DS n° 04 — 2h

On veillera à présenter *très clairement* sa copie, on attachera un soin particulier à la rédaction et **on encadrera les réponses**. Des points pourront être accordés à la présentation, à la lisibilité et au soin accordé à la rédaction. Il est impératif d'utiliser un brouillon. Les programmes seront rédigés clairement et proprement, en mettant en valeur l'indentation et en choisissant des noms de variables explicites. **Les commentaires des programmes doivent dans tous les cas être dans une autre couleur que le programme lui-même**. Il est impératif de respecter l'indentation usuelle des fonctions OCaml et C.

Lorsque l'on pose des hypothèses sur les arguments, il n'est pas nécessaire dans vos fonctions de le vérifier ni de prévoir le comportement si l'hypothèse est violée. L'introduction et l'utilisation de fonctions auxiliaires est autorisée, et même encouragée, d'autant plus si cela améliore la lisibilité et la compréhension. Les fonctions auxiliaires **doivent être précédées de leur type (en OCaml) et commentées**.

Le sujet est composé d'une première partie de questions de cours qu'il faut impérativement traiter en premier, puis de deux parties indépendantes, que vous pouvez traiter dans l'ordre de votre choix.

I Questions de cours

I.1 Lemme de l'étoile

Q 1) Énoncer précisément le *lemme de l'étoile*.

Q 2) On pose $\Sigma = \{a, b\}$. Déterminer parmi les langages suivants lesquels sont réguliers. Justifier.

- $L_1 = \{a^n b^2 \mid n \in \mathbb{N}^*\}$
- $L_2 = \{a^n b^{2n} \mid n \in \mathbb{N}^*\}$
- $L_3 = \{uv \mid u, v \in \Sigma^*\}$
- $L_4 = \{uu \mid u \in \Sigma^*\}$

I.2 Terminaison automatique d'un programme

Dans cette sous-section, un programme est un fichier texte `.ml` correspondant à un programme OCaml valide, qui compile sans erreur. On suppose que l'on se place toujours dans le répertoire courant.

Q 3) Donner la commande UNIX permettant de lister le contenu du répertoire courant.

Q 4) Est-il possible de vérifier par une procédure effective qu'un fichier texte correspond bien à un programme ?

On cherche à montrer qu'il est impossible d'écrire en OCaml une fonction `termine` de type `string -> bool` telle que si `nom_fichier` est le nom d'un programme, `termine nom_fichier` s'arrête toujours et s'évalue à `true` si et seulement si l'exécution du programme contenu dans le fichier correspondant termine. On procède par l'absurde et on suppose qu'une telle fonction existe.

Q 5) Écrire une fonction `terminaison_inverse : string -> unit` telle que si `nom_fichier` est un nom de programme, alors `terminaison_inverse nom_fichier` termine si et seulement si le programme contenu dans le fichier ne termine pas.

On enregistre le code source de la fonction `termine`, celui de la fonction de la question précédente `terminaison_inverse` ainsi que la ligne ci-dessous dans un fichier de nom `"paradoxe.ml"`.

paradoxe.ml

```
let termine nom_fichier =
  let fichier_source = open_in nom_fichier in
  (* code source de la fonction termine *)
  ...

let terminaison_inverse nom_fichier =
  (* code écrit à la question précédente *)
  ...

let () = terminaison_inverse "paradoxe.ml"
```

Q 6) Que se passe-t-il si on compile ce programme ? Que se passe-t-il si on exécute ce programme ?

Q 7) Conclure.

I.3 Élagage $\alpha\beta$

Q 8) Déterminer le score des nœuds explorés étant donné les valeurs de l'heuristique aux feuilles pour l'arbre de la figure 1, *avec élagage* $\alpha - \beta$. On suppose que la racine est contrôlée par le joueur Max. On ne recopiera sur la feuille que la portion de l'arbre réellement explorée, en indiquant clairement les élagages réalisés.

