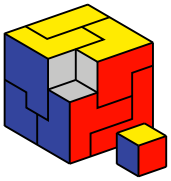
Prologin 2022



Concours national d'informatique

Épreuve écrite d'algorithmique

Fort de café

Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre épreuve régionale. Sa durée est de 3 heures. Par la suite, vous passerez un entretien (20 minutes) et une épreuve de programmation sur machine (4 heures).

Conseils

— Lisez bien tout le sujet avant de commencer.
— **Soignez la présentation** de votre copie.
— N'hésitez pas à poser des questions.
— Si vous avez fini en avance, relisez bien.
— N'oubliez pas de passer une bonne journée.

Remarques

— Le barème est donné à titre indicatif uniquement.
— Indiquez lisiblement vos nom et prénom, la ville où vous passez l'épreuve et la date en haut de votre copie.
— Tous les langages sont autorisés, veuillez néanmoins préciser celui que vous utilisez.
— Ce sont des humains qui lisent vos copies : laissez une marge, aérez votre code, ajoutez des commentaires (**seulement** lorsqu'ils sont nécessaires) et évitez au maximum les fautes d'orthographe.
— Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.
— Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le-nous, nous pouvons ajouter des questions plus difficiles.

1 Mise en bouche

Dans cette première partie d'introduction on explique des concepts utiles pour la résolution des problèmes finaux, en deuxième partie de ce sujet.

On s'intéresse aux problèmes de reconnaissance des langages, et en particulier des langages dits « rationnels ».

1.1 Alphabet, langage...

On défini un alphabet (noté le plus souvent Σ) comme un ensemble fini d'élements appelés « lettres ». La concaténation [1] permet de former, à partir des lettres de ce langage, des mots qui sont simplement des suites de lettres. Le mot qui ne contient aucune lettre est appelé « mot vide » et est noté ε.

L'ensemble des mots [2] sur un alphabet Σ est noté Σ^*. Il contient tous les mots que l'on peut former à partir des lettres de l'alphabet Σ, ainsi que le mot vide. Un tel ensemble est appelé un langage.

Par exemple, si l'on considère comme alphabet $\Sigma = \{0, 1\}$.

Alors le langage Σ^* contient 0 et 1 mais aussi 101010, 1111, 000, ε etc...

1.2 Automates

La question centrale autour des langages rationnels est d'être en mesure d'écrire un programme qui vérifie si un mot appartient à un langage donné. Par exemple je peux décrire mon langage comme l'ensemble des mots sur l'alphabet latin qui terminent par un a. Afin de répondre à cette question d'appartenance, on décrit des modèles de calcul bien spécifiques qui sont les **automates**.

Ici on ne traitera que la notion d'automate fini (machines avec un nombre fini d'états) et déterministes [3].

1.2.1 Automate fini déterministe

Un **automate fini déterministe** est un quintuplet $A = (\Sigma, Q, \delta, q_0, F)$ où :

- Σ est un alphabet
- Q est un ensemble fini d'états
- δ est une fonction de $Q \times \Sigma \longrightarrow Q$ appelée fonction de transition. Cette fonction, à partir d'un état et d'une lettre permet de donner, s'il existe, l'état suivant dans l'automate. Par exemple, dans l'automate en Figure 1 on a $\delta(q_0, 1) = q_1$.
- q_0 l'état initial de l'automate.
- F, une partie de Q appelée états finaux.

Globalement, cette idée est plus facilement visualisée par un graphe qui permet de représenter cet automate. Par exemple, si je considère l'alphabet $\Sigma = \{0, 1\}$ et que je veux décrire un automate qui accepte seulement les mots qui se terminent par un 1 [4] alors on aurait un automate qui ressemblerait à ce qu'on peut voir sur la Figure 1.

La flèche indique l'état d'entrée de l'automate, les transitions sont décrites par les arcs du graphe. Par exemple quand on se situe dans l'état q_0 et que l'on lit un 1, on passe à l'état q_1. Cet état est entouré deux fois ce qui siginifie qu'il est final. Ainsi, si l'on se retrouve sur un état final lorsqu'on a terminé de lire le mot, celui-ci est accepté, sinon il est refusé. Notez qu'il est

1. opération qui permet de coller deux lettres les unes à la suite des autres, notée $\cdot$
2. qui n'est pas nécessairement fini
3. cette notion s'éclairciera plus tard
4. les nombres binaires impairs par exemple

