

COMPOSITION D'INFORMATIQUE (CORRECTION<sup>1</sup>)

(Durée : 4 heures)

\* \* \*

## Graphes ET/OU et automates alternants

## Partie I. Préliminaires

**Question 1** [2 pts] La fonction `appartient` est de complexité  $O(|S|)$ .

```
let rec appartient e x = match e with
| [] -> false
| t::q -> (t=x) or (appartient q x);;
```

**Question 2** [4 pts] La fonction `ens_egaux` est de complexité  $O(|S_1| \times |S_2|)$ .

```
let rec ensInclus e1 e2 = match e1 with
| [] -> true
| t::q -> (appartient e2 t) & (ensInclus q e2);;
```

```
let ensEgaux e1 e2 = (ensInclus e1 e2) & (ensInclus e2 e1);;
```

**Question 3** [3 pts] La fonction `union` est de complexité  $O(|S_1| \times |S_2|)$ .

```
let rec union e1 e2 = match e2 with
| [] -> e1
| t::q -> if (appartient t e1) then union e1 q
           else t::(union e1 q);;
```

**Question 4** [3 pts] D'après la question précédente, il existe une constante  $k$  (indépendante des entrées) telle que le coût de l'union de  $S_1$  et  $S_2$  soit majoré par  $k|S_1| \times |S_2|$ . Si on note  $\mathbf{e}=[\mathbf{e}_n; \dots; \mathbf{e}_1]$ , et  $u_n$  la taille de `gunion g [e_n; ...; e_1]` et  $c_n$  son coût de calcul, on obtient  $c_n \leq k|g(e_n)| \times u_{n-1} + c_{n-1} + k'$  ( $k'$  constante indépendante de  $n$ ) ie  $c_n \leq k|g|^2 + c_{n-1} + k' \leq (k|g|^2 + k')n$  d'où une complexité en  $O(|S||g|^2)$ .

```
let rec gunion g e = match e with
| [] -> []
| t::q -> union (g.(t)) (gunion g q);;
```

## Partie II. Graphes ET/OU

**Question 5** [10 pts] La fonction `inter` est de complexité  $O(|S_1| \cdot |S_2|)$  et `ginter` de complexité  $O(|S| \cdot |g|^2)$ . Dans la fonction `est_compatible` :

- le calcul d'une étape est de complexité  $O(|e(s)| \cdot |g|^2)$  et retourne un ensemble de taille  $O(|g|)$ ,
- le calcul d'égalité d'ensembles est de complexité  $(|g|^2)$ ,
- on en déduit donc une complexité totale en  $O(|\mathcal{G}| \cdot |g|^2)$ .

---

<sup>1</sup>cf correction de <http://www.dptinfo.ens-cachan.fr/concours-ens/Ressources/> (BATOG Guillaume)

```

let rec inter e1 e2 = match e2 with
| [] -> []
| t::q -> if (appartient t e1) then t::(inter e1 q)
           else inter e1 q;;

let rec ginter g e = match e with
| [] -> failwith "intersection vide"
| [x] -> g.(x)
| t::q -> inter (g.(t)) (ginter g q);;

let it_vect f u0 tab =
let n = vect_length tab and res = ref u0 in
for i = 0 to (n-1) do
  res := f (!res) (tab.(i))
done; !res;;

let est_compatible graphe g =
let uneEtape graphe g s =
  let opGs = if graphe.f.(s) then union g (graphe.e.(s))
            else ginter g (graphe.e.(s))

  in union opGs [s] in
let it_f u s = u & ( ensEgaux (g.(s)) (uneEtape graphe g s) )
in it_vect it_f true g;;

```

### Question 6

a) [3 pts]

