Devoir surveillé n° 04 — 2h

LA CALCULATRICE N'EST PAS AUTORISÉE.

Les questions de programmation doivent être traitées en langage CAML ou C selon ce qui est demandé par l'énoncé. En CAML, et sauf mention du contraire, on autorisera toutes les fonctions des modules `Array` et `List`, ainsi que les fonctions de la bibliothèque standard (celles qui s'écrivent sans nom de module, comme `max`, `incr` ainsi que les opérateurs comme `@`). Sauf mention contraire, l'utilisation d'autres modules est interdite. En C, on supposera que les bibliothèques `stdlib.h` et `stdbool.h` ont été chargées.

Lorsque le candidat écrira une fonction, il pourra faire appel à des fonctions définies dans les questions précédentes, même si elles n'ont pas été traitées. Il pourra également définir des fonctions auxiliaires, mais devra préciser leurs rôles ainsi que les types et significations de leurs arguments. Les candidats sont encouragés à expliquer les choix d'implémentation de leurs fonctions lorsque ceux-ci ne découlent pas directement des spécifications de l'énoncé. Si les paramètres d'une fonction à coder sont supposés vérifier certaines hypothèses, il ne sera pas utile dans l'écriture de cette fonction de tester si les hypothèses sont bien satisfaites.

On identifiera une même grandeur écrite dans deux polices de caractères différentes, en italique du point de vue mathématique (par exemple n) et à chasse fixe du point de vue informatique (par exemple `n`).

Sans précision supplémentaire, lorsqu'une question demande la complexité d'une fonction, il s'agira de la complexité temporelle asymptotique dans le pire des cas. La complexité sera exprimée sous la forme $\mathcal{O}(f(n,m))$ où n et m sont des tailles des paramètres d'entrée, et f une expression la plus simple possible. Les calculs de complexité seront justifiés succinctement.

I Machines et mots synchronisants

On appelle **machine** un triplet (Q, Σ, δ) où Q est un ensemble fini non vide d'**états**, Σ un alphabet et δ une application de $Q \times \Sigma$ dans Q appelée **fonction de transition**.

Une machine peut donc être vue comme un automate fini déterministe complet sans notion d'état initial ou final.

Comme pour les automates finis, on utilisera la notion de fonction de transition étendue définie par :

- pour tout $q \in Q$, $\delta^*(q, \varepsilon) = q$;
- pour tout $q \in Q$, $u \in \Sigma^*$ et $a \in \Sigma$, $\delta^*(q, ua) = \delta(\delta^*(q, u), a)$.

Un mot $u \in \Sigma^*$ est dit **synchronisant** pour une machine (Q, Σ, δ) s'il existe $q_s \in Q$ tel que $\forall q \in Q, \delta^*(q, u) = q_s$. L'existence de tels mots permet de ramener une machine dans un état particulier connu en lisant un mot donné, donc en pratique de réinitialiser une machine réelle. La figure 1 représente une machine M_0. On pourra remarquer que ba et bb sont des mots synchronisants pour M_0, qui à partir de n'importe quel état mènent respectivement dans l'état q_0 et dans l'état q_2.

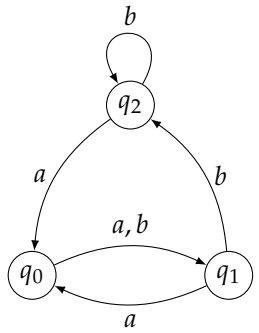

FIGURE 1 – La machine M_0 sur l'alphabet $\{a, b\}$.

Q 1) Que dire de l'ensemble des mots synchronisant pour une machine ayant un seul état ?

Dans toute la suite du problème, on supposera que les machines ont au moins deux états.

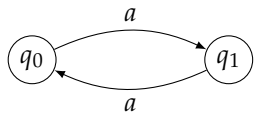

FIGURE 2 – La machine M_1 sur l'alphabet $\{a\}$.

