RICM3 – Automates et Grammaires

Durée : 2h00, sans documents.

- Tous les appareils électroniques sont interdits à l'exception des montres mécaniques
- Le barème est donné à titre indicatif
- Le sujet comporte 5 exercices indépendants.
- Le sujet est sur 25.5 mais il suffit d'avoir 20 pour avoir la note maximale, ce qui vous donne le choix et vous permet de faire l'impasse sur certaines questions.

Exercice 1 : Questions de cours (20 min)

Q1. Pour un alphabet Σ donné et une longueur de mot ℓ donnée, est-il possible de constuire un automate (à nombre) d'états fini que reconnaît les mots de longueur ℓ ? SI OUI *expliquez le principe de construction de l'automate.* SI NON *expliquez pourquoi cela est impossible.*

Q2. Décrivez par une phrase en français le langage correspondant à l'expression régulière $(a|b)^* \cdot c$. Est-il fini ou inifini ?

Q3. **Complétez (répondez sur votre copie)**

(1) $\{a, b\} \cdot \{c, d\} =$

(2) $\{a, b\} \cdot \{a, e\} =$

(3) $\{a\}^* \cdot \{\} =$

(4) $\{\}^* \cdot \{a\} =$

(5) $\{a, \epsilon\} \cdot \{a\}^* =$

Q4. Comment savoir si un automate (à nombre) d'états fini est déterministe ou non-déterministe ?

Q5. Est-il possible qu'un automate (à nombre) d'états fini reconnaisse un langage infini ? SI OUI *donnez un exemple.* SI NON *expliquez pourquoi cela est impossible.*

Q6. **Complétez (répondez sur votre copie)** Deux automates sont équivalents *si et seulement si ...*

Q7. Les automates (à nombre) d'états fini déterministes sont-ils capables de reconnaitre les mêmes langages que les automates (à nombre) d'états fini non-déterministes ? *Justifiez votre réponse.*

Exercice 2 : Définition des identificateurs par différence d'expressions régulières (20 min)

5.5 pt

Dans la plupart des langages de programmation les noms de variables (appelés identificateurs) doivent respecter les contraintes suivantes :

1. les identificateurs sont formés de lettres (`'a'`,...,`'z'`), de chiffres (`'0'`,...,`'9'`), et du caractère souligné `'_'`
2. un identificateur doit commencer par une lettre (`'a'`, ..., `'z'`) ou le caractère `'_'`

   Exemple : `x` et `_y` sont des identificateurs, `1_x` n'en est pas un.
3. un identificateur ne doit pas se terminer par un souligné `'_'`

   Exemple : `x_1` est un identificateur valide, `x_` n'en est pas un.
4. un identificateur doit contenir au moins une lettre

   Exemple : `_1_x` est un identificateur valide, `_1_2` n'en est pas un.
5. un identificateur ne doit pas contenir deux soulignés consécutifs

   Exemple : `_1_x` est un identificateur valide, `__x` n'en est pas un.

Le but de cet exercice est de décrire les identificateurs valides par la différence de deux expressions régulières, $e_1 \setminus e_2$. Autrement dit, un identificateur sera valide s'il est reconnu par e_1 et rejetté par e_2. L'intérêt de cette description c'est qu'il est difficile d'écrire directement l'expression régulière qui tient compte de toutes les contraintes alors qu'il est facile d'écrire une expression régulière e_1 qui traite une partie des contraintes et de corriger ce qu'elle accepte en trop par une expression régulière e_2, elle aussi simple à écrire.

0.25 pt

Q8. Donnez les expressions régulières (e_l, e_c, Σ) qui reconnaissent respectivement une lettre, un chiffre, et l'alphabet considéré dans cet exercice.

0.75 pt

Q9. Donnez une expression régulière e_1 qui reconnaît les mots qui respectent les contraintes 1,2,3.

0.75 pt

Q10. Dessinez l'automate A_1 correspondant à e_1.

0.25 pt

Q11. Donnez deux mots de longueur 4 qui ne satisfont pas la contraintes 4 et deux mots de longueur 4 qui ne satisfont pas la contrainte 5.

