.
```
let lexico u v =
  let nu = String.length u in
  let nv = String.length v in
  let rec parcours i =
    if i = nu
    then 1
    else (if i = nv
          then 0
          else (if u.[i]=v.[i]
                then parcours (i+1)
```

```
                    else (if u.[i]<v.[i]
                          then 1
                          else 0)))
    in
    parcours 0;;
```

(c) i. On veut montrer que $\prec$ est une relation d'ordre.

- Reflexivité : Pour $u \in A^\star$, $|u| = |u|$ et $u \leq_l u$ donc $u \prec u$.
- Anti-symétrie : Soit $u, v$ dans $A^\star$ tels que $u \prec v$ et $v \prec u$. Comme $u \prec v$ implique que $|u| \leq |v|$ et que $v \prec u$ implique $|v| \leq |u|$ alors $|u| = |v|$. Cela implique que $u \leq_l v$ et $v \leq_l u$. En utilisant ce qui précède, on en déduit que $u = v$.
- Transitivité : Soit $u, v$ et $w$ dans $A^\star$ tels que $u \prec v$ et $v \prec w$. Là encore, on a $|u| \leq |v| \leq |w|$. Si l'une des deux inégalités est stricte, on a bien $u \prec w$. D'autre part, si $|u| = |v| = |w|$ alors $u \leq_l v$ et $v \leq_l w$ donc $u \leq_l w$ d'après ce qui précède et finalement $u \prec w$.

L'ordre est total.

ii.
```
let radiciel u v =
  let nu = String.length u in
  let nv = String.length v in
  if nu < nv
  then 1
  else (if nv < nu
        then 0
        else lexico u v)
;;
```

iii. Étant donnée une longueur de mot $k$, il y a seulement $|A|^k$ mots de cette longueur. Or si l'on prend un mot $w$, seuls des mots de longueur inférieure ou égale à celle de $w$ peuvent lui être inférieur pour cette relation d'ordre. Ainsi, si l'on prend un mot $w$ de longueur $k$, il y a moins de $\sum_{i=0}^{k} |A|^i$ mots inférieurs à $w$. Cet ordre est donc bien fondé.

4. `Distances dans` $A^*$

(a) Soit $u, v, w \in A^*$.

- $|u| \geq |\texttt{plpc}(u,v)|$ et $|v| \geq |\texttt{plpc}(u,v)|$, donc on a bien $d(u,v) \geq 0$.
- $u = v$ ssi ($|u| = |\texttt{plpc}(u,v)|$ et $|v| = |\texttt{plpc}(u,v)|$) ssi $d(u,v) = 0$.
- Par définition, $|\texttt{plpc}(u,v)| = |\texttt{plpc}(v,u)|$. Donc $d(u,v) = |u|+|v|-2|\texttt{plpc}(u,v)| = |v| + |u| - 2|\texttt{plpc}(v,u)| = d(v,u)$.
- $\Delta = d(u,v)+d(v,w)-d(u,w) = |u|+|v|-2|\texttt{plpc}(u,v)|+|v|+|w|-2|\texttt{plpc}(v,w)|-(|u| + |w| - 2|\texttt{plpc}(u,w)|)$
  Donc $\Delta = |v| - 2|\texttt{plpc}(u,v)| + |v| - 2|\texttt{plpc}(v,w)| + 2|\texttt{plpc}(u,w)|$.
  Or $\min(|\texttt{plpc}(u,v)|, |\texttt{plpc}(v,w)|) \leq |\texttt{plpc}(u,w)|$ et $\max(|\texttt{plpc}(u,v)|, |\texttt{plpc}(v,w)|) \leq |v|$.
  Ainsi $\Delta \geq 0$, ce qui implique $d(u,v) + d(v,w) \geq d(u,w)$.

(b) i. Soit $u, v, w \in A^*$.