Q 2) La machine M_1 représentée sur la figure 2 admet-elle un un mot synchronisant ? Justifier la réponse.

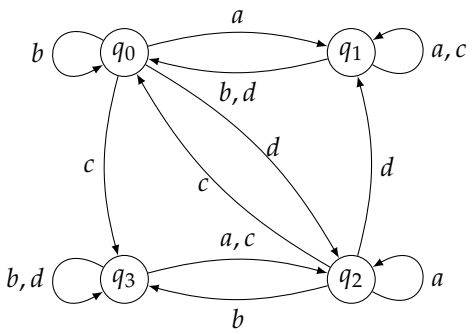

FIGURE 3 – La machine M_2 sur l'alphabet $\{a, b, c, d\}$.

Q 3) Donner un mot synchronisant de trois lettres pour la machine M_2 représentée à la figure 3 ainsi que l'état obtenu à partir de la lecture de ce mot. On ne demande pas de justifier la réponse.

II Implémentation (en C)

On suppose que $Q = \{0, 1, \ldots, |Q| - 1\}$ et que Σ correspond aux 256 caractères représentables par le type `char`. Un mot de Σ^* sera alors représenté par une chaîne de caractères de type `char*`. On rappelle que le caractère de fin de chaîne est `'\0'`.

On définit la constante :

C
```c
const int nb_lettres = 256;
```

Une machine sera représentée par un objet de type `machine` défini par :

C
```c
struct machine {
    int nb_etats;
    int** delta;
};

typedef struct machine machine;
```

On représente une machine $M = (Q, \Sigma, \delta)$ par un objet `m` de type `machine*`.

- le nombre d'états $|Q|$ est représenté par l'entier `m->nb_etats;`
- l'ensemble des transitions est donné par une matrice de taille $|Q| \times |\Sigma|$. À la ligne correspondant à l'état $q \in Q$ et à la colonne correspondant à la lettre $a \in \Sigma$ on trouve l'état d'arrivé $\delta(q, a)$. En C, on représente cette matrice par un tableau de tableaux `delta` de type `int**`. Si `int q` représente un état $q \in Q$ et si `char a` représente une lettre $a \in \Sigma$ de code ASCII `int c = (int)a;` alors `delta[q][c]` vaut $\delta(q, a)$.

On rappelle que l'on peut obtenir l'entier correspondant au code ASCII d'un caractère[1] `char a` par un transtypage `(int)a`, ce qui nous permet d'indexer les colonnes de la matrice de transition par des entiers.

Q 4) Écrire une fonction `machine* init_machine(int nb_etats)` qui crée une machine à $|Q|$ états telle que pour tout $q \in Q$ et $a \in \Sigma$, $\delta(q, a) = q$.

Q 5) Écrire une fonction `void liberer_machine(machine* m)` qui libère l'espace mémoire occupé par une machine.

Q 6) Écrire une fonction `int delta_etoile(machine* m, int q, char* u)` qui prend en arguments une machine $M = (Q, \Sigma, \delta)$, un état $q \in Q$ et un mot $u \in \Sigma^*$ et qui renvoie l'état $\delta^*(q, u)$.

Q 7) Écrire une fonction `bool est_synchronisant(machine* m, char* u)` qui prend en argument une machine M et un mot u et renvoie le booléen `true` si u est synchronisant pour M et `false` sinon.

III Considérations générales

Q 8) Montrer que si une machine admet un mot synchronisant alors il existe $a \in \Sigma$ et $q \neq q' \in Q$ tels que $\delta(q, a) = \delta(q', a)$.

Soit $LS(M)$ le langage des mots synchronisants d'une machine $M = (Q, \Sigma, \delta)$. On introduit la machine des parties $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\delta})$ où $\widehat{Q} = \mathcal{P}(Q)$ et $\widehat{\delta}$ est définie par :

$$\forall P \subseteq Q, \forall a \in \Sigma, \widehat{\delta}(P, a) = \{\delta(p, a), p \in P\}$$

Q 9) Montrer que l'existence d'un mot synchronisant pour M se ramène à un problème d'accessibilité de certain(s) état(s) depuis certain(s) état(s) de $\widehat{M}$.

