DS 7B d'option informatique - MP* - Réduction d'automates - Sujet Mines-Ponts 2019 - Correction

Question 1 – Le langage reconnu par $\mathcal{A}_1$ est l'ensemble des mots de longueur impaire.

Question 2 – Le langage reconnu par $\mathcal{A}_2$ est l'ensemble des mots contenant un nombre impair de b.

Question 3 – Le langage reconnu par $\mathcal{A}_1$ est dénoté par l'expression rationnelle $(a^2 \mid ab \mid ba \mid b^2)^*(a \mid b)$.

Question 4 – Le langage reconnu par $\mathcal{A}_2$ est dénote par l'expression rationnelle $(a \mid ba^*b)^*ba^*$.

Question 5 –

```
let (aut_fig2: automate) = (2, [|(0,1); (1,0)|], [|false; true|]);;
```

Question 6 –

```
(* k est l'indice de "List.hd li" dans a *)
let numero (n: int) (a: int list): int array =
  let t = Array.make n (-1) in
  let rec aux li k = match li with
    | [] -> ()
    | i :: q -> t.(i) <- k;
                aux q (k+1) in
  aux a 0;
  t;;
```

Question 7 –

```
(* li est la liste des prochains états à visiter *)
let etats_accessibles (aut: automate): int list =
  let (n, delta, _) = aut in
  let vu = Array.make n false in
  let rec parc_prof (li: int list): int list = match li with
    | [] -> []
    | q :: tl when vu.(q) -> parc_prof tl
    | q :: tl -> vu.(q) <- true;
                 let (succ_a, succ_b) = delta.(q) in
                 q :: parc_prof (succ_a :: succ_b :: tl) in
  parc_prof [0];;
```

Pour la complexité, on s'intéresse à l'arbre des appels récursifs de la fonction `parc_prof` noté A. On remarque que :

→ Chaque appel à `parc_prof` produit 0 ou 2 appels récursifs. Donc A est binaire strict.

→ Grâce au tableau `vu`, chaque état de l'automate correspond à 0 ou 1 noeud interne de A.

Ainsi le nombre de noeuds internes dans A est au plus n et comme il est strict, son nombre de noeuds est au plus $2n + 1$. De plus, sans compter les appels récursifs, chaque appel à `parc_prof` s'exécute en temps constant. En conclusion :

Le temps d'exécution de `etats_accessibles` est $\mathcal{O}(n)$ avec n le nombre d'états dans l'automate.

Question 8 – Soit n le nombre d'états de l'automate et $n' \geqslant 1$ le nombre d'états accessibles. La difficulté est de trouver un moyen simple de renuméroter les états accessibles entre 0 et $n'-1$. Pour cela on utilise les deux fonctions précédentes :

→ Un appel à « `etats_accessibles aut` » renvoie une liste `access` contenant les états accessibles.

→ Un appel à « `numero n access` » renvoie un tableau `num` tel que `num.(q)` est le nouveau numéro de l'état `q`.

En particulier, comme 0 est le premier élément de `access`, l'état initial reste numéroté par 0 dans le nouvel automate.

```
let partie_accessible (aut: automate): automate =
  let (n, delta, f) = aut in
  let access = etats_accessibles aut in
  let num = numero n access in
  let n2 = List.length access in
  let delta2 = Array.make n2 (-1,-1) in
  let f2 = Array.make n2 false in
  List.iter (fun q ->
      let succ_a, succ_b = delta.(q) in
      delta2.(num.(q)) <- (num.(succ_a), num.(succ_b));
      f2.(num.(q)) <- f.(q)
    ) access;
  (n2, delta2, f2);;
```

Question 9 – On obtient :

Q	$\varphi(q)$
E	C
F	C
G	D

Question 10 – On obtient :

Q	$\varphi(q)$
H	C
I	C
J	D
K	D

Question 11 – Supposons par l'absurde qu'il existe $\varphi : \{A, B\} \to \{C, D\}$ un morphisme de $\mathcal{A}_1$ vers $\mathcal{A}_2$. On remarque que :

→ Par le point (2) de la définition de morphisme : $\varphi(A) = C$.

→ Par le point (1) de la définition de morphisme : φ est surjective, donc $\varphi(B) = D$.

En utilisant le point (3) avec $q = A$ et $\sigma = a$:

$$\varphi(\delta_1(A, a)) = \delta_2(\varphi(A), a) \qquad \text{donc} \qquad \varphi(B) = \delta_2(C, a) \qquad \text{donc} \qquad D = C$$

Ce qui constitue une contradiction.

Question 12 – Supposons par l'absurde qu'il existe $\varphi : \{L, M, N\} \to \{C, D\}$ un morphisme de $\mathcal{A}_5$ vers $\mathcal{A}_2$. On remarque que :

→ Par le point (2) de la définition de morphisme : $\varphi(L) = C$.

→ Par le point (4) de la définition de morphisme : $\varphi(M) \in F_2$ donc $\varphi(M) = D$.

→ Par le point (4) de la définition de morphisme : $\varphi(N) \notin F_2$ donc $\varphi(N) = C$.

En utilisant le point (3) avec $q = N$ et $\sigma = b$:

$$\varphi(\delta_5(N, b)) = \delta_2(\varphi(N), b) \qquad \text{donc} \qquad \varphi(L) = \delta_2(C, b) \qquad \text{donc} \qquad C = D$$

Ce qui constitue une contradiction.

Question 13 – Soient $\mathcal{A} = (Q_\mathcal{A}, i_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A})$ et $\mathcal{B} = (Q_\mathcal{B}, i_\mathcal{B}, \delta_\mathcal{B}, F_\mathcal{B})$ deux automates. Supposons qu'il existe $\varphi : Q_\mathcal{A} \to Q_\mathcal{B}$ un morphisme de $\mathcal{A}$ vers $\mathcal{B}$ et montrons que $\mathcal{A}$ et $\mathcal{B}$ acceptent le même langage. Pour cela, on montre d'abord par récurrence sur $|w|$ que si $w \in \{a, b\}^*$ alors :

$$\forall q \in Q_\mathcal{A} : \varphi(\delta^*_\mathcal{A}(q, w)) = \delta^*_\mathcal{B}(\varphi(q), w)$$

Initialisation. Pour $w = \varepsilon$:

$$\varphi(\delta^*_\mathcal{A}(q, \varepsilon)) = \varphi(q) = \delta^*_\mathcal{B}(\varphi(q), \varepsilon)$$