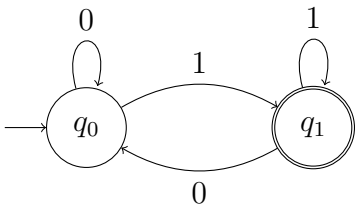


FIGURE 1 – Automate acceptant les mots de $\{0, 1\}$ qui finissent par un 1

possible également que la fonction δ soit une fonction partielle [5] et à ce moment là, il faudra également ajouter comme critère de rejection d'un mot le fait qu'il n'existe pas de transitions pour une lettre au moment de la lecture. Par exemple, si je décide d'étendre mon alphabet en rajoutant une lettre tel que $\Sigma = \{0, 1, 2\}$. Si l'on garde le même automate et que l'on cherche à vérifier si le mot 210 est accepté par l'automate. Alors on se rends compte qu'il n'existe pas de transition pour 2 à l'état q_0 et on peut donc directement rejeter le mot.

1.3 Questions

On pose $\Sigma = \{a, b, c\}$ Soit A l'automate suivant :

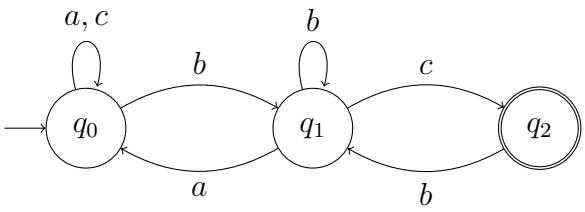


FIGURE 2 – Automate A

Question 1 (1 point)

> Donner un exemple d'un mot qui est accepté et d'un mot refusé par l'automate A.

Question 2 (2 points)

> Décrire une structure de donnée qui permetrait d'implémenter un automate fini déterministe.
>
> Si vous utilisez des sous-structures de données, faites bien attention à les préciser (vous pouvez les écrires directement dans le langage de votre choix si vous pensez cela plus pertinent).

Question 3 (2 points)

> A l'aide de la structure de donnée précédente, écrire une fonction `accept` qui prend un automate et un mot et qui renvoie un booléen. Cette fonction devra retourner `true` si l'automate passé en paramètre accepte le mot.

5. Une fonction partielle est une fonction qui n'a pas de résultat pour certaines entrées, on dit qu'elle n'est pas définie pour ces entrées là

Question 4 (1 point)

> À l'aide d'une phrase en français ou d'une expression rationnelle si vous les maîtrisez, décrire « avec les mains » le langage représenté par l'automate A.

Question 5 (2 points)

> Soit $\Sigma = \{a, b\}$. Écrire un automate qui reconnaît tous les mots qui comprennent un nombre pair de a, suivis d'un nombre impair de b.

On dit qu'un automate est « minimal » si c'est le plus petit automate[6] qui reconnaît un langage.

Question 6 (2 points)

> Cet automate, au nommage des états prêt, est-il unique ? Montrez le.

Question 7 (2 points)

> Dessinez un automate fini déterministe de votre choix qui n'est pas minimal puis minimisez le. Vous expliquerez votre processus de pensée.

2 Ticket de métro

Dans cette partie on modélise un portique de métro à l'aide d'un automate.
La machine possède les contraintes définies par le tableau suivant :

État du portique	Entrée	État suivant	Action
verrouillé	jeton	déverrouillé	Déverrouille le portique pour passer
	pousser	verrouillé	Rien
déverrouillé	jeton	déverrouillé	Rien
	pousser	verrouillé	Ferme le portique, après passage

Question 8 (1 point)

> Dessiner l'automate qui le représente. Cet automate est un peu spécial vu qu'il n'a pas vraiment d'état final et sert à lire des mots en continu. On ne va donc pas lui donner d'état final.
>
> Utilisez des noms pertinents pour nommer les états.

Question 9 (1 point)

> Énumérez l'alphabet sur lequel travaille cet automate.

6. au sens du nombre d'états

Question 10 (1 point)

> Donner l'ensemble Q des états, ainsi que l'état de départ.

Question 11 (2 points)

> Construire à partir de toutes ces données, la table de transition de l'automate.
> Une table de transition est un tableau à double entrée qui en fonction de l'état (en ligne) et d'une lettre (en colonne) permet de déterminer le prochain état de l'automate.

Question 12 (2 points)

> Utilisez cette table pour écrire un programme qui tourne en continu et qui fait le travail du portique. Vous utiliserez l'API du portique qui comporte une fonction `getInput` qui renvoie la prochaine lettre lue par le portique, et une fonction `setLocked` qui permet de verrouiller le portique à l'aide d'un booléen.

On considère maintenant que notre portique possède un digicode secret dont la valeur est le numéro du pass carmillon de son créateur, un membre de Prologin qui en avait marre de payer ses tickets de métro. Son numéro est le 4212. On aimerait que le clavier secret accepte l'ouverture du portillon si les quatre derniers chiffres tapés sont bons, sinon il ne fait rien.

