Sujet Centrale 2022 : correction

Q1.
L_1 est le langage associé à l'expression rationnelle $(a+b)^*.a.b.a^*$. Autrement dit, c'est le langage $\{maba^n \,/\, m \in \{a,b\}^*,\ n \in \mathbb{N}\}$. Son langage miroir L_1^* est associé à $a^*b.a.(a+b)^*$. C'est aussi le langage $\{a^nbam \,/\, n \in \mathbb{N}, m \in \{a,b\}^*\}$.
Q2.
On inverse l'automate $\mathcal{A}_1$: on choisit 3 états $0,1,2$ tels que 2 est état initial, 0 état final et tel qu'il y a les transitions : $(2,a,2)$, $(2,b,1)$, $(1,a,0)$, $(0,a,0)$, $(0,b,0)$.
Q3.
Soit $I' = F$, $F' = I$ et $T' = {}^tT$, c'est-à-dire que $(q,a,q') \in T'$ si et seulement si $(q',a,q) \in T$.
Alors $m = a_0 \cdots a_{n-1} \in L_{\tilde{\mathcal{A}}}$ si et seulement s'il existe $q_0 \in I'$, $q_n \in F'$ et une succession de transitions, pour tout $0 \leq i \leq n-1$, $q_i \xrightarrow{a_i} q_{i+1}$ dans T'. Par définition de $\tilde{\mathcal{A}}$, ceci équivaut à ce qu'il existe $q_0 \in F$, $q_n \in I$ et une succession de transition, pour tout $0 \leq i \leq n-1$, $q_{i+1} \xrightarrow{a_i} q_i$ dans T. Ceci équivaut à ce que $a_n \cdots a_0 \in L_{\mathcal{A}}$, c'est-à-dire à ce que $\tilde{m} \in L_{\mathcal{A}}$. En conclusion, $m \in L_{\tilde{\mathcal{A}}}$ si et seulement si $\tilde{m} \in L_{\mathcal{A}}$ donc, comme $\tilde{\tilde{w}} = w$, $m \in L_{\mathcal{A}}$ si et seulement si $\tilde{\tilde{m}} \in L_{\mathcal{A}}$ si et seulement si $\tilde{m} \in L_{\tilde{\mathcal{A}}}$. $L_{\tilde{\mathcal{A}}}$ est donc l'ensemble des mots miroirs de $L_{\mathcal{A}}$.
Q4.

```
let transpose a =
    let rec aux l = match l with
       | [] -> []
       | (n,l,m)::q -> (m,l,n)::(aux q) in
    { nb=a.nb ; init=a.final ; final=a.init : trans=aux a.trans };;
```

Q5.
La fonction auxiliaire parcourt la liste et chaque appel récursif s'ajoute à deux opérations en temps constant (création d'un élément "(m,l,n)" et ajout à une liste) ; cette fonction est donc en $O(n)$ où n est la longueur de la liste. La fonction "transpose" fait appel à la fonction auxiliaire et effectue un nombre borné d'opérations autres ; elle a donc une complexité en $O(1)+O(n) = O(n)$.
Q6.

```
let rec palindrome mot = let n = String.length mot and b = ref true in
   while !b && !i <n/2 do
      b:=mot.[i]=mot.[n-1-i] done;
   !b;;
```

