Composition d'informatique n°3

Sujet 1 : algorithmique et programmation (Durée : 4 heures)

L'utilisation de la calculatrice **n'est pas autorisée** pour cette épreuve.

⋆ ⋆ ⋆

Jeu de synchronisation

Le sujet traite de synchronisation dans des automates, ainsi qu'un jeu à deux joueurs sur la synchronisation. Il comporte quatre parties. La première présente les machines et la notion de synchronisation dans ces dernières, ainsi que quelques exemples et propriétés. La deuxième traite de l'existence de mots synchronisants dans des machines. La troisième partie présente le jeu de synchronisation et énonce quelques résultats, notamment sous certaines conditions de machines. Enfin, la quatrième et dernière partie présente un algorithme de recherche de stratégie gagnante dans une machine quelconque.

Consignes

Les questions de programmation doivent être traitées en langage C uniquement. On supposera que les bibliothèques `stdlib.h`, `stdbool.h` et `string.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 en Computer Modern 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 dans le pire des cas. La complexité sera exprimée sous la forme $\mathcal{O}(f(n,m))$ où n et m sont les tailles des arguments de la fonction, et f une expression la plus simple possible. Les calculs de complexité seront justifiés succinctement.

1 Machines synchronisées

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 tous $q \in Q$, $u \in \Sigma^*$ 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 $q_f \in Q$ tel que $\forall q \in Q$, $\delta^*(q, u) = q_f$. 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 machine M est dite **synchronisée** si elle possède un mot synchronisant.

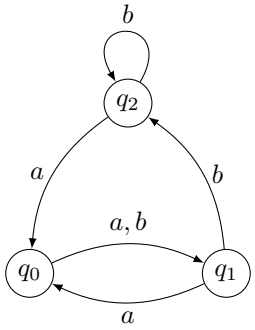

FIGURE 1 – La machine M_0.

La figure 1 représente une machine M_0. On pourra remarquer que ba et bb sont des mots synchronisants pour M_0.

On représente une lettre de Σ en C par un objet de 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'`.

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

```
struct Machine{
    int Q;
    int Sigma;
    int** delta;
};

typedef struct Machine machine;
```

tel que si $M = (Q, \Sigma, \delta)$ est une machine représenté par un objet `M` de type `machine`, alors :

- `M.Q` représente $|Q|$, on suppose $Q = [\![0, |Q| - 1]\!]$;
- `M.Sigma` représente $|\Sigma|$;
- `M.delta` est un tableau de $|Q|$ tableaux tels que si $q \in Q$ et $a \in \Sigma$, alors `M.delta[q][code(a)]` vaut $\delta(q, a)$, où `int code(char a)` est une bijection de Σ dans $[\![0, |\Sigma| - 1]\!]$, qu'on suppose déjà définie et qu'on ne demande pas d'expliciter.

Question 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.

Question 2 On considère la machine M_1 représentée figure 2, sur l'alphabet $\Sigma = \{a\}$. Donner, en justifiant, un mot synchronisant pour M_1 s'il en existe un, ou prouver qu'il n'en existe pas.

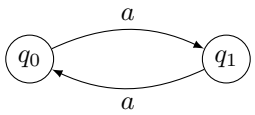

FIGURE 2 – La machine M_1.

Question 3 Donner un mot synchronisant de trois lettres pour la machine M_2 représentée figure 3. On ne demande pas de justifier la réponse.

Question 4 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 et un mot u et renvoie $\delta^*(q, u)$.

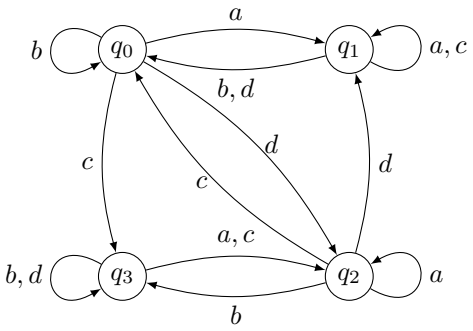

FIGURE 3 – La machine M_2.

Question 5 Écrire une fonction `bool 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.

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 :

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

Question 6 Montrer que l'existence d'un mot synchronisant pour M se ramène à un problème d'accessibilité à certain(s) état(s) depuis certain(s) état(s) de $\widehat{M}$.

Question 7 En déduire que le langage $LS(M)$ des mots synchronisants de M est reconnaissable.

Question 8 Déterminer puis représenter graphiquement un automate fini déterministe (pas nécessairement complet) reconnaissant $LS(M_0)$.

2 Existence de mot synchronisant

On considère le problème de décision `SYNCHRONISÉE` :

* **Instance** : une machine $M = (Q, \Sigma, \delta)$.
* **Question** : la machine M est-elle synchronisée ?