Hérédité. Supposons la propriété vraie pour un mot $w \in \{a, b\}^*$. Soit $\sigma \in \{a, b\}$. Alors pour tout $q \in Q_\mathcal{A}$:

$$\begin{aligned}
\varphi(\delta^*_\mathcal{A}(q, \sigma w)) &= \varphi(\delta^*_\mathcal{A}(\delta_\mathcal{A}(q, \sigma), w)) \\
&= \delta^*_\mathcal{B}(\varphi(\delta_\mathcal{A}(q, \sigma)), w) && \text{d'après l'hypothèse de récurrence appliquée à } \delta_\mathcal{A}(q, \sigma) \\
&= \delta^*_\mathcal{B}(\delta_\mathcal{B}(\varphi(q), \sigma), w) && \text{d'après le point (3) de la définition de morphisme} \\
&= \delta^*_\mathcal{B}(\varphi(q), \sigma w).
\end{aligned}$$

Conclusion. Pour tout mot $w \in \{a, b\}^*$:

$$\begin{aligned}
w \in \mathscr{L}(\mathcal{A}) &\Leftrightarrow \delta^*_\mathcal{A}(i_\mathcal{A}, w) \in F_\mathcal{A} \\
&\Leftrightarrow \varphi(\delta^*_\mathcal{A}(i_\mathcal{A}, w)) \in F_\mathcal{B} && \text{par le point (4)} \\
&\Leftrightarrow \delta^*_\mathcal{B}(\varphi(i_\mathcal{A}), w) \in F_\mathcal{B} && \text{d'après ce qui précède} \\
&\Leftrightarrow \delta^*_\mathcal{B}(i_\mathcal{B}, w) \in F_\mathcal{B} && \text{par le point (2)} \\
&\Leftrightarrow w \in \mathscr{L}(\mathcal{B})
\end{aligned}$$

Question 14 – Soient $\mathcal{A} = (Q_\mathcal{A}, i_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A})$ et $\mathcal{B} = (Q_\mathcal{B}, i_\mathcal{B}, \delta_\mathcal{B}, F_\mathcal{B})$ deux automates ayant le même nombre d'états. Supposons qu'il existe $\varphi : Q_\mathcal{A} \to Q_\mathcal{B}$ un morphisme de $\mathcal{A}$ vers $\mathcal{B}$. D'après le point (1) de la définition de morphisme, φ est surjective. Comme $Q_\mathcal{A}$ et $Q_\mathcal{B}$ sont des ensembles finis et que $|Q_\mathcal{A}| = |Q_\mathcal{B}|$:

$\boxed{\varphi \text{ est bijective.}}$

Montrons que φ^{-1} est un morphisme de $\mathcal{B}$ vers $\mathcal{A}$ en vérifiant les points (1) à (4) :

(1) φ^{-1} est bijective, donc elle est surjective.

(2) $\varphi(i_\mathcal{A}) = i_\mathcal{B}$ donc $i_\mathcal{A} = \varphi^{-1}(i_\mathcal{B})$.

(3) Pour tout $q \in Q_\mathcal{B}$ et $\sigma \in \{a, b\}$, si on note $q' = \varphi^{-1}(q)$, alors :

$$\begin{aligned}
\varphi^{-1}(\delta_\mathcal{B}(q, \sigma)) &= \varphi^{-1}(\delta_\mathcal{B}(\varphi(q'), \sigma)) \\
&= \varphi^{-1}(\varphi(\delta_\mathcal{A}(q', \sigma))) \\
&= \delta_\mathcal{A}(q', \sigma) \\
&= \delta_\mathcal{A}(\varphi^{-1}(q), \sigma)
\end{aligned}$$

(4) Pour tout $q \in Q_{\mathcal{B}}$, si on note $q' = \varphi^{-1}(q)$:

$$\begin{aligned} q \in F_{\mathcal{B}} &\Leftrightarrow \varphi(q') \in F_{\mathcal{B}} \\ &\Leftrightarrow q' \in F_{\mathcal{A}} \\ &\Leftrightarrow \varphi^{-1}(q) \in F_{\mathcal{A}} \end{aligned}$$

Finalement :

$$\boxed{\varphi^{-1} \text{ est un morphisme de } \mathcal{B} \text{ vers } \mathcal{A}}$$

Question 15 – Soient $\mathcal{A} = (Q_{\mathcal{A}}, i_{\mathcal{A}}, \delta_{\mathcal{A}}, F_{\mathcal{A}})$, $\mathcal{B} = (Q_{\mathcal{B}}, i_{\mathcal{B}}, \delta_{\mathcal{B}}, F_{\mathcal{B}})$ et $\mathcal{C} = (Q_{\mathcal{C}}, i_{\mathcal{C}}, \delta_{\mathcal{C}}, F_{\mathcal{C}})$ trois automates. Soient $\varphi_1 : \mathcal{A} \to \mathcal{B}$ et $\varphi_2 : \mathcal{B} \to \mathcal{C}$ deux morphismes d'automates. Montrons que $\varphi_2 \circ \varphi_1$ est un morphisme de $\mathcal{A}$ vers $\mathcal{C}$ en vérifiant les points (1) à (4) de la définition :

(1) $\varphi_2 \circ \varphi_1$ est surjective en tant que composée de fonctions surjectives.

(2) $\varphi_2 \circ \varphi_1(i_{\mathcal{A}}) = \varphi_2(i_{\mathcal{B}}) = i_{\mathcal{C}}$.

(3) Pour tout $q \in Q_{\mathcal{A}}$ et tout $\sigma \in \{a, b\}$:

$$\varphi_2 \circ \varphi_1(\delta_{\mathcal{A}}(q, \sigma)) = \varphi_2(\delta_{\mathcal{B}}(\varphi_1(q), \sigma)) = \delta_{\mathcal{C}}(\varphi_2 \circ \varphi_1(q), \sigma)$$

(4) Pour tout $q \in Q_{\mathcal{A}}$:

$$\begin{aligned} q \in F_{\mathcal{A}} &\Leftrightarrow \varphi_1(q) \in F_{\mathcal{B}} \\ &\Leftrightarrow \varphi_2 \circ \varphi_2(q) \in F_{\mathcal{C}} \end{aligned}$$

Question 16 – Soient $\mathcal{A} = (Q_{\mathcal{A}}, i_{\mathcal{A}}, \delta_{\mathcal{A}}, F_{\mathcal{A}})$ et $\mathcal{B} = (Q_{\mathcal{B}}, i_{\mathcal{B}}, \delta_{\mathcal{B}}, F_{\mathcal{B}})$ deux automates accessibles et $\varphi : Q_{\mathcal{A}} \to Q_{\mathcal{B}}$ une fonction vérifiant les points (2), (3) et (4) de la définition de morphisme. Il s'agit de montrer que φ est surjective.