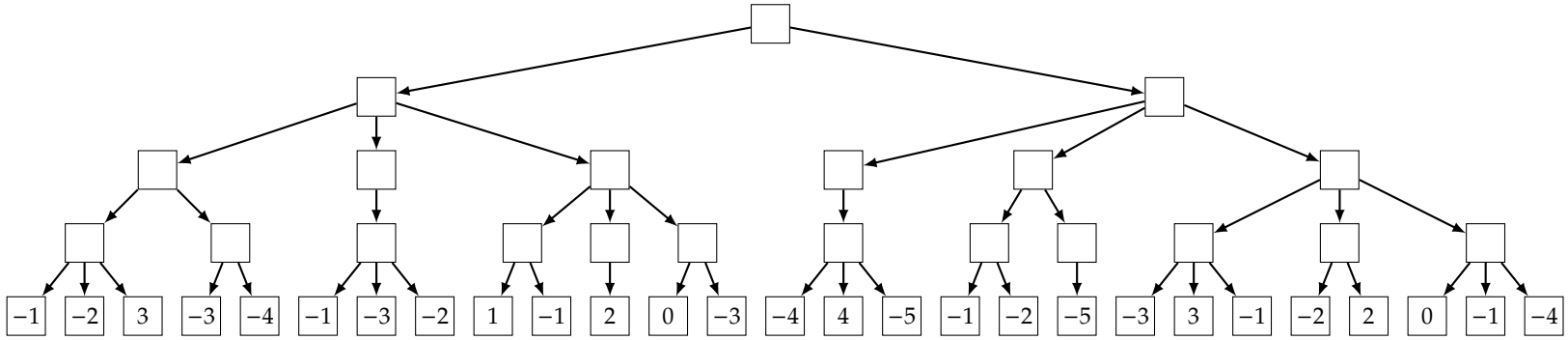
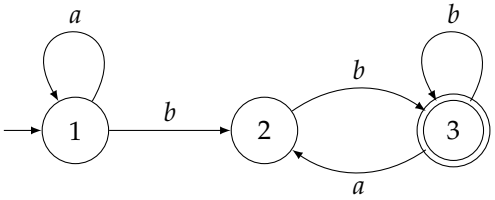


FIGURE 1 – Arbre des configurations jusqu'à une profondeur 4 avec valeur de l'heuristique aux feuilles. La racine appartient au joueur 1 qui cherche à maximiser le score.

I.4 Algorithmes sur les automates

I.4.1 Élimination des états

Q 9) Appliquer la méthode d'élimination des états à l'automate suivant en éliminant successivement dans cet ordre les états 3, puis 2 puis 1, sans opérer aucune simplification au niveau des expressions régulières.



I.4.2 Algorithme de Berry-Sethi

Q 10) Appliquer l'algorithme de Berry-Sethi pour déterminer l'automate de Glushkov associé à l'expression régulière $(a|b)^\star a$. On donnera les résultats des étapes intermédiaires.

Q 11) Déterminiser l'automate obtenu.

I.4.3 Automates de Thompson

Q 12) Appliquer la construction de Thompson pour déterminer un automate reconnaissant le langage dénoté par l'expression régulière $(a|b)^\star a$.

Q 13) Éliminer les ϵ-transitions de l'automate obtenu. On ne représentera que les états co-accessibles.

Q 14) Déterminiser l'automate ainsi obtenu.

II Le jeu de Nim

Le jeu de Nim est un jeu à deux joueurs où chaque joueur doit à son tour retirer une ou plusieurs allumettes dans un même tas parmi plusieurs tas d'allumettes. Le joueur 1 commence. On distingue deux variantes de jeu :

- la variante **classique** où le gagnant est celui qui retire la dernière allumette ;
- la variante **misère** où celui qui retire la dernière allumette est le perdant.

Dans sa version la plus simple, il n'y a qu'un tas d'allumettes : n allumettes sont placées côte à côte et tour à tour, chaque joueur doit retirer une ou plusieurs allumettes. On note $*n$ un jeu de Nim avec un seul tas de n allumettes. Lorsqu'on considère un jeu de Nim à plusieurs tas, on note de manière additive le jeu considéré. Par exemple, la figure 2 représente le jeu de Nim $*1 + *3 + *5 + *7$. À chaque tour de jeu, un joueur doit prendre une ou plusieurs allumettes sur *une même ligne*. Il est toujours nécessaire de prendre au moins une allumette, et on peut au plus prendre toutes les allumettes d'une même ligne.

