Le problème comporte 13 questions. Il est conseillé de traîter les questions dans l'ordre de l'énoncé, cependant on pourra aborder une question en admettant les résultats des précédentes. Les algorithmes demandés pourront être écrits dans un langage aux choix du candidat, en utilisant les structures de contrôle usuelles. On pourra par exemple utiliser un langage semblable à celui qui est décrit dans l'annexe A.

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 $|E|$. La *taille* d'une liste est sa longueur. On pourra supposer que les sommets sont numérotés et donc que $V = \{1, \ldots, N\}$. Les éléments de $V$ sont de taille 1.

Si $S$ est un ensemble, on note $2^S$ l'ensemble des sous-ensembles de $S$. On pourra choisir de représenter un ensemble $S$ d'éléments de $V$ par une liste de taille $|S|$ ou par un tableau de booléens de taille $|V|$.

On utilisera la structure de donnée des *tableaux indicés par $V$ à valeurs dans un type de donnée $Y$* (tableaux de $Y$ en abrégé) qui représentent les applications de $V$ dans $Y$. On dispose pour de telles données

- d'une fonction d'accès : $t(x)$ est la valeur de $t$ en $x$ si $t$ est un tableau de $Y$ et $x \in V$. Un tel accès s'effectue en temps unitaire
- d'une fonction d'initialisation init qui, étant donné $y \in Y$, renvoie un tableau représentant la fonction constante égale à $y$. Le coût de cette fonction est $|V|$.
- de primitives de modification : $t(x) := y$ modifie le tableau $t$ en remplaçant sa valeur en $x$ par $y$. Cette opération a un coût unitaire. (Dans un langage fonctionnel, on pourra supposer l'existence d'une fonction de coût unitaire qui renvoie le tableau modifié).

Le temps nécessaire à l'exécution d'un algorithme sur la donnée $d$ sera supposé être le nombre d'opérations élémentaires effectuées : accès à un tableau, test d'égalité sur les données atomiques (éléments de $V$), test de vide d'une liste, accès au premier élément d'une liste (car), accès au reste d'une liste (cdr), ajout d'un élément à une liste (cons, : :), ajouter ou retrancher 1 à un entier, test à 0 d'un entier, or , and sur des données Booléennes. On utilisera la notation $O$ : par exemple, la complexité d'un algorithme est $O(1)$ si son temps d'exécution ne dépend pas de la donnée ; il est $O(|d|)$ si son temps d'exécution est, dans le cas le pire, linéaire dans la taille $|d|$ de la donnée $d$.

## 1 Accessibilité dans les graphes ET/OU

Dans cette partie, $\mathcal{G} = (V, E, f)$ désignera un graphe ET/OU. 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 1
Montrer que l'application qui à tout sommet de $\mathcal{G}$ associe $V$ est compatible avec $\mathcal{G}$.

### Question 2
Montrer que, si $g_1$ et $g_2$ sont deux applications compatibles avec $\mathcal{G}$, l'application $g_1 \cap g_2$ définie par :
$$g_1 \cap g_2(s) \quad = \quad g_1(s) \cap g_2(s)$$
est compatible avec $\mathcal{G}$

### Question 3
Montrer qu'il existe une unique application $A_{\mathcal{G}}$ compatible avec $\mathcal{G}$ telle que pour toute application $g$ compatible avec $\mathcal{G}$ et pour tout sommet $s$ de $\mathcal{G}$, $A_{\mathcal{G}}(s) \subseteq g(s)$

L'application $A_{\mathcal{G}}$ est appelée *relation d'accessibilité* de $\mathcal{G}$. L'objet de cette partie est de construire des algorithmes qui permettent de calculer $A_{\mathcal{G}}$

On représente les graphes ET/OU à l'aide de deux tableaux $\mathsf{G}$ et $\mathsf{f}$ indicés par les sommets du graphe : $\mathsf{G}(s)$ est l'ensemble des sommets $s'$ de $\mathcal{G}$ tels que $(s, s') \in E$ et $\mathsf{f}(s)$ est égal à $f(s)$.

Les ensembles de sommets seront représentés par des listes sans répétition ou des tableaux de Booléens, au choix du candidat.

### Question 4

Donner des algorithmes (ou programmes) qui réalisent les fonctions suivantes :

1. le test d'appartenance à un ensemble de sommets
2. le test d'égalité de deux ensembles de sommets
3. l'union de deux ensembles de sommets
4. étant donné un ensemble de sommets $S$ et un tableau $g$ indicé par $V$ à valeurs dans les ensembles de sommets, calcule l'union des $g(s)$ pour $s \in S$.
5. étant donné un tableau $g$ indicé par $V$, à valeurs dans les ensembles de sommets, et un graphe ET/OU $\mathcal{G}$ (donné par $\mathsf{G}$ et $\mathsf{f}$), le test de compatibilité de $g$ avec $\mathcal{G}$.