Soit $q \in Q_{\mathcal{B}}$. Comme $\mathcal{B}$ est accessible, il existe un mot $w \in \{a, b\}^*$ tel que $\delta^*_{\mathcal{B}}(i_{\mathcal{B}}, w) = q$. Posons $q' = \delta^*_{\mathcal{A}}(i_{\mathcal{A}}, w)$. D'après la propriété montrée par récurrence dans la question 13 (qui a été montrée sans le point (1) de la définition de morphisme), on a :

$$\varphi(\delta^*_{\mathcal{A}}(i_{\mathcal{A}}, w)) = \delta^*_{\mathcal{B}}(\varphi(i_{\mathcal{A}}), w) \qquad \text{donc} \qquad \varphi(q') = q$$

Ainsi, tout état $q \in Q_{\mathcal{B}}$ possède un antécédent par φ, c'est à dire que φ est surjective.

Question 17 – Soient $\mathcal{A} = (Q_{\mathcal{A}}, i_{\mathcal{A}}, \delta_{\mathcal{A}}, F_{\mathcal{A}})$ et $\mathcal{B} = (Q_{\mathcal{B}}, i_{\mathcal{B}}, \delta_{\mathcal{B}}, F_{\mathcal{B}})$ deux automates accessibles. Pour écrire `existe_morphisme`, on modifie la fonction `etats_accessibles` de la question 7. Pour cela, on lance en parallèle deux parcours en profondeur sur $\mathcal{A}$ et $\mathcal{B}$: ces deux parcours partent de $i_{\mathcal{A}}$ et $i_{\mathcal{B}}$ en suivant les mêmes étiquettes en parallèle.

Supposons qu'il existe un morphisme $\varphi : \mathcal{A} \to \mathcal{B}$. À chaque étape du double parcours, si on note $q_{\mathcal{A}} \in Q_{\mathcal{A}}$ et $q_{\mathcal{B}} \in Q_{\mathcal{B}}$ les états courants, on doit avoir $\varphi(q_{\mathcal{A}}) = q_{\mathcal{B}}$. Le morphisme $\varphi : \mathcal{A} \to \mathcal{B}$ est donc construit au fur et à mesure du double parcours :

→ Si $\varphi(q_{\mathcal{A}})$ n'est pas encore défini, on pose $\varphi(q_{\mathcal{A}}) = q_{\mathcal{B}}$.

→ Si $\varphi(q_{\mathcal{A}})$ est défini, mais que $\varphi(q_{\mathcal{A}}) \neq q_{\mathcal{B}}$, alors il n'existe pas de morphisme de $\mathcal{A}$ vers $\mathcal{B}$.

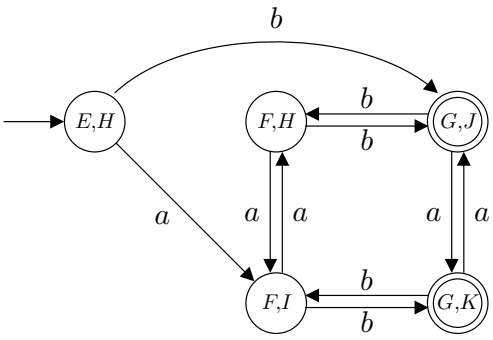
→ Sinon, $\varphi(q_{\mathcal{A}})$ est défini et $\varphi(q_{\mathcal{A}}) = q_{\mathcal{B}}$. Dans ce cas, on peut continuer le parcours sans modifier φ.

Si cette procédure parvient à construire une fonction $\varphi : Q_{\mathcal{A}} \to Q_{\mathcal{B}}$, alors φ vérifie les points (2) et (3) de la définition de morphisme. Il reste alors à tester si le point (4) est vrai, c'est le rôle de la fonction `verif4`. Notons que le point (1) est automatiquement vérifié en vertu de la question 16.

```
(* Vérifie la propriété (4) d'un morphisme *)
let verif4 (aut1: automate) (aut2: automate) (phi: morphisme): bool =
  let (n1, _, f1) = aut1 in
  let (n2, _, f2) = aut2 in
  let q = ref 0 in
  let test() =
    f1.(!q) && f2.(phi.(!q)) || not f1.(!q) && not f2.(phi.(!q)) in
  while !q < n1 && test() do
    incr q;
  done;
  !q = n1;;
```

```
let existe_morphisme (aut1: automate) (aut2: automate): bool * morphisme =
  let (n1, delta1, f1) = aut1 in
  let (n2, delta2, f2) = aut2 in
  let phi = Array.make n1 (-1) in
  let rec parc_prof (li: (int * int) list): bool = match li with
    | [] -> true
    | (q1, q2) :: tl when phi.(q1) = q2 -> parc_prof tl
    | (q1, q2) :: tl when phi.(q1) <> q2 && phi.(q1) <> -1  -> false
    | (q1, q2) :: tl ->
       phi.(q1) <- q2;
       let (succ_a1, succ_b1) = delta1.(q1) in
       let (succ_a2, succ_b2) = delta2.(q2) in
       parc_prof ((succ_a1,succ_a2) :: (succ_b1,succ_b2) :: tl) in
  let b1 = parc_prof [(0,0)] in
  (b1 && verif4 aut1 aut2 phi, phi);;
```

Question 18 – On obtient :



Question 19 –

```
(* L'état (q1, q2) est numéroté par q1 + q2*n1 *)
let produit (aut1: automate) (aut2: automate): automate =
  let (n1, delta1, f1) = aut1 in
  let (n2, delta2, f2) = aut2 in
  let n = n1 * n2 in
  let delta = Array.make n (-1,-1) in
  let f = Array.make n false in
  for q = 0 to n-1 do
    let q1 = q mod n1 in
    let q2 = q/n1 in
    let (succ_a1, succ_b1) = delta1.(q1) in
    let (succ_a2, succ_b2) = delta2.(q2) in
    delta.(q) <- (succ_a1 + succ_a2 * n1, succ_b1 + succ_b2 * n1);
    f.(q) <- f1.(q1) && f2.(q2)
  done;
  (n, delta, f);;
```