Q7
Si $\Sigma = \{a\}$, $\Sigma^* = \{a^n \,/\, n \in \mathbb{N}\}$ donc tout mot est un palindrome donc $Pal(\Sigma) = \Sigma^*$. Σ^* est reconnu par l'automate à un état (à la fois initial et final) et à une transition entre 0 et 0 étiquetée par a. Par conséquent, $Pal(\Sigma)$ est rationnel.
Q8.
Soit $\mathcal{A}$ un automate de nombre d'état n. a^nba^n est dans $Pal(\Sigma) \cap a^*ba^*$. Supposons que $\mathcal{A}$ reconnaisse a^nba^n. Alors il existe des transitions $q_i \xrightarrow{a} q_{i+1}$, pour tout $i \in [\![0, n-1]\!]$, $q_n \xrightarrow{b} q_{n+1}$, $q_i \xrightarrow{a} q_{i+1}$, pour tout $i \in [\![n+1, 2n]\!]$. Comme il y a n états, il existe $0 \leq i < j \leq n$ tel que $q_i = q_j$. Par conséquent, le chemin $q_i \xrightarrow{a} \cdots \xrightarrow{a} q_j$ est un cycle étiqueté par le mot a^{j-i}. Par conséquent, en ajoutant ce cycle au chemin étiqueté par le mot a^nba^n, on voit que le mot $a^{n+j-i}ba^n$ est aussi reconnu par $\mathcal{A}$. Ce mot n'était pas un palindrome, l'automate $\mathcal{A}$ ne reconnait pas $Pal(\Sigma)$. Ceci étant vrai pour tout automate, $Pal(\Sigma)$ n'est pas reconnaissable.
Q9.
Soit $\mathcal{A}_{q,q'} = (Q, \{q\}, \{q'\}, T)$. Un mot $w = a_0 \cdots a_n$ est reconnu par $\mathcal{A}_{q,q'}$ si et seulement si il existe des transitions $q_i \xrightarrow{a_i} q_{i+1}$ dans T tel que $q_0 = q$ et $q_n = q'$, c'est-à-dire si et seulement si w étiquette un chemin de q à q', c'est-à-dire si et seulement si $w \in L_{q,q'}$. $L_{q,q'}$ est donc reconnu par $\mathcal{A}_{q,q'}$ et est donc rationnel.
Un mot w est reconnu par $\mathcal{A}$ si et seulement s'il existe $i \in I$, $f \in F$ et un chemin de i à f étiqueté par w si et seulement s'il existe $i \in I$, $f \in F$ tel que $w \in L_{i,f}$. On a donc $L_{\mathcal{A}} = \bigcup_{(i,f) \in I \times F} L_{i,f}$.
Q10.
Soit $m \in Pal(\Sigma) \cap (\Sigma^2)^*$. $m \in (\Sigma^2)^*$ donc $m = m_1 \cdots m_n$ où, pour tout $i \in [\![1,n]\!]$, $m_i \in \Sigma^2$. Ainsi, $|m| = 2n$ est pair. Notons $m = a_0 \cdots a_{2n-1}$. Soit $u = a_0 \cdots a_{n-1}$. Comme m est un palindrome, $a_0 \cdots a_{n-1}a_n \cdots a_{2n-1} = a_{2n-1} \cdots a_n a_{n-1} \cdots a_0$ donc, par égalité des n dernières lettres, $a_n \cdots a_{2n-1} = a_{n-1} \cdots a_0 = \tilde{u}$. On a donc $m = u\tilde{u}$.
Réciproquement, soit $m = u\tilde{u}$ où $u \in \Sigma^*$. Alors $\tilde{m} = \tilde{u\tilde{u}} = \tilde{\tilde{u}}\tilde{u} = u\tilde{u} = m$ donc $u \in Pal(\Sigma)$. En outre, $|m| = 2|u|$ donc m a un nombre pair de lettres. Soit, pour tout $i \in [\![0, |u|-1]\!]$, $m_i = a_{2i}a_{2i+1}$. Alors, pour tout $i \in [\![0, |u|-1]\!]$, $m_i \in \Sigma^2$ et $m = m_0 \cdots m_{|u|-1}$ donc $m \in (\Sigma^2)^*$. En conclusion, $m \in Pal(\Sigma) \cap (\Sigma^2)^*$.
Finalement, $Pal(\Sigma) \cap (\Sigma^2)^* = \{u\tilde{u} \,/\, u \in \Sigma^*\}$.
Q11.
Soit $m \in D(a^*b)$. Alors $m = w\tilde{w}$ où $w \in L(a^*b)$ donc $w = a^nb$ où $n \in \mathbb{N}$. Par conséquent, $m = a^nb^2b^n$. Réciproquement, s'il existe $n \in \mathbb{N}$ tel que $m = a^nb^2a^n$, alors $m = (a^nb)\widetilde{a^nb}$ donc $m \in D(a^*b)$. On a donc $D(a^*b) = \{a^nb^2a^n \,/\, n \in \mathbb{N}\}$.
Soit $m \in R(a^*b^*a^*)$. Alors $m\tilde{m} \in a^*b^*a^*$ donc il existe $(n,p,q) \in \mathbb{N}^3$ tels que $m\tilde{m} = a^nb^pa^q$. On notera $|m|_l$ le nombre de lettre l dans le mot m. On a donc $|m|_b = |\tilde{m}|_b$ donc $p = |a^nb^pa^q| = |m|_b + |\tilde{m}|_b = 2|m|_b$. Ainsi, p est pair. En outre, le nombre de p dans m et dans $\tilde{m}$ est le même. m est un préfixe de $a^nb^{2|m|_b}a^q$ qui contient $|m|_b$ lettres b ; c'est donc $a^nb^{|m|_b}$. On a donc $m = a^nb^{|m|_b}$ et

$m\tilde{m} = a^n b^{|m|_b} b^{|m|_b} a^q$ donc $\tilde{m} = b^{|m|_b} a^q$ et $\tilde{m} = b^{|m|_b} a^n$ donc $n = q$ et $m = a^n b^{|m|_b}$. Par conséquent, $m \in L(a^*b^*)$. Réciproquement, soit $m \in L(a^*b^*)$; alors $m = a^n b^p$ donc $m\tilde{m} = a^n b^{2p} b^n \in L(a^*b^*a^*)$ donc $m \in R(a^*b^*a^*)$. Ainsi, $R(a^*b^*a^*) = L(a^*b^*)$.

Q12.

Si $L = \{ab\}$, $D(L) = \{abba\}$ est reconnaissable. En revanche, si $L = a^*b$, alors $D(L)$ n'est pas reconnaissable. En effet, $D(L) = \{a^n b^2 a^n \,/\, n \in \mathbb{N}\}$. Si $\mathcal{A}$ est un automate à n états, considérons $a^{n+1}b^2a^{n+1}$. Supposons que $\mathcal{A}$ reconnaît $a^n b^2 a^n$. Il existe un chemin étiqueté par $a^n b^2 a^n$. Comme démontré à la question 8, le chemin des n premières transitions contient nécessairement un cycle de taille $k > 0$; par conséquent, le mot $a^{n+k}b^2a^n$ est reconnu par $\mathcal{A}$ mais n'est pas dans $D(L)$. Ainsi, $\mathcal{A}$ ne peut reconnaître $D(L)$.
En conclusion, $D(L)$ peut être reconnaissable ou non.