	1	2	3	4	5	6	7	8	9
0	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}	{9}
1	{1, 3}	{1, 2}	{3, 4, 6}	{4}	{5, 8}	{6}	{7, 8}	{3, 8}	{6, 8, 9}
2	{1, 3, 4, 6}	{1, 2, 3}	{3, 4, 6}	{4}	{3, 5, 8}	{6, 8}	{3, 7, 8}	{3, 4, 6, 8}	{3, 6, 8, 9}
3	{1, 3, 4, 6}	{1, 2, 3, 4, 6}	{3, 4, 6, 8}	{3, 4}	{3, 4, 5, 6, 8}	{3, 6, 8}	{3, 4, 6, 7, 8}	{3, 4, 6, 8}	{3, 4, 6, 8, 9}
4	{1, 3, 4, 6, 8}	{1, 2, 3, 4, 6}	{3, 4, 6, 8}	{3, 4, 6}	{3, 4, 5, 6, 8}	{3, 4, 6, 8}	{3, 4, 6, 7, 8}	{3, 4, 6, 8}	{3, 4, 6, 8, 9}
5	{1, 3, 4, 6, 8}	{1, 2, 3, 4, 6, 8}	{3, 4, 6, 8}	{3, 4, 6}	{3, 4, 5, 6, 8}	{3, 4, 6, 8}	{3, 4, 6, 7, 8}	{3, 4, 6, 8}	{3, 4, 6, 8, 9}
6	{1, 3, 4, 6, 8}	{1, 2, 3, 4, 6, 8}	{3, 4, 6, 8}	{3, 4, 6, 8}	{3, 4, 5, 6, 8}	{3, 4, 6, 8}	{3, 4, 6, 7, 8}	{3, 4, 6, 8}	{3, 4, 6, 8, 9}

- b) [4 pts] Notons  $A_n = (A_n(0), \dots, A_n(N-1))$  et  $A_{n+1} = \psi(A_n)$ . La suite  $(A_n)_{n \geq 0}$  est croissante (pour l'inclusion, car ses composantes le sont) majorée par  $\{0, \dots, N-1\}^N$  donc converge et stationne vers  $A_G$ . De plus si  $A_M$  est point fixe de  $\psi$ , alors  $A_n = A_M$  pour tout  $n \geq M$  (par récurrence sur  $n$ ) donc l'ensemble des  $n$  tel que  $A_n$  n'est pas point fixe de  $\psi$  est majoré par  $|A_G| \leq N^2$  donc on peut choisir  $M = N^2$  (même  $N^2 - N$  car  $|A_0| = N$ ).
- c) [6 pts] Montrons par récurrence sur  $n$  que pour tout  $s$ ,  $A_{n+1}(s) = \{s\} \cup \bigoplus_{(s,s') \in E} A_n(s')$  (où  $\bigoplus = \cup$  ou  $\cap$ ). Pour  $n = 0$ , c'est la définition de  $A_1$ . Supposons l'égalité vraie au rang  $n$ .

$$A_{n+2}(s) = A_{n+1}(s) \cup \bigoplus_{(s,s') \in E} A_{n+1}(s') = \{s\} \cup \bigoplus_{(s,s') \in E} A_n(s') \cup \bigoplus_{(s,s') \in E} A_{n+1}(s')$$

en appliquant l'hypothèse de récurrence à  $A_{n+1}(s)$ . Or la croissance des  $A_n(s)$  permet d'écrire que  $\bigoplus_{(s,s') \in E} A_n(s') \subseteq \bigoplus_{(s,s') \in E} A_{n+1}(s')$  d'où  $A_{n+2}(s) = \{s\} \cup \bigoplus_{(s,s') \in E} A_{n+1}(s)$ . En choisissant  $n = M$ , on obtient que  $A_G$  est compatible pour  $\mathcal{G}$ .