La question 6 montre que ce problème est décidable, mais suggère la création de l'automate des parties, de taille exponentielle en le nombre d'états d'une machine M. Dans cette partie, on cherche à montrer qu'on peut résoudre ce problème en temps polynomial.

2.1 Machine des paires

On considère une machine $M = (Q, \Sigma, \delta)$. Pour $X \subseteq Q$ et $u \in \Sigma^*$, on note $\delta^*(X, u) = \{\delta^*(q, u), q \in X\}$.

Question 9 Soit u un mot synchronisant de M et u_0, u_1, …, 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 $|\delta^*(Q, u_i)|$?

Soient $p, q \in Q^2$, $p \neq q$. La paire d'états $\{p, q\}$ est dite **synchronisée** s'il existe un mot $u_{p,q} \in \Sigma^*$ tel que $\delta^*(p, u_{p,q}) = \delta^*(q, u_{p,q})$.

Question 10 Montrer que M est synchronisée si et seulement si toute paire d'états de Q est synchronisée.

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 **des paires** $\widetilde{M} = (\widetilde{Q}, \Sigma, \widetilde{\delta})$ définie par :

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

La machine $\widetilde{M}$ est donc une machine à $\widetilde{n} = \frac{n(n+1)}{2}$ états (où $n = |Q|$). On suppose disposer d'une bijection $\varphi : \widetilde{Q} \to [\![0, \widetilde{n} - 1]\!]$ sous la forme d'une fonction `int phi(int p, int q)` telle que pour $p, q \in Q^2$, `phi(p, q)` renvoie $\varphi(\{p, q\})$ (y compris dans le cas où $p = q$).

Question 11 Écrire une fonction `bool paire_synchronisee(machine M, int p, int q)` qui prend en argument une machine M et deux états p et q dans la machine M et renvoie un booléen qui indique si la paire $\{p, q\}$ est synchronisée.

On ne demande pas nécessairement de construire la machine des paires explicitement, mais on pourra l'utiliser dans le raisonnement.

Question 12 En déduire une fonction `bool synchronisee(machine M)` qui détermine si une machine est synchronisée ou non.

Question 13 Déterminer la complexité temporelle de la fonction précédente en fonction du nombre n d'états de la machine M.

Question 14 Montrer qu'on peut peut résoudre le problème `SYNCHRONISÉE` en complexité temporelle $\mathcal{O}(|Q|^2|\Sigma|)$. On détaillera en français l'algorithme correspondant, mais on ne demande pas de l'implémenter.

On considère le problème `ÉTAT DE PASSAGE` :

* **Instance** : une machine $M = (Q, \Sigma, \delta)$ et un état $q_0 \in Q$.
* **Question** : Existe-t-il un mot $u \in \Sigma^*$ tel que pour tout $q \in Q$, le calcul de q à $\delta^*(q, u)$ passe par q_0 ?

Question 15 Montrer que `ÉTAT DE PASSAGE` se réduit many-one à `SYNCHRONISÉE`.

2.2 Bornes sur les mots synchronisants

Dans cette partie, on considère une machine synchronisée $M = (Q, \Sigma, \delta)$ avec $n = |Q|$. On s'intéresse à des bornes sur la taille d'un mot synchronisant de M en fonction de n.

Question 16 Montrer que M possède un mot synchronisant de taille au plus $\frac{n(n-1)^2}{2}$.

Indication : on pourra s'intéresser à la taille d'un mot synchronisant une paire d'états.

Pour $n \in \mathbb{N}^*$, on définit la machine de Černý d'ordre n, notée $\mathcal{C}_n$, par $\mathcal{C}_n = (Q = [\![0, n-1]\!], \Sigma = \{a, b\}, \delta)$ où, pour $q \in Q$:

- $\delta(q, a) = \begin{cases} 1 & \text{si } q = 0 \\ q & \text{sinon} \end{cases}$;
- $\delta(q, b) = (q + 1) \bmod n$.

La figure 4 représente la machine $\mathcal{C}_3$.

Question 17 Montrer que pour $n \in \mathbb{N}^*$, $\mathcal{C}_n$ est synchronisée et donner l'expression d'un mot synchronisant de taille $(n-1)^2$.

Dans la machine $\mathcal{C}_n$, pour $X \subseteq Q$, on note $\Delta(X)$ le plus grand « trou » de X, c'est-à-dire la plus grande suite d'états consécutifs (modulo n) de $Q \setminus X$. Par exemple, pour $n = 8$ et $X = \{2, 4, 5\}$, $\Delta(X) = \{6, 7, 0, 1\}$. Si plusieurs ensembles $\Delta(X)$ peuvent convenir, on en choisit un arbitrairement.