Soit A un automate qui reconnaît L. Alors $w \in \bigcup_{(i,q,f)\in I\times Q\times F} L_{i,q} \cap \widetilde{L_{q,f}}$ si et seulement s'il existe $(i,q,f) \in I \times Q \times F$ tel que $w \in L_{i,q}$ et $w \in \widetilde{L_{q,f}}$ si et seulement s'il existe $i \in I$, $q \in Q$, $f \in F$ tel que $w \in L_{i,q}$ et $\tilde{w} \in L_{q,f}$ si et seulement s'il existe $i \in I$, $f \in F$ tel que $w\tilde{w} \in L_{i,f}$ si et seulement si $w\tilde{w} \in \bigcup_{(i,f)\in I\times F} L_{i,f}$ si et seulement si $w\tilde{w} \in L(\mathcal{A})$ si et seulement si $w \in R(L)$. En conclusion, $R(L) = \bigcup_{(i,q,f)\in I\times Q\times F} L_{i,q} \cap \tilde{L_{q,f}}$. Pour tout $(i,q,f) \in I\times Q\times F$, $L_{i,q}$ et $L_{q,f}$ sont reconnaissables donc $L_{i,q}$ et $\tilde{L_{q,f}}$ sont reconnaissables donc, de par la stabilité de ces langages par intersection finie, $L_{i,q} \cap \tilde{L_{q,f}}$ est reconnaissable donc, en vertu de la stabilité par union finie de ces langages, $\bigcup_{(i,q,f)\in I\times Q\times F} L_{i,q} \cap \tilde{L_{q,f}}$ est reconnaissable. Finalement, $R(L)$ est reconnaissable.

Q13.

L'automate $Q = \{0,1,2,3\}$, $I = \{0\}$, $F = \{3\}$ et $\delta = \{(0,b,0),(0,a,1),(0,a,1),(1,b,0),(1,b,2),(2,a,3)\}$ reconnaît le langage $(b+ab)^*ba$.

Q14.

Cet automate est $Q = \{e_0,e_1,e_2,e_3\}$, $I = \{e_0\}$, $F = \{e_2,e_3\}$ et $\delta = \{(e_0,a,e_1),(e_1,b,e_2),(e_2,b,e_3),(e_3,b,e_3),(e_3,a,e_2)\}$.

Q15.

Cet automate est $Q = \{q_0,q_1,q_2,q_3\}$, $I = \{q_0\}$, $F = \{q_3\}$ et $\delta = \{(q_0,a,q_1),(q_0,b,q_2),(q_1,b,q_0),(q_2,b,q_2),(q_2,a,q_3),(q_3,b,q_0)\}$.

Q16.

$\mathcal{A}_2$ reconnaît L_2 donc $\tilde{\mathcal{A}}_2$ reconnaît $\tilde{L_2}$ donc $\mathcal{A}_3 = (\tilde{\mathcal{A}}_2)_{det}$ reconnaît $\tilde{L_2}$ donc $\tilde{\mathcal{A}}_3$ reconnaît $\tilde{\tilde{L_2}} = L_2$ donc $\mathcal{A}_4 = (\tilde{\mathcal{A}}_3)_{det}$ reconnaît aussi L_2.

Q17.

```
let rec supprimer l =
    let rec supprime_elmt x liste = match liste with
        | [] -> []
        | y::q -> if x=y then supprime_elmt x q else y::(supprime_elmt x q)
    in match l with
    | [] -> []
    | x::q -> x:: supprimer (supprime_elmt x q);;
```

Q18.

Soit C_n la complexité pour une liste de taille n de "supprimer". La complexité de "supprime_elmt" sur une liste de taille n est en $O(n)$. Un appel à "supprimer" sur une liste de taille n crée un appel d'une complexité $O(n)$ (la fonction "supprime_elmt") et un appel sur une liste de taille au plus $n-1$. Par conséquent, $C_n = C_{n-1} + O(n)$. Ceci donne une complexité quadratique : $C_n = O(n^2)$.

Q19.

```
let est_dans q k = (k/pow.(q)) mod 2=1;;
```

Q20.

```
let numero l = let rec aux l x = match l with
                   | [] -> x
                   | y::q -> if est_dans y x then aux q x else aux q (x+pow.(y))
    in aux l 0;;
```

Q21.

```
let intersecte l x = match l with
    | [] -> false
    | q::l0 -> (est_dans q x) || intersecte l0 x;;
```

Q22.

```
let etat_suivant k t = let rec aux t ka kb =
                         | [] -> (ka,kb)
                         | (x,'a',y)::q -> if (est_dans x k) && not (est_dans y ka) then aux q (ka+pow.(y)) kb else aux q ka kb
                         | (x,'b',y)::q -> if (est_dans x k) && not (est_dans y kb) then aux q ka (kb+pow.(y)) else aux q ka kb
              in aux t 0 0;;
```

Q23.

```
let cherche k l = match l with
    | [] -> -1
    | (x,v)::q -> if x=k then v else cherche k q;;
```

