DS 3 : Langages et Logique

Consignes à lire attentivement

Ce sujet comporte 6 pages. Durée du devoir : **4 heures**.

Ce sujet est divisé en 3 problèmes indépendants, qui peuvent être traitées dans un ordre quelconque. **Vous devez cependant rendre 2 copies différentes : une première copie contenant vos réponses au problème 1, et une seconde copie contenant vos réponse aux problèmes 2 et 3.**

⋆ ⋆ ⋆

Problème 1 : *Mots de Lyndon et de de Bruijn*

Les définitions et notations utilisées tout au long du problème sont les suivantes :

- Σ désigne un alphabet. **Sauf mention contraire, cet alphabet est $\{0,1\}$, y compris dans les définitions ci-dessous.** En particulier, il fait sens d'écrire $x < y$ si x et y sont des lettres de Σ
- Si $u, v \in \Sigma^*$, on dit que u est un *début* de v si u est un préfixe non trivial (c'est-à-dire, non vide et non égal à v) de v. De même, u est une *fin* de v si u est un suffixe non trivial de v.
- On définit sur Σ^* la relation $\prec$ de la façon suivante : si $u = u_1...u_m$ et $v = v_1...v_n$,

$$u \prec v \quad \text{ si et seulement si } \quad u \text{ est un début de } v \text{ ou } \exists k \in [\![1, m]\!], \begin{cases} \forall i \in [\![1, k-1]\!], u_i = v_i \\ u_k < v_k \end{cases}$$

  On définit ensuite la relation $\preceq$ par : $u \preceq v$ si et seulement si $u = v$ ou $u \prec v$. Remarquons que cette relation n'est rien d'autre que l'ordre lexicographique.
- Si $u = u_0...u_{n-1} \in \Sigma^*$, les *conjugués* du mot u sont tous les mots $v \in \Sigma^*$ pour lesquels il existe $i \in [\![0, n-1]\!]$ tel que $v = u_i u_{i+1}...u_{n-1}u_0...u_{i-1}$. Autrement dit, ce sont les mots obtenus à partir de u par permutation circulaire des lettres de ce mot. On dit que v est un conjugué *non trivial* de u si $i \neq 0$.
- On définit sur Σ^* une relation C de la façon suivante : u et v sont en relation via C, ce qu'on note $C(u, v)$, si et seulement si u est un conjugué de v.

Partie 1 : *Préliminaires*

Question 1. Écrire une fonction `inferieur : string → string → int` telle que `inferieur u v` renvoie 0 si les chaînes u et v sont égales, une valeur strictement négative si $u \prec v$ et une valeur strictement positive si $v \prec u$.

Question 2. Montrer que pour tout $u, v \in \Sigma^*$, si $u \neq v$ alors ($u \prec v$ ou $v \prec u$).

Question 3. Indiquer si $\prec$ est compatible avec la concaténation à droite. Autrement dit, dire s'il est vrai que :

$$\forall u, v, w \in \Sigma^*, \text{ si } u \prec v \text{ alors } uw \prec vw$$

Question 4. Écrire une fonction `conjugue : string → int → string` telle que `conjugue m i` renvoie la chaîne de caractères correspondant au conjugué de `m` commençant par le `i`-ème caractère de `m`. On vérifiera que `i` est compris entre 0 et $|\texttt{m}| - 1$.

Question 5. Considérons l'ensemble des conjugués d'un mot de taille n. Quelle est la taille minimale de cet ensemble ? Quelle est sa taille maximale ? Dans les deux cas, donner un exemple d'alphabet et de mot sur cet alphabet pour lequel il y a atteinte des bornes proposées.

Question 6. Expliquer brièvement pourquoi C est une relation d'équivalence sur Σ^*.