Question 20 – Soient $\mathcal{A} = (Q_\mathcal{A}, i_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A})$ et $\mathcal{A}' = (Q_{\mathcal{A}'}, i_{\mathcal{A}'}, \delta_{\mathcal{A}'}, F_{\mathcal{A}'})$ deux automates. À l'aide d'une récurrence sur $|w|$, on peut montrer que si $w \in \{a,b\}^*$, alors :

$$\forall (q,q') \in Q_\mathcal{A} \times Q_{\mathcal{A}'} : \delta^*_{\mathcal{A}\times\mathcal{A}'}((q,q'),w) = (\delta^*_\mathcal{A}(q,w), \delta^*_{\mathcal{A}'}(q',w))$$

Soit $(q,q') \in Q_\mathcal{A} \times Q_{\mathcal{A}'}$ un état accessible de $\mathcal{A} \times \mathcal{A}'$. Par définition, il existe un mot $w \in \{a,b\}^*$ tel que :

$$(q,q') = \delta^*_{\mathcal{A}\times\mathcal{A}'}((i_\mathcal{A}, i_{\mathcal{A}'}), w)$$

D'après ce qui précède :

$$q = \delta^*_\mathcal{A}(i_\mathcal{A}, w) \qquad \text{et} \qquad q' = \delta^*_{\mathcal{A}'}(i_{\mathcal{A}'}, w)$$

Donc :

$$q \in F_\mathcal{A} \Leftrightarrow w \in \mathscr{L}(\mathcal{A}) \qquad \text{et} \qquad q' \in F_{\mathcal{A}'} \Leftrightarrow w \in \mathscr{L}(\mathcal{A}')$$

Ainsi, si on suppose que $\mathscr{L}(\mathcal{A}) = \mathscr{L}(\mathcal{A}')$, alors $q \in F_\mathcal{A} \Leftrightarrow q' \in F_{\mathcal{A}'}$.

Question 21 – Soient $\mathcal{A} = (Q_\mathcal{A}, i_\mathcal{A}, \delta_\mathcal{A}, F_\mathcal{A})$ et $\mathcal{A}' = (Q_{\mathcal{A}'}, i_{\mathcal{A}'}, \delta_{\mathcal{A}'}, F_{\mathcal{A}'})$ deux automates accessibles acceptant le même langage. Soit $\mathcal{B} = (Q_\mathcal{B}, i_\mathcal{B}, \delta_\mathcal{B}, F_\mathcal{B})$ la partie accessible de $\mathcal{A} \times \mathcal{A}'$. Montrons qu'il existe un morphisme $\varphi : \mathcal{B} \to \mathcal{A}$ (l'existence d'un morphisme $\varphi' : \mathcal{B} \to \mathcal{A}'$ se montre de manière analogue). Soit :

$$\varphi : \begin{cases} Q_\mathcal{B} \to Q_\mathcal{A} \\ (q,q') \mapsto q \end{cases}$$

Montrons que φ est un morphisme de $\mathcal{B}$ vers $\mathcal{A}$ en vérifiant les points (1) à (4) de la définition :

(1) Soit $q \in Q_\mathcal{A}$. Comme $\mathcal{A}$ est accessible, il existe $w \in \{a,b\}^*$ tel que $\delta^*_\mathcal{A}(i_\mathcal{A}, w) = q$. On pose :

$$q_0 = \delta^*_\mathcal{B}(i_\mathcal{B}, w) \qquad \text{alors} \qquad q_0 = (\delta^*_\mathcal{A}(i_\mathcal{A}, w), \delta^*_{\mathcal{A}'}(i_{\mathcal{A}'}, w)) = (q, \delta^*_{\mathcal{A}'}(i_{\mathcal{A}'}, w))$$

Donc $\varphi(q_0) = q$. Ainsi, tout état $q \in Q_\mathcal{A}$ a un antécédent par φ qui est donc surjective.

(2) On a :

$$\varphi(i_\mathcal{B}) = \varphi((i_\mathcal{A}, i_{\mathcal{A}'})) = i_\mathcal{A}$$

(3) Pour tout $(q,q') \in Q_\mathcal{B}$ et tout $\sigma \in \{a,b\}$:

$$\varphi(\delta_\mathcal{B}((q,q'),\sigma)) = \varphi(\delta_\mathcal{A}(q,\sigma), \delta_{\mathcal{A}'}(q',\sigma)) = \delta_\mathcal{A}(q,\sigma) = \delta_\mathcal{A}(\varphi(q,q'),\sigma)$$

(4) Soit $(q,q') \in Q_{\mathcal{B}}$. Montrons $(q,q') \in F_{\mathcal{B}} \Leftrightarrow q \in F_{\mathcal{A}}$ par double implication.
($\Rightarrow$) Si $(q,q') \in F_{\mathcal{B}}$, alors $q \in F_{\mathcal{A}}$ car $F_{\mathcal{B}} \subset F_{\mathcal{A}} \times F_{\mathcal{A}'}$.
($\Leftarrow$) Réciproquement, si $q \in F_{\mathcal{A}}$ alors le résultat de la question précédente stipule que $q' \in F_{\mathcal{A}'}$ et donc :

$$(q,q') \in (F_{\mathcal{A}} \times F_{\mathcal{A}'}) \cap Q_{\mathcal{B}} = F_{\mathcal{B}}$$

Question 22 – Dans la suite, on dira qu'une suite $(q_0, q_1, q_2, \dots, q_k)$ est un (φ,ψ)-trajet entre p et q si elle vérifie les propriétés de l'énoncé :

$$\begin{cases} p = q_0, q_k = q \\ \forall 0 \leqslant j < k : \Big[\varphi(q_j) = \varphi(q_{j+1}) \text{ ou } \psi(q_j) = \psi(q_{j+1})\Big] \end{cases}$$

Montrons que $\equiv$ est réflexive, symétrique et transitive :