- par définition, $\delta(u, v) \geq 0$ ;
- $\delta(u, v) = 0$ ssi aucune opération élémentaire n'est nécessaire pour passer de $u$ à $v$, donc ssi $u = v$ ;
- si en enchaînant des suppressions de lettres, des ajouts de lettres et des remplacements d'une lettre par une autre pour passer de $u$ à $v$, on peut passer de $v$ à $u$ avec le même nombre d'opérations, en les faisant dans l'ordre inverse et en faisant des suppressions à la place des ajouts et des ajouts à la place des suppressions. On obtient ainsi l'égalité $\delta(u, v) = \delta(v, u)$.
- si on passe de $u$ à $v$ avec $n$ opérations et de $v$ à $w$ avec $m$ opérations, on peut enchaîner les deux suites d'opérations et passer de $u$ à $w$ en $n + m$ opérations.

ii.
```
let distance u v =
  let nu = String.length u in
  let nv = String.length v in
  let d = Array.make_matrix nu nv 0 in
  if u.[0] <> v.[0] then d.(0).(0) <- 1;
  for i = 1 to nu-1 do
    if u.[i] = v.[0] then d.(i).(0) <- i
    else d.(i).(0) <- d.(i-1).(0) + 1
  done;
  for j = 1 to nv-1 do
    if u.[0] = v.[j] then d.(0).(j) <- j
    else d.(0).(j) <- d.(0).(j-1) + 1
  done;
  for i = 1 to nu-1 do
    for j = 1 to nv-1 do
      d.(i).(j) <- min d.(i-1).(j) (min d.(i).(j-1) d.(i-1).(j-1)) +1;
      if u.[i] = v.[j] then d.(i).(j) <- min d.(i).(j) d.(i-1).(j-1);
    done;
  done;
  d.(nu-1).(nv-1)
;;
```

5. (a)
   - nombre de préfixes : $n + 1$ en comptant le mot vide et le mot lui-même. ($n - 1$ préfixes propres)
   - nombre de suffixes : $n + 1$ en comptant le mot vide et le mot lui-même. ($n - 1$ suffixes propres)
   - nombre de facteurs : $\binom{n+1}{2} = (n + 1) * n/2$, on coupe le mot en deux endroits et on garde le facteur du milieu.

   (b)
   - nombre de facteurs non-vides de $a^m$ : $m$ en comptant le mot vide et le mot lui-même.
   - nombre de facteurs non-vides de $b^n$ : $n$
   - nombre de facteurs contenant au moins un $a$ et un $b$ : $m * n$

   En ajoutant le mot vide, on obtient $m + n + m * n + 1 = (m + 1) * (n + 1)$ en tout.

6. `Lemme de Lévi`

(a) Soit $u, v, x, y \in A^*$ et notons $u = u_1...u_{|u|}$, $v = v_1...v_{|v|}$, $x = x_1...x_{|x|}$ et $y = y_1...y_{|y|}$.

$\Rightarrow$ Supposons que $uv = xy$.

- Si $|u| = |x|$, alors $u_1...u_{|u|}v_1...v_{|v|} = uv = xy = x_1...x_{|x|}y_1...y_{|y|}$ implique directement $u = u_1...u_{|u|} = x_1...x_{|x|} = x$ et $v = v_1...v_{|v|} = y_1...y_{|y|} = y$. Donc en prenant $t = \varepsilon$, on obtient la propriété attendue.
- Si $|u| > |x|$, alors $u = u_1...u_{|u|} = x_1...x_{|x|}y_1...y_{|u|-|x|} = xy_1...y_{|u|-|x|}$. On a donc égalité entre les facteurs $u_{|x|+1}...u_{|u|}$ et $y_1...y_{|u|-|x|}$. En nommant $t$ ce facteur, on obtient bien $u = xt$ et $y = tv$.
- Si $|u| < |x|$, le raisonnement est le même en inversant les notations $u, v$ et $x, y$.

$\Leftarrow$ Supposons qu'il existe $t \in A^*$ tel que $u = xt$ et $tv = y$. Alors $uv = xtv = xy$. De la même façon, s'il existe $t \in A^*$ tel que $ut = x$ et $v = ty$, alors $uv = uty = xy$.