Question 13 (2 points)

> Écrire un programme qui permet de faire tourner, sans automates, le portique du métro avec les nouvelles contraintes.

Question 14 (1 point)

> Dessiner le nouvel automate obtenu, pour représenter des classes de nombres vous pouvez utiliser des opérations ensemblistes pour décrire une transition.
> Par exemple, $\Sigma \setminus \{a\}$ sera empruntée pour toutes les valeurs différentes de a.

Question 15 (2 points)

> Écrire le programme qui permet de faire tourner, à l'aide de la table de transition de l'automate que l'on vient de dessiner, le portique du métro.

Question 16 (1 point)

> Comparez les deux programmes. Tirez-en une conclusion sur l'utilité ou l'inutilité des automates.

2.1 Automate fini non déterministe

Dans les automates déterministes que l'on a rencontré, on pouvait voir qu'il n'existait qu'une seule entrée possible et qu'un seul choix possible pour chaque lettre dans un état. Ainsi, on « n'hésite pas » lors qu'il faut effectuer une transition dans l'automate.

On introduit la notion de transition instantanée ou ε-transition. Cette transition est dite spontanée car elle permet à un automate de changer d'état ou de configuration spontanément, sans consommer un symbole d'entrée. De telles transitions sont définies pour de nombreux modèles d'automate, et des algorithmes d'élimination existent ou pas selon les modèles.

Ces transitions détruisent l'aspect linéaire de vérification des mots car elle peuvent nécessiter l'exploration d'un chemin qui n'aboutit à rien. Par exemple :

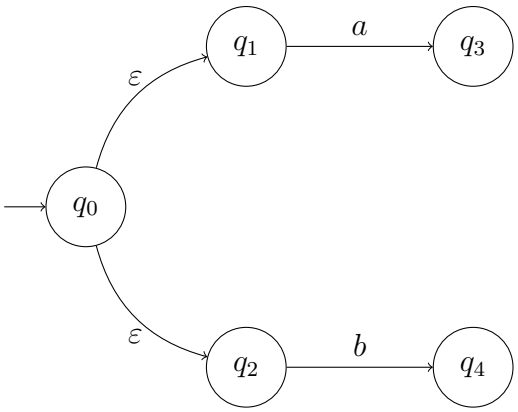


FIGURE 3 – Automate fini non déterministe

Cet automate non déterministe accepte aussi bien les mots a et b. Mais nécessite le parcours de la branche a et b pour réfuter c.

Question 17 (1 point)

> Quelle est la complexité, au pire, du parcours d'un automate fini non déterministe ? On note N le nombre d'états et T le nombre de transitions.

Question 18 (1 point)

> Sur quel principe algorithmique repose la vérification de l'acceptation d'un mot dans un automate fini non déterministe ?

Question 19 (2 points)

> Implémenter dans le langage de votre choix, la fonction `accept` précédente mais pour un automate fini non déterministe. Vous êtes libre de changer les structures de données que vous utilisez tant que vous le précisez.
>
> N'oubliez pas qu'un automate non déterministe peut posséder autant de sommets de départ que de sommets.

Question 20 (1 point)

> Quelle(s) possible(s) optimisation(s) pour rendre le temps d'exécution quasi linéaire en pratique peuvent être effectuée(s) ?

3 Pour aller plus loin

Cette section contient des questions pour aller plus loin sur les automates et qui sont particulièrement difficiles. Nous vous conseillons de terminer le sujet et d'y revenir à la fin si vous avez encore le temps.

Question 21 (2 points)

> Pour tout automate fini non déterministe, il existe un automate qui ne possède pas d'ε-transitions, proposez une méthode qui permet de retirer les ε-transitions d'un automate. [7]

Question 22 (5 points)

> Un des théorèmes fondamentaux de la théorie des langages est le théorème de Rabin-Scott qui montre l'équivalence entre la classe les automates fini déterministes et non déterministes. Ainsi, pour tout automate fini non déterministe, il existe un automate fini déterministe qui reconnaît le même langage.
>
> Démontrer ce théorème. [8]

Question 23 (3 points)

> On dit que deux mots u et v d'un langage L sont indistinguables si quelque soit $w \in L$, $u \cdot w \in L$ et $v \cdot w \in L$.
>
> Montrer que cette propriété permet de créer une relation d'équivalence sur L. [9]
>
> Que représente le nombre de classes d'équivalence d'un langage ?