0.75 pt

Q12. Donnez une expression régulière e_2 qui accepte les mots qui respectent la contrainte 1 mais pas la contrainte 4, ni la contrainte 5 ; de sorte que $e_1 \setminus e_2$ correspondent exactement aux identificateurs valides, c'est-à-dire aux mots qui respectent les contraintes 1,2,3,4,5.

0.75 pt

Q13. Dessinez l'automate A_2 correspondant à e_2.

Transformation en une expression régulière sans opérateur «\» Sur des ensembles E et F, l'opération «privé de», notée $E \setminus F$, correspond à $E \cap \overline{F}$. Sachant cela, on peut définir le langage reconnu par $e_1 \setminus e_2$:

$$\mathscr{L}(e_1 \setminus e_2) = \mathscr{L}(e_1) \cap \overline{\mathscr{L}(e_2)}$$

2 pt

Q14. Décrivez les (≥ 6) étapes qui permettent, à partir des automates A_1 et A_2, d'obtenir une expression régulière équivalente à $e_1 \setminus e_2$ mais qui n'utilise plus l'opérateur «\». **On vous demande ni calcul, ni algorithme, mais une description précise, en français, de la méthode à suivre en faisant référence à des algorithmes vus en cours.**

Exercice 3 : Minimisation (20 min)

3.5 pt

On considère l'automate suivant :

A	${}^i1^a$	2	3	4	5	6	7	8^a	9
a	4	9	9	8	8	1	9	6	6
b	3	2	2	3	7	2	7	2	2
c	1	4	4	4	6	5	5	8	8

0.5 pt

Q15. Le langage reconnu par l'automate ci-dessus est-il vide ? **Justifiez votre réponse.**

2 pt

Q16. Minimisez l'automate en justifiant chaque étape de l'algorithme. Les $\frac{2}{3}$ des points sont attribués aux justifications.

1 pt

Q17. Dessinez l'automate minimisé.

6 pt

Exercice 4 : Grammaire des listes de diffusion (30 min)

Les adresses email sont formées d'un nom d'utilisateur et d'un nom de domaine séparées par `'@'`. Les noms d'utilisateur et de domaines sont formés d'une séquence d'identificateurs séparés par des `'.'`. Les identificateurs sont des mots comportant des lettres, des chiffres et des `'_'`.

> **Indication :** Dans cet exercice on suppose que les identificateurs sont déjà définis par $Ident$. Vous pouvez donc l'utiliser.

Les adresses email sont définies par l'expression régulière

$$email \stackrel{def}{=} Ident \cdot (\texttt{'.'} \cdot Ident)^* \cdot \texttt{'@'} \cdot Ident \cdot (\texttt{'.'} \cdot Ident)^*$$

Ne pas confondre «·» qui est l'opération de concaténation avec le caractère `'.'` qui fait partie de l'alphabet Σ.

0.25 pt

Q18. Donnez deux mots qui appartiennent à $\mathscr{L}(email)$ et deux mots qui n'appartiennent pas $\mathscr{L}(email)$

1 pt

Q19. Donnez une grammaire avec pour germe le non-terminal Email qui reconnaît le même langage que l'expression régulière *email*.

0.5 pt

Q20. Complétez la grammaire précédente avec un non-terminal Diff afin qu'elle engendre les listes de diffusion définies comme séquence (éventuellement vide) d'adresses email séparées par des virgule `','` et terminée par une virgule `','`.

1.5 pt

Q21. Complétez la grammaire précédente avec un attribut $n \in \mathbb{N}$ afin qu'elle retourne le nombre d'adresse dans la liste de diffusion.

1.5 pt

Q22. Modifiez la grammaire pour lui ajouter un attribut *fréquence* : $Liste(Ident \times \mathbb{N})$ afin qu'elle retourne la fréquence de chaque nom d'hébergeur dans la liste de diffusion.
La partie hébergeur d'un domaine est son premier identificateur.

> **Exemple :** les domaines gmail.com et gmail.fr ont le même hébergeur : gmail

1.25 pt

Q23. Donnez le code Caml de la fonction qui permet d'incrémenter la fréquence d'un hébergeur.