On a donc bien l'équivalence annoncée.

(b) Procédons par double implication.

$\Leftarrow$ Supposons qu'il existe $w \in A^*$ et $p, q \in \mathbb{N}$ tq $u = w^p$ et $v = w^q$. Alors $uv = w^pw^q = w^{p+q} = w^{q+p} = w^qw^p = vu$, donc $u$ et $v$ commutent.

$\Rightarrow$ Prouvons par récurrence forte sur le maximum des longueurs de $u$ et $v$ que si $u$ et $v$ commutent alors il existe $w \in A^*$ et $p, q \in \mathbb{N}$ tq $u = w^p$ et $v = w^q$.

(I) si $\max(|u|, |v|) = 0$, alors $u = v = \varepsilon$ et $w, p, q = \varepsilon, 1, 1$ conviennent.

(H) soit $n \in \mathbb{N}$ tel que la propriété soit vraie pour tout $k \leq n$.
Soit $u, v \in A^*$ tels que $\max(|u|, |v|) = n + 1$ et notons $u = u_1...u_{|u|}$ et $v = v_1...v_{|v|}$.

- Si $|u| = |v|$, alors $uv = vu$ implique $u = v$, donc en posant $w = u$ et $p = q = 1$, on obtient bien la propriété attendue.
- Si $\min(|u|, |v|) = 0$, alors les deux propriétés de l'implication sont toujours vraies. L'implication est donc bien vraie.
- Étudions à présent le cas où $u$ et $v$ sont de longueurs différentes et strictement positives. On suppose sans perte de généralité que $|u| > |v|$, alors $|u| = n + 1$.
  Supposons que $uv = vu$ alors il existe $t \in A^*$ tel que $|t| > 0$ et $u = tv$. De plus $tvv = uv = vu = vtv$, donc $v$ et $t$ commutent. Or $0 < |v| < |u| = n + 1$, donc on a également $0 < |t| < |u| = n + 1$. Ainsi, par hypothèse de récurrence, il existe $w \in A^*$ et $p', q \in \mathbb{N}$ tq $t = w^{p'}$ et $v = w^q$.
  En posant $p = p' + q$, on obtient enfin l'existence de $w \in A^*$ et $p, q \in \mathbb{N}$ tq $u = tv = w^{p'}w^q = w^p$ et $v = w^q$.

La propriété est donc héréditaire.

Par le principe de récurrence, la propriété est donc vraie pour tout $n \in \mathbb{N}$, donc pour tout couple $u, v \in A^*$.

L'équivalence annoncée est donc vraie.

# 2 Les langages

7. `Lemme d'Arden`

   (a) $L.L^*.M + M = (L.L^* + \varepsilon).M = L^*.M$, donc $L^*.M$ est bien solution de l'équation.

   (b) Soit $X$ une solution.
   Prouvons par récurrence sur $k \in \mathbb{N}$ que pour tout $k \in \mathbb{N}, L^k.M \subset X$
   (I) $\varepsilon.M = M$ et $M$ est inclus dans $X$, donc la propriété est initialisée.
   (H) Soit $k \in \mathbb{N}$ tq $L^k.M \subset X$. Alors $X = L^k.M + X_k$.
   Comme $X$ est solution de l'équation, on a : $X = L^k.M + X_k = L.(L^k.M + X_k) + M = L^{k+1}.M + L.X_k + M$, donc $L^{k+1}.M$ est bien inclus dans $X$.
   (C) D'après le principe de récurrence, la propriété est vraie pour tout $k \in \mathbb{N}$.
   On a donc bien $L^*.M \subset X$.