Puisque la relation C est une relation d'équivalence, il fait sens de parler des classes d'équivalences selon cette relation. Une telle classe d'équivalence s'appelle un *collier*. Par exemple, la classe d'équivalence de 001 est le collier $C_1 = \{001, 010, 100\}$ et celle de 1010 est $C_2 = \{0101, 1010\}$. On dit qu'un collier est apériodique si pour tout mot u dans le collier, il n'existe pas de conjugué non trivial de u qui soit égal à u. Par exemple, C_1 est apériodique mais pas C_2 car en notant $u_0 = 1$, $u_1 = 0$, $u_2 = 1$, $u_3 = 0$, on a que $u = u_0u_1u_2u_3 = 1010$ et que $u_2u_3u_0u_1$ est un conjugué non trivial de u qui est égal à $1010 = u$.

Partie 2 : *Mots de Lyndon*

Un mot $u \in \Sigma^*$ est un *mot de Lyndon* si tout conjugué non trivial v de u vérifie $u \prec v$.

Question 7. Indiquer dans chacun des cas si le mot considéré est un mot de Lyndon. Justifier brièvement :
1. 0010011 2. 010011 3. 001001

Question 8. Les mots de longueur 1 sont-ils des mots de Lyndon ?

Question 9. Ecrire une fonction `lyndon : string → bool` telle que `lyndon m` renvoie `true` si un mot `m`, supposé non vide, est un mot de Lyndon et `false` sinon.

La suite de cette partie explore deux méthodes pour construire des mots de Lyndon. La première exploite le théorème suivant (qu'on ne demande pas de démontrer) : un mot est un mot de Lyndon si et seulement si c'est le plus petit élément (pour l'ordre $\preceq$) d'un collier apériodique.

Question 10. Déterminer les classes d'équivalence selon la relation C (c'est-à-dire, les colliers) des mots de longueur 4 sur l'alphabet $\{0, 1\}$.

Question 11. En déduire la liste des mots de Lyndon de longueur 4.

Question 12. Pour $n \in \mathbb{N}$, quelle structure de données permettrait de représenter de façon efficace l'ensemble des colliers de taille n ? On n'attend pas d'implémentation de cette structure.

Question 13. En utilisant la structude de données de la question précédente, décrire en pseudo-code comment calculer les colliers de taille $n \in \mathbb{N}$.

La seconde méthode repose sur un algorithme permettant de générer tous les mots de Lyndon de taille bornée par un entier n. Le pseudo-code de cet algorithme est le suivant :

```
1:  fonction Génère-Lyndon(n ∈ ℕ) :  // Sortie : L'ensemble des mots de Lyndon de taille inférieure ou égale à n
2:      soit m = "0"  // le mot constitué de la lettre 0
3:      soit E = {m}
4:      tant que m ≠ ε faire :
5:          m ← m concaténé à lui-même jusqu'à avoir un mot de taille n. La dernière occurrence
6:                de m sera éventuellement tronquée pour arriver à un mot de taille exactement n.
7:          tant que la dernière lettre de m est 1 faire :
8:              m ← m privé de sa dernière lettre
9:          si m ≠ ε alors :
10:             Remplacer la dernière lettre de m par 1
11:             E ← E ∪ {m}
12:     renvoyer E
```

Question 14. Si le mot m qui entre dans la boucle « tant que » de la ligne 4 est le mot 00111 et que $n = 9$, indiquer quel est le mot qui sera ajouté à l'ensemble E suite à l'exécution des lignes 3 à 9.

Question 15. Construire sans justifier l'ensemble des mots de Lyndon de taille au plus 4. On écrira ces mots dans l'ordre dans lequel l'algorithme précédent les produit.

Question 16. Expliquer le principe assurant la terminaison de cet algorithme.

Question 17. Majorer grossièrement sa complexité pire cas en considérant qu'ajouter ou supprimer une lettre à un mot se fait en temps constant, de même que l'ajout d'un élément à un ensemble.

Partie 3 : *Factorisations de Lyndon*

Soit $u \in \Sigma^*$. Une *factorisation de Lyndon* de u est une suite $\left(u^{(1)}, u^{(2)}, ..., u^{(n)}\right)$ de mots de Lyndon telle que $u^{(1)} \succeq u^{(2)} \succeq ... \succeq u^{(n)})$ et $u = u^{(1)}u^{(2)}...u^{(n)}$. Autrement dit, une factorisation de Lyndon de u consiste en un découpage de u en mots de Lyndon de plus en plus petits (au sens de l'ordre lexicographique). Par exemple, le mot $m_{ex} = 011100010$ admet comme factorisation de Lyndon :

$$m_{ex} = (0111)(0001)(0)$$

On admet le théorème suivant : tout mot $u \in \Sigma^*$ admet une factorisation de Lyndon. On cherche dans les questions suivantes à construire algorithmiquement une telle factorisation. Pour ce faire, on considère l'algorithme $\mathcal{A}$ suivant. On souhaite factoriser le mot u.

