

COMPOSITION D'INFORMATIQUE¹

(Durée : 4 heures)

L'utilisation des calculatrices **n'est pas autorisée** pour cette épreuve.

* * *

Graphes ET/OU et automates alternants

*On attachera une grande importance à la concision, à la clarté et à la précision de la rédaction.**La troisième partie du problème est quasiment indépendante des deux autres.*

Partie I. Préliminaires

Si S est un ensemble, on note 2^S l'ensemble des sous-ensembles de S . On choisit ici de représenter un ensemble S d'éléments de V par une liste de taille $|S|$:

```
type ensemble == int list ;;
```

Question 1 Écrire une fonction `appartient` qui teste si un entier n apparaît dans un ensemble S . Quelle est sa complexité ?

```
appartient : ensemble -> int -> bool
```

Question 2 Écrire une fonction `ens_egaux` qui teste si deux ensembles S_1 et S_2 sont égaux. Quelle est sa complexité ?

```
ensEgaux : ensemble -> ensemble -> bool
```

Question 3 Écrire une fonction `union` qui retourne l'union de deux ensembles S_1 et S_2 . Quelle est sa complexité ?

```
union : ensemble -> ensemble -> ensemble
```

Question 4 Notons $V = \{0, \dots, n-1\}$. Écrire une fonction `gunion` qui, étant donné un ensemble $S \subseteq V$ et un tableau g indicé par V à valeurs dans 2^V , calcule l'union des $g(s)$ pour $s \in S$. Exprimer sa complexité en fonction de $|S|$ et $|g|$ ($|g|$ étant la somme des tailles des listes que contient g).

```
gunion : ensemble vect -> ensemble -> ensemble
```

Partie II. Graphes ET/OU

Un graphe ET/OU \mathcal{G} est un triplet (V, E, f) où V est un ensemble fini (les sommets), E est un sous-ensemble de $V \times V$ (les arêtes) et f est une application de V dans $\{\wedge, \vee\}$. La taille $|\mathcal{G}|$ d'un graphe ET/OU est la somme du nombre de sommets $|V|$ et du nombre d'arêtes $|E|$. On pourra supposer que les sommets sont numérotés et donc que $V = \{0, \dots, N-1\}$. V sera représenté par l'entier N , E par un tableau t d'ensembles de sommets défini par $s' \in t(s)$ si et seulement si $(s, s') \in E$ et enfin f sera représenté par un tableau de booléens où \vee (resp. \wedge) est représenté par *true* (resp. *false*) :

¹Largement inspirée de l'épreuve écrite des ENS 2004 (BATOG Guillaume)

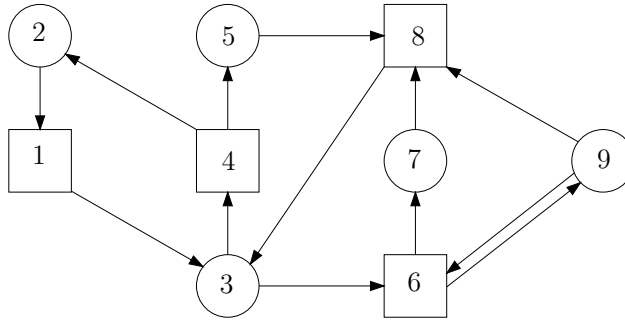


FIG. 1 – Un graphe ET/OU

```
type grapheETOU = { v : int ; e : ensemble vect ; f : bool vect } ;;
```

Si g est une application de V dans 2^V , on dira que g est compatible avec \mathcal{G} si, pour tout $s \in V$

– ou bien $f(s) = \vee$ et

$$g(s) = \{s\} \cup \bigcup_{(s,s') \in E} g(s')$$

– ou bien $f(s) = \wedge$ et

$$g(s) = \{s\} \cup \bigcap_{(s,s') \in E} g(s').$$

Question 5 Écrire une fonction `est_compatible` qui, étant donné un tableau g indicé par V à valeurs dans les ensembles de sommets et un graphe ET/OU \mathcal{G} , teste si g est compatible avec \mathcal{G} . Exprimer sa complexité en fonction de $|\mathcal{G}|$ et $|g|$.

```
est_compatible : grapheETOU -> ensemble vect -> bool
```

On définit la suite d'applications $A_n(s)$ qui associent à chaque sommet un ensemble de sommets :

- $A_0(s) = \{s\}$ pour tout s
- si $f(s) = \vee$, $A_{n+1}(s) = A_n(s) \cup \bigcup_{(s,s') \in E} A_n(s')$
- si $f(s) = \wedge$, $A_{n+1}(s) = A_n(s) \cup \bigcap_{(s,s') \in E} A_n(s')$