   (c) Supposons que $\varepsilon \notin L$. Soit $X$ une solution. Supposons par l'absurde qu'il existe $u \in X \setminus L^*.M$. Soit un tel $u$ de longueur minimum.
   On a $u \in X$, donc $u \in L.X + M$. Par hypothèse, $u \notin L^*M$, donc en particulier $u \notin M$. Ainsi, $u \in L.X$. Comme $\varepsilon \notin L$, il existe un mot non vide $v \in L$ et un mot $w \in X$ tels que $u = v.w$.
   - si $w = \varepsilon$, alors $u \in L$, ce qui est absurde.
   - si $w \in L^*M$ alors $u \in L^*M$, ce qui est absurde.
   - si $w \in X \setminus L^*.M$, alors que $|w| < |u|$ nous met face à une nouvelle absurdité.

   On a donc bien $X = L^*.M$.

   (d) Supposons que $\varepsilon \in L$. Remarquons tout d'abord que dans ce cas, $L.L^* = L^*$.
   Soit $N$ un langage tq $M \subset N$.
   Calculons : $L.X + M = L.L^*.N + M = L^*.N + M = L^*N = X$.
   Le langage $X = L^*.N$ est donc solution.

   (e) En reprenant le raisonnement des questions a), b) et c), on prouve que $X = M.L^*$ est l'unique solution de cette équation.

8. `Anagramme`

   (a) Il y a 10! anagrammes d'un mot de longueur 10. Or pour afficher un mot de longueur 10, il faut $10^{-5}$s. On obtient donc un temps global de $10! * 10^{-5} = 36.288$s. Pour 20, on trouve $20! * 2 * 10^{-5} = 4.865804 * 10^{13}$s.

   (b) On propose ici une solution de complexité linéaire :

```
let detecteAnagramme m n =
  let occ = Array.make 26 0 in
  let lm = String.length m in
  let ln = String.length n in
  let b = ref (ln=lm) in
  if !b then
    (for i = 0 to ln-1 do
       let ind = int_of_char(m.[i])-97 in
```

```
      occ.(ind) <- occ.(ind) +1
    done);
  let j = ref 0 in
  while !b && !j<ln-1 do
    let ind = int_of_char(m.[!j])-97 in
    occ.(ind) <- occ.(ind) -1;
    b := occ.(ind) >= 0;
    incr j
  done;
  !b;;
```

(c) On commence par écrire une fonction `ajoute_lettre : char -> string -> string list` telle que `ajoute_lettre c mot` renvoie la liste de tous les mots que l'on obtient en insérant la lettre `c` à une position dans le mot `mot`.

On va utiliser les deux fonctions suivantes :

- La fonction `String.sub` telle que `String.sub mot i k` renvoie la sous-chaine de `mot` qui commence au caractère d'indice $i$ et qui est de longueur $k$.
- La fonction `Char.escaped` qui transforme un caractère en une chaine de longueur 1.

```
let ajoute_lettre c mot =
  let rec parcours i =
    if i > String.length mot then []
    else (String.sub mot 0 i)^Char.escaped(c)^(String.sub mot i (n-i))
                :: (parcours (i+1))
  in parcours 0;;
```

On va maintenant *généraliser* la fonction précédente, en une fonction `ajoute_lettre_list : char -> string list-> string list` telle que `ajoute_lettre c lmot` renvoie la liste obtenue en concaténant les listes obtenues par la fonction précédente pour chaque mot de la liste `lmot`

```
let rec ajoute_lettre_list c lmot = match lmot with
  | [] -> []
  | mot :: q -> (ajoute_lettre c mot) @ ( ajoute_lettre_list c q);;
```

On peut ensuite écrire notre fonction récursive.

Pour calculer les anagrammes d'un mot `mot` ayant au moins une lettre, on sépare le mot en considérant sa première lettre `c` et le reste du mot `queue`. Il suffit alors de calculer tous les anagrammes de `queue` et d'y ajouter la lettre `c` par la fonction précédente.

```
let rec anagramme mot =
  let n = String.length mot in
  if n = 0 then []
```

```
else ajoute_lettre_list mot.[0] (anagramme (String.sub mot 1 (n-1)));;
```

9. `Centrale 2003 : langage des parenthèsages`

   (a) Comme $\varepsilon \in L_0$, $ab \in L_1$. On en déduit que $ab \in L_2$ (car $L_1 \subset L_2$) et que $a(ab)b \in L_2$. Ainsi, $ababbaa = ab.aabb$ est élément de $L_2^2$, donc de $L_3$ : $abaabb \in \mathcal{L}_P$.

