Commencé le	mercredi 7 juin 2023, 13:13
État	Terminé
Terminé le	mercredi 7 juin 2023, 14:24
Temps mis	1 heure 11 min
Points	22,00/22,00
Note	**26,00** sur 26,00 (**100**%)

Question 1

Non répondue

Non noté

Si une question vous semble comporter des erreurs ou imprécisions, vulgairement parlant des bugs, ne posez pas de question oralement, mais signalez-le ci-dessous en précisant :

- le numéro de la question concernée
- vos interrogations sur cette question
- éventuellement l'interprétation ou les choix faits pour votre (vos) réponse(s) à cette question.

Question 2

Correct

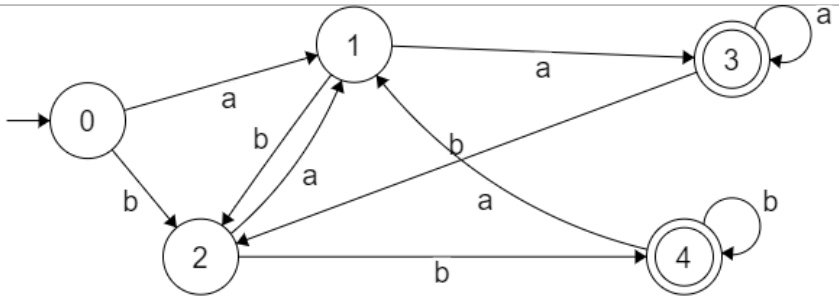
Note de 2,00 sur 2,00

Sur l'alphabet {a,b}, construire l'automate **déterministe complet minimal** qui reconnait l'ensemble L des mots de longueur au moins deux, dont les 2 dernières lettres sont les mêmes. Autrement dit L = (a+b)*(aa+bb).
(votre réponse n'est pas évaluée durant le contrôle, donc toute réponse allume vert)

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

Help Clear



	Test	Résultat attendu	Résultat obtenu	
✔	`dfa_s_is_complete`	`True`	`True`	✔
✔	`dfa_s_c._states`	`[0, 1, 2, 3, 4]`	`[0, 1, 2, 3, 4]`	✔
✔	`dfa_s_c._accept_states`	`[3, 4]`	`[3, 4]`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 0 in k))`	`(1, 2)`	`(1, 2)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 1 in k))`	`(3, 2)`	`(3, 2)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 2 in k))`	`(1, 4)`	`(1, 4)`	✔
✔	`dfa_s_c._state_transitions.get((3,'a'))`	`3`	`3`	✔
✔	`dfa_s_c._state_transitions.get((3,'b'))`	`2`	`2`	✔
✔	`dfa_s_c._state_transitions.get((4,'a'))`	`1`	`1`	✔
✔	`dfa_s_c._state_transitions.get((4,'b'))`	`4`	`4`	✔

Tous les tests ont été réussis ! ✔

Il faut bien gérer les *retours*, par exemple depuis l'état 1 (ou 3) : quand on lit un 'b', on doit aller en 2 (et pas en 0).

▸ **Montrer / masquer la solution de l'auteur de la question (Python3)**

Correct

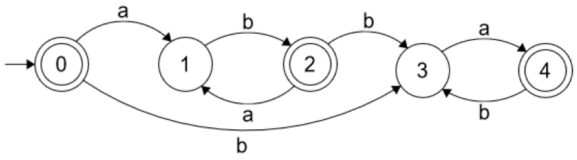
Note pour cet envoi : 2,00/2,00.

Question 3

Correct

Note de 2,00 sur 2,00

En partant de l'automate suivant :



donner l'automate **déterministe complet minimal** obtenu par application de l'algorithme vu en cours/TD.