Soit  $g$  compatible pour  $\mathcal{G}$ . Montrons par récurrence sur  $n$  que pour tout  $s$ ,  $A_n(s) \subseteq g(s)$ . Avec  $n = M$ , on obtient le résultat souhaité. Pour  $n = 0$ ,  $\{s\} \subseteq g(s)$  car  $g$  est compatible. Supposons  $A_n(s) \subseteq g(s)$ , alors

$$A_{n+1}(s) = A_n(s) \cup \bigoplus_{(s,s') \in E} A_n(s') \subseteq g(s) \cup \bigoplus_{(s,s') \in E} g(s') \subseteq g(s)$$

la dernière inégalité venant du fait que  $g$  est compatible donc  $g(s)$  contient  $\bigoplus_{(s,s') \in E} g(s')$ .

d) [10 pts]

- l'initialisation de `aG` prend un temps  $O(|V|)$ ,
- le calcul d'une étape se fait en  $O(|e(s)| \cdot |g|^2)$  (cf question 5),
- à chaque itération de boucle, la réinitialisation de `aG` demande  $O(|E| \cdot |aG|^2)$  opérations avec  $|aG| \leq |V|^2$  et  $|E| \leq |V|^2$  d'où  $O(|V|^4)$  opérations, et le test  $A_n = A_{n+1}$  s'effectue en  $O(|V|^3)$  opérations,
- on obtient alors une complexité finale en  $O(|V|^6)$ .

```
let gEgaux t1 t2 =
let n = vect_length t1 and res = ref true in
for i = 0 to (n-1) do
  res := !res & (ensEgaux (t1.(i)) (t2.(i)))
done; !res;;
```

```
let rel_acc graphe =
let n = graphe.v in
let aG = ref ( init_vect n (fun x -> [x]) ) and test = ref true in
let uneEtape graphe g s =
  let opGs = if graphe.f.(s) then gunion g (graphe.e.(s))
              else ginter g (graphe.e.(s))
  in union opGs (g.(s)) in
while !test do
  let aG2 = init_vect n (fun s -> uneEtape graphe !aG s) in
  if gEgaux aG aG1 then test := false
  else aG := aG2
done; !aG;;
```

**Question 7** [5 pts] La construction de `aG` prend un temps  $O(|V|^6)$  d'après la question précédente. Pour chacune des étapes de `parcours_aG`, le test de vacuité se calcule en  $O(|aG(j)| \cdot |S|)$  d'où une complexité totale du parcours en  $O(|aG| \cdot |S|) = O(|V|^3)$ . La fonction `rel_acc_inv` est donc de complexité  $O(|V|^6)$  ( $O(|V|^3)$  si on dispose de `AG`).

```
let rel_acc_inv graphe e =
let n = graphe.v and aG = rel_acc graphe in
let it_f u s =
  if (inter (aG.(j)) e) = [] then u else j::u
in it_vect it_f [] aG;;
```

### Question 8

- a) [4 pts] La création de `pre` s'effectue en  $O(|V|)$  opérations et la fonction `parcours_aretes` est linéaire en le nombre d'arêtes donc `pred` est de complexité  $O(|G|)$ .

```
let pred graphe =
let n = graphe.v in
let pre = make_vect n [] in
let rec parcours_aretes = function
  | -1 -> ()
  | j -> let f_do_list a = pre.(a) <- j::(pre.(a)) in
          do_list f_do_list (graphe.e.(j));
          parcours_aretes (j-1)
in parcours_aretes (n-1); pre;;
```

- b) [2 pts]

```
let nb_succ graphe =
let f i = if graphe.f.(i) then 1 else list_length (graphe.e.(i)) in
init_vect (graphe.v) f;;
```

### Question 9

a) [5 pts] Les initialisations des lignes 1-3 se font  $O(|\mathcal{G}|)$  d'après la question précédente. Pour tout  $s$  dans  $V$ ,  $s$  ne peut être ajouté qu'au plus une fois dans  $S$  :  $n(s)$  est décrémenté à chaque visite dans la boucle 7-12 et  $s$  est ajouté à  $S$  si  $n(s) = 0$ . Ainsi la boucle 4-13 est exécutée au plus  $|V|$  fois et les instructions 8-11 sont globalement exécutées au plus  $\sum_{s \in V} |\text{pred}(s)| = |E|$ . Au total, le coût de l'algorithme est au plus  $O(|V|) + O(|E|) = O(|\mathcal{G}|)$ .

b) [15 pts] Montrons l'invariant de boucle suivant (pour la boucle externe) :

1. pour tout élément  $s \in S \cup T$ ,  $s_0 \in A_{\mathcal{G}}(s)$ ,
2. pour tout  $s \notin S \cup T$  tel que  $f(s) = \wedge$ ,  $n(s) = |G(s)| - |G(s) \cap T|$ .

