Sujet Centrale 2017 : correction

I.A. Notons q cet état. Pour tout $m \in \Sigma^*$, $m.q = q$ donc m est synchronisant : tout mot est synchronisant.
I.B. Si $m = a^n$ et n est impair, $m.1 = 2$, $m.2 = 1$; si n est pair, $m.1 = 1$ et $m.2 = 2$; pour tout m, $m.1 \neq m.2$. De ce fait, il n'existe pas de mot m et d'état q_0 tel que, pour tout état q, $m.q = q_0$. La machine n'a pas de mot synchronisant.
I.C. *bcb* est synchronisant.
I.D.

```
let delta_etoile m q u = match u with
    | [] -> q
    | x::v -> delta_etoile m (m.delta q x) v;;
```

I.E.

```
let est_synchronisant m u = let q=delta_etoile m 0 u in
    let rec aux i = if i=m.n_etats then true else (delta_etoile m i u)=q && aux(i+1) in
    aux 1;;
```

I.F. Supposons que, pour tout $q \neq q'$, pour toute lettre x, $q.x \neq q'.x$. Un mot synchronisant ne peut être vide car, pour tout $q \neq q'$, $q.\epsilon = q \neq q' = q'.\epsilon$. Si M admet un mot synchronisant, il en admet un de taille minimal que l'on notera m. Soit x sa dernière lettre : $m = m'x$. Il existe q_0 tel que, pour tout q, $q.m = q_0$ donc $(q.m').x = q_0$. Supposons que, pour tout q, q', $q.m' = q'.m'$; alors q' est synchronisant et de taille strictement inférieure à celle de m. Par conséquent, il existe $q \neq q'$ tel que $q.m' \neq q.m$. Notons $q_1 = q.m'$ et $q_2 = q'.m'$. On a donc $q_1 \neq q_2$ et $q_1.x = q.m = q.m' = q_2.x$.
I.G.1. Pour tout ensemble P, $\hat{\delta}^*(P, m) = \{\delta^*(p, m) \,/\, p \in P\}$. Supposons qu'il existe un mot synchronisant m. Notons q_0 tel que, pour tout $q \in Q$, $q.m = q_0$. Alors $\hat{\delta}^*(Q, m) = \{\delta^*(p, m) \,/\, p \in Q\} = \{q_0\}$. Ainsi, il existe un ensemble singleton accessible depuis l'état Q de la machine des parties. Réciproquement, supposons qu'il existe un état singleton $\{q_0\}$ accessible depuis l'état Q. Alors, il existe un mot m tel que $\hat{\delta}^*(Q, m) = \{q_0\}$ donc $\{\delta^*(q, m) \,/\, q \in Q\} = \{q_0\}$ donc, pour tout $q \in Q$, $m.q = q_0$ donc m est synchronisant. Ainsi, il existe un mot synchronisant si et seulement s'il existe un état singleton accessible depuis l'état Q dans la machine des parties.
I.G.2. Pour une machine $M = (Q, \Sigma, \delta)$, on pose $A = (\mathcal{P}(Q), Q, F, \hat{\delta})$ où $F = \{\{q\} \,/\, q \in Q\}$. Alors un mot m est reconnu par A si et seulement s'il existe $q \in Q$ tel que $\hat{\delta}^*(Q, m) = \{q\}$ ce qui équivaut à ce que m est synchronisant. $LS(M)$ est donc reconnaissable.
I.G.3.