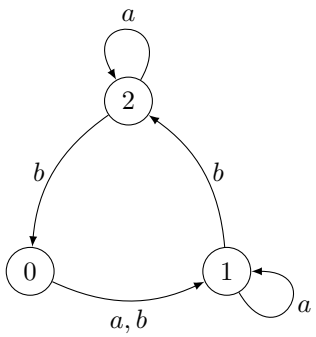

FIGURE 4 – La machine $\mathcal{C}_3$.

Question 18 Soit $X \subseteq Q$ et $\alpha \in \Sigma$ tels que $|\Delta(X)| = j$ et $|\Delta(\delta(X, \alpha))| = j + 1$. Quelles peuvent être les valeurs possibles de α et $\Delta(X)$?

Question 19 En déduire qu'il n'existe pas, pour $n \in \mathbb{N}^*$, de mot synchronisant pour $\mathcal{C}_n$ de taille strictement inférieure à $(n-1)^2$.

3 Jeu de synchronisation

On considère le jeu à deux joueurs suivant dans une machine $M = (Q, \Sigma, \delta)$, appelé **jeu de synchronisation** :

- initialement, il y a une pièce sur chaque état de M ;
- les joueurs jouent tour à tour, en commençant par le joueur 1. Le tour d'un joueur se déroule comme suit :
  - le joueur choisit une lettre a ;
  - pour chaque état q qui contient une pièce, le joueur déplace la pièce de l'état q vers $\delta(q, a)$;
  - si après les déplacements un état contient plusieurs pièces, le joueur les enlève toutes sauf une (de telle sorte qu'à la fin de son tour, un état contient au plus une pièce) ;
- la partie se termine s'il ne reste qu'un seul état qui contient une pièce, auquel cas c'est le joueur 1 qui remporte la partie. Si le joueur 2 peut empêcher la partie de se terminer indéfiniment, c'est lui qui remporte la partie.

Pour M une machine fixée, une **configuration** du jeu est un ensemble $X \subseteq Q$ qui représente les états de M qui contiennent une pièce. Une **partie** peut alors être vue soit comme une suite de lettres $(a_i)_{i \in \mathbb{N}^*}$ (correspondant aux lettres choisies par les joueurs), soit comme une suite de parties $(X_i)_{i \in \mathbb{N}}$ (correspondant aux configurations de la partie), avec $X_0 = Q$. On remarque qu'on peut faire le lien entre ces deux représentations : pour $i \in \mathbb{N}$, $X_{i+1} = \delta(X_i, a_{i+1})$. La partie est gagnée par le joueur 1 s'il existe $i \in \mathbb{N}$ tel que $|X_i| = 1$.

Une **stratégie** pour un joueur est alors une fonction $f : \mathcal{P}(Q) \to \Sigma$. Si f et g sont des stratégies pour les joueurs 1 et 2 respectivement, on appelle (f, g)**-partie** depuis une configuration $X \subseteq Q$ la partie $(X_i)_{i \in \mathbb{N}}$ telle que $X_0 = X$ et :

- pour $i \in 2\mathbb{N}$, $X_{i+1} = \delta(X_i, f(X_i))$;
- pour $i \in 2\mathbb{N} + 1$, $X_{i+1} = \delta(X_i, g(X_i))$.

Une stratégie f est **gagnante** depuis une configuration X pour le joueur 1 si quelle que soit la stratégie g pour le joueur 2, la (f, g)-partie depuis X est gagnante pour 1. On définit de même une stratégie gagnante pour le joueur 2. Lorsqu'on parle de stratégie gagnante sans préciser la configuration X, il sera sous-entendu qu'il s'agit d'une stratégie gagnante depuis la configuration $X = Q$.

Question 20 Montrer que s'il existe une stratégie gagnante pour 1 dans une machine M, alors M est synchronisée.

Question 21 Montrer que pour une machine M, il existe une stratégie gagnante pour le joueur 1 si et seulement si il existe une stratégie gagnante pour le joueur 1 depuis toute configuration X de cardinal 2.

La réciproque à la question 20 est fausse, et on peut utiliser à nouveau les machines de Černý.

Question 22 Montrer que pour $n \geqslant 4$, il existe une stratégie gagnante pour le joueur 2 dans le jeu de synchronisation dans la machine $\mathcal{C}_n$.

Soient $p, q \in Q^2$, on dit que q est **accessible** depuis p, noté $p \preccurlyeq q$, s'il existe $u \in \Sigma^*$ tel que $\delta^*(p, u) = q$. Un état **puits** est un état maximal pour la relation $\preccurlyeq$, c'est-à-dire un état $q_m \in Q$ tel que pour tout $q \in Q \setminus \{q_m\}$, $q_m \not\preccurlyeq q$.

M est dite **faiblement acyclique** si la relation $\preccurlyeq$ est antisymétrique.