Remarquons que les configurations terminales sont celles ne comportant plus aucune allumette et qu'elles sont :

- gagnantes pour le joueur qui ne les contrôle pas dans la variante classique ;
- gagnantes pour le joueur qui les contrôle dans la variante misère.

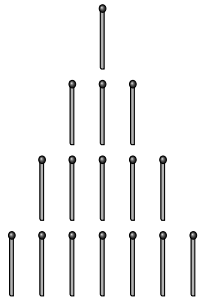


FIGURE 2 – Le jeu $*1 + *3 + *5 + *7$.

Q 15) Montrer que quelle que soit la variante et quel que soit le nombre et la taille des tas, il existe toujours un des deux joueurs qui possède une stratégie gagnante au jeu de Nim.

Q 16) Dessiner le graphe des configurations pour le jeu $*1 + *2$ dans la variante misère. On représentera les sommets contrôlés par le premier joueur par un carré et ceux contrôlés par le deuxième joueur par un cercle. On ne représentera que les sept sommets accessibles à partir de la configuration initiale. Pour chaque sommet, indiquer clairement s'il est dans l'attracteur du joueur 1, du joueur 2 ou dans aucun des deux.

Q 17) Dans cette question, on considère le jeu $*n$ pour $n \geq 1$. Pour chacune des deux variantes, caractériser les attracteurs des deux joueurs. Justifier.

Q 18) En déduire, en fonction de n et de la variante, quel joueur possède une stratégie gagnante.

Q 19) Pour $n \in \mathbb{N}^*$, proposer une stratégie gagnante facile à appliquer pour le joueur 2 dans la variante classique du jeu $*n + *n$. Justifier soigneusement votre proposition, par exemple à l'aide d'un raisonnement par récurrence.

Q 20) Soit $1 \leq m < n$ deux entiers. Montrer qu'il existe une stratégie gagnante pour le joueur 1 dans la variante classique du jeu $*m + *n$.

Q 21) Déterminer, en justifiant, quel joueur possède une stratégie gagnante dans la variante misère pour les configurations initiales :

- $*1 + *n$ pour $n \geq 1$;
- $*2 + *2$;
- $*2 + *n$ pour $n \geq 3$

Q 22) Déterminer sans justifier quel joueur possède une stratégie gagante dans la variante misère du jeu $*m + *n$ pour $n, m \in \mathbb{N}^*$.

III Automates probabilistes

On fixe dans cet exercice un alphabet $\Sigma = \{\mathbf{0}, \mathbf{1}\}$.

Un *automate probabiliste* sur l'alphabet Σ est un quadruplet $\mathcal{A} = (Q, q_0, F, \mathbb{P})$ où :

- Q est un ensemble fini non vide dont les éléments sont appelés *états* ;
- $q_0 \in Q$ est appelé *état initial* ;
- $F \subseteq Q$ est un ensemble dont les éléments sont appelés *états finals* ;
- $\mathbb{P} : Q \times \Sigma \times Q \to [0,1]$ est une application appelée *fonction probabiliste de transition*. On suppose, que pour tout $q \in Q$, pour tout $\alpha \in \Sigma$, on a

$$\sum_{q' \in Q} \mathbb{P}(q, \alpha, q') = 1$$

On note $\mathbb{P}(q \xrightarrow{\alpha} q')$ pour $\mathbb{P}(q, \alpha, q')$.

Une *transition* est un triplet $(q, \alpha, q') \in Q \times \Sigma \times Q$ avec $\mathbb{P}(q \xrightarrow{\alpha} q') > 0$. On la note $q \xrightarrow{\alpha} q'$.