- Initialement, on factorise $u = u_1 u_2 ... u_n$ de taille n en $u^{(1)}, ..., u^{(n)}$ où pour tout $i \in [\![1, n]\!]$, $u^{(i)}$ est la i-ème lettre de u, à savoir u_i. Autrement dit, initialement chaque facteur est une lettre.
- Si à une étape de l'algorithme on a obtenu une factorisation $u = u^{(1)}u^{(2)}...u^{(p)}$ en mots de Lyndon, et qu'il existe $k \in [\![1, p-1]\!]$ tel que $u^{(k)} \prec u^{(k+1)}$), on modifie la factorisation obtenue en une nouvelle factorisation plus courte $v^{(1)}v^{(2)}...v^{(p-1)}$ définie par :

$$\begin{cases} v^{(j)} = u^{(j)} & \text{pour tout } j \in [\![1, k-1]\!] \\ v^{(k)} = u^{(k)}u^{(k+1)} & \\ v^{(j)} = u^{(j+1)} & \text{pour tout } j \in [\![k+1, p-1]\!] \end{cases}$$

Question 18. En utilisant l'algorithme décrit ci-dessus, donner une factorisation de Lyndon de 10100101100100.

Question 19. On admet le résultat suivant : si u et v sont deux mots de Lyndon tels que $u \prec v$ alors uv reste un mot de Lyndon. Montrer que l'algorithme $\mathcal{A}$ fournit bien une factorisation de Lyndon de son entrée.

Pour implémenter cet algorithme en OCaml, on choisit de représenter la factorisation d'un mot par un tableau de booléens avec les conventions suivantes : la case i contient true si et seulement si la i-ème lettre de u est la fin d'un des facteurs. Par exemple, la factorisation du mot m_{ex} sera représentée par le tableau :

`false`	`false`	`false`	`true`	`false`	`false`	`false`	`true`	`true`

Question 20. Écrire une fonction `extraire_chaine : string → int → int → string` telle que `extraire_chaine mot debut fin` qui renvoie la chaîne extraite de la chaîne `mot` comprise entre les caractères indicés par `debut` (inclus) et `fin` (inclus).

Question 21. Écrire une fonction `est_trie : string → bool array → (bool * int)` `bool est_trie(char* mot, bool* facto, int* defaut_de_tri)` prenant en entrée un mot non vide et une factorisation de ce mot avec les conventions ci-dessus. Si la factorisation proposée est une suite décroissante de mots, elle renverra le couple `(true, -1)`. Sinon, elle renverra le couple `(false, i)`, où `i` est l'indice de fin d'un des mots strictement plus petit que son successeur. On expliquera clairement le fonctionnement de la fonction proposée.

Question 22. En déduire une fonction `factorisation : string → bool array` qui calcule une factorisation de Lyndon de son entrée.

Partie 4 : *Mots de de Bruijn*

Dans cette partie, Σ désigne un alphabet quelconque. Un *mot de de Bruijn* d'ordre $n \in \mathbb{N}^*$ est un mot de Σ^* qui contient tous les mots de Σ^n (c'est-à-dire les mots de longueur exactement n) une et une seule fois lorsqu'on le considère circulairement.

Par exemple, $abacbbcca$ est un mot de de Bruijn d'ordre 2 sur $\{a, b, c\}$, car il contient une et une seule fois chaque mot de longueur 2 de $\{a, b, c\}$, le mot aa étant obtenu en considérant que la lettre suivant le a final est la première lettre du mot, ici a.

Question 23. Le mot $abba$ est-il un mot de de Bruijn sur $\{a, b\}$? Si oui, quel est son ordre ?