$LS(M_0)$ est décrit par $11(0|1)^*$.
I.H. On considère la machine $\tilde{M} = (Q, \Sigma, \tilde{\delta})$ obtenue à partir de $M = (Q, \Sigma, \delta)$ en modifiant toute transition issue de q_0 par une transition de même étiquette de destination q_0. Comme toute transition dans cette nouvelle machine issue de q_0 atterrit en q_0, le seul état destination d'un mot synchronisant de $\tilde{M}$ est q_0. De plus, pour tout $q \in Q$, pour tout mot u, $\tilde{\delta}^*(q, u) = q_0$ si le chemin de q à $q.u$ dans M passe par q_0 et $\tilde{\delta}^*(q, u) = \delta^*(q, u)$ sinon (et dans ce cas $\delta^*(q, u) \neq q_0$). Par conséquent, u est un mot synchronisant pour $\tilde{M}$ si et seulement si, pour tout $q \in Q$, le chemin de q à $u.q$ dans M passe par q_0.
II.A.1.

```
let ajoute f x =
    if (f.fin=f.deb && not f.vide) then failwith "File pleine";
```

```
    f.vide <- false;
    f.tab.(f.fin) <- x;
    if f.fin < Array.length f.tab-1 then f.fin <- f.fin+1 else f.fin <- 0;;
```

II.A.2.

```
let retire f =
    if f.vide then failwith "File vide";
    let x=f.tab.(f.deb) in
    if f.deb=Array.length f.tab -1 then f.deb <- 0 else f.deb <- f.deb+1;
    x;;
```

II.A.3. Les deux ne font qu'un nombre borné d'opérations (quelque soit l taille de la liste) donc elles ont une complexité bornée en $O(1)$.

II.B. La seule instruction pouvant ne pas se terminer est la boucle "tant que". Tout sommet inséré initialement dans F a une valeur de D finie. Si, au début d'un passage de boucle, tout sommet de F a une valeur finie, si s est le sommet en tête de la file F, s est retiré de F ; les sommets s' insérés lors de ce passage ont une valeur finie puisque $D[s']$ prend la valeur $D[s]+1$ qui est finie. Ainsi, tout sommet de F a une valeur finie. Pour être inséré dans F, par définition de la boucle, un sommet s' doit avoir, avant son insertion, valeur infinie par D. De ce fait, un sommet ne peut être inséré qu'une fois dans la file F. A chaque passage de boucle, un sommet est sorti de cette file. Par conséquent, il peut y avoir qu'au plus $|S|$ passages dans la boucle, un pour chaque sommet. Cette boucle se termine bien donc l'algorithme aussi.

II.C. Les créations des tableaux D et P sont en $O(|S|)$. La première boucle sur E effectue, à chaque passage de boucle, un nombre borné (4) d'opérations donc sa complexité est $O(|E|) = O(|S|)$.

Chaque passage de boucle de la boucle "tant que" parcourt tous les arcs adjacents au sommet de la tête de la file et effectue pour chacun de ces arcs un nombre au plus borné d'opérations. La complexité du passage de boucle avec s en tête est donc $|A_s|$ où A_s est l'ensemble des sommets adjacents à s. Il y a au plus un passage de boucle pour chaque sommet donc la complexité totale de cette boucle est en $O(\sum_{s\in S} |A_s|) = O(|A|)$.

La complexité totale de l'algorithme est donc $O(|A| + |S|)$.

II.D. Avant le premier passage de boucle, tous les sommets s dans F vérifient $D[s] = 0$ donc la propriété est vraie.

Supposons cette propriété au début d'un passage de boucle. Tout sommet s' inséré dans F le sera en prenant la valeur $D[s'] = D[s_1]+1$ donc ces sommets s' nouvellement insérés auront tous la même valeur et cette valeur sera $D[s'] = D[s_1]+1 \geq D[s_r]$ donc, enfin, les sommets de la file $s_2, \cdots, s_r, s'_1, \cdots, s'_q$ sont bien rangés par ordre croissant de D. Enfin, pour tout sommet s' nouvellement inséré, $D[s'] - D[s_2] \leq D[s'] - D[s_1] = 1$. La propriété reste donc vraie à la fin de ce passage de boucle.

Par récurrence, cette propriété est vraie au début de chaque passage de boucle.