Q24.

```
let determinise a = let k=numero a.init in liste=ref [(k,0)] and t=ref [] and f= ref [] in
    let rec aux k =
        let (ka,kb)=etat_suivant k a.trans in
        if cherche ka !liste =-1 then begin
           liste:=(ka,Liste.length !liste);
           if intersecte ka a.final then f:=(cherche ka !liste)::!f end;
        if cherche kb !liste =-1 then begin
           liste:=(kb,Liste.length !liste);
           if intersecte kb a.final then f:=(cherche kb !liste)::!f end;
        let ta=(cherche k !liste ,'a',cherche ka !liste) and tb=(cherche k !liste ,'b',cherche kb !liste) in
        if not (List.mem ta !t) then (t:=ta::!t; aux ka);
        if not (List.mem tb !t) then (t:=tb::!t; aux kb)
    in aux k;;
    { nb = List.length !liste ; init=[0] ; final =!f ; trans=!t};;
```

Q25.
Le calcul des états suivants pour un ensemble X demande de parcourir la liste de toutes les transitions, qui est en $O(n^2)$. Ce calcul est effectué pour chaque état de A_{det} ; les autres calculs effectués pour chaque état de A_{det} sont de moindre coût. Cela donne une complexité de $O(Nn^2)$.
Q26.
L'automate miroir $\tilde{A}$ est accessible ; q est un état de A donc de $\tilde{A}$; il existe donc une suite de transition dans $\tilde{A}$ de f vers q, c'est-à-dire, par définition de $\tilde{A}$, une suite de transitions de q vers f dans A. Cette suite de transitions correspond à un mot w. On a donc $q \in \delta^\star(I, u)$ et $f \in \delta^\star(q, w)$ donc $f \in \delta^\star(I, uw)$ donc uw est reconnu par A donc $uw \in L$.
Q27.
Soit $q \in \delta^\star(I, u)$. D'après la question précédente, il existe $w \in \Sigma^*$ tel que $uw \in L$. On a donc $w \in u^{-1}L = v^{-1}L$. Par conséquent, $vw \in L$; il y a un chemin dans A étiqueté par vw. En outre, $\tilde{w}\tilde{v}$ est reconnu par $\tilde{A}$. L'automate $\tilde{A}$ étant déterministe, le chemin de f vers q étiqueté par $\tilde{w}$ est le seul chemin partant de f dans $\tilde{A}$ étiqueté par $\tilde{w}$; par conséquent, le seul chemin aboutissant à f étiqueté par w est celui partant de q. Par conséquent, le chemin partant de I étiqueté par vw doit se terminer par le chemin étiqueté par w aboutissant à f depuis w. Ainsi, $q \in \delta^\star(I, v)$. Par conséquent, $\delta^\star(I, u) \subset \delta^\star(I, v)$. Par symétrie des rôles de u et v, $\delta^\star(I, u) = \delta^\star(I, v)$.
Q28.
$\tilde{A}$ reconnaît $\tilde{L}$ donc $(\tilde{A})_{det} = B$ reconnaît aussi $\tilde{L}$. $\tilde{B}$ reconnaît $\tilde{\tilde{L}} = L$ donc $(\tilde{B})_{det}$ reconnaît aussi L.
En outre, $\tilde{B}$ est un automate dont le miroir est B ; or $B = (\tilde{A})_{det}$ est déterministe et accessible ; le miroir de $\tilde{B}$ est donc déterministe et accessible. D'après la question précédente, $(\tilde{B})_{det}$ vérifie automatiquement la proprité $(*)$.
Q29.

```
let minimal a = determinise (transpose (determinise (transpose a) ));;
```

Q30.

```
let rec lettre e = match e with
    | Vide -> 0
    | Epsilon -> 0
    | Lettre(_) -> 1
    | Union(e1,e2) -> lettre e1 + lettre e2
    | Concat(e1,e2) -> lettre e1 + lettre e2
    | Etoile(e1) -> lettre e1;;
```

Q31.

```
let rec est_vide e = match e with
    | Vide -> true
    | Epsilon -> false
    | Lettre(_) -> false
    | Union(e1,e2) -> est_vide(e1) && est_vide(e2)
    | Concat(e1,e2) -> est_vide(e1) || est_vide(e2)
    | Etoile(_) -> false;;
```

Q32.

```
let se e = match e with
    | Etoile(Vide) -> Epsilon
    | Etoile(Epsilon) -> Epsilon
    | Etoile(Etoile(e1)) -> Etoile(e1)
    | _ -> e;;
```

Q33.
L'application de la simplification $b.\emptyset \equiv \emptyset$ diminue de 1 le nombre de 1. On l'applique n jusqu'à obtenir $a + \emptyset$ qui, en une simplification, nous donne a. Il y a donc $n + 1$ simplifications à effectuer.
Q34.

```
let rec simplifie e = match e with
    | Union(e1,e2) -> su Union(simplifie e1,simplifie e2)
    | Concat(e1,e2) -> sc Concat(simplifie e1,simplifie e2)
    | Etoile(e1) -> se Etoile(simplifie e1)
    | _ -> e;;
```

Q35.

```
let somme m1 m2 = let n=Array.length m1 and p=Array.length m1.(0) in
    let m=Array.make_matrix n p Vide in
    for i=0 to n-1 do for j=0 to p-1 do
            m.(i).(j) <- Union(m1.(i).(j),m2.(i).(j))
    done done; m;;
```

Comme il y a deux boucles imbriquées de taille n et p chacune de complexité $O(1)$, la complexité est $O(n \times p)$.