Dans chaque cas, justifier brièvement la correction de l'algorithme et donner sa complexité.

### Question 5

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 $f(s) = \wedge$, $A_{n+1}(s) = A_n(s) \cup \bigcap_{(s,s') \in E} A_n(s')$

1. 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.
2. 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 $A(s)$ la limite ainsi obtenue.
3. Montrer que $A = A_{\mathcal{G}}$
4. En déduire un algorithme de calcul de $A_{\mathcal{G}}$ dont on précisera la complexité.

### Question 6

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 $S$). Donner un algorithme qui, étant donnés $\mathcal{G}$ et $S$, calcule $A_{\mathcal{G}}^{-1}(S)$ et préciser sa complexité.

### Question 7

On associe à chaque sommet $s$ la liste de ses prédecesseurs $\mathsf{prec}(s)$ et le nombre $n(s)$ qui est égal au nombre de successeurs de $s$ si $f(s) = \wedge$ et égal à 1 sinon.

Soient $S, T$ des listes de sommets. Initialement $T$ est vide et $S = \{s_0\}$. On considère l'algorithme donné dans la figure 2.

1. Montrer que la complexité de cet algorithme est $O(|\mathcal{G}|)$
2. Montrer que, après exécution, $T = A_{\mathcal{G}}^{-1}(\{s_0\})$. (Ind : on pourra montrer que, si $f(s) = \wedge$, $n(s) = |\mathsf{G}(s)| - |\mathsf{G}(s) \cap T|$ à chaque entrée dans la boucle externe).
3. Donner un algorithme qui calcule toutes les listes sans répétitions $\mathsf{prec}[s]$ et tous les entiers $n(s)$, pour $s \in V$, en temps (total) $O(|\mathcal{G}|)$.
4. En déduire qu'il existe un algorithme linéaire qui résoud le problème d'accessibilité : étant donnés $\mathcal{G}$ et $s, t \in V$, l'algorithme répond **true** si $t \in A_{\mathcal{G}}(s)$ et **false** sinon.

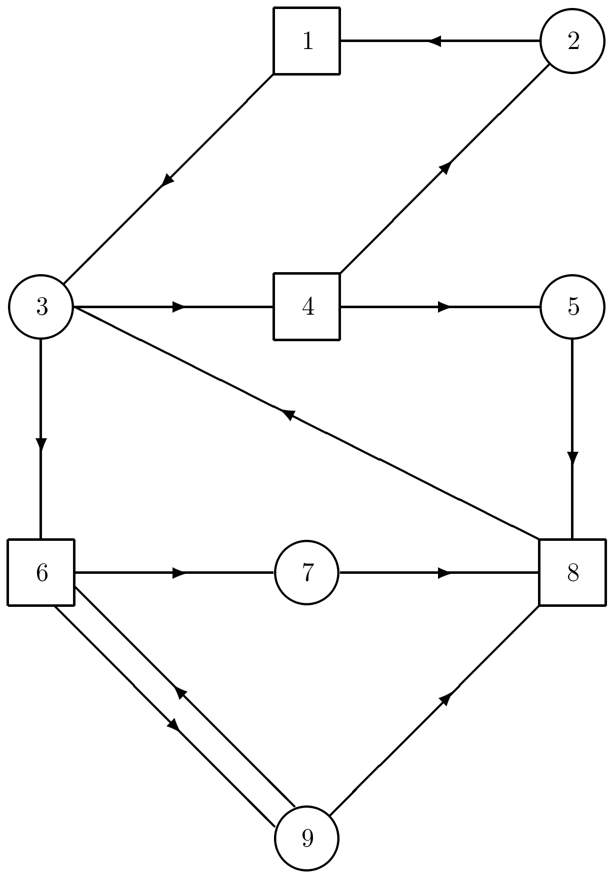


Fig. 1 – Un graphe ET/OU

Pour $s \in S$ faire $n(s) := 0$
Tant que non vide$(S)$ faire
    $s :=$ car$(S)$ ; $S :=$ cdr$(S)$
    $L :=$ prec$(s)$ ; $T :=$ cons$(s, T)$
    Tant que non vide$(L)$ faire
        $s_1 :=$ car$(L)$
        $L :=$ cdr$(L)$
        $n(s_1) := n(s_1) - 1$
        Si $n(s_1) = 0$ alors $S :=$ cons$(s_1, S)$
    Fin Tq
Fin Tq

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

## 2 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 $\delta : Q \times A \to 2^Q$ et une fonction de type $\tau : Q \times A \to \{\wedge, \vee\}$. On écrira en abrégé $\delta(q, a) = q_1 \vee \ldots \vee q_m$ (resp. $\delta(q, a) = q_1 \wedge \ldots \wedge q_m$) si $\tau(q, a) = \vee$ (resp. $\tau(q, a) = \wedge$). On écrira de plus $\delta(q, a) = \mathbf{false}$ (resp. $\delta(q, a) = \mathbf{true}$ ) lorsque $\delta(q, a) = \emptyset$ et $\tau(q, a) = \vee$ (resp. $\tau(q, a) = \wedge$). Dans toute la suite, on supposera que $A \subseteq \{0, 1\}$.