Q 10) En déduire que le langage $LS(M)$ des mots synchronisants de M est reconnaissable.

1. En réalité, en C, un caractère *est* un entier et le transtypage n'est pas nécessaire.

Q 11) Déterminer puis représenter graphiquement un automate fini déterministe (pas nécessairement complet) reconnaissant $LS(M_0)$.

Q 12) Montrer que si l'on sait résoudre le problème de l'existence d'un mot synchronisant pour une machine quelconque, on sait alors également résoudre le problème consistant à dire, pour une machine M et un état q_0 de M, s'il existe un mot u tel que pour tout état q de Q le chemin menant de q à $\delta^*(q,u)$ passe forcément par q_0.

IV Graphe d'automate (en CAML)

On appelle **graphe d'automate** tout couple (S,A) où S est un ensemble dont les éléments sont appelés **sommets** et A est une partie de $S\times\Sigma\times S$ dont les éléments sont appelés **arcs**. Remarquons que les *boucles* sont possibles. Pour un arc (q,a,q'), on dit que a est l'**étiquette** de l'arc, q son origine et q' son extrémité. Un graphe d'automate correspond donc à un automate non déterministe sans notion d'état initial ou final.

Par exemple, avec $\Sigma=\{a,b\}$, $S_0=\{0,1,2,3,4,5\}$ et $A_0=\{(0,a,1),(0,a,3),(0,b,0),(0,b,2),(1,a,1),(1,b,2),(2,a,1),(2,b,3),(2,b,4),(3,a,2),(4,a,1),(4,b,5),(5,a,1)\}$, le graphe d'automate $G_0=(S_0,A_0)$ est représenté à la figure 4.

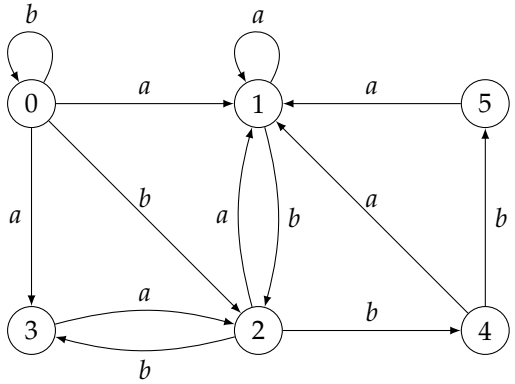

FIGURE 4 – Le graphe d'automate G_0.

On prendra toujours pour l'ensemble de sommets S un intervalle d'entiers de la forme $\{0,|S|-1\}$, pour Σ l'ensemble des caractères du type `char` et on suppose que l'ensemble des arcs étiquetés par Σ est représenté par un tableau de listes d'adjacences `g`, c'est-à-dire que pour tout $s\in S$, `g.(s)` est la liste (dans un ordre arbitraire) de tous les couples (t,a) tel que $(s,a,t)\in A$.

Par exemple, le graphe G_0 est ainsi représenté par :

OCAML
```
type graphe = (int * char) list array

let g0 = [|
  [(1, 'a'); (3, 'a'); (0, 'b'); (2, 'b')];
  [(1, 'a'); (2, 'a')];
  [(1, 'b'); (3, 'b'); (4, 'b')];
  [(2, 'a')];
  [(1, 'a'); (5, 'b')];
  [(1, 'a')]
|]
```

Q 13) Écrire une fonction `accessibles : graphe -> int list -> int` prenant en entrée un graphe d'automate $G=(S,A)$ et un ensemble de sommets $S'\subseteq S$ représenté sous la forme d'une liste de sommets et qui renvoie le nombre de sommets accessibles à partir d'au moins un sommet de S'.