   (b) Montrons par récurrence sur $n$ que pour tout mot $w$ de $L_n$, $|w|_a = |w|_b$.

   - Comme $L_0$ est réduit au mot vide, l'hypothèse de récurrence est vérifiée au rang 0.
   - Fixons $n \geq 0$ et supposons la propriété vérifiée au rang $n$. Soit alors $w \in L_{n+1}$, que nous pouvons supposer non vide. Nous sommes donc dans l'un des trois cas suivants :
     - $w \in L_n$ et l'hypothèse de récurrence affirme que $w$ contient autant de $a$ que de $b$.
     - $w = uv$ avec $u, v \in L_n \setminus \{\varepsilon\}$ (si l'un des mots est vide, on se retrouve dans le cas précédent). Alors $|u|_a = |u|_b$, $|v|_a = |v|_b$ : on en déduit que $|w|_a = |u|_a + |v|_a = |u|_b + |v|_b = |w|_b$.
     - $w = aub$ avec $u \in L_n$. Comme $|u|_a = |u|_b$, on a encore $|w|_a = 1 + |u|_a = 1 + |u|_b = |w|_b$.

     La propriété est donc vérifiée au rang $n+1$, ce qui achève la preuve par récurrence.

   (c) Montrons par récurrence sur $n$ que pour tout mot $w\mathbb{N}eq\varepsilon$ de $L_n$, $w_1 = a$ et $w_{|w|} = b$.

   - Comme $L_0$ est réduit au mot vide, l'hypothèse de récurrence est vérifiée au rang 0.
   - Fixons $n \geq 0$ et supposons la propriété vérifiée au rang $n$. Soit alors $w \in L_{n+1}$, que nous pouvons supposer non vide. Nous sommes donc dans l'un des trois cas suivants :
     - $w \in L_n$ et l'hypothèse de récurrence affirme qu'il commence par un $a$ et qu'il se termine par un $b$.
     - $w = uv$ avec $u, v \in L_n \setminus \{\varepsilon\}$ (si l'un des mots est vide, on se retrouve dans le cas précédent). Alors $u$ et $v$ commencent par $a$ et se terminent par $b$ : on en déduit que $w$ commence par $a$ et se termine par $b$.
     - $w = aub$ avec $u \in L_n$. La première (resp. dernière) lettre de $w$ est donc un $a$ (resp. un $b$).

     La propriété est donc vérifiée au rang $n+1$, ce qui achève la preuve par récurrence.

   (d) Prouvons une nouvelle fois ce résultat par récurrence.

   - si $w \in L_0$, $w = \varepsilon$ et la propriété est vérifiée, puisqu'il n'existe pas de $i$ tel que $w_i = a$.
   - soit $n \geq 0$ et supposons que la propriété soit vérifiée pour tout mot $w \in L_n$. Fixons alors un mot $w \in L_{n+1}$ et un $i$ tel que $w_i = a$. Nous sommes une nouvelle fois dans l'un des trois cas suivants :
     - $w \in L_n$ et il suffit d'apliquer la propriété au rang $n$.

- $w = uv$ avec $u, v \in L_n \setminus \{\varepsilon\}$. Si $i \leq |u|$, on sait qu'il existe $j$ tel que $i < j \leq |u|$ et $u[i \dots j] \in \mathcal{L}_P$. On en déduit donc que $w[i \dots j] \in \mathcal{L}_P$. Si $i > |u|$, alors $v_{i-|u|} = a$ et il existe $j' > i - |u|$ tel que $v[i - |u| \dots j'] \in \mathcal{L}_P$. En posant $j = j' + |u|$, nous avons $i < j \leq |w|$ et $w[i \dots j] = v[i - |u| \dots j']$ est élément de $\mathcal{L}_P$.
- $w = aub$ avec $u \in L_n$. Si $i = 1$, $j = |u|$ donne $w[i \dots j] = w \in \mathcal{L}_P$. Sinon, $u_{i-1} = a$ et il existe $j' > i - 1$ tel que $u[i - 1 \dots j'] \in \mathcal{L}_P$. En posant $j = j' + 1 > i$, nous avons donc $w[i \dots j] = u[i - 1, j'] \in \mathcal{L}_P$.