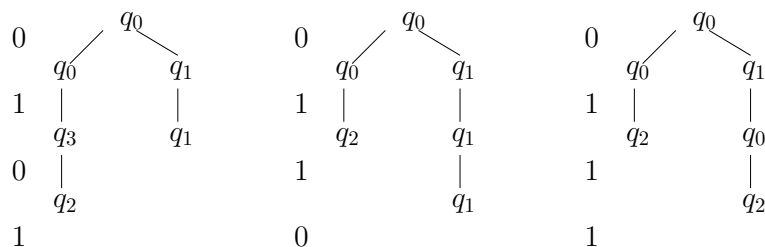
Pour cela, on raisonne par récurrence sur le nombre d'exécutions de la boucle externe. C'est vrai initialement. Supposons donc l'invariant avant une exécution de la boucle externe. Soient  $T', S', n'$  les nouvelles valeurs de  $T, S, n$  à la sortie de cette boucle : on obtient  $T' = s :: T$  et, pour tout  $s_1 \in \text{prec}(s)$ ,  $n'(s_1) = n(s_1) - 1$ . Si  $f(s_1) = \wedge$ , on a donc  $G(s_1) \cap T' = (G(s_1) \cap T) \cup \{s\}$  et donc  $|G(s_1)| - |G(s_1) \cap T'| = |G(s_1)| - |G(s_1) \cap T| - 1 = n(s_1) - 1$  ( $T'$  ne contient pas de répétition), soit  $n'(s_1) = |G(s_1)| - |G(s_1) \cap T'|$ .

Par ailleurs, si  $s_1 \in S' \setminus S$ , alors  $n(s_1) = 1$  et, ou bien  $f(s_1) = \vee$  et dans ce cas, comme  $(s_1, s) \in E$  et  $s_0 \in A_{\mathcal{G}}(s)$ , on a aussi  $s_0 \in A_{\mathcal{G}}(s_1)$ , par compatibilité de  $A_{\mathcal{G}}$ . Ou bien  $f(s) = \wedge$ . Dans ce cas,  $n'(s_1) = 0$  entraîne  $G(s_1) = G(s_1) \cap T'$  et donc  $G(s_1) \subseteq T'$ . Or, par hypothèse de récurrence, pour tout  $s_2 \in T'$ ,  $s_0 \in A_{\mathcal{G}}(s_2)$  donc pour tout  $s_2 \in G(s_1)$ ,  $s_0 \in A_{\mathcal{G}}(s_2)$ . Par compatibilité de  $A_{\mathcal{G}}$ ,  $A_{\mathcal{G}}(s_1) \supseteq \bigcap_{s_2 \in G(s_1)} A_{\mathcal{G}}(s_2)$  et donc  $s_0 \in A_{\mathcal{G}}(s_1)$ . Réciproquement, si  $s \in A_{\mathcal{G}}(s_0)$ , soit  $n$  le plus petit entier tel que  $s \in A_n(s_0)$  (il existe d'après la question 5.3). On montre par récurrence sur  $n$  que  $s \in T$  après exécution de l'algorithme. Pour cela, il suffit de montrer que  $s$  est ajouté à  $S$  à l'une des étapes de calcul. Si  $n = 0$ , alors  $s = s_0$  et donc  $s \in S$  (au départ). Si la propriété est vraie pour  $p < n$  et  $n > 0$  supposons d'abord que  $f(s) = \vee$ . Dans ce cas,  $A_n(s) = A_{n-1}(s) \cup \bigcup_{(s, s') \in E} A_{n-1}(s')$  et, par minimalité de  $n$ ,  $s_0 \in A_{n-1}(s')$  pour au moins un sommet  $s'$  tel que  $(s, s') \in E$ . Par hypothèse de récurrence,  $s'$  est ajouté à  $S$  à une étape de l'algorithme. D'autre part, par définition,  $s \in \text{prec}(s')$ . Lorsque  $s'$  est ajouté à  $T$ , la boucle suivante ajoute  $s$  à  $S$  (puisque  $n(s)$  est initialement à 1 et n'est jamais passé par 0). Si maintenant  $f(s) = \wedge$ ,  $A_n(s) = A_{n-1}(s) \cup \bigcap_{(s, s') \in E} A_{n-1}(s')$  et, par minimalité de  $n$ ,  $s_0 \in A_{n-1}(s')$  pour tout  $s$  tel que  $(s, s') \in E$ . Par hypothèse de récurrence, chacun des sommets  $s'$  est ajouté à  $S$  à l'une des étapes de l'algorithme. Chaque fois que l'un des sommets  $s'$  est ajouté à  $T$ ,  $n(s)$  est décrémenté (et seulement à ces occasions). Lorsque le dernier des sommets  $s'$  est ajouté à  $T$ ,  $n(s) = 0$  et  $s$  est ajouté à  $S$ .