Question 24 (2 points)

> Les langages dit rationels sont les langages reconnus par les automates finis. Donner une condition nécessaire et suffisante à l'aide de la relation définie précédamment pour montrer qu'un langage est rationel.

7. cette opération est appellée émondage d'un automate

8. une piste de démonstration consiste en la construction d'un tel automate à l'aide d'une construction par ensemble des parties

9. On appelle cette relation la relation de Myhill-Nerode.

4 Fort de café

Le café (de l'arabe qahwa, « boisson stimulante ») est une boisson énergisante psychotrope stimulante, obtenue à partir des graines torréfiées de diverses variétés de caféier, de l'arbuste caféier, du genre Coffea. Le café est également la ressource la plus importante d'un programmeur. Et quand la machine ne fonctionne plus, c'est la panique au local Prologin. Bien que la machine ait simplement un bouton « faire du café » il arrive souvent que son manque de message d'erreur (indiqué par un simple « ✷ ») la rende difficilement opérable. Afin de comprendre comment utiliser la cafetière, un diagramme est accrochée au mur juste à côté. Ce diagramme est une machine de Mealy, représentation du système de la cafetière que vous pouvez voir en annexe. Une action qui n'entraîne pas une erreur est représenté par «✓ » et une action qui entraîne la production d'un café est notée « ☕ ».

4.1 Machine de Mealy

Une machine de Mealy est un transducteur à états finis. C'est un automate qui prends en entrée un mot et qui produit un mot en sortie, via l'application de différentes transitions. C'est donc la donnée de 6 éléments :

- Un ensemble fini d'états noté Q
- Un état initial noté q_0, élément de Q
- Un ensemble fini de lettres A nommé alphabet d'entrée
- Un ensemble fini de lettres B nommé alphabet de sortie
- Une fonction de transition $\delta : Q \times A \longrightarrow Q$ qui permet en fonction de l'état sur lequel on se situe et d'une entrée de donner l'état suivant de la machine.
- Une fonction de transition $\lambda : Q \times A \longrightarrow B$ qui permet en fonction de l'état sur lequel on se situe et d'une entrée de retourner la lettre produite par la machine.

Prenons comme exemple une machine de Mealy qui produit à chaque fois qu'elle rencontre la séquence 01 la lettre a autrement elle produit la lettre b.

Par exemple :

- 101 produit *bba*
- 0101 produit *baba*

On a la machine suivante :

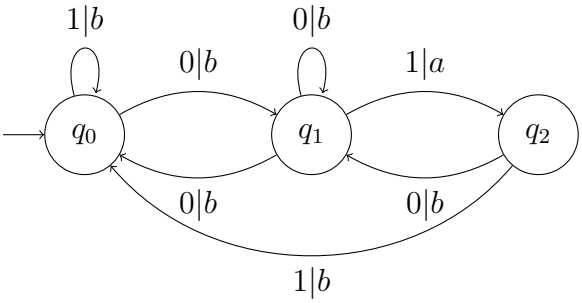


FIGURE 4 – Machine de Mealy qui produit a quand elle rencontre la séquence 01 et b sinon

4.2 Questions

Question 25 (1 point)

> Donner une séquence de mots de A permettant d'aboutir à un café. Vous noterez également quel est le mot produit par cette séquence dans la machine de Mealy [10].
>
> Pour éviter d'avoir à dessiner les tasses et les étoiles, vous pouvez utiliser le code suivante :
>
> — C pour un ☕
> — E pour une ✷
> — O pour un ✓

Question 26 (1 point)

> Décrivez l'alphabet A et B de l'automate de Mealy de notre cafetière.

Question 27 (1 point)

> Représenter la fonction de transition δ sous la forme d'un tableau à double entrée avec les éléments de Q en ligne et les éléments de A en colonne.

Question 28 (2 points)

> Décrire une structure de donnée qui permettrait d'implémenter une machine de Mealy.
>
> Si vous utilisez des sous-structures de données, faites bien attention à les préciser (vous pouvez les écrires directement dans le langage de votre choix si vous pensez cela plus pertinent).

Question 29 (1 point)

> Aidons nos programmeurs fatigués à résoudre leurs problèmes. Écrivez une fonction qui permet en cas d'erreur de connaître les raisons possibles de la panne (l'ordre n'importe pas) en renvoyant les lettres de l'alphabet qui permettrait d'accéder à un café.
>
> Par exemple, si je me situe dans l'état c de la machine, la fonction doit retourner la liste $\{pod\}$

Question 30 (1 point)

> Écrire un algorithme qui vérifie si un enchaînement d'actions sur la machine permet d'aboutir à un café. Vous réutiliserez les structures de données décrites plus tôt.