Q 14) Justifier que votre fonction termine.

Q 15) Donner, sans justifier, la complexité de votre fonction.

V Existence d'un mot synchronisant (en CAML)

On reprend dans cette partie le problème de l'existence d'un mot synchronisant pour une machine $M=(Q,\Sigma,\delta)$. Pour $X\subseteq Q$ et $u\in\Sigma^*$, on note de manière abusive $\delta^*(X,u)=\{\delta^*(q,u),q\in X\}$.

Q 16) Soit u un mot synchronisant de M et $u_0,u_1,\dots,u_r$ une suite de préfixes de u, rangés dans l'ordre croissant de leur longueur et telle que $u_r=u$. Que peut-on dire de la suite des cardinaux $|\delta^*(Q,u_i)|$?

Q 17) Montrer qu'il existe un mot synchronisant si et seulement s'il existe pour tout couple d'états (q,q') de Q^2 un mot $u_{q,q'}$ tel que $\delta^*(q,u_{q,q'})=\delta^*(q',u_{q,q'})$.

On veut se servir du critère établi ci-dessus pour déterminer s'il existe un mot synchronisant. Pour cela, on associe à la machine M la machine $\widetilde{M}=(\widetilde{Q},\Sigma,\tilde{\delta})$ définie par :

- $\widetilde{Q}=\mathcal{P}_1(Q)\cup\mathcal{P}_2(Q)$ est formé des parties à un ou deux éléments de Q ;
- $\tilde{\delta}$ est définie par $\forall(X,a)\in\widetilde{Q}\times\Sigma,\tilde{\delta}(X,a)=\{\delta(q,a),q\in X\}$.

Q 18) Si $n = |Q|$, que vaut $\tilde{n} = |\widetilde{Q}|$?

On rappelle que pour la modélisation informatique, l'ensemble d'états d'une machine est modélisé par un intervalle $[\![0, n-1]\!]$. $\widetilde{Q}$ doit donc être modélisé par $[\![0, \tilde{n}-1]\!]$.

Q 19) Choisir une bijection φ_n de $\widetilde{Q}$ sur $[\![0, \tilde{n}-1]\!]$, que vous détaillerez et écrire une fonction `phi : int -> int list -> int` qui prend en argument l'entier n correspondant au nombre d'état, ainsi qu'une liste ℓ à un élément ou à deux éléments distincts et qui renvoie $\varphi_n(\{q\})$ si ℓ est constituée du seul élément q et $\varphi_n(\{q_1, q_2\})$ si ℓ est constituée de deux éléments $q_1 \neq q_2$.

On admettra disposer de sa fonction réciproque `phi_inv : int -> int -> int list`.

Par la suite, on représente une machine $M = (Q, \Sigma, \delta)$ par la donnée d'une matrice d'entiers `m : int array array` telle que `m.(q).(int_of_char a)` représente l'état $\delta(q, a)$.

Q 20) Écrire une fonction `delta_tilde : machine -> int -> lettre -> int` qui prend en argument une machine M, $X \in \widetilde{Q}$ et $a \in \Sigma$ et renvoie $\tilde{\delta}(X, a)$.

Il est clair qu'à la machine $\widetilde{M}$, on peut associer un graphe d'automate $\widetilde{G}$ dont l'ensemble des sommets est $\widetilde{Q}$ et dont l'ensemble des arcs est $\{(X, a, \tilde{\delta}(X, a)), (X, a) \in \widetilde{Q} \times \Sigma\}$. On associe alors à $\widetilde{G}$ le graphe transposé $\widetilde{G}^T$ qui a les mêmes sommets que $\widetilde{G}$ mais dont les arcs sont retournés (i.e. (X, a, Y) est un arc de $\widetilde{G}^T$ si et seulement si (Y, a, X) est un arc de $\widetilde{G}$).