$\epsilon$ désignera le mot vide , $w_1 \cdot w_2$ la concaténation des mots $w_1$ et $w_2$, $w(i)$ la ième 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 étiqueté par $Q$ et tel que :

- La racine de l'arbre est étiquetée par $q_0$. (C'est le noeud de l'arbre de profondeur 0).
- Si un noeud $n$ de l'arbre, de profondeur $k$, est étiqueté par $q$ et si $\tau(q) = \vee$, $|w| \geq k + 1$, $w(k + 1) = a$ et $\delta(q, a) = \{q_1, \ldots, q_m\}$, alors $n$ a exactement un fils (de profondeur $k + 1$) étiqueté par l'un des états $q_1, \ldots, q_m$. Noter que l'on doit avoir $m > 0$ et donc qu'un calcul ne peut pas utiliser de transition $\delta(q, a) = \mathbf{false}$ .
- Si un noeud $n$ de l'arbre, de profondeur $k$, est étiqueté par $q$ et si $\tau(q) = \wedge$, $|w| \geq k + 1$, $w(k + 1) = a$ et $\delta(q, a) = \{q_1, \ldots, q_m\}$, alors $n$ a exactement $m$ fils (de profondeur $k + 1$), étiquetés respectivement par $q_1, \ldots, q_m$. Noter que, si $m = 0$ (c'est à dire $\delta(q, a) = \mathbf{true}$ ), $n$ est une feuille.
- Tous les noeuds de $T$ sont de profondeur inférieure ou égale à $|w|$.

Un calcul de $\mathcal{A}$ sur $T$ 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 $\mathcal{A}$ est l'ensemble des mots $w$ de $A^*$ tels qu'il existe un calcul réussi de $\mathcal{A}$ sur $w$.

### Question 8

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

| $\delta$ | 0 | 1 |
|---|---|---|
| $q_0$ | $q_1 \wedge q_0$ | $q_2 \vee q_3$ |
| $q_1$ | **true** | $q_0 \vee q_1$ |
| $q_2$ | **false** | **true** |
| $q_3$ | $q_2$ | **false** |

| $\tau$ | 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.

La *taille* d'un automate non-déterministe 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 tous 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 9

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 10**
Démontrer que, étant donné un automate alternant $\mathcal{A}$, on peut calculer en temps $O(|\mathcal{A}|)$ un automate alternant $\mathcal{A}^c$ qui accepte le complémentaire du langage accepté par $\mathcal{A}$.(Ind : on pourra considérer l'automate dual obtenu en échangeant $\vee$ et $\wedge$ d'une part et les états finaux et non finaux d'autre part)

**Question 11**
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 $\mathcal{A}$.

**Question 12**
Montrer que tout langage reconnu par un automate alternant $\mathcal{A}_1$ est aussi reconnu par un automate non-déterministe $\mathcal{A}_2$.

Quelle est la complexité d'un algorithme calculant $\mathcal{A}_2$ à partir de $\mathcal{A}_1$ ?

**Question 13**

1. 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.
2. Donner un automate alternant qui accepte $L_n$, et de taille $O(n^2)$. (Ind : on pourra considérer les états $q_i$ à partir desquels sont acceptés les mots de la forme $1^{2k+1}$ où le ième bit dans l'écriture en base 2 de $k$ est 0.).

## A Exemple d'un petit langage algorithmique

**instructions élémentaires** : l'affectation $e := e'$, l'affichage print $e$, les appels de procédures $p(e_1, \ldots, e_n)$, l'identité skip .

**structures de contrôle** :

$i$ ; $i'$ : composition séquentielle des instructions

begin $i$ end : parenthèsage (on peut aussi utiliser une indentation)

if $e$ then $i$ else $i'$ : conditionnelle

if $e$ then $i$ : abréviation de if $e$ then $i$ else skip.

for $x$= $e$ to $e'$ do $i$ : itération (dont les bornes sont calculées avant entrée dans la boucle ; $x, e, e'$ sont de type entier)

Tant que $e$ faire $i$ Fin Tq : boucle, le test d'arrêt $e$ étant évalué à chaque itération.