c) [2 pts] On calcule d'abord **pred** et **succ** en temps  $O(|\mathcal{G}|)$  d'après la question 8. Puis on applique l'algorithme de la figure 2 avec  $s_0 = s$  qui s'exécute en temps  $O(|\mathcal{G}|)$  d'après la question a). D'après la question b), il suffit alors de tester l'appartenance de  $t$  à l'ensemble  $T = A_{\mathcal{G}}^{-1}(\{s\})$  (test linéaire).

### Partie III. Automates alternants

#### Question 10 [3 pts]



Question 11 [3 pts] À l'aide de la définition d'un calcul réussi, on montre par double inclusion :

$$\mathcal{L}(\mathcal{A}[q]) = \bigcup_{a \in \{0,1\}} \{a\} \cdot \bigoplus_{q' \in \delta(q,a)} \mathcal{L}(\mathcal{A}[q'])$$

où  $\oplus$  représente une intersection si  $\tau(q, a) = \wedge$  et une réunion si  $\tau(q, a) = \vee$  avec la convention  $\cap_{\emptyset} = \{\epsilon\}$  (ensemble réduit au mot vide) et  $\cup_{\emptyset} = \emptyset$ .

**Question 12** [6 pts] Soit  $\mathcal{A}_1 = (Q, A, I, F, \delta)$  un automate fini non-déterministe donné. On construit l'automate alternant  $\mathcal{A}_2 = (Q, A, I, F, \delta, \tau)$  où on définit pour tout  $q$  et  $a$ ,  $\tau(q, a) = \vee$ .  $\mathcal{A}_2$  est bien de taille  $O(|\mathcal{A}_1|)$ .

Montrons par récurrence sur  $|w|$  que pour tout  $q \in Q$ ,  $w \in \mathcal{L}(\mathcal{A}_1[q])$  si et seulement si  $w \in \mathcal{L}(\mathcal{A}_2[q])$ . Le cas de base est immédiat par identité des états de  $\mathcal{A}_1$  et  $\mathcal{A}_2$ . Puisque tous les états de  $\mathcal{A}_2$  sont étiquetés par  $\vee$ , les calculs réussis de  $w$  de  $\mathcal{A}_2$  sont des listes de longueur  $|w|$  (arbres linéaires) dont on montre la correspondance directe avec les calculs réussis de  $\mathcal{A}_1$  à l'aide de la récurrence énoncée plus haut.