Question 31 (1 point)

> Décrire et montrer une relation d'équivalence sur les états d'une machine de Mealy.
> Pour rappel une relation d'ordre doit être :
> — Réfléxive : pour tout élément de $q \in Q$ on a $q \sim q$
> — Symétrique : pour tout élément de $(q_1, q_2) \in Q^2$ si on a $q_1 \sim q_2$ alors on a $q_2 \sim q_1$
> — Transitive : pour tout élément de $(q_1, q_2, q_3) \in Q^3$ $q_1 \sim q_2$ et $q_2 \sim q_3$ implique $q_1 \sim q_3$

10. Comme la séquence d'exemple doit produire un café, le mot produit doit se terminer par un café.

Question 32 (2 points)

> La machine de Mealy de notre cafetière n'est pas minimale. À l'aide de l'équivalence entre les états que vous avez défini à la question précédente, proposer une machine de Mealy minimale représentant la cafetière.

Question 33 (3 points)

> Proposer un algorithme de minimisation des machines de Mealy.

5 Questions bonus

N'abordez ces questions bonus qui si vous pensez que les organisateurs de Prologin ne connaissent pas l'informatique théorique ou que vous avez terminé le sujet obligatoire en vous étant relu 42 fois.

Question bonus 34 (1 point)

> Donner le pseudo du concepteur du sujet.

Question bonus 35 (2 points)

> Montrer qu'une machine de Mealy est au moins aussi expressive qu'un automate fini.

Question bonus 36 (1000 points)

> Décrire Enigma sous la forme d'une machine de Mealy, revenez dans le passé pour la donner à Alan Turing et écourter la guerre.
> Le soin et la présentation seront pris en compte.

Question bonus 37 (3 points)

> La machine de Mealy de la cafetière est extraite d'un article sur l'apprentissage actif d'automates.
> Citez cet article ainsi que son auteur (co-auteurs compris) et sa date de parution.

Question bonus 38 (1 point)

> Écrire une blague sur les automates.

Question bonus 39 (2 points)

> Décrire un autre objet du quotidien sous la forme d'une machine de Mealy.

6 Annexes

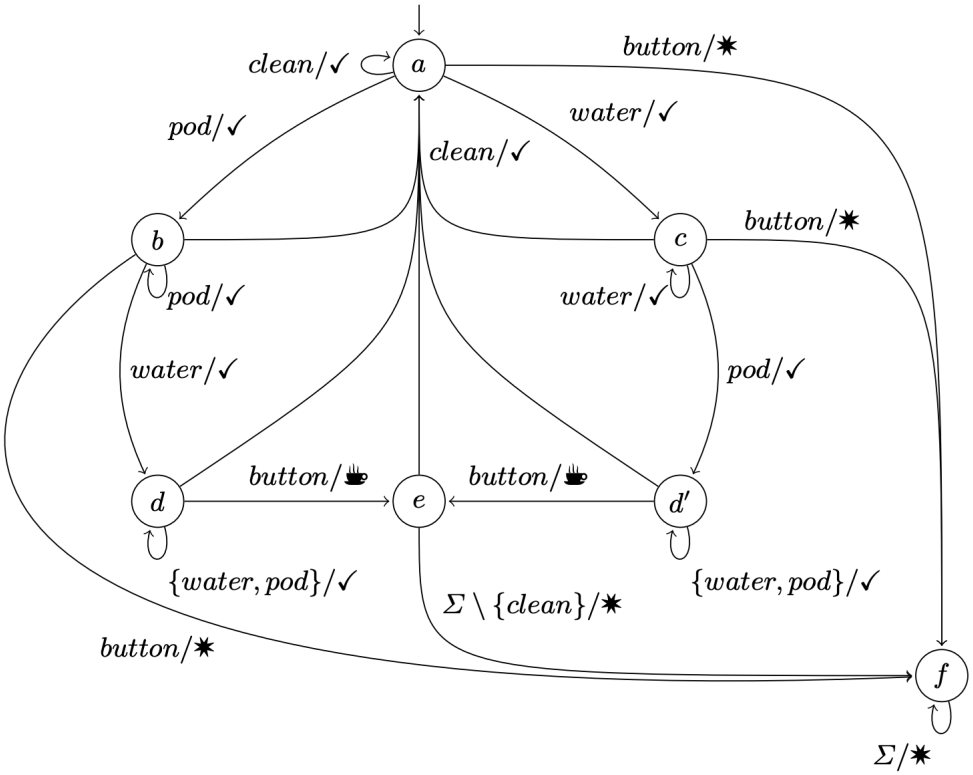


FIGURE 5 – Machine de Mealy de la cafetière du local Prologin