II.E.1. Par définition de l'algorithme, si $D[s] < \infty$, s a été inséré dans la file F. Tout sommet inséré dans F est un sommet de E ou un voisin d'un sommet déjà inséré dans la file. De ce fait, il existe une suite $s_0, \cdots, s_p = s$ de S tel que $s_0 \in E$ et, pour tout $0 \leq i \leq p-1$, (s_i, s_{i+1}) est un arc. Il existe donc un chemin de s_0 à s de taille p donc s est accessible depuis E et $d_s \leq p$. En outre, par définition de l'algorithme, pour tout $0 \leq i \leq p$, $D[s_{i+1}] = D[s_i] + 1$ et $D[s_0] = 0$ donc $D[s] = p$. Ainsi, s est accessible et $d_s \leq D[s]$.

Réciproquement, supposons qu'il existe un sommet accessible tel que $D[s] = +\infty$. Choisissons un tel sommet s à distance minimal de E, distance que l'on notera p. Si $p = 0$, $s \in E$ ce qui est absurde car tout sommet de E a valeur $D[s] = 0$. Ainsi $p \geq 1$ et s est voisin d'un sommet s' à distance $p-1$ de E. Comme $d_{s'} < d_s$, s' vérifie $D[s'] < \infty$; l'algorithme devrait alors fixer $D[s]$ à $D[s']+1$ ce qui contredit que $D[s] = \infty$. Finalement, tout sommet tel que $D[s] = \infty$ n'est pas accessible.

c est diminué de 1 à chaque fois qu'une valeur de D passe de ∞ à une valeur finie donc à chaque sommet accessible ; de plus, la valeur de D d'un sommet n'est modifiée et ne devient finie qu'à condition qu'elle soit au préalable infinie ; cette modification n'a donc lieu qu'une et une seule fois pour chaque sommet accessible. Par conséquent, c est le nombre de sommets non accessibles.

II.E.2. Supposons qu'il existe s tel que $D[s] > d_s$. Choisissons s un tel sommet à distance minimal p de E. Comme, pour tout $e \in E$, $D[e] = 0 = d_e$, $d_s > 0$. s est donc voisin d'un sommet s' à distance $p-1$ de E. On a donc $D[s'] = d_{s'} = p-1$. s' est inséré dans la file F, nécessairement avant s car les sommets y sont insérés par ordre croissante de D (question II.D.). De ce fait, s doit être inséré lors du parcours de s' ou d'un autre de ses voisins à distance $p-1$ de E. On aura donc $D[s] = D[s'] + 1 = p = d_s$ ce qui contredit l'hypothèse effectuée. Ainsi, $D[s] = d_s$ pour tout sommet s accessible depuis E.

$P[s]$ est alors l'origine et l'étiquette d'un arc aboutissant à s faisant partie d'un chemin minimal de E à s.

II.F.

```
let accessibles v e = let n=Array.length v in let f={tab=Array.make n 0; deb=0; fin=0; vide=true}
   and d=Array.make n -1 and p=Array.make n (-2,-1) and c=ref n;
   let rec boucle1 e = matche e with
       | [] -> ()
       | s::e0 -> ajoute f s; d.(s) <- 0; p.(s) <- (-1,-1); c:= !c-1; aux e0
   in boucle1 e;
   let rec auxboucle2 s l = match l with
      | [] -> ()
      | (t,x)::l0 -> if d.(t)=-1 then begin d.(t) <- d.(s)+1; p.(t) <- (s,x); ajoute f t; c:= !c-1 end in
```

```
   while not (f.vide) do
       let s=retire f in auxboucle2 s v.(s)
   done;
   (!c, d, p);;
```

II.G.

```
let chemin s p = let l= ref [] and t=ref s in
   if p.(s)=(-2,-1) then failwith "état non accessible";
   while p.( !t ) != (-1,-1) do
      let (u,x)=p.( !t) in l:= x::(!l);
      t:=u
   done;
   !l;;
```

III.A.

III.B. $(1,1,0,0)$ est une distribution de vérité satisfaisant F_1. Le mot 1100 est synchronisant.