Q36.

```
let produit m1 m2 = let n=Array.length m1 and p=Array.length m2 and q=Array.length m2.(0) in
    let m=Array.make_matrix n q Vide in
    for i=0 to n-1 for j=0 to q-1 do for k=0 to p-1 do
        m.(i).(j) <- Union(m.(i).(j), Concat(m1.(i).(k),m2.(k).(j)))
    done done done; m;;
```

Il y a trois boucle imbriquées de taille n, p, q, chacune de complexité $O(1)$. La complexité est donc $O(npq)$.

Q37.

$L_{0,0} = (a + bd^*ca)^*$, $L_{0,1} = a^*b(d + ca^*b)^*$, $L_{1,0} = d^*c(a + bd^*c)^*$, $L_{1,1} = (d + ca^*b)^*$.

Q38.

Le calcul de D^* exige une complexité de $C(n-1)$. Le produit $B \times D^*$ exige une complexité $O((n-1)^2) = O(n^2)$; de même le produit $(BD^*)C$ exige une complexité $O(n^2)$. La complexité du calcul de $a + BD^*C$ exige donc une complexité $C(n-1) + O(n^2)$. Comme $a + BD^*C$ est de taille 1, son calcul est en $O(1)$. Le calcul de A' est donc en $C(n-1) + O(n^2)$.

Le calcul de Ca^*B est en $O((n-1)^2) = O(n^2)$, celui de $D + Ca^*B$ est en $O(n^2)$. Le calcul de $D' = (D + Ca^*B)^*$ ajoute ensuite une complexité $C(n-1)$. Le calcul de D' est donc en $C(n-1) + O(n^2)$.

Le calcul de C' demande le calcul de $(a + BD^*C)^*$, qui a déjà été fait pour le calcul de A' ; s'il a été gardé en mémoire, il n'ajoute pas de complexité ; le calcul de D^* a déjà été fait donc n'ajoute pas de complexité ; le calcul de $C' = D^*C(a + BD^*C)^*$ ajoute une complexité de $O((n-1)^2) = O(n^2)$ par rapport aux calculs précédents.

Le calcul de B' demande de réutiliser le calcul de D'. On ajoute à celui-ci le calcul de a^*BD' qui est en $O((n-1)^2) = O(n^2)$.

La complexité totale est donc $C(n) = C(n-1) + O(n^2) + C(n-1) + O(n^2) + O(n^2) + O(n^2) = 2C(n-1) + O(n^2)$.

Pour tout $n \in \mathbb{N}$, $\frac{C(n)}{2^n} = \frac{C(n-1)}{2^{n-1}} + O(\frac{n^2}{2^n})$ donc $\frac{C(n)}{2^n} - \frac{C(n-1)}{2^{n-1}} = O(\frac{n^2}{2^n})$.

La série $\sum \frac{n^2}{2^n}$ converge (car $\frac{n^2}{2^n} = o((\frac{2}{3})^n)$). La série télescopique $\sum \frac{C(n)}{2^n} - \frac{C(n-1)}{2^{n-1}}$ converge donc $\frac{C(n)}{2^n} = O(1)$ donc $C(n) = O(2^n)$.

Q39.

Le calcul de D^* exige $C(n/2)$, celui de BD^* $O((\frac{n}{2})^3) = O(n^3)$, celui de BD^*C également $O(n^3)$, celui de $A + BD^*C$ $O(n^2)$. Le calcul de $(A + BD^*C)^*$ ajoute ensuite une complexité $C(n/2)$. En conclusion, le calcul de A' exige $C(n/2) + O(n^3) + O(n^3) + O(n^2) + C(n/2) = 2C(n/2) + O(n^3)$.

Le calcul de D' exige le calcul de A^* qui est en $C(n/2)$ puis ceux de CA^* et de CA^*B qui ajoutent $O(n^3)$ chacun ; le calcul de $D + CA^*B$ ajoute $O(n^2)$; enfin le calcul de $(D + CA^*B)^*$ qui est en $C(n-1)$. Le calcul de B' exige donc $C(n/2) + 2O(n^3) + O(n^2) + C(n/2) = 2C(n/2) + O(n^3)$.

Le calcul de B' exige de réutiliser celui de D' auquel on ajoute les calculs de A^* - qui a déjà été fait - A^*B en $O(n^3)$ puis de A^*BD' en $O(n^3)$. Le calcul de B' ajoute une complexité $O(n^3)$.

Le calcul de C' exige de réutiliser le calcul de A' et d'y ajouter les calculs de D^* -déjà fait-, de D^*C et de D^*CA' qui ajoutent chacun $O(n^3)$. Ce calcul ajoute une complexité de $O(n^3)$.

Au total, la complexité est donc $C(n) = 2C(n/2) + O(n^3) + 2C(n/2) + O(n^3) + O(n^3) + O(n^3) = 4C(n/2) + O(n^3)$.