- $\rightarrow$ $\equiv$ est réflexive. En effet pour tout $p \in Q_{\mathcal{B}}$, la suite (p) est un (φ,ψ)-trajet entre p et p de longueur $k = 0$. Donc $p \equiv p$.
- $\rightarrow$ $\equiv$ est symétrique. Soient $(p,q) \in Q_{\mathcal{B}}^2$ tels que $p \equiv q$. Par définition, il existe un (φ,ψ)-trajet $(q_0, q_1, q_2, \dots, q_k)$ entre p et q. On vérifie facilement que la suite écrite à l'envers $(q_k \dots, q_2, q_1, q_0)$ est un (φ,ψ)-trajet entre q et p. Donc $q \equiv p$.
- $\rightarrow$ $\equiv$ est transitive. Soient $(p,q,r) \in Q_{\mathcal{B}}^3$ tels que $p \equiv q$ et $q \equiv r$. Par définition, il existe un (φ,ψ)-trajet $(q_0, q_1, q_2, \dots, q_{k_1})$ de longueur $k_1 \in \mathbb{N}$ entre p et q et un (φ,ψ)-trajet $(q'_0, q'_1, q'_2, \dots, q'_{k_2})$ de longueur $k_2 \in \mathbb{N}$ entre q et r. Ces (φ,ψ)-trajets vérifient $q_{k_1} = q = q'_0$ et on vérifie facilement que leur concaténation est un (φ,ψ)-trajet entre p et r de longueur $k_1 + k_2$:

  $$(q_0, q_1, q_2, \dots, q_{k_1}, q'_1, q'_2, \dots, q'_{k_2})$$

  Donc $p \equiv r$.

Question 23 – Soient $(p,q) \in Q_{\mathcal{B}}^2$ tels que $p \equiv q$ et soit $\sigma \in \{a,b\}$. Par définition, il existe un (φ,ψ)-trajet $\gamma = (q_0, q_1, q_2, \dots, q_k)$ entre p et q. Soit $\gamma' = (q'_0, q'_1, q'_2, \dots, q'_k)$ le uplet défini par :

$$\forall j \in [\![\, 0\,;k \,]\!] : q'_j = \delta_{\mathcal{B}}(q_j, \sigma)$$

D'après le point (3) de la définition de morphisme :

$$\forall j \in [\![\, 0\,;k \,]\!] : \begin{cases} \varphi(q'_j) = \delta_{\mathcal{A}}(\varphi(q_j), \sigma) \\ \psi(q'_j) = \delta_{\mathcal{A}'}(\psi(q_j), \sigma) \end{cases}$$

Ainsi pour tout $j \in [\![\, 0\,;k-1 \,]\!]$:

$$\Big[\varphi(q_j) = \varphi(q_{j+1}) \text{ ou } \psi(q_j) = \psi(q_{j+1})\Big] \qquad \text{donc} \qquad \Big[\varphi(q'_j) = \varphi(q'_{j+1}) \text{ ou } \psi(q'_j) = \psi(q'_{j+1})\Big]$$

On en déduit que γ' est un (φ,ψ)-trajet entre $q'_0 = \delta_{\mathcal{B}}(q_0,\sigma)$ et $q'_k = \delta_{\mathcal{B}}(q_k,\sigma)$. Donc $\delta_{\mathcal{B}}(q_0,\sigma) \equiv \delta_{\mathcal{B}}(q_k,\sigma)$.

Question 24 – Supposons d'abord que $\varphi(p) = \varphi(q)$. Le point (4) de la définition de morphisme donne :

$$\begin{aligned} p \in F_{\mathcal{B}} &\Leftrightarrow \varphi(p) \in F_{\mathcal{A}} \\ &\Leftrightarrow \varphi(q) \in F_{\mathcal{A}} \\ &\Leftrightarrow q \in F_{\mathcal{B}} \end{aligned}$$

De même, si $\psi(p) = \psi(q)$ alors $p \in F_{\mathcal{B}} \Leftrightarrow q \in F_{\mathcal{B}}$.

Soient $(p,q) \in Q_{\mathcal{B}}^2$ tels que $p \equiv q$. Par définition, il existe un (φ,ψ)-trajet $\gamma = (q_0, q_1, q_2, \dots, q_k)$ entre p et q. Pour tout $j \in [\![\, 0\,;k-1 \,]\!]$:

$$\Big[\varphi(q_j) = \varphi(q_{j+1}) \text{ ou } \psi(q_j) = \psi(q_{j+1})\Big] \qquad \text{donc} \qquad q_j \in F_{\mathcal{B}} \Leftrightarrow q_{j+1} \in F_{\mathcal{B}}$$

Par transitivité de $\Leftrightarrow$:

$$q_0 \in F_{\mathcal{B}} \Leftrightarrow q_k \in F_{\mathcal{B}} \qquad \text{c'est à dire} \qquad p \in F_{\mathcal{B}} \Leftrightarrow q \in F_{\mathcal{B}}$$

Question 25 – On pose $\mathcal{C} = (Q_\mathcal{C}, i_\mathcal{C}, \delta_\mathcal{C}, F_\mathcal{C})$ l'automate défini par :

$$Q_\mathcal{C} = \{S_0, S_1, \ldots, S_{\ell-1}\} \qquad i_\mathcal{C} = S_0 = [i_\mathcal{B}] \qquad F_\mathcal{C} = \bigcup_{q \in F_\mathcal{B}} \{[q]\}$$

et pour tout $q \in Q_\mathcal{B}$ et $\sigma \in \{a, b\}$:

$$\delta_\mathcal{C}([q], \sigma) = [\delta_\mathcal{B}(q, \sigma)]$$

D'après la question 23, la classe $[\delta_\mathcal{B}(q, \sigma)]$ ne dépend pas du représentant q choisi. Ainsi :

La définition de $\delta_\mathcal{B}$ et donc de $\mathcal{C}$ est licite.

Par récurrence sur $|w|$, on peut montrer que si $w \in \{a, b\}^*$ alors :

$$\forall q \in Q_\mathcal{B} : \delta^*_\mathcal{C}([q], w) = [\delta^*_\mathcal{B}(q, w)]$$

Comme $\mathcal{B}$ est accessible, pour tout $q \in Q$ il existe un mot $w \in \{a, b\}^*$ tel que :

$$q = \delta^*_\mathcal{B}(i_\mathcal{B}, w) \qquad \text{donc} \qquad [q] = \delta^*_\mathcal{B}(i_\mathcal{C}, w)$$

Donc :

$\mathcal{C}$ est accessible.

il reste à montrer que η est un morphisme de $\mathcal{B}$ vers $\mathcal{C}$:

(1) Soit $S \in Q_\mathcal{C}$. Comme S est non vide, on peut considérer $q \in S$. Alors $\eta(q) = S$ et donc η est surjective.

(2) Par définition, $\eta(i_\mathcal{B}) = i_\mathcal{C}$.