Le $j$ n'est en général pas unique : avec $w = abaabb$ et $i = 1$, on peut choisir $j = 2$ ou $j = 6$.

(e) Notons $(\mathcal{P})$ la condition : $|w|_a = |w|_b$ et $|u|_a \geq |u|_b$ pour tout préfixe $u$ de $w$.
Montrons tout d'abord que $(\mathcal{P})$ est nécessaire pour que $w$ appartienne à $\mathcal{L}_P$. Comme on a déjà vu que $|w|_a = |w|_b$ pour tout $w \in \mathcal{L}_P$, il reste à montrer (par induction) que $|u|_a \geq |u|_b$ pour tout préfixe $u$ de $w \in \mathcal{L}_P$.

- Si $w \in L_0$, $w = \varepsilon$ et le seul préfixe de $w$ est $\varepsilon$, qui vérifie bien $|\varepsilon|_a \geq |\varepsilon|_b$.
- Soit $n \in \mathbb{N}$ et supposons la propriété vraie pour tout mot $w \in L_n$. Fixons $w \in L_{n+1}$, que nous pouvons supposer ne pas appartenir à $L_n$, et soit $u$ un préfixe de $w$. Nous avons alors trois cas possibles :
  - $w = w_1 w_2$ avec $w_1, w_2 \in L_n$ et $u$ est un préfixe de $w_1$ : par hypothèse de récurrence, $|u|_a \geq |u|_b$.
  - $w = w_1 w_2$ avec $w_1, w_2 \in L_n$ et $u = w_1 v$ où $v$ est un préfixe (non vide) de $w_2$ : par hypothèse de récurrence, on sait que $|v|_a \geq |v|_b$, et donc
  $$|u|_a = |w_1|_a + |v|_a = |w_1|_b + |v|_a \geq |w_1|_b + |v|_b = |u|_b.$$
  - $w = avb$ avec $v \in L_n$. Si $u$ est vide ou si $u = w$, on a bien $|u|_a \geq |u|_b$. Sinon, on peut écrire $u = at$ avec $t$ préfixe de $v$. Par hypothèse de récurrence, $|t|_a \geq |t|_b$ et donc
  $$|u|_a = 1 + |t|_a \geq 1 + |t|_b > |t|_b = |u|_b.$$

  Ainsi, tout mot de $\mathcal{L}_P$ vérifie la propriété $(\mathcal{P})$.

Montrons ensuite que la condition est suffisante par induction sur la longueur du mot $w$.

- Si $w$ est le mot vide, il est bien élément de $\mathcal{L}_P$.
- Soit $n \in \mathbb{N}$ et supposons que tout mot $w$ de longueur au plus $n$ vérifiant $(\mathcal{P})$ est élément de $\mathcal{L}_P$. Soit alors un mot $w$ de longueur $n + 1$ vérifiant $(\mathcal{P})$. Notons $\phi : \{0, 1, \dots, n + 1\} \to \mathbb{Z}$ qui à tout $i$ associe $|u|_a - |u|_b$, où $u$ est le préfixe de $w$ de longueur $i$ : $u = w[1 \dots i]$. Par hypothèse, $\phi$ est à valeurs dans $\mathbb{N}$ et $\phi(0) = \phi(n + 1) = 0$. Comme la différence entre $\phi(i)$ et $\phi(i + 1)$ est toujours égale à 1 ou -1, ceci impose en particulier les conditions $\phi(1) = 1$ et $\phi(n) = 1$, i.e. $w_1 = a$ et $w_{n+1} = b$. Deux cas se présentent alors :