Q 21) Écrire une fonction `transpose : machine -> (int * char) list array` qui à partir d'une machine M calcule le tableau des listes d'adjacences de $\widetilde{G}^T$.

Q 22) Justifier qu'il suffit d'appliquer la fonction `accessibles` de la partie IV au graphe $\widetilde{G}^T$ et à l'ensemble des sommets de $\widetilde{G}^T$ correspondant à des singletons pour déterminer si la machine M possède un mot synchronisant.

Q 23) Écrire une fonction `existe_synchronisant : machine -> bool` qui détermine si une machine possède un mot synchronisant.

VI Réduction depuis SAT

Cette partie ne doit être abordée que si vous avez traité intégralement toutes les parties précédentes, elle ne rapporte que très peu de points et n'est nécessaire que pour atteindre la note maximale.

Dans cette dernière partie, on cherche à montrer que le problème consistant à décider si une machine M possède un mot synchronisant de taille au moins m est un problème au moins aussi difficile que le problème SAT. On s'intéresse dans cette partie à la satisfiabilité d'une formule logique sous forme normale conjonctive. Par exemple,

$$\varphi_0 = (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (x_2 \vee \overline{x_3} \vee x_4)$$

est une formule sous forme normale conjonctive formée de trois clauses et portant sur un ensemble de 4 variables $\mathcal{V}_0 = \{x_1, x_2, x_3, x_4\}$.

Soit φ une formule sous forme normale conjonctive, composée de n clauses et faisant intervenir m variables d'un ensemble $\mathcal{V}$. On suppose les clauses numérotées $c_1, c_2, \ldots, c_n$. On veut ramener le problème de la satisfiabilité d'une telle formule au problème de la recherche d'un mot synchronisant de longueur inférieure ou égale à m sur une certaine machine.

On introduit pour cela la machine $M_\varphi = (Q, \Sigma, \delta)$ associée à φ par :

- Q est formé de $mn + n + 1$ états, à savoir un état particulier noté f et $n(m+1)$ autres états qu'on notera $q_{i,j}$ avec $(i, j) \in [\![1, n]\!] \times [\![1, m+1]\!]$;
- $\Sigma = \{0, 1\}$;
- δ est définie par :
  - f est un états puits, c'est-à-dire que $\delta(f, 0) = \delta(f, 1) = f$;
  - pour tout $i \in [\![1, n]\!]$, $\delta(q_{i,m+1}, 0) = \delta(q_{i,m+1}, 1) = f$;
  - pour tout $i \in [\![1, n]\!]$ et $j \in [\![1, m]\!]$,

$$\delta(q_{i,j}, 0) = \begin{cases} f & \text{si le littéral } \overline{x_j} \text{ apparaît dans la clause } c_i \\ q_{i,j+1} & \text{sinon} \end{cases}$$

$$\delta(q_{i,j}, 1) = \begin{cases} f & \text{si le littéral } x_j \text{ apparaît dans la clause } c_i \\ q_{i,j+1} & \text{sinon} \end{cases}$$

Q 24) Représenter graphiquement M_{φ_0} où φ_0 est la formule donnée précédemment.

Q 25) Donner un modèle $\mu \in \{0, 1\}^{\mathcal{V}_0}$ de φ_0. Le mot $\mu(x_1)\mu(x_2)\mu(x_3)\mu(x_4)$ est-il synchronisant ?

Q 26) Montrer que tout mot u de longueur $m+1$ est synchronisant. À quelle condition sur les $\delta^*(q_{i,1}, u)$ un mot u de longueur m est-il synchronisant ?

Q 27) Montrer que si une formule φ est satisfiable, tout modèle de φ donne un mot de longueur m synchronisant pour M_φ. On détaillera la construction de ce mot.

Q 28) Réciproquement, montrer que si M_φ possède un mot synchronisant de longueur inférieure ou égale à m, alors φ est satisfiable. Indiquer comment il est possible de construire un modèle de φ.