(3) Pour tout $q \in Q_\mathcal{B}$ et tout $\sigma \in \{a, b\}$, la définition de $\delta_\mathcal{C}$ donne :

$$\eta(\delta_\mathcal{B}(q, \sigma)) = \delta_\mathcal{C}(\eta(q), \sigma)$$

(4) Pour tout $q \in Q_\mathcal{B}$, le résultat de la question 24 donne :

$$\begin{aligned} q \in F_\mathcal{B} &\Leftrightarrow \exists q' \in F_\mathcal{B}, q \equiv q' \\ &\Leftrightarrow \eta(q) \in F_\mathcal{C} \end{aligned}$$

Ainsi :

$\eta : \mathcal{B} \to \mathcal{C}$ est un morphisme.

Question 26 – Construisons le morphisme $\varphi' : \mathcal{A} \to \mathcal{C}$ (le morphisme $\psi' : \mathcal{A}' \to \mathcal{C}$ se construit de manière analogue).

⋆ Pour tout $p_\mathcal{A} \in Q_\mathcal{A}$, on pose :

$$E(p_\mathcal{A}) = \Big\{\eta(p) : p \in Q_\mathcal{B}, \varphi(p) = p_\mathcal{A}\Big\}$$

On remarque que pour tout $(p, q) \in Q^2_\mathcal{B}$, si $\varphi(p) = \varphi(q)$, alors (p, q) est un (φ, ψ)-trajet de longueur 1 entre p et q, donc $p \equiv q$ et donc $\eta(p) = \eta(q)$. Ainsi $|E(p_\mathcal{A})| \leqslant 1$ et comme φ est surjective, $|E(p_\mathcal{A})| = 1$. Le morphisme φ' est alors défini par :

$$\varphi' : Q_\mathcal{A} \to Q_\mathcal{C} \text{ est la fonction telle que } \varphi'(p_\mathcal{A}) \text{ est l'unique élément de } E(p_\mathcal{A})$$

⋆ Montrons que φ' est un morphisme :

(1) Soit $S \in Q_{\mathcal{C}}$. Comme S est non vide, on peut considérer $p \in S$ et poser $p_{\mathcal{A}} = \varphi(p)$. Ainsi :

$$\eta(p) \in E(p_{\mathcal{A}}) \qquad \text{donc} \qquad \varphi'(p_{\mathcal{A}}) = \eta(p) = S$$

Donc φ' est surjective.

(2) Comme $\varphi(i_{\mathcal{B}}) = i_{\mathcal{A}}$:

$$\eta(i_{\mathcal{B}}) \in E(i_{\mathcal{A}}) \qquad \text{donc} \qquad \varphi'(i_{\mathcal{A}}) = \eta(i_{\mathcal{B}}) = i_{\mathcal{C}}$$

(3) Soit $p_{\mathcal{A}} \in Q_{\mathcal{A}}$ et $\sigma \in \{a, b\}$. On a $\varphi'(p_{\mathcal{A}}) = \eta(p)$ où $p \in Q_{\mathcal{B}}$ et $\varphi(p) = p_{\mathcal{A}}$. Donc :

$$\delta_{\mathcal{C}}(\varphi'(p_{\mathcal{A}}), \sigma) = \delta_{\mathcal{C}}(\eta(p), \sigma) = \eta(\delta_{\mathcal{B}}(p, \sigma)) \qquad \text{et} \qquad \varphi(\delta_{\mathcal{B}}(p, \sigma)) = \delta_{\mathcal{A}}(\varphi(p), \sigma) = \delta_{\mathcal{A}}(p_{\mathcal{A}}, \sigma)$$

Ainsi :

$$\delta_{\mathcal{C}}(\varphi'(p_{\mathcal{A}}), \sigma) \in E(\delta_{\mathcal{A}}(p_{\mathcal{A}}, \sigma)) \qquad \text{donc} \qquad \delta_{\mathcal{C}}(\varphi'(p_{\mathcal{A}}), \sigma) = \varphi'(\delta_{\mathcal{A}}(p_{\mathcal{A}}, \sigma))$$

(4) Soit $p_{\mathcal{A}} \in Q_{\mathcal{A}}$. On a $\varphi'(p_{\mathcal{A}}) = \eta(p)$ où $p \in Q_{\mathcal{B}}$ et $\varphi(p) = p_{\mathcal{A}}$. Donc :

$$\begin{aligned} p_{\mathcal{A}} \in F_{\mathcal{A}} &\Leftrightarrow \varphi(p) \in F_{\mathcal{A}} \\ &\Leftrightarrow p \in F_{\mathcal{B}} \\ &\Leftrightarrow \eta(p) \in F_{\mathcal{B}} \\ &\Leftrightarrow \varphi'(p_{\mathcal{A}}) \in F_{\mathcal{B}} \end{aligned}$$

Question 27 –

```
(* Renvoie 0 si le tableau est vide *)
let maxi (t: int array): int =
  Array.fold_left max 0 t;;
```

```
(* Pour chaque élément e de t, on remplace e par f.(e). *)
let renomme (t: int array): int array =
  let n = Array.length t in
  let k = ref 0 in
  let f = Array.make (maxi t + 1) (-1) in
  for i = 0 to n-1 do
    match f.(t.(i)) with
      | -1 -> f.(t.(i)) <- !k;
              incr k
      | _ -> ()
  done;
  Array.map (fun e -> f.(e)) t;;
```

On remarque que :

Le complexité de `renomme` est en $\mathcal{O}(n+m)$ où n la taille du tableau en entrée et m sa valeur maximale.

Question 28 –

```
(* inv.(k) est la liste des i tels que phi.(i) = k *)
let inv_morph (phi: morphisme): int list array =
  let n = Array.length phi in
  let inv = Array.make n [] in
  for i = 0 to n-1 do
    inv.(phi.(i)) <- i :: inv.(phi.(i))
  done;
  inv;;
```

```
(* Pour chaque état q0, on recherche tous les états ayant la même image que q0
   par le morphisme \eta. Pour cela, on utilise une procédure similaire au
   parcours en profondeur. *)
let relation (phi: morphisme) (psi: morphisme): morphisme =
  let n = Array.length phi in
  let inv_phi = inv_morph phi in
  let inv_psi = inv_morph psi in
  let t = Array.make n (-1) in
  let rec explorer (q0: int) (li: int list): unit = match li with
    | [] -> ()
    | q :: tl when t.(q) <> -1 -> explorer q0 tl
    | q :: tl -> t.(q) <- q0;
                 explorer q0 tl;
                 explorer q0 inv_phi.(phi.(q));
                 explorer q0 inv_psi.(psi.(q)) in
  for q0 = 0 to n-1 do
    explorer q0 [q0]
  done;
  renomme t;;
```

Question 29 – Pour le montrer on utilise les questions précédentes. Soit $\mathcal{B}$ la partie accessible de $\mathcal{A} \times \mathcal{A}'$. D'après la question 21, il existe deux morphismes $\varphi : \mathcal{B} \to \mathcal{A}$ et $\psi : \mathcal{B} \to \mathcal{A}'$. D'après les questions 25 et 26, il existe un automate $\mathcal{C}$ et deux morphismes $\varphi' : \mathcal{A} \to \mathcal{C}$ et $\psi' : \mathcal{A}' \to \mathcal{C}$.