Question 24. Quelle est la longueur d'un mot de de Bruijn sur Σ en fonction de son ordre n et de $|\Sigma|$?

Les questions suivantes explorent deux méthodes permettant de construire des mots de de Bruijn. La première repose sur la notion de *graphe de de Bruijn*. Le graphe de de Bruijn d'ordre n sur un alphabet Σ à k lettres est un graphe étiqueté orienté $B(k, n)$ tel que :

- Les sommets de $B(k, n)$ sont étiquetés par les mots de Σ^n.
- Les arêtes sont données par l'ensemble $\left\{ am \xrightarrow{b} mb \mid a, b \in \Sigma \text{ et } m \in \Sigma^{n-1} \right\}$. Par exemple $001 \xrightarrow{0} 010$ est une arête dans $B(2, 3)$ lorsque l'alphabet est $\{0, 1\}$: elle va du sommet 001 vers le sommet 010 et est étiquetée par 0.

Question 25. Dessiner le graphe de de Bruijn $B(2, 3)$ lorsque l'alphabet est $\{0, 1\}$.

Question 26. Montrer que le degré sortant et le degré entrant de chaque sommet dans $B(k, n)$ est égal à k.

Question 27. En déduire le nombre d'arcs dans le graphe $B(k,n)$.

Un circuit eulérien dans un graphe orienté G est un circuit passant une et une seule fois par chaque arc de G. On admet le résultat suivant : un graphe orienté G admet un circuit eulérien si et seulement si il est connexe et pour tout sommet s de G, les degrés entrant et sortant de s sont égaux.

Question 28. Construire $B(2,2)$ et déterminer un circuit eulérien dans ce graphe. Constater que le mot obtenu par concaténation des étiquettes des arêtes parcourues lors de ce circuit est un mot de de Bruijn. Quel est son ordre ?

Ce résultat est en fait général : un circuit eulérien dans le graphe $B(k,n)$ produit un mot de de Bruijn d'ordre $n+1$ sur k symboles. En admettant ce fait :

Question 29. Montrer que pour tout alphabet Σ, et tout $n \geq 2$, il existe un mot de de Bruijn sur Σ d'ordre n.

Une autre façon de construire un mot de de Bruijn est de construire… des mots de Lyndon, dont la génération a été étudiée en partie précédemment. Si $n \in \mathbb{N}$, concaténer dans l'ordre lexicographique tous les mots de Lyndon sur Σ dont la longueur divise n produit un mot de de Bruijn d'ordre n sur Σ. Le mot de de Bruijn obtenu est même le plus petit parmi tous les mots de de Bruijn d'ordre n sur Σ.

Question 30. Calculer le plus petit mot de de Bruijn d'ordre 4 sur $\{0,1\}$.

L'accès à votre immeuble est protégé par un digicode. Pour entrer, il faut taper une séquence de n chiffres entre 0 et $k-1$. Le digicode fonctionne de la façon suivante : à chaque nouveau chiffre entré après le n-ème, il vérifie si les n derniers chiffres correspondent au code d'entrée. Par exemple, si $n=4$ et qu'on tape 021201, le digicode teste successivement les codes 0212, 2120 puis 1201.

Question 31. Combien de codes faudrait-il a priori tester pour pouvoir entrer dans l'immeuble ? Si on les essaie tous successivement, combien de chiffres faudrait-il taper dans le digicode ?

Question 32. Vu le fonctionnement du digicode, proposer une séquence astucieuse de chiffres qui garantit l'ouverture de la porte tout en diminuant le nombre total de chiffres (qu'on précisera) à entrer dans le digicode. Expliquer brièvement votre raisonnement.

⋆ ⋆ ⋆

Problème 2 : *Autour du Lemme d'Arden*

L'objectif de ce problème est d'étudier des (systèmes d') équations dont les inconnues sont des langages.

Partie 1 : *Le Lemme d'Arden*

Dans cette partie, Σ est un alphabet et $K, L \subset \Sigma^*$ sont deux langages sur cet alphabet. On y étudie l'équation (E) ci-dessous, dont l'inconnue est le langage X.

$$(E) : \quad X = KX \mid L$$

Question 33. Montrer que le langage K^*L est solution de l'équation (E).