On représente un automate probabiliste de manière graphique, de façon similaire à la représentation des automates non déterministes classiques : les états sont représentés par des cercles, l'état initial par une flèche arrivant sur le cercle correspondant, les états finals par des cercles doubles. La fonction probabiliste de transition est représentée par une flèche entre états :

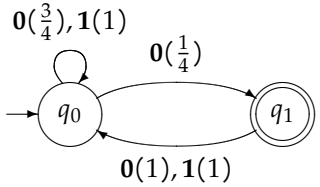
- si $\mathbb{P}(q \xrightarrow{\alpha} q')$ est un nombre $p > 0$, on met une flèche de l'état q à l'état q', étiquetée par $\alpha(p)$.

Ainsi, l'automate $\mathcal{A}_0 = (\{q_0, q_1\}, q_0, \{q_1\}, \mathbb{P})$ représenté à la figure 3 à gauche a pour fonction probabiliste de transition $\mathbb{P}$ la fonction dont la table est représentée sur cette même figure à droite (seules les valeurs non nulles sont mentionnées).

Etant donné un automate probabiliste $\mathcal{A} = (Q, q_0, F, \mathbb{P})$ sur Σ, un *chemin* ρ est une suite finie de transitions $q_{i_1} \xrightarrow{\alpha_1} q_{i_2} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} q_{i_{n+1}}$. On dit que ρ a pour *étiquette* le mot $\alpha_1 \dots \alpha_n \in \Sigma^*$, pour *état de départ* l'état $q_{i_1} \in Q$ et pour *état d'arrivée* l'état $q_{i_{n+1}} \in Q$. La *probabilité* de ρ, notée $\mathbb{P}(\rho)$, est définie par

$$\mathbb{P}(\rho) = \prod_{k=1}^{n} \mathbb{P}(q_{i_k} \xrightarrow{\alpha_k} q_{i_{k+1}})$$

Un état peut être vu comme un chemin de longueur nulle et la probabilité d'un chemin de longueur nulle est égale à 1. Un *chemin pour le mot* $u \in \Sigma^*$ est un chemin dont



q	a	q'	$\mathbb{P}(q \xrightarrow{a} q')$
q_0	$\mathbf{0}$	q_0	$3/4$
q_0	$\mathbf{0}$	q_1	$1/4$
q_0	$\mathbf{1}$	q_0	1
q_1	$\mathbf{0}$	q_0	1
q_1	$\mathbf{1}$	q_0	1

FIGURE 3 – L'automate $\mathcal{A}_0$.

l'étiquette est u et l'état de départ est q_0. Ce chemin est *acceptant* si l'état d'arrivée est un état de F et *non acceptant* sinon. On note $C(u)$ l'ensemble des chemins pour u et $C^+(u)$ l'ensemble des chemins acceptants pour u. La *probabilité d'un mot* $u \in \Sigma^*$, notée $\mathbb{P}(u)$ est définie comme la somme des probabilités de tous les chemins acceptants pour le mot $u \in \Sigma^*$:

$$\mathbb{P}(u) = \sum_{\rho \in C^+(u)} \mathbb{P}(\rho)$$

Q 23) Calculer les probabilités $\mathbb{P}(\varepsilon)$, $\mathbb{P}(\mathbf{0})$, $\mathbb{P}(\mathbf{010})$ pour l'automate $\mathcal{A}_0$.

Q 24) Quels sont les mots u dont la probabilité $\mathbb{P}(u)$ pour $\mathcal{A}_0$ est égale à 0 ? Quels sont ceux dont la probabilité est égale à 1 ?

Q 25) Proposer sans justifier une expression régulière simple pour le langage des mots u dont la probabilité $\mathbb{P}(u)$ pour $\mathcal{A}_0$ est non nulle.

Q 26) Montrer soigneusement que pour tout automate probabiliste $\mathcal{A} = (Q, q_0, F, \mathbb{P})$, il existe un automate non nécessairement déterministe $\mathcal{A}'$ qui accepte exactement les mots u dont la probabilité $\mathbb{P}(u)$ pour $\mathcal{A}$ est non nulle.