– pour tout $i$ compris entre 1 et $n$, $\phi(i) \geq 1$. En notant $v = w[2 \dots n]$, nous montrons facilement que $v$ vérifie la propriété $(\mathcal{P})$. Par exemple, si $u$ est un préfixe de $v$ de longueur $i$ :

$$|u|_a - |u|_b = |au|_a - 1 - |au|_b = \phi(i+1) - 1 \geq 0.$$

Comme $|v| \leq n$, l'hypothèse de récurrence permet d'affirmer que $v$ est élément de $\mathcal{L}_P$, puis que $w = aub$ l'est également.

– il existe $i$ tel que $1 \leq i \leq n$ et $\phi(i) = 0$. Posons alors $w_1 = w[1 \dots i]$ et $w_2 = w[i+1 \dots n+1]$. On montre une nouvelle fois facilement que $w_1$ et $w_2$ vérifient $(\mathcal{P})$. Comme ces mots sont de longueurs au plus $n$, ils sont par hypothèse de récurrence éléments de $\mathcal{L}_P$, de même que $w = w_1 w_2$.

(f) La fonction ci-dessous calcule $\phi(i)$ pour $i$ croissant de 1 jusqu'à $|w|$, éventuellement en s'arrêtant quand on obtient une valeur négative. Si la boucle est entièrement parcourue, le mot $w$ est bien parenthésé si et seulement si $\phi(|w|)$ est nul. Sinon, le mot est mal parenthésé.

```
let parenthese w =
    let l = String.length w in
    let rec aux a b n =
        if n=l then a=b
        else (a>=b) && (if w.[n]='a'
                        then aux (a+1) b (n+1)
                        else aux a (b+1) (n+1))
    in aux 0 0 0;;
```

La complexité est clairement linéaire (dans le pire des cas) en la taille de la chaîne $w$, puisque la boucle porte sur l'indice $i$ qui décrit, dans le pire des cas, l'intervalle $[\![0, |w|]\!]$.

(g) i.
```
let association w =
    let l = String.length w in
    let t = Array.make l 0 in
    for i = 0 to l-2 do
        if w.[i] = 'a' then
        (let rec aux a b j =
            if a=b then j
            else if w.[j+1]='a' then
                 aux (a+1) b (j+1)
            else aux a (b+1) (j+1)
         in let j = aux 1 0 i in
        t.(i) <- j; t.(j) <- i)
    done;
    t;;
```

La complexité est un $O(n^2)$, cela correspond au pire cas qui arrive pour le mot $a^n b^n$.

ii.
```
let creer_pile () = ref [];;
let est_vide p = !p = [];;
```

```
let empiler i p = p := i:: !p;;
let depiler p = let s = List.hd(!p) in p := List.tl(!p); s;;

let association w =
    let p = creer_pile () in
    let l = String.length s in
    let t = Array.make l 0 in
    for i = 0 to l-1 do
        if w.[i] = 'a' then
          empiler i p
        else
          let k = depiler p in t.(i) <- k; t.(k) <- i
    done;
    t;;
```

(h)
- $c_0 = 1$
- $c_{2k+1} = 0$ car tous les mots bien parenthésés sont de longueur paire
- $c_{2k} = \sum_{i=0}^{k} c_{2i} * c_{2(k-1-i)}$, car une fois l'indice nécessairement pair associé à la parenthèse ouvrante par lequel commence le mot, il reste à choisir le mot bien parenthésé entre les deux parenthèses et le mot bien parenthésé qui suit la parenthèse fermante.