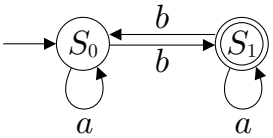
Question 30 – Soit $\mathcal{B} = (Q_\mathcal{B}, i_\mathcal{B}, \delta_\mathcal{B}, F_\mathcal{B})$ l'automate de la question 18, c'est à dire la partie accessible de $\mathcal{A}_3 \times \mathcal{A}_4$. La réponse à la question 21 permet de construire deux morphismes $\varphi : \mathcal{B} \to \mathcal{A}_3$ et $\psi : \mathcal{B} \to \mathcal{A}_4$:

q	$\varphi(q)$	$\psi(q)$
(E,H)	E	H
(F,H)	F	H
(F,I)	F	I
(G,J)	G	J
(G,K)	G	K

On en déduit que $Q_\mathcal{B}$ possède deux classes d'équivalences (voir la définition partie 4.2) :

$$S_0 = \Big\{(E,H);(F,H);(F,I)\Big\} \qquad\qquad S_1 = \Big\{(G,J),(G,K)\Big\}$$

L'automate $\mathcal{C}$ demandé par l'énoncé est :



Pour $\varphi' : \mathcal{A}_3 \to \mathcal{C}$ et $\psi' : \mathcal{A}_4 \to \mathcal{C}$ on obtient :

q	$\varphi'(q)$
E	S_0
F	S_0
G	S_1

q	$\psi(q)$
H	S_0
I	S_0
J	S_1
K	S_1

Question 31 – Soient $\mathcal{A}$ et $\mathcal{A}'$ deux automates de $\mathfrak{K}_L$ ayant m_L états. Le résultat de la question 29 garantit l'existence d'un automate $\mathcal{C}$ et deux morphisme $\varphi' : \mathcal{A} \to \mathcal{C}$ et $\psi' : \mathcal{A}' \to \mathcal{C}$. D'après la question 13 :

$$\mathscr{L}(\mathcal{C}) = \mathscr{L}(\mathcal{A}) = \mathscr{L}(\mathcal{A}')$$

Étant donné qu'un morphisme est surjectif (point (1) de la définition de morphisme), le nombre d'états dans $\mathcal{C}$ est au plus m_L. Ainsi, par minimalité de m_L, le nombre d'états de $\mathcal{C}$ est égal à m_L. D'après la question 14, φ' et ψ' sont bijectives et φ'^{-1} et ψ'^{-1} sont des morphismes. Par la question 15, la fonction $\gamma = \psi'^{-1} \circ \varphi'$ est un morphisme $\gamma : \mathcal{A} \to \mathcal{A}'$ et il s'agit d'un isomorphisme par la question 14.

Question 32 – La démonstration est similaire à celle de la question précédente. Soit $\mathcal{M}_L$ un automate de $\mathfrak{K}_L$ ayant m_L états. Le résultat de la question 29 garantit l'existence d'un automate $\mathcal{C}$ et deux morphisme $\varphi' : \mathcal{A} \to \mathcal{C}$ et $\psi' : \mathcal{M}_L \to \mathcal{C}$. D'après la question 13 :

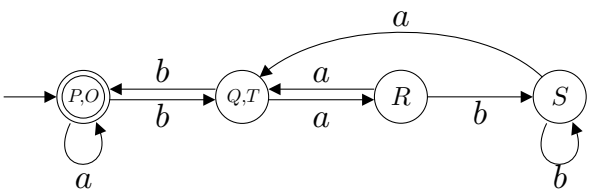
$$\mathscr{L}(\mathcal{C}) = \mathscr{L}(\mathcal{A}) = \mathscr{L}(\mathcal{M}_L)$$

Étant donné qu'un morphisme est surjectif (point (1) de la définition de morphisme), le nombre d'états dans $\mathcal{C}$ est au plus m_L. Ainsi, par minimalité de m_L, le nombre d'états de $\mathcal{C}$ est égal à m_L. D'après la question 14, ψ' est bijective et ψ'^{-1} est un morphisme. Par la question 15, la fonction $\varphi = \psi'^{-1} \circ \varphi'$ est un morphisme $\varphi : \mathcal{A} \to \mathcal{M}_L$.

Question 33 – Soit φ le morphisme recherché. Comme on doit avoir $\varphi(O) = \varphi(P)$, d'après le point (3) de la définition de morphisme appliquée à $(q, \sigma) = (O, b)$ et $(q, \sigma) = (P, b)$:

$$\varphi(T) = \varphi(Q)$$

On obtient :



q	$\varphi(q)$
P	(P, O)
Q	(Q, T)
R	R
S	S
T	(Q, T)
O	(P, O)

Question 34 – Supposons par l'absurde que $A_6^{Q,R}$ et ψ existent. On pose :

$$\mathcal{A}_6 = (Q_6, i_6, \delta_6, F_6) \qquad \text{et} \qquad \mathcal{A}_6^{Q,R} = (Q_6', i_6', \delta_6', F_6')$$

D'après le point (3) de la définition de morphisme :

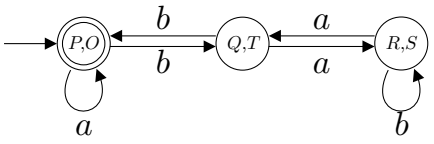
$$\psi(O) = \psi(\delta_6(Q, b)) = \delta_6'(\psi(Q), b) = \delta_6'(\psi(R), b) = \psi(\delta_6(R, b)) = \psi(S)$$

D'après le point (4) de la définition de morphisme :

$$\begin{aligned} O \in F_6 &\Leftrightarrow \psi(O) \in F_6' \\ &\Leftrightarrow \psi(S) \in F_6' \\ &\Leftrightarrow S \in F_6 \end{aligned}$$

Or $O \in F_6$ et $S \notin F_6$, d'où la contradiction.