Exercice 5 : Modélisation à l'aide d'automates – les «*fuzzy password* »

7 pt

(30 min)

L'objectif de cet exercice est de proposer une nouvelle forme de mot de passe basée sur les automates. Un «*fuzzy password* » est défini par un code secret, **4137** dans l'exemple, et pour le rendre imprévisible (*fuzzy*) on permet de mêler des symboles quelconque au code secret.
On considère l'alphabet $\Sigma = \mathcal{L} \cup \mathcal{C}$ constitué des lettres $\mathcal{L} = \{a, ..., z\}$ et des chiffres $\mathcal{C} = \{0, ..., 9\}$.

0.25 pt

Q24. On considère le langage L_1 formé des mots sur l'alphabet Σ qui contiennent les chiffres du code **4137** dans cet ordre. Autrement dit, les mots de L_1 sont de la forme ...**4**...**1**...**3**...**7**....
Donnez trois mots qui appartiennent à L_1 et trois mots qui n'appartiennent pas à L_1.

0.75 pt

Q25. Dessinez un automate (à nombre) d'états fini *non-déterministe* A_1 qui reconnaît L_1.

Résistance d'un tel *fuzzy password* Ce mot de passe est très facile à casser [1] : il suffit d'essayer toutes les symboles de l'alphabet autant de fois qu'il y a de symboles dans le code **4137**. Pour le rendre plus résistant aux attaques, on décide de limiter à 12 le nombre de symboles qu'on peut taper.

1.25 pt

Q26. Expliquez comment construire, à partir de l'automate A_1 de la question précédente, un automate A_2 qui reconnaît le langage L_2 constitué des mots de L_1 de longueur ≤ 12. **On ne vous demande pas de construire l'automate mais juste de décrire la méthode pour le construire.**

0.75 pt

Q27. Donnez une borne supérieure sur le nombre d'états de l'automate A_2. **Justifiez votre réponse**.

Amélioration Afin d'empêcher un attaquant d'essayer systématiquement tous les symboles de Σ, on souhaite se restreindre au langage L_3 des mots sur Σ qui ne contiennent pas plusieurs occurences d'un même symbole de Σ.

> **Exemple :** $aa... \notin L_3$ car il contient deux occurences de a, $01234567890... \notin L_3$ est rejeté puisqu'il contient deux occurences de 0

1.5 pt

Q28. Expliquez comment construire un *automate (à nombre) d'états fini déterministe* A_3 qui reconnaît le langage L_3. **On ne vous demande pas l'automate mais la méthode de contruction**.

> **Indication :** Vous pourrez nommer les états de A_3 par les symboles qu'on a encore le droit de taper.

0.5 pt

Q29. Donnez le nombre exact d'états de l'automate A_3. **Justifiez votre réponse**.

0.5 pt

Q30. Expliquez comment construire un automate A_4 qui reconnaît les mots de passe valide de L_1 qui sont sans occurence multiple de symboles de Σ. **Justifiez votre réponse.**

Autre proposition de *fuzzy password* On considère à présent un autre type d'automate qui permet de glisser le code secret **4137** choisi par l'utilisateur au milieu de lettres qui semblent aléatoires mais qui doivent en réalité respecter des contraintes connues seulement de l'utilisateur.

Prenons un exemple, on s'intéresse au langage $L_5 = \{l.\mathbf{4}.c.\mathbf{1}.c.\mathbf{3}.l.\mathbf{7} \mid l \in \mathcal{L}, c \in \mathcal{C}\}$

Autrement dit, les mots de passe valides contiennent le code secret avec en plus une lettre l et un chiffre c que l'utilisateur choisira librement au moment de rentrer son mot de passe et qu'il doit repéter au bon emplacement dans le mot de passe. L'utilisateur est bien sûr le seul à savoir quand placer ces deux symboles libres.

1.5 pt

Q31. Dessinez un automate A_5 qui reconnaît le langage L_5

1. 01234567890123456789 casse le mot de passe puisque le 012**3**45678901**2**3456**7** est accepté