(i) Ce sont les nombres de Catalan. Posons la série génératrice $c(x) = \sum_{n=0}^{+\infty} c_{2n}x^n$.
Le rayon de convergence de $c$ est strictement positif car $\sum \frac{c_n}{4^n}$ est bornée par 2.
En appliquant la relation de récurrence, on prouve que $c(x) = 1 + xc(x)^2$. En résolvant cette équation, on trouve $c(x) = \frac{1-(1-4x)^{1/2}}{2x}$.
On conclut en développant cette expression et on trouve :

$$c_{2n} = \frac{1}{n+1}\sum_{i=0}^{n}\binom{n}{i}^2 = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n+1}$$

10. Soit $w \in (LM)^*L$, alors il existe $(w_0, w_1) \in (LM)^* \times L$ tels que $w = w_0.w_1$. De la même façon, il existe $k \in \mathbb{N}$ et $(v_i)_{0\leq i<k} \in (LM)^k$ tels que $w_0 = v_0...v_{k-1}$. Enfin, pour tout $0 \leq i < k$, il existe $l_i \in L$ et $m_i \in M$ tels que $v_i = l_i m_i$. On obtient donc que $w = l_0.m_0.....l_{k-1}.m_{k-1}.w_1$. Or $m_0.l_1.m_1.....l_{k-1}.m_{k-1}.w_1 \in (ML)^k \subset (ML)^*$. Donc $w \in L(ML)^*$. L'autre inclusion se montre de la même façon.

11. (a) $\overline{\sigma(0)} = \overline{01} = 10 = \sigma(1) = \sigma(\overline{0})$ et $\overline{\sigma(1)} = \overline{10} = 01 = \sigma(0) = \sigma(\overline{1})$, comme les deux applications sont compatibles avec la concaténation, par récurrence directe, on obtient pour tout $m \in A^*$, $\sigma(\overline{m}) = \overline{\sigma(m)}$.

(b) On tâtonne : 0, 01, 0110, 01101001,... Montrons que $m_{n+1} = m_n.\overline{m_n}$ :

Initialisation : c'est vrai pour les premières étapes

Hérédité : Soit $n \in \mathbb{N}$ tel que $m_n$ vérifie la propriété. Prouvons que $m_{n+1}$ aussi.
$m_{n+1} = \sigma(m_n) = \sigma(m_{n-1}.\overline{m_{n-1}}) = \sigma(m_{n-1}).\sigma(\overline{m_{n-1}}) = m_n.\overline{\sigma(m_{n-1})} = m_n.\overline{m_n}$

Conclusion : la propriété est donc vérifiée pour tout $n \in \mathbb{N}$.

(c) `let sigma list = match list with`

```
  | [] -> []
  | 0 :: q -> 0::1::(sigma q)
  | 1 :: q -> 1 :: 0 :: (sigma q)
;;

let rec itere n =
  if n =0 then [0]
  else sigma (itere (n-1));;
```

(d) Commençons par une version naïve. Une récurrence immédiate montre que le mot $m_n$ est de longueur $2^n$. Afin de calculer le $n$-ième terme de la suite $m_\infty$, il faut donc calculer $m_p$ où $n \leq 2^p$ c'est-à-dire $p \geq \lceil \log_2(n) \rceil$.

```
let rec log2 n =
  if n <= 2 then 1 else 1+log2 ((n+1)/2);;
```

On en déduit donc la fonction

```
let nieme n =
  let p = log2 n in
  element n (itere p);;
```

où

```
let rec element n list = match l with
  |[] -> failwith "liste trop courte"
  |t::q -> if n = 0 then t else (element (n-1) q);;
```

On peut faire mieux. On commence la numérotation de $m_\infty$ à 0. On a $m_\infty = 01101001100\ldots$, on veut donc construire une suite $u_n$ telle que $u_0 = 0, u_1 = 1, u_2 = 1, u_3 = 0, \ldots$.
On remarque que le mot $m_n$ est de longueur $2^n$. Comme on sait de plus que $m_{n+1} = m_n.\overline{m_n}$, on obtient que si $n = 2^p + r$ où $r < 2^p$ alors $u_n = 1 - u_r$
On en déduit que $u_n$ est égal à la somme (modulo 2) des chiffres de la décomposition en base 2 de $n$.

```
let rec nieme n =
  if n = 0 then 0
  else
  if n mod 2 = 0
  then nieme (n / 2)
  else 1 - nieme (n / 2);;
```