Pour tout $n \in \mathbb{N}$, $C(2^n) = 4C(2^{n-1}) + O(n^3)$ donc $\frac{C(2^n)}{4^n} = \frac{C(2^{n-1})}{4^{n-1}} + O(\frac{n^3}{4^n})$. La série $\sum \frac{n^3}{4^n}$ converge (car $\frac{n^3}{4^n} = o(\frac{1}{2^n})$). La série télescopique $\sum \frac{C(2^n)}{4^n} - \frac{C(2^{n-1})}{4^{n-1}}$ converge donc. Ainsi, $\frac{C(2^n)}{4^n} = O(1)$ donc $C(2^n) = O(4^n)$ donc $C(2^n) = O((2^n)^2)$. Si n est une puissance de 2, $C(n) = O(n^2)$.

Q40.

On effectue un découpage avec A de taille $\lfloor n/2 \rfloor$ et D de taille $\lceil n/2 \rceil$. On aura $C(n) = 2C(\lfloor n/2 \rfloor) + 2C(\lceil n/2 \rceil) + O(n^3)$. On aura alors, par récurrence sur k, si $2^k \leq n < 2^{k+1}$, $C(2^k) \leq C(n) \leq C(2^{k+1})$ donc, comme $C(n) \leq C(2^{k+1}) = O(2^{k+1} \times 2^{k+1}) = O(2^k \times 2^k) = O(n \times n) = O(n^2)$. En conclusion, $C(n) = O(n^2)$. Cette complexité quadratique est bien meilleure que la complexité exponentielle de la question 38.

Q41.

```
let etoile m = let n=Array.length m in match n with
    | 1 -> Array.make 1 Array.make 1 Etoile(m.(0).(0))
    | _ -> let (a,b,c,d)=decoupe m (n/2) (n-n/2) in
    let ae=etoile a and de=etoile d in
    let ap=etoile (somme a (produit b (produit de c)))
    and dp=etoile (somme d (produit c (produit ae b)))
    in recoller ap (produit ae (produit b dp)) (produit de (produit c ap)) dp;;
```

Q42.

On pose $x_i = \epsilon$ si $i \in I$, $x_i = \emptyset$ sinon et $y_i = \epsilon$ si $i \in F$, $y_i = \emptyset$ sinon.

Alors $\mathcal{L}([XM_A^*Y])_{0,0} = \bigcup_{0 \leq i,j \leq n-1} \mathcal{L}(x_i).\mathcal{L}([M_A^*]_{i,j}).(y_j) = \bigcup_{0 \leq i,j \leq n-1} x_i L_{i,j} y_j$. Si $x_i = \emptyset$ ou $y_j = \emptyset$, $x_i L_{i,j} y_j = \emptyset$; si $x_i = y_j = \epsilon$, $x_i.L_{i,j}y_j = L_{i,j}$. Par conséquent, $\mathcal{L}([XM_A^*Y])_{0,0} = \bigcup_{i \in I, j \in F} L_{i,j}$. Comme $L_A = \bigcup_{i \in I, j \in F} L_{i,j}$, $\mathcal{L}([XM_A^*Y]_{0,0}) = L_A$.

Q43.

```
let langage a = let n=a.nb in let x=Array.make_matrix 1 n Vide and y=Array.make_matrix n 1 Vide
    and m=Array.make n n Vide in
```

```
let rec aux1 l = match l with
    | [] -> ()
    | i::q -> x.(0).(i) <- Epsilon; aux1 q
and aux2 l = match l with
    | [] -> ()
    | i::q -> y.(i).(0) <- Epsilon; aux2 q
and aux3 l = match l with
    | [] -> ()
    | (i,c,j)::q -> m.(i).(j) <- Union(Lettre(c),m.(i).(j)); aux3 q
    in aux1 a.init; aux2 a.final; aux3 a.trans;
(produit x (produit (etoile m) y)).(0).(0);;
```

Les calculs de X et de Y sont de complexité $O(n)$, celui de M_A $O(n^2)$. Le calcul de M_A^* est $O(n^2)$, celui de XM_A^* $O(n^2)$ et celui de XM_A^*Y encore $O(n^2)$. La complexité de cette fonction est donc $O(n^2)$.

Q44.

$\partial_a(E) = \partial_a\big((ab+b)^*\big).\{ba\} \cup \partial_a(ba) = \big(\partial_a(ab+b).\{(ab+b)^*\}\big).\{ba\} \cup \partial_a(b).\{a\} = \big((\partial_a(ab) \cup \partial_a(b)\big).\{(ab+b)^*\}\big).\{ba\} \cup \emptyset = \big((\partial_a(a).\{b\} \cup \emptyset).\{(ab+b)^*\}\big).\{ba\} = \big(\{\epsilon\}.\{b\}.\{(ab+b)^*\}\big).\{ba\} = \{b(ab+b)^*ba\}$

$\partial_b(E) = \partial_b\big((ab+b)^*\big).\{ba\} \cup \partial_b(ba) = \partial_b(ab+b).\{(ab+b)^*\}.\{ba\} \cup \partial_b(b).\{a\} = \big(\partial_b(ab) \cup \partial_b(b)\big).\{(ab+b)^*\}.\{ba\} \cup \{\epsilon\}.\{a\} = \big(\partial_b(a).\{b\} \cup \{\epsilon\}\big).\{(ab+b)^*.ba\} \cup \{a\} = \big(\emptyset.\{b\} \cup \{\epsilon\}\big).\{(ab+b)^*.ba\} \cup \{a\} = \{\epsilon, b\}.\{(ab+b)^*.ba\} \cup \{a\} = \{(ab+b)^*ba, b(ab+b)^*ba\} \cup \{a\} = \{a, (ab+b)^*ba, b(ab+b)^*ba\}$.