**Question 13** [10 pts]

- a) On montre par récurrence sur  $|w|$  que pour tout  $q \in Q$ , si  $w \in \mathcal{L}(\mathcal{A}[q])$  alors  $w \notin \mathcal{L}(\mathcal{A}^c[q])$ . Le cas de base se déduit de la complémentarité des états finaux et non finaux de  $\mathcal{A}$ . Considérons  $w$  de taille  $n$  vérifiant l'hypothèse de récurrence, soient  $q \in Q$  et  $a \in A$ , supposons  $aw \in \mathcal{L}(\mathcal{A}[q])$ . Utilisons le résultat de la question 11 :
- Si  $\tau(q, a) = \vee$ , alors il existe  $q' \in \delta(q, a)$  tel que  $w \in \mathcal{L}(\mathcal{A}[q'])$ . Par hypothèse de récurrence,  $w \notin \mathcal{L}(\mathcal{A}^c[q'])$  pour tout  $q' \in I$ . Puisque  $\tau^c(q, a) = \wedge$ , d'après la question 11,  $aw \notin \mathcal{L}(\mathcal{A}^c[q])$  d'où le résultat.
  - Si  $\tau(q, a) = \wedge$ , alors pour tout  $q' \in \delta(q, a)$ ,  $w \in \mathcal{L}(\mathcal{A}[q'])$ . Par hypothèse de récurrence,  $w \notin \mathcal{L}(\mathcal{A}^c[q'])$  pour tout  $q' \in \delta(q, a)$  donc  $aw \notin \mathcal{L}(\mathcal{A}^c[q])$  d'après la question 11.
- Ainsi en appliquant ce résultat pour  $q = q_0$  (état initial), on obtient que les langages  $\mathcal{L}(\mathcal{A})$  et  $\mathcal{L}(\mathcal{A}^c)$  sont disjoints.
- b) On montre par récurrence sur  $|w|$  que pour tout  $q \in Q$ , si  $w \notin \mathcal{L}(\mathcal{A}[q])$  alors  $w \in \mathcal{L}(\mathcal{A}^c[q])$ . Le cas de base se déduit de la complémentarité des états finaux et non finaux de  $\mathcal{A}$ . Considérons  $w$  de taille  $n$  vérifiant l'hypothèse de récurrence, soient  $q \in Q$  et  $a \in A$ , supposons  $aw \notin \mathcal{L}(\mathcal{A}[q])$ . Utilisons le résultat de la question 11 :
- Si  $\tau(q, a) = \vee$ , alors pour tout  $q' \in \delta(q, a)$ ,  $w \notin \mathcal{L}(\mathcal{A}[q'])$ . Par hypothèse de récurrence,  $w \in \mathcal{L}(\mathcal{A}^c[q'])$  pour tout  $q' \in I$ . Puisque  $\tau^c(q, a) = \wedge$ , d'après la question 11,  $aw \in \mathcal{L}(\mathcal{A}^c[q])$  d'où le résultat.
  - Si  $\tau(q, a) = \wedge$ , alors il existe  $q' \in \delta(q, a)$  tel que  $w \notin \mathcal{L}(\mathcal{A}[q'])$ . Par hypothèse de récurrence,  $w \in \mathcal{L}(\mathcal{A}^c[q'])$  donc  $aw \in \mathcal{L}(\mathcal{A}^c[q])$  d'après la question 11.
- Ainsi en appliquant ce résultat pour  $q = q_0$  (état initial), on obtient que  $\mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{A}^c) = A^*$ .
- c) Finalement  $\mathcal{L}(\mathcal{A}^c) = (\mathcal{L}(\mathcal{A}))^c$  et l'automate  $\mathcal{A}^c$  est de même taille que  $\mathcal{A}$ .