III.C. Pour toute lettre x, $\delta(q_{i,j}) = q_{i,j+1}$ ou $\delta(q_{i,j}) = f$. Par récurrence, pour un mot u de taille p, $\delta^*(q_{i,j}, u) = f$ ou $\delta^*(q_{i,j}, u) = q_{i,j+p}$. Nécessaire, si $p \geq m+1$, comme $j+p > m+1$, $\delta^*(q_{i,j}, u) = f$. De plus, f étant stable par toute lettre, $\delta^*(f, u) = f$. Ainsi, pour tout mot u de taille au moins $m+1$, pour tout état q, $\delta^*(q, u) = f$. Le mot u est donc synchronisant. Pour tout mot u, $\delta^*(f, u) = f$ donc u est synchronisant si et seulement si, pour tout i, j, $\delta^*(q_{i,j}, u) = f$. En particulier, si u est synchronisant, pour tout i, $q_{i,1}.u = f$. Réciproquement, supposons que, pour tout i, $q_{i,1}.u = f$. Pour tout $j \geq 2$, $\delta^*(q_{i,j}, m) = f$ ou $\delta^*(q_{i,j}, u) = q_{i,j+m}$; comme $j+m > m+1$, $\delta^*(q_{i,j}, u) = f$. Ainsi, u est synchronisant.

III.D. Soit $(v_j)_{1\leq j\leq m}$ une distribution de vérité satisfaisant F. Soit $i \in [\![1, n]\!]$. Considérons la i-ème clause. Cette clause est satisfaite par la distribution donc il existe un littéral x_j ou $\overline{x_j}$ de cette clause qui est satisfait par la distribution. Par ailleurs, $\delta^*(q_{i,1}, v_1 \cdots v_{j-1})$ vaut f ou $q_{i,j}$. Si le littéral satisfait est x_j, $v_j = 1$ et, par définition de δ, $\delta(q_{i,j}, v_j) = f$; si c'est $\overline{x_j}$, $v_j = 0$ et $\delta(q_{i,j}, v_j) = f$. Dans tous les cas, $\delta^*(q_{i,1}, v_1 \cdots v_j) = \delta(\delta^*(q_{i,1}, v_1 \cdots v_{j-1}), v_j) = f$. Pour tout mot u, $\delta^*(f, u) = f$ donc $\delta^*(q_{i,1}, v_1 \cdots v_m) = \delta^*(f, v_{j+1} \cdots v_m) = f$. Ceci vaut pour tout $1 \leq i \leq n$ .D'après la question précédente, $v_1 \cdots v_m$ est un mot synchronisant de longueur m.

III.E. Soit $v_1 \cdots v_m$ un mot de longueur m synchronisant. Soit $1 \leq i \leq n$. Supposons que, pour tout $1 \leq j \leq m$, $\delta(q_{i,j}, v_j) \neq f$. Alors $\delta(q_{i,j}, v_j) = q_{i,j+1}$ donc $\delta^*(q_{i,1}, v_1 \cdots v_m) = q_{i,m+1} \neq f$. Ceci contredit la condition de la question III.C. De ce fait, il existe $1 \leq j \leq m$ tel que $\delta(q_{i,j} v_j) = f$ donc, par définition de δ, un littéral x_j ou $\overline{x_j}$ apparaît dans la clause numéro i ; si c'est x_j, $v_j = 1$, si c'est $\overline{x_j}$, $v_j = 0$; dans les deux cas, v_j satisfait ce littéral donc la i-ème clause est satisfaite par $(v_j)_{1\leq j\leq m}$. Ceci étant vrai pour toute clause de F, la formule F est satisfaite par $(v_j)_{1\leq j\leq m}$.

IV.A.1. Supposons que $q.u_i = q'.u_i$ pour $(q, q') \in Q^2$. Comme u_{i+1} est un préfixe de u_i, $q.u_{i+1} = q'.u_{i+1}$. On a donc une application surjective $Q.u_i \to Q.u_{i+1}$, $q.u_i \mapsto q.u_{i+1}$ qui est surjective. De ce fait, $|Q.u_{i+1}| \leq |Q.u_i|$. La suite des cardinaux $(|Q.u_i|)$ est donc décroisante.