Q 27) Appliquer la construction précédente à l'automate $\mathcal{A}_0$ pour obtenir un automate non-déterministe qui accepte exactement les mots u dont la probabilité $\mathbb{P}(u)$ pour $\mathcal{A}_0$ est non nulle. Déterminiser cet automate.

Q 28) Montrer, par récurrence sur la longueur du mot u, que, pour tout automate probabiliste $\mathcal{A} = (Q, q_0, F, \mathbb{P})$ et pour tout mot $u \in \Sigma^*$, on a :

$$\sum_{\rho \in C(u)} \mathbb{P}(\rho) = 1$$

Pour $a \in \Sigma$, on pourra noter $Q_0 = \{q \in Q \mid \mathbb{P}(q_0 \xrightarrow{a} q) > 0\}$ et pour $q \in Q$ on pourra considérer l'automate $A_q = (Q, q, F, \mathbb{P})$.

Soit $\mathcal{A} = (Q, q_0, F, \mathbb{P})$ un automate probabiliste sur Σ. Pour un réel $\eta \in [0,1[$, le *η-langage reconnu par* $\mathcal{A}$, noté $\mathcal{L}_\eta(\mathcal{A})$, est défini par

$$\mathcal{L}_\eta(\mathcal{A}) = \{u \in \Sigma^* \mid \mathbb{P}(u) > \eta\}$$

On dit qu'un langage $L \subseteq \Sigma^*$ est *stochastique* s'il existe un automate probabiliste $\mathcal{A}$ et un réel $\eta \in [0,1[$ tel que $L = \mathcal{L}_\eta(\mathcal{A})$.

Q 29) Démontrer que tout langage régulier est stochastique.

Etant donné un mot $a_1 \dots a_n$ sur l'alphabet $\{\mathbf{0}, \mathbf{1}\}$, on dit que l'expression $\overline{0, a_1 \dots a_n}^2$ est une *écriture (finie) en base* 2 du nombre réel $\sum_{i=1}^n 2^{-i} a_i$. On note alors

$$\sum_{i=1}^n 2^{-i} a_i = \overline{0, a_1 \dots a_n}^2$$

Ainsi, $\frac{1}{4} = 2^{-2} = \overline{0, \mathbf{01}}^2 = \overline{0, \mathbf{0100}}^2$.

On considère maintenant l'automate $\mathcal{A}_1 = (\{q_0, q_1\}, q_0, \{q_1\}, \mathbb{P})$ ci-dessous :

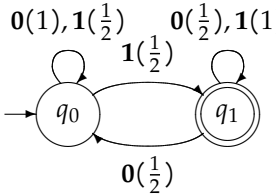


FIGURE 4 – L'automate $\mathcal{A}_1$.

Q 30) Dans l'automate $\mathcal{A}_1$, calculer $\mathbb{P}(q_0 \xrightarrow{\mathbf{1}} q_0 \xrightarrow{\mathbf{1}} q_1 \xrightarrow{\mathbf{1}} q_1 \xrightarrow{\mathbf{0}} q_0 \xrightarrow{\mathbf{1}} q_1)$ et en donner une écriture finie en base 2.

Q 31) Dans l'automate $\mathcal{A}_1$, calculer $\mathbb{P}(\mathbf{10})$ et en donner une écriture finie en base 2.

Q 32) Dans l'automate $\mathcal{A}_1$, calculer $\mathbb{P}(\mathbf{1101})$ et en donner une écriture finie en base 2.

Q 33) Soit $u \in \Sigma^*$ un mot arbitraire sur Σ. Montrer que $\mathbb{P}(u)$ pour $\mathcal{A}_1$ admet une écriture finie en base 2 et en donner une expression. Prouver que cette écriture est correcte.

Q 34) Soit $\eta \in [0,1[$. Prouver l'égalité suivante :

$$\mathcal{L}_\eta(\mathcal{A}_1) = \{a_1 \dots a_n \in \Sigma^* \mid \overline{0, a_n \dots a_1}^2 > \eta\}$$

Q 35) En déduire qu'il existe des langages stochastiques qui ne sont pas réguliers.

— FIN DU SUJET —