Q45.

Q46.

Soit $m \in (uv)^{-1}L$. $uvm \in L$ donc $u(vm) \in L$ donc $vm \in u^{-1}L$ donc $m \in v^{-1}u^{-1}L$. Réciproquement, soit $m \in v^{-1}u^{-1}L$. Alors $vm \in u^{-1}L$ donc $uvm \in L$ donc $m \in (uv)^{-1}L$. En conclusion, $(uv)^{-1}L = v^{-1}u^{-1}L$.

Q47.

Montrons cette propriété par récurrence sur le nombre de lettres de w.

Si $w = \epsilon$, $\mathcal{L}(\partial_\epsilon(S)) = \mathcal{L}(\bigcup_{E\in S} \partial_\epsilon(E)) = \bigcup_{E\in S} \mathcal{L}(\partial_\epsilon(E)) = \bigcup_{E\in S} \mathcal{L}(\{E\}) = \bigcup_{E\in S} \mathcal{L}(E) = \mathcal{L}(S) = \epsilon^{-1}\mathcal{L}(S)$.

Si w est une lettre, cette propriété est admise par l'énoncé.

Supposons qu'elle est vraie pour un mot w. Montrons qu'elle l'est pour wa où a est une lettre. $\mathcal{L}(\partial_{wa}(S)) = \bigcup_{E\in S} \mathcal{L}(\partial_{wa}(E)) = \bigcup_{E\in S} \mathcal{L}(\partial_a(\partial_w(E)))$. D'après la propriété admise par l'énoncé, $\mathcal{L}(\partial_{wa}(S)) = \bigcup_{E\in S} a^{-a}\mathcal{L}(\partial_w(E))$. D'après l'hypothèse de récurrence, $\mathcal{L}(\partial_{wa}(S)) = \bigcup_{E\in S} a^{-1}w^{-1}\mathcal{L}(E)$. D'après la question précédente, $\mathcal{L}(\partial_{wa}(S)) = \bigcup_{E\in S} (wa)^{-1}\mathcal{L}(E)$. En outre, pour tout mot m, $m^{-1}(L_1 \cup L_2) = \{m' \,/\, mm' \in L_1 \cup L_2\} = \{m' \,/\, mm' \in L_1 \text{ ou } mm' \in L_2\} = \{m' \,/\, mm' \in L_1\} \cup \{m' \,/\, mm' \in L_2\} = m^{-1}L_1 \cup m^{-1}L_2$. En conclusion, $\mathcal{L}(\partial_{wa}(S)) = \bigcup_{E\in S} (wa)^{-1}\mathcal{L}(E) = (wa)^{-1} \bigcup_{E\in S} \mathcal{L}(E) = (wa)^{-1}\mathcal{L}(S)$. Ceci prouve la propriété pour wa.

En conclusion, par récurrence, pour tout $w \in \Sigma^*$, $\mathcal{L}(\partial_w(S)) = w^{-1}\mathcal{L}(S)$.

Q48.

Montrons ce résultat par récurrence sur le nombre de lettres de w.

$\partial_\epsilon(E) = \{E\}$ est bien l'ensemble des états accessibles depuis E en lisant le mot ϵ.

Supposons que ce résultat soit vrai pour w. Soit a une lettre. Montrons que ce résultat est vrai pour wa. $\partial_{wa}(E) = \partial_a(\partial_w(E))$. Or $\partial_w(E)$ est l'ensemble des états accessibles depuis l'état E. Ainsi, $\partial_{wa}(E) = \bigcup_{E'\in\partial_w(E)} \partial_a(E') = \bigcup_{w:E->E'} \partial_a(E')$ où la notation $w : E- > E'$ correspond à l'ensemble des états E' accessibles depuis E en lisant le mot w. Par définition de T, $\partial_a(E')$ est l'ensemble des états accessibles depuis E' par l'automate A en lisant a. En outre, un état E'' est accessible depuis E en lisant wa si et seulement si il existe E' accessible depuis E en lisant w et tel que E'' est accessible depuis E' en lisant a. En conclusion, $\bigcup_{w:E->E'} \partial_a(E')$ est l'ensemble des état accessibles depuis E en lisant wa donc, $\partial_{wa}(E)$ est l'ensemble des états accessibles depuis E en lisant wa.

Ceci prouve la propriété par récurrence sur le nombre de lettres de w.