Pour la suite de ce problème, on admet le *lemme d'Arden* :

Lemme 1 : *Lemme d'Arden.* Pour tous $K, L \subset \Sigma^*$ tels que $\varepsilon \notin K$, l'équation (E) admet pour **unique** solution K^*L.

Partie 2 : *Systèmes d'équations aux langages*

Dans cette partie, on fixe l'alphabet $\Sigma = \{a, b\}$. On note L_p le langage des mots ayant un nombre pair de b et L_i le langage des mots ayant un nombre impair de b. Si m est un mot et L un langage, on s'autorise les abus de notation « mL » pour désigner la concaténation de langages $\{m\}L$, et « $m \mid L$ » pour désigner l'union de langages $\{m\} \mid L$.

Question 34. Expliquer brièvement pourquoi (L_p, L_i) est solution du système d'équations suivant :

$$(S) : \begin{cases} L_p = aL_p \mid bL_i \mid \varepsilon & (1) \\ L_i = aL_i \mid bL_p & (2) \end{cases}$$

Question 35. À l'aide du lemme d'Arden, résoudre le système (S). En déduire une expression régulière pour L_p et L_i.
Indication : utiliser le lemme d'Arden sur (2) *puis substituer dans* (1).

Question 36. Montrer que les expressions régulières $a^*b(a \mid ba^*b)^*$ et $(a \mid ba^*b)^*ba^*$ sont équivalentes et dénotent toutes les deux le langage L_2.
Indication : on s'appuiera sur une résolution différente du système (S).

Partie 3 : *Langage reconnu par un automate*

Le but de cette partie est de décrire une méthode permettant de déterminer le langage reconnu par un automate.

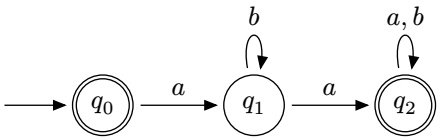
On rappelle qu'un *automate non déterministe* est la donnée de $\mathcal{A} = (Q, \Sigma, q_0, F, \delta)$ où Q un ensemble fini d'états, Σ est un alphabet, $q_0 \in Q$ est l'état initial, $F \subset Q$ est l'ensemble des états acceptants de $\mathcal{A}$ et δ est une fonction de transition sur $Q \times \Sigma$ associant à (q, c) l'état atteint en lisant la lettre c dans l'état q.

On rappelle également que le *langage reconnu* par un automate A est le langage

$$\mathcal{L}(\mathcal{A}) = \{u \in \Sigma^* \mid \delta^*(q_0, u) \in F\}$$

des mots u qui permettent de passer de l'état initial q_0 à l'un de ses états acceptants en lisant u ; en étendant au passage la fonction de transition δ des lettres aux mots. *($\delta^*(q_0, u)$ est alors l'état atteint en lisant u depuis q_0).*

On considère l'automate $\mathcal{A} = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \{q_0, q_2\}, \delta)$ ci-dessous.

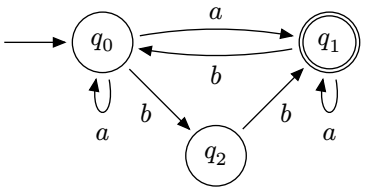


Pour $i \in \{0, 1, 2\}$, on note $L_i = \{u \in \Sigma \mid \delta^*(q_i, u) \in F\}$, qui n'est autre que le langage des mots qui font aboutir à un état acceptant à partir de l'état q_i. Remarquons qu'ainsi, déterminer $\mathcal{L}(\mathcal{A})$ revient à déterminer L_0.

Question 37. La lecture de l'automate donne des liens entre les langages L_0, L_1 et L_2. Par exemple, $L_0 = \varepsilon \mid aL_1$.

1. Déterminer sans justification un système de trois équations liant les langages L_0, L_1 et L_2
2. Résoudre ce système et en déduire le langage reconnu par $\mathcal{A}$.