Question 6

- a) Calculer la suite $A_n(s)$ ($n \leq 6$) pour chacun des 9 sommets du graphe \mathcal{G} donné dans la figure 1, dans lequel $f(s) = \vee$ pour les sommets représentés par des cercles et $f(s) = \wedge$ pour les sommets représentés par des carrés.
- b) Montrer que, pour tout graphe ET/OU \mathcal{G} , il existe un entier M que l'on précisera tel que, pour tout $n \geq M$, pour tout s , $A_{n+1}(s) = A_n(s)$. On notera alors $A_{\mathcal{G}}(s)$ la limite obtenue.
- c) Montrer que $A_{\mathcal{G}}$ est compatible avec \mathcal{G} et que pour tout g compatible avec \mathcal{G} et pour tout sommet s de \mathcal{G} , on a $A_{\mathcal{G}}(s) \subseteq g(s)$.
- d) En déduire une fonction `rel_acc` qui construit cette application $A_{\mathcal{G}}$, appelée relation d'accessibilité de \mathcal{G} . Exprimer sa complexité en fonction de $|V|$.

```
rel_acc : grapheETOU -> ensemble vect
```

Question 7 Si S est un ensemble de sommets de \mathcal{G} , on note $A_{\mathcal{G}}^{-1}(S) = \{t \in V \mid S \cap A_{\mathcal{G}}(t) \neq \emptyset\}$ (ensemble des sommets desquels on peut accéder à un sommet de S). Écrire une fonction `rel_acc_inv` qui, étant donné \mathcal{G} et S , calcule $A_{\mathcal{G}}^{-1}(S)$.

```
rel_acc_inv : grapheETOU -> ensemble -> ensemble
```

```

T := liste vide ; S := liste réduite à s0 ;
prec := pred(G) ; n := nb_succ(G) ;
n(s0) := 0 ;
TANT QUE non_vide(S) faire
    s := tête(S) ; S := queue(S) ;
    L := prec(s) ; T := cons(s,T) ;
    TANT QUE non_vide(L) faire
        s1 := tête(L)
        L := queue(L)
        n(s1) := n(s1) - 1
        SI n(s1) = 0 alors S := cons(s1,S)
    Fin tant que
Fin tant que

```

FIG. 2 – Un algorithme sur les graphes ET/OU

Question 8

- a) Construire une fonction `pred` qui, étant donné un graphe \mathcal{G} , retourne en temps $O(|\mathcal{G}|)$ un tableau dont l'élément d'indice $s \in V$ est la liste des prédécesseurs de s dans \mathcal{G} .

`pred : grapheETOU -> ensemble vect`

- b) Construire une fonction `nb_succ` qui, étant donné un graphe \mathcal{G} , retourne en temps $O(|\mathcal{G}|)$ un tableau dont l'élément d'indice $s \in V$ est un entier égal au nombre de successeurs de s si $f(s) = \wedge$ et égal à 1 sinon.

`nb_succ : grapheETOU -> int vect`

Question 9 On va maintenant étudier l'algorithme de la figure 2 écrit en pseudo-code. `cons(s,T)` retourne la liste obtenue en plaçant l'élément s en tête de la liste T .

- a) Montrer que la complexité de cet algorithme est $O(|\mathcal{G}|)$.
- b) Montrer qu'après exécution de l'algorithme, $T = A_{\mathcal{G}}^{-1}(\{s_0\})$. *Indication : on pourra montrer que, si $f(s) = \wedge$, $n(s) = |G(s)| - |G(s) \cap T|$ à chaque entrée dans la boucle externe.*
- c) Décrire alors un algorithme linéaire en $|\mathcal{G}|$ qui résout le problème d'accessibilité : étant donnés \mathcal{G} et $s, t \in V$, l'algorithme retourne vrai si $t \in A_{\mathcal{G}}(s)$ et faux sinon.

Partie III. Automates alternants

Un automate alternant est donné par un ensemble fini d'états Q , un état initial $q_0 \in Q$, un ensemble d'états finaux $Q_f \subseteq Q$, un alphabet d'entrée A , une fonction de transition $\delta : Q \times A \rightarrow 2^Q$ et une fonction de type $\tau : Q \times A \rightarrow \{\wedge, \vee\}$. On écrira en abrégé $\delta(q, a) = q_1 \vee \dots \vee q_m$ (resp. $\delta(q, a) = q_1 \wedge \dots \wedge q_m$) si $\tau(q, a) = \vee$ (resp. $\tau(q, a) = \wedge$). On écrira de plus $\delta(q, a) = false$ (resp. $\delta(q, a) = true$) lorsque $\delta(q, a) = \emptyset$ et $\tau(q, a) = \vee$ (resp. $\tau(q, a) = \wedge$). Dans toute la suite, on supposera que $A \subseteq \{0, 1\}$.

ϵ désignera le mot vide, $w_1 \cdot w_2$ la concaténation des mots w_1 et w_2 , $w(i)$ la i^e lettre du mot w (si elle est définie) et $|w|$ la longueur de w .