Q49.

w est un mot reconnu par l'automate d'Antimirov si et seulement s'il existe un état $E_1 \in F$ accessible depuis l'état initial E en lisant le mot w. Ceci équivaut à ce qu'il existe E_1 un état accessible depuis l'état initial E en lisant le mot w tel que $\epsilon \in \mathcal{L}(E_1)$. D'après la question précédente, les états accessibles depuis l'état initial E en lisant le mot w sont ceux de $\partial_w(E)$. Par conséquent, w est reconnu par l'automate d'Antimirov si et seulement s'il existe un état E_1 de $\partial_w(E)$ tel que $\epsilon \in \mathcal{L}(E_1)$. Ceci équivaut à $\epsilon \in \mathcal{L}(\partial_w(E))$. D'après la question 47, ceci équivaut à ce que $\epsilon \in w^{-1}\mathcal{L}(E)$, c'est-à-dire à ce que $w \in \mathcal{L}(E)$. En conclusion, un mot w est reconnu par l'automate d'Antimirov si et seulement si $w \in \mathcal{L}(E)$. L'automate d'Antimirov reconnaît donc le langage associé à l'expression E.

Q.50.

Si E est du type $\emptyset$ ou ϵ, par définition de ∂_w, si $w = a$, $\partial_a(E) = \emptyset$ donc $\partial_w(E) = \emptyset$ donc $\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_a(E) = \emptyset$ a au plus $||E|| = 0$ éléments.

Si $E = b$, si $w = a$, $\partial_a(E)$ est vide sauf pour $a = b$. Si w est non vide, $\partial_w(\partial_b(E)) = \partial_w(\{\epsilon\}) = \emptyset$. On a donc $\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(E) = \partial_b(b) = \{\epsilon\}$ a bien au plus $||E|| = 1$ élément.

Supposons que cette propriété soit vraie pour deux expressions E et F.

On montre, par induction sur la taille des mots, que $\partial_w(E+F) = \partial_w(E) \cup \partial_w(F)$. Par conséquent, $\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(E+F) = (\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(E)) \cup (\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(F))$. Par hypothèse $\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(E)$ a au plus $||E||$ éléments et $\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(F)$ a au plus $||F||$ éléments. Par conséquent, $(\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(E)) \cup (\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(F))$ a au plus $||E|| + ||F||$ éléments donc $\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(E+F)$ au plus $||E|| + ||F|| = ||E+F||$ éléments.

On montre, par induction sur la taille des mots, que $\partial_w(E.F) \subset \partial_w(E).\{F\} \cup \bigcup v \in S^+(w)\partial_v(F)$. Ainsi, $\bigcup_{w\in\Sigma^*, w\neq\epsilon} \partial_w(E.F) \subset$

$(\bigcup_{w\in\Sigma^*,w\neq\epsilon}\partial_w(E).\{F\})\cup\bigcup_{w\in\Sigma^*,w\neq\epsilon}\bigcup_{v\in S^+(w)}\partial_v(F)$; $\bigcup_{w\in\Sigma^*,w\neq\epsilon}\partial_w(E).\{F\}$ a autant d'éléments que $\bigcup_{w\in\Sigma^*,w\neq\epsilon}\partial_w(E)$ éléments donc au plus $||E||$ éléments ; $\bigcup_{w\in\Sigma^*,w\neq\epsilon}\bigcup_{v\in S^+(w)}\partial_v(F)=\bigcup_{v\in\Sigma^*,v\neq\epsilon}\partial_v(F)$ a au plus $||F||$ éléments. Par conséquent, $(\bigcup_{w\in\Sigma^*,w\neq\epsilon}\partial_w(E).\{F\})\cup\bigcup_{w\in\Sigma^*,w\neq\epsilon}\bigcup_{v\in S^+(w)}\partial_v(F)$ a au plus $||E||+||F||$ éléments ; $\bigcup_{w\in\Sigma^*,w\neq\epsilon}\partial_w(E.F)$ a donc au plus $||E||+||F||=||E.F||$ éléments.

Enfin, on montre par induction sur le nombre de lettres de w que $\partial_w(E^*)\subset\bigcup_{v\in S^+(w)}\partial_w(E).\{E^*\}$. $\bigcup_{w\in\Sigma^*,w\neq\epsilon}\partial_w(E^*)\subset\bigcup_{w\in\Sigma^*,w\neq\epsilon}\bigcup_{v\in S^+(w)}\partial_v(E).\{E^*\}=\big(\bigcup_{v\in\Sigma^*,v\neq\epsilon}\partial_v(E)\big).\{E^*\}$; comme $\bigcup_{v\in\Sigma^*,v\neq\epsilon}\partial_v(E)$ a au plus $||E||$ éléments, $\big(\bigcup_{v\in\Sigma^*,v\neq\epsilon}\partial_v(E)\big).\{E^*\}$ a au plus $||E||$ éléments donc $\bigcup_{w\in\Sigma^*,w\neq\epsilon}\bigcup_{v\in S^+(w)}\partial_v(E)$. a au plus $||E||=||E^*||$ éléments donc $\bigcup_{w\in\Sigma^*,w\neq\epsilon}\partial_w(E^*)$ a au plus $||E^*||$ éléments.

Par induction structurelle sur les expressions rationnelles, ceci prouve le résultat demandé pour toute expression rationnelle.
Il existe certaines expressions rationnelles E pour lesquels on a besoin au minimum du nombre $||E||$ de lettres présentes dans E, c'est par exemple le cas pour une expression constitué d'une suite de lettres distinctes $a_1\cdots a_n$. La taille de l'automate est donc optimale pour les pires cas.