Question 35 – On peut fusionner les états R et S pour obtenir :



Question 36 –

```
(* tdp = table des prédécesseurs *)
let table_de_predecesseurs (aut: automate): bool array array =
  let (n, delta, f) = aut in
  let tdp = Array.make_matrix n n false in
  let rec parc_prof (p: int) (q: int): unit = match () with
    | () when tdp.(p).(q) -> ()
    | () -> tdp.(p).(q) <- true;
            let (succ_ap, succ_bp) = delta.(p) in
            let (succ_aq, succ_bq) = delta.(q) in
            parc_prof succ_ap succ_aq;
            parc_prof succ_bp succ_bq in
  for p = 0 to n-1 do
    for q = 0 to n-1 do
      if f.(p) && not f.(q) then (parc_prof p q; parc_prof q p)
    done;
  done;
  tdp;;
```

Remarque. Il semble y avoir une erreur d'énoncé dans la question 36 : au vu de la question 37, il aurait été plus logique de considérer les (φ, ψ)-trajets entre (p, q) et (p_0, q_0) à la place des (φ, ψ)-trajets entre (p_0, q_0) et (p, q).

Question 37 – Soit $\mathcal{A} = (Q, i, \delta, F)$ un automate et P le graphe de la question 36. Notons $\equiv$ la relation d'équivalence sur $Q_{\mathcal{A}}$ définie par :

$$\forall (p,q) \in Q_{\mathcal{A}}^2 : p \equiv q \Leftrightarrow \neg \mathcal{P}(p,q) \qquad \text{où} \qquad \mathcal{P}(p,q) : \begin{cases} \text{Il existe un couple } (p_0,q_0) \in Q_{\mathcal{A}}^2 \text{ et un } (\varphi,\psi)\text{-} \\ \text{trajet de } (p,q) \text{ vers } (p_0,q_0) \text{ dans } P \text{ tel que } p_0 \in F \\ \text{et } q_0 \notin F \text{ ou } p_0 \notin F \text{ et } q_0 \in F. \end{cases}$$

En suivant la même construction que dans la question 25, on obtient un automate $\mathcal{C} \in \mathfrak{K}_L$ dont les états sont les classes d'équivalence de $\equiv$, ainsi qu'un morphisme $\eta : \mathcal{A} \to \mathcal{C}$.

Montrons que $\mathcal{C}$ possède m_L états. D'après la question 32, il existe un morphisme $\varphi : \mathcal{C} \to \mathcal{B}$ où $\mathcal{B}$ est un automate de $\mathfrak{K}_L$ à m_L états. Soient $(p, q) \in Q_{\mathcal{A}}^2$ tel que $\mathcal{P}(p, q)$. En utilisant une récurrence sur la longueur du (φ, ψ)-trajet entre (p, q) et (p_0, q_0), on peut montrer que $\varphi(\eta(p)) \neq \varphi(\eta(q))$. En d'autres termes :

$$\forall (p,q) \in Q_{\mathcal{A}}^2 : \varphi(\eta(p)) = \varphi(\eta(q)) \Rightarrow \eta(p) = \eta(q)$$

On en déduit que φ est injective et donc bijective. D'où le résultat.

```
type graphe_oriente = (int * int) list array array;;

(* Renvoie le graphe P défini dans la question 36 *)
let inv_aut (aut: automate): graphe_oriente =
  let (n, delta, _) = aut in
  let g = Array.make_matrix n n [] in
  for p = 0 to n-1 do
    for q = 0 to n-1 do
      let (succ_ap, succ_bp) = delta.(p) in
      let (succ_aq, succ_bq) = delta.(q) in
      g.(succ_ap).(succ_aq) <- (p,q) :: g.(succ_ap).(succ_aq);
      g.(succ_bp).(succ_bq) <- (p,q) :: g.(succ_bp).(succ_bq);
    done;
  done;
  g;;

(* eq.(p).(q) vaut true lorsque p et q sont équivalents *)
let classe_eq (aut: automate): bool array array =
  let (n, _, f) = aut in
  let g = inv_aut aut in
  let eq = Array.make_matrix n n true in
  let rec parc_prof (p,q): unit = match eq.(p).(q) with
    | false -> ()
    | true ->  eq.(p).(q) <- false;
               List.iter parc_prof g.(p).(q) in
  for p = 0 to n-1 do
    for q = 0 to n-1 do
      if f.(p) && not f.(q) then (parc_prof (p,q); parc_prof (q,p))
    done;
  done;
  eq;;

(* Cette fonction construit le morphisme entre A et l'automate réduit.
   m est le nombre de classes d'équivalence *)
let get_morph (aut: automate): int * morphisme =
  let (n, _ , _) = aut in
  let eq = classe_eq aut in
  let m = ref 0 in
  let phi = Array.make n (-1) in
  for p = 0 to n-1 do
    if phi.(p) = -1 then begin
        incr m;
        for q = 0 to n-1 do
          if eq.(p).(q) then phi.(q) <- !m-1
        done;
      end;
  done;
  !m, phi;;
```

```
let reduit (aut: automate): automate =
  let (n1, delta1 , f1) = aut in
  let n2, phi = get_morph aut in
  (***** Construction des classes *****)
  let classes = Array.make n2 [] in
  for q1 = 0 to n1-1 do
    classes.(phi.(q1)) <- q1 :: classes.(phi.(q1));
  done;
  (***** Construction de la fonction de transition *****)
  let delta2 = Array.make n2 (-1, -1) in
  for q2 = 0 to n2-1 do
    let succ_a, succ_b = delta1.(List.hd classes.(q2)) in
    delta2.(q2) <- phi.(succ_a), phi.(succ_b);
    if not (List.for_all (fun q1 ->
                let s, r = delta1.(q1) in
                delta2.(q2) = (phi.(s) , phi.(r))
              ) classes.(q2)) then failwith "reduit: erreur 1";
  done;
  (***** Construction des états finaux *****)
  let f2 = Array.make n2 false in
  for q2 = 0 to n2-1 do
    f2.(q2) <- f1.(List.hd classes.(q2));
    if not (List.for_all (fun q1 ->
                f2.(q2) = f1.(q1)
              ) classes.(q2)) then failwith "reduit: erreur 2";
  done;
  (**********)
  (n2, delta2, f2);;
```