Question 38. En utilisant une méthode similaire, déterminer le langage reconnu par l'automate suivant :



Partie 4 : *Les langages reconnus sont réguliers*

L'objectif de cette partie est de prouver le résultat suivant : tout langage reconnaissable par un automate est régulier.

Question 39. Montrer par récurrence sur $n \in \mathbb{N}^*$ le théorème suivant :
Soit $\left(K_{i,j}\right)_{i,j \in [\![0,n-1]\!]}$ un n^2-uplet de langages sur un alphabet Σ dont aucun ne contient ε. Soit $(L_0, ..., L_{n-1})$ un n-uplet de langages quelconques. Alors le système d'équations :

$$X_i = \left(\bigcup_{j=0}^{n-1} K_{i,j} X_j \right) \mid L_i \quad \text{pour } i \in [\![0, n-1]\!]$$

d'inconnues les langages $(X_0, ..., X_{n-1})$ admet une unique solution. De plus si les langages $K_{i,j}$ et L_i sont tous réguliers alors les composantes X_i de cette unique solution sont également des langages réguliers.

Question 40. Soit L un langage reconnaissable par un automate. En s'inspirant de la partie 3, montrer que L est l'une des composantes d'une solution d'un système d'équations aux langages qu'on déterminera et en déduire que L est nécessairement régulier.

⋆ ⋆ ⋆

Problème 3 : Logique et calcul des propositions

Dans un futur lointain, l'espèce humaine a découvert une autre espèce consciente. L'étude de cette espèce a permis de découvrir qu'elle est capable de percevoir si quelqu'un dit la vérité ou un mensonge. Les membres de cette espèce respectent les règles de politesse suivantes lors des discussions au sein d'un groupe : « Les orateurs doivent rester constants au cours d'une discussion : soit ils disent toujours la vérité, soit ils mentent toujours. De plus, si un orateur dit la vérité alors l'orateur suivant doit également dire la vérité. Si le sujet de la discussion change, les orateurs sont libres de changer leurs comportements. ».
Vous assistez à une discussion sur les moyens d'attaque et de défense que peut posséder la faune de cette planète entre trois membres de cette espèce que nous appellerons A, B et C.

A : « Le kjalt peut avoir un dard ou des griffes. »

B : « Non, il n'a pas de dard. »

C : « Il a des pinces et des griffes. »

Nous noterons D, G et P les variables propositionnelles associées au fait qu'un kjalt possède respectivement un dard, des griffes et des pinces.
Nous noterons A_1, B_1 et C_1 les formules propositionnelles associées aux déclarations de A, B et C dans cette première discussion.
C quitte le groupe et la discussion change de sujet pour parler de la flore de la planète.

A : « Un lyop peut être de couleur mauve mais pas de couleur jaune. »

B : « Il ne peut pas être de couleur verte. »

A : « Il ne peut être de couleur verte que s'il peut être de couleur jaune. »

Nous noterons J, M et V les variables propositionnelles associées au fait qu'un lyop peut être respectivement de couleur jaune, mauve et verte.
Nous noterons A_2, A_3 et B_2 les formules propositionnelles associées aux déclarations de A et B dans cette seconde discussion.

Question 41. *Représenter les règles de politesse appliquées à la première discussion sous la forme d'une formule du calcul des propositions dépendant des formules A_1, B_1 et C_1.*

Question 42. *Représenter les informations données par les participants de la première discussion sous la forme de formules du calcul des propositions A_1, B_1 et C_1 dépendant des variables D, G et P.*

Question 43. *En utilisant le calcul des propositions (résolution avec les formules de De Morgan), déterminer le (ou les) moyen(s) d'attaque et de défense que peut posséder un kjalt.*

Question 44. *Représenter les règles de politesse appliquées à la seconde discussion sous la forme d'une formule du calcul des propositions dépendant des formules A_2, A_3 et B_2.*

Question 45. *Représenter les informations données par les participants lors de la seconde discussion sous la forme de trois formules du calcul des propositions A_2, A_3 et B_2 dépendant des variables J, M et V.*

Question 46. *En utilisant le calcul des propositions (résolution avec les tables de vérité), déterminer la (ou les) couleur(s) possible(s) pour un lyop.*