**Question 14** [17 pts] On va réduire le problème d'acceptation des automates alternants au problème d'accessibilité des graphes ET/OU. On dispose d'un automate alternant  $\mathcal{A} = (Q, \{0, 1\}, q_0, Q_f, \delta, \tau)$  et d'un mot  $w = w_1 \dots w_n$  de taille  $n$ . On construit le graphe ET/OU  $\mathcal{G}(w, \mathcal{A}) = (V, E, f)$  avec  $V$  comme dans l'indication de l'énoncé. On définit  $E$  comme étant l'ensemble des transitions suivantes :

- $(q, i) \rightarrow (q', i + 1)$  pour tout  $q \in Q$  et  $0 \leq i \leq n - 1$  et  $q' \in \delta(q, w_{i+1})$  ;
- $(q, i) \rightarrow r$  pour tout  $q, i$  tels que  $\delta(q, w_{i+1}) = true$  ;
- $(q, n) \rightarrow r$  pour tout  $q \in Q_f$ .

Enfin  $f$  est définie de la manière suivante :  $f((q, i)) = \tau(q, w_{i+1})$  pour tout  $q$  et  $0 \leq i \leq n - 1$  et  $f((q, n)) = f(r) = \wedge$  pour tout  $q$ . Notons  $A_{\mathcal{G}}$  la relation d'accessibilité de  $\mathcal{G}(w, \mathcal{A})$ .

Montrons par récurrence le résultat  $H_0$  ( $0 \leq k \leq n$ ) : si  $W_k = w_{n-k+1} \dots w_n$ , alors pour tout  $q \in Q$ ,  $W_k \in \mathcal{L}(\mathcal{A}[q])$  si et seulement si  $r \in A_{\mathcal{G}}((q, n - k))$ . Pour  $k = 0$ ,  $W_0 = \epsilon \in \mathcal{L}(\mathcal{A}[q])$  si et seulement si  $q \in Q_f$  et  $(q, n) \rightarrow r$  dans  $E$  si et seulement si  $q' = r$  et  $q \in Q_f$  donc par compatibilité de  $A_{\mathcal{G}}$ ,  $r \in A_{\mathcal{G}}((q, n))$  si et seulement si  $q \in Q_f$  donc  $(H_0)$  est vraie.

Supposons  $(H_k)$  vraie pour un  $k$  dans  $\{0, \dots, n - 1\}$ . Notons  $i = n - k$ ,  $W_{k+1} = w_i \cdot W_k$ . Soit  $q$  quelconque dans  $Q$ . Supposons  $\tau(q, w_i) = \wedge$  et notons  $I = \delta(q, w_i)$ .

- Si  $I$  est non vide, d'après la question 11,  $W_{k+1} \in \mathcal{L}(\mathcal{A}[q])$  si et seulement si pour tout  $q' \in \delta(q, w_i) = I$ ,  $W_k \in \mathcal{L}(\mathcal{A}[q'])$ . Par hypothèse  $(H_k)$ ,  $r \in A_{\mathcal{G}}((q', i))$  pour tout  $q' \in I$ . Par

construction de  $\mathcal{G}(w, \mathcal{A})$ ,  $\text{succ}((q, i - 1)) = \delta(q, w_i) \times \{i\}$ . Par compatibilité de  $A_{\mathcal{G}}$ ,

$$A_{\mathcal{G}}((q, i - 1)) = \{(q, i - 1)\} \cup \bigcap_{q' \in \delta(q, w_i)} A_{\mathcal{G}}((q', i)) \ni r$$

donc  $(H_{k+1})$  est vraie.

- Si  $I$  est vide,  $W_{k+1} \in \mathcal{L}(\mathcal{A}[q])$  (définition d'un calcul réussi) et  $r \in A_{\mathcal{G}}((q, i - 1))$  (par construction du graphe) donc l'équivalence est vraie.

Si  $\tau(q, w_i) = \vee$ , la démonstration est symétrique. Ainsi on montre que  $(H_n)$  est vraie par récurrence, d'où le résultat attendu en spécialisant  $q = q_0$ .