Question 23 Soit $M = (Q, \Sigma, \delta)$ une machine faiblement acyclique. Montrer qu'il y a équivalence entre les trois propriétés suivantes :

(a) M a un unique état puits.

(b) M est synchronisée.

(c) Le joueur 1 a une stratégie gagnante dans M.

4 Recherche de stratégie gagnante

On revient dans le cas général d'une machine $M = (Q, \Sigma, \delta)$ quelconque. On cherche à déterminer un algorithme de complexité polynomiale permettant de déterminer l'existence d'une stratégie gagnante pour le joueur 1 (et par extension permettant de déterminer cette stratégie).

Étant donné une machine $M = (Q, \delta, \Sigma)$ et sa machine des paires $\widetilde{M} = (\widetilde{Q}, \Sigma, \widetilde{\delta})$, on définit par récurrence les ensembles Attr_i par :

- $\mathrm{Attr}_0 = \{\{q\} \mid q \in Q\} \times \{1, 2\}$;
- pour $i \in \mathbb{N}$, $\mathrm{Attr}_{i+1} = \mathrm{Attr}_i \cup \{(X, 1) \mid X \in \widetilde{Q} \text{ et } \widetilde{\delta}(X, \Sigma) \times \{2\} \cap \mathrm{Attr}_i \neq \emptyset\} \cup \{(X, 2) \mid X \in \widetilde{Q} \text{ et } \widetilde{\delta}(X, \Sigma) \times \{1\} \subseteq \mathrm{Attr}_i\}$

Question 24 Décrire en français à quoi correspond l'ensemble Attr_i, pour $i \in \mathbb{N}$. On ne demande pas de justification.

Question 25 Montrer que la suite $(\mathrm{Attr}_i)_{i \in \mathbb{N}}$ converge vers un ensemble qu'on notera Attr.

Question 26 Montrer que le joueur 1 a une stratégie gagnante dans le jeu de synchronisation sur M si et seulement si $\mathcal{P}_2(Q) \times \{1\} \subseteq \mathrm{Attr}$.

On pose $G_M = (S, A)$ le graphe orienté défini par :

- $S = \widetilde{Q} \times \{1, 2\}$;
- pour $X, Y \in \widetilde{Q}$ et $j \in \{1, 2\}$, $((X, j), (Y, 3 - j)) \in A$ si et seulement s'il existe $a \in \Sigma$ tel que $\widetilde{\delta}(Y, a) = X$.

Autrement dit, le graphe G_M correspond au transposé de la machine des paires, sans les étiquettes des transitions et dans lequel on a dupliqué chaque état et créé une alternance des joueurs.

On représente un graphe orienté en C par un objet du type `graphe` défini par :

```
struct Graphe {
    int S;
    int* deg;
    int** A;
};

typedef struct Graphe graphe;
```

tel que si $G = (S, A)$ est représenté par un objet `G` de type `graphe`, alors $S = [\![0, n-1]\!]$ avec $n =$ `G.S`, `G.deg` est un tableau d'entiers de taille n tel que pour $s \in S$, `G.deg[s]` est le degré **sortant** de s et `G.A` est un tableau de taille n tel que pour $s \in S$, `G.A[s]` est un tableau de taille `G.deg[s]` contenant les voisins de s, dans un ordre quelconque.

On s'autorise, pour la suite, l'utilisation de la fonction `calloc` qui permet la création de tableaux de `0` ou de `false`, alloués sur le tas. Son utilisation peut se faire de la manière suivante :

```
int* tab_zeros = calloc(n, sizeof(int)); // tableau de n zéros
bool* tab_false = calloc(n, sizeof(bool)); // tableau de n false
```

Question 27 Écrire une fonction `graphe construire_GM(machine M)` qui construit le graphe G_M défini précédemment pour une machine M. On choisira une numérotation adaptée des sommets, à préciser.

On accepte la construction d'un multigraphe, c'est-à-dire un graphe où un sommet peut avoir plusieurs fois le même voisin.

Question 28 Écrire une fonction `int gagnant(machine M)` qui prend en argument la machine M renvoie un entier valant 1 ou 2 selon le joueur qui possède une stratégie gagnante dans le jeu de synchronisation dans la machine M.

Indication : on pourra construire un tableau de booléens `attr` *de taille* $|S|$ *tel que pour* $s \in S$, `attr[s]` *vaut* `true` *si et seulement si* $s \in Attr$. *On supposera disposer d'une fonction* `void liberer_graphe(graphe G)` *permettant de libérer la mémoire allouée pour un graphe* G.

Question 29 Quelle est la complexité temporelle de la fonction `gagnant` ?

Partie bonus

Question 30 Soit $M = (Q, \Sigma, \delta)$ une machine synchronisée à n états. Montrer que M admet un mot synchronisant de longueur $(n-1)^2$.

⋆ ⋆ ⋆