IV.A.2. Soit m un mot synchronisant. Alors, pour tout q, q', $q.m = q'.m$. Réciproquement, supposons que, pour tout $(q, q') \in Q^2$, il existe un mot $u_{q,q'}$ tel que $q.u_{q,q'} = q'.u_{q,q'}$. Montrons que, pour tout $Q' \subset Q$, il existe un mot u et un état q_0 tels que, pour tout $q \in Q'$, $q.u = q_0$. Cette propriété est tautologique si $\text{card}(Q') = 1$. Supposons la propriété vraie pour tout $Q' \subset Q$ de cardinal p. Soit $Q' \subset Q$ de cardinal $p+1$. Soit $q \in Q'$. Par hypothèse de récurrence, il existe un mot u et un état q_0 tel que, pour tout état $q' \in Q' \backslash \{q\}$, $u.q' = q_0$. De par l'hypothèse faite (appliquée aux états $u.q$ et q_0), il existe v tel que $v.(u.q) = v.q_0$. Alors, pour tout $q' \in Q'$, si $q' \neq q$, $vu.q' = v.(u.q') = v.q_0$ et, si $q' = q$, $vu.q = v.q_0$. En conclusion, par récurrence, pour tout $Q' \subset Q$, il existe un mot u et un état q_0 tels que, pour tout $q \in Q'$, $u.q = q_0$. En particulier, il existe un mot m et un état q_0 tels que, pour tout $q \in Q$, $m.q = q_0$. m est donc un mot synchronisant.

IV.B. $\tilde{n} = \binom{n}{1} + \binom{n}{2} = n + \frac{n(n-1)}{2} = \frac{n(n+1)}{2}$.

IV.C.

```
let delta2 m q_barre x = let n=m.n_etats in match nb_to_set n q_barre with
   | [i] -> set_to_nb n [delta i x]
   | [i,j] -> let k=delta i x and l=delta j x in
       if k=l then set_to_nb n [k] else set_to_nb n [k,l];;
```

IV.D.

```
let retourne_machine m = let ne=m.n_etats and nl=m.n_lettres in
    let n0=ne*(ne+1)/2 in let t=Array.make n0 [] in
    for q=0 to n0-1 do
       for x=0 to nl-1 do
          let r=delta2 m q x in t.(r) <- (q,x) :: t.(r)
       done
    done;
    t;;
```

IV.E. Soient q, q' deux états. Alors il existe un mot u tel que $q.u = q'.u$ si et seulement s'il existe un singleton $\{q_0\}$ et un mot u tels que $\tilde{\delta}^*(\{q, q'\}, u) = \{q_0\}$ si et seulement s'il existe un chemin de $\{q, q'\}$ vers un singleton $\{q_0\}$ dans $\tilde{G}$ si et seulement s'il existe un chemin d'un singleton $\{q_0\}$ vers $\{q, q'\}$ dans $\tilde{G}_R$ si et seulement si $\{q, q'\}$ est accessible dans $\tilde{G}_R$ depuis l'ensemble des singletons. D'après la question IV.A., il existe un mot synchronisant dans la machine M si et seulement si tous les états $\{q, q'\}$ de $\tilde{G_R}$ sont accessibles depuis l'ensemble des singletons. Il suffit donc de vérifier via la fonction "accessibles" appliquée au graphe $\tilde{G}_R$ et à l'ensemble des singletons si tous les états de $\tilde{G}_R$ sont accessibles (les singletons l'étant tautologiquement).

IV.F.

```
let existe_synchronisant m = let n=m.n_etats and s= ref [] in
   for i=0 to n-1 do
      s:= (set_to_nb n [i]) :: !s
   done;
   match accessibles (retourne_machine m) !s with p,_,_ -> p=0;;
```