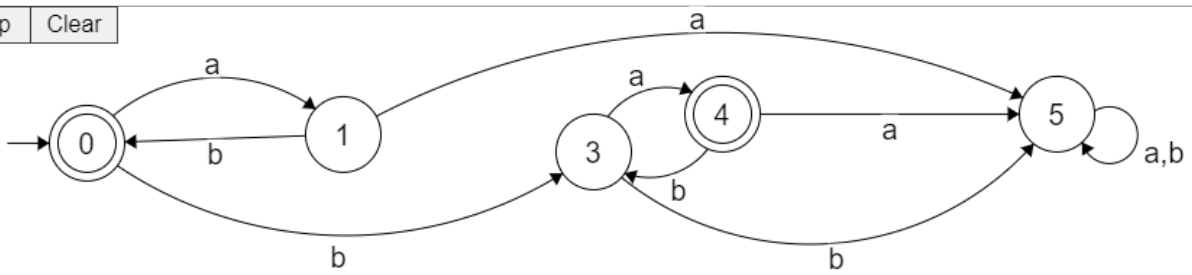
Dans votre réponse, les numéros/noms donnés aux états ne sont pas importants.

(votre réponse n'est pas évaluée durant le contrôle, donc toute réponse allume vert)

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

Help Clear



	Test	Résultat attendu	Résultat obtenu	
✔	`dfa_s_is_complete`	`True`	`True`	✔
✔	`dfa_s_c._states`	`[0, 1, 2, 3, 4]`	`[0, 1, 2, 3, 4]`	✔
✔	`dfa_s_c._accept_states`	`[0, 4]`	`[0, 4]`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 0 in k))`	`(1, 2)`	`(1, 2)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 1 in k))`	`(3, 0)`	`(3, 0)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 2 in k))`	`(4, 3)`	`(4, 3)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 3 in k))`	`(3, 3)`	`(3, 3)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 4 in k))`	`(3, 2)`	`(3, 2)`	✔

Tous les tests ont été réussis ! ✔

Il s'agit de minimiser le DFA donné.
On *voit* que les états 0 et 2 sont *équivalents* (avec 'a' depuis 0|2 on va en 1, et avec 'b' on va en 3). Et en fait, via l'algo de minimisation ce sont les 2 seuls états qui vont rester regroupés.
Le DFA donné n'est pas complet (par exemple, il n'y a pas de transition 'a' depuis 1).
Le DFA complet minimum renuméroté est :

1. **état initial 0** = regroupement {0, 2} du DFA donné

2. **état 1** = transition (0,'a') : état 1 du DFA donné

3. **état 2** = transition (0,'b') : état 3 du DFA donné

4. **état 3** = transition (1,'a') : état *poubelle* ajouté par complétion du DFA donné
5. **état 0** = transition (1,'b') : état 2 du DFA donné donc **état 0**

6. **état 4** = transition (2,'a') : état 4 du DFA donné
7. **état 3** = transition (2,'b') : état *poubelle* ajouté par complétion du DFA donné

8. **état 3** = transition (3,'a') : état *poubelle* ajouté par complétion du DFA donné
9. **état 3** = transition (3,'b') : état *poubelle* ajouté par complétion du DFA donné

10. **état 3** = transition (4,'a') : état *poubelle* ajouté par complétion du DFA donné
11. **état 2** = transition (4,'b') : état 3 du DFA donné donc **état 0**

Votre réponse est correcte si et seulement si votre DFA renuméroté selon le même principe, est celui qui vient d'être construit.

▸ **Montrer / masquer la solution de l'auteur de la question (Python3)**

Correct

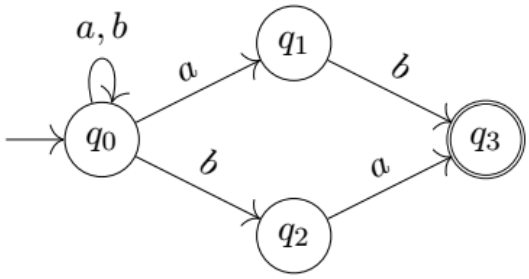
Note pour cet envoi : 2,00/2,00.

Question 4

Correct

Note de 2,00 sur 2,00

En partant de l'AFI suivant sur l'alphabet $\{a, b\}$:



(unique état acceptant: q_3)

donner l'AFD obtenu par application de l'algorithme vu en cours/TD.

(vous n'êtes pas obligé de minimiser votre AFD).