Finalement,  $\mathcal{G}(w, \mathcal{A})$  possède  $(n + 1)|Q| + 1$  sommets et au plus  $n(|\delta| + 1) + |Q|$  arêtes donc de taille  $O(|w| \times |\mathcal{A}|)$ . Sa construction est linéaire en sa taille et le test d'accessibilité est linéaire en sa taille d'après la question 9. Étant donnés  $w$  et  $\mathcal{A}$ , le test  $w \in \mathcal{L}(\mathcal{A})$  peut être décidé par un algorithme de complexité  $O(|w| \times |\mathcal{A}|)$ .

**Question 15** [14 pts] On dispose d'un automate alternant  $\mathcal{A}_1 = (Q, \{0, 1\}, q_0, Q_f, \delta, \tau)$ . On construit l'automate fini non déterministe  $\mathcal{A}_2 = (2^Q, \{0, 1\}, \{q_0\}, 2^{Q_f}, \delta_2)$  qui simule les calculs de  $\mathcal{A}_1$  par un parcours "parallèle" de la façon suivante :  $(S, a, S') \in \delta_2$  si et seulement si

- pour tout  $q \in S_{\wedge}(a) = \{q \in S \mid \tau(q, a) = \wedge\}$ ,  $\delta(q, a) \subseteq S'$ ,
- pour tout  $q \in S_{\vee}(a) = \{q \in S \mid \tau(q, a) = \vee\}$ ,  $|\delta(q, a) \cap S'| = 1$ .

Montrons par récurrence sur  $|w|$  que pour tout état  $S \subseteq Q$  de  $\mathcal{A}_2$ ,  $w \in \mathcal{L}(\mathcal{A}_2[S])$  si et seulement si  $w \in \bigcap_{q \in S} \mathcal{L}(\mathcal{A}_1[q])$ . Initialisation :  $w = \epsilon \in \mathcal{L}(\mathcal{A}_2[S])$  si et seulement si  $S \subseteq Q_f$ . Supposons l'hypothèse de récurrence vraie pour  $w$  de taille  $n$ . Soit  $S \subseteq Q$  quelconque,  $a \in \{0, 1\}$ , par construction de  $\mathcal{A}_2$ ,  $aw \in \mathcal{L}(\mathcal{A}_2[S])$  si et seulement s'il existe  $S' \in \delta_2(S, a)$  tel que  $w \in \mathcal{L}(\mathcal{A}_2[S'])$ . Par hypothèse de récurrence, ceci équivaut à (\*) : il existe  $j_q \in \delta(q, a)$  pour tout  $q \in S_{\vee}(a)$  tels que

$$w \in \bigcap_{q' \in S'} \mathcal{L}(\mathcal{A}_1[q']) = \bigcap_{q \in S_{\wedge}(a)} \left( \bigcap_{q' \in \delta(q, a)} \mathcal{L}(\mathcal{A}_1[q']) \right) \cap \bigcap_{q \in S_{\vee}(a)} \mathcal{L}(\mathcal{A}_1[j_q]).$$

D'après la question 11 (et une récurrence sur  $|S|$ ), (\*) est équivalent à  $aw \in \bigcap_{q \in S} \mathcal{L}(\mathcal{A}_1[q])$ .

Finalement, en spécialisant  $S = \{q_0\}$ , on obtient que  $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$ . La construction de  $\mathcal{A}_2$  à partir de  $\mathcal{A}_1$  est exponentielle.

**Question 16** [3 pts] Raisonnons par l'absurde et supposons que ce langage soit reconnu par un automate ayant au plus  $2^n$  états. Alors  $1^{2^n}$ , de longueur  $2^n$ , peut s'écrire  $w_1 \cdot w_2 \cdot w_3$  avec  $w_2 \neq \epsilon$  et  $w_1 \cdot w_2^k \cdot w_3$  dans le langage d'après le lemme de l'étoile. Or c'est absurde puisque le langage ne contient qu'un mot.

\* \*  
\*