Un calcul de l'automate $\mathcal{A} = (Q, q_0, A, \delta, \tau)$ sur le mot w est un arbre T étiqueté par Q et tel que :

- La racine de l'arbre est étiquetée par q_0 . (C'est le nœud de l'arbre de profondeur 0).
- Si un nœud n de l'arbre, de profondeur k , est étiqueté par q et si $\tau(q) = \vee$, $|w| \geq k + 1$, $w(k + 1) = a$ et $\delta(q, a) = \{q_1, \dots, q_m\}$, alors n a exactement un fils (de profondeur $k + 1$) étiqueté par l'un des états q_1, \dots, q_m . Noter que l'on doit avoir $m > 0$ et donc qu'un calcul ne peut pas utiliser de transition $\delta(q, a) = false$.

- Si un nœud n de l'arbre, de profondeur k , est étiqueté par q et si $\tau(q) = \wedge$, $|w| \geq k + 1$, $w(k + 1) = a$ et $\delta(q, a) = \{q_1, \dots, q_m\}$, alors n a exactement m fils (de profondeur $k + 1$) étiquetés respectivement par q_1, \dots, q_m . Noter que, si $m = 0$ (c'est-à-dire $\delta(q, a) = true$), n est une feuille.
- Tous les nœuds de l'arbre sont de profondeur inférieure ou égale à $|w|$.

Un calcul T de \mathcal{A} sur w est réussi si toute feuille n de T de profondeur $|w|$ est étiquetée par un état final. (Noter que si le calcul ne comprend aucune feuille de profondeur $|w|$, il est toujours réussi). Le langage accepté par \mathcal{A} , noté $\mathcal{L}(\mathcal{A})$, est l'ensemble des mots w de A^* tels qu'il existe un calcul réussi de \mathcal{A} sur w .

Question 10 On considère l'automate alternant dont la fonction de transition est donnée par :

δ	0	1
q_0	$q_0 \wedge q_1$	$q_2 \vee q_3$
q_1	$true$	$q_0 \vee q_1$
q_2	$false$	$true$
q_3	q_2	$false$

τ	0	1
q_0	\wedge	\vee
q_1	\wedge	\vee
q_2	\vee	\wedge
q_3	\wedge	\vee

L'état initial est q_0 . Il n'y a pas d'état final. Donner des calculs réussis de l'automate sur les mots 0101, 0110, 0111.

Question 11 Soit \mathcal{A} un automate alternant, pour tout $q \in Q$, on note $\mathcal{A}[q]$ l'automate alternant \mathcal{A} dont q est l'unique état initial. Pour tout $q \in Q$, exprimer $\mathcal{L}(\mathcal{A}[q])$ en fonction de l'ensemble des $\mathcal{L}(\mathcal{A}[q'])$ pour $q' \in Q$.

La taille d'un automate non-déterministe classique est égale à la somme de son nombre d'états et de son nombre de transitions. La taille d'un automate alternant est la somme, pour tout les états q et pour toutes les lettres de l'alphabet a , du cardinal de $\delta(q, a)$ et du nombre d'états :

$$|\mathcal{A}| = |Q| + \sum_{q \in Q} \sum_{a \in A} |\delta(q, a)|.$$

Question 12 Montrer que tout langage reconnu par un automate fini non-déterministe \mathcal{A}_1 est aussi reconnu par un automate alternant \mathcal{A}_2 de taille $O(|\mathcal{A}_1|)$.

Question 13 Étant donné un automate alternant \mathcal{A} , on considère l'automate dual noté \mathcal{A}^c obtenu en échangeant \vee et \wedge d'une part et les états finaux et non finaux d'autre part.

- Montrer que $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}^c) = \emptyset$.
- Montrer que tout mot de A^* est accepté par \mathcal{A} ou \mathcal{A}^c .
- En déduire qu'on peut calculer en $O(|\mathcal{A}|)$ un automate qui accepte le complémentaire de $\mathcal{L}(\mathcal{A})$.

Question 14 Donner un algorithme de complexité $O(|w| \times |\mathcal{A}|)$ qui, étant donnés un mot w et un automate alternant \mathcal{A} , détermine si w est accepté par \mathcal{A} . *Indication : on construira un graphe ET/OU $\mathcal{G}(w, \mathcal{A})$ à partir de w et \mathcal{A} de sommets $V = \{(q, i) \mid q \in Q, 1 \leq i \leq |w|\} \cup \{(q_0, 0)\} \cup \{r\}$ tel que r est accessible à partir de $(q_0, 0)$ dans $\mathcal{G}(w, \mathcal{A})$ si et seulement si w est accepté par \mathcal{A} .*

Question 15 Montrer que tout langage reconnu par un automate alternant \mathcal{A}_1 est aussi reconnu par un automate fini non-déterministe \mathcal{A}_2 . Quel est la complexité d'un algorithme calculant \mathcal{A}_2 à partir de \mathcal{A}_1 ?

Question 16 On considère le langage L_n qui contient le mot unique 1^{2^n} . Montrer que tout automate non-déterministe acceptant L_n comporte au moins $2^n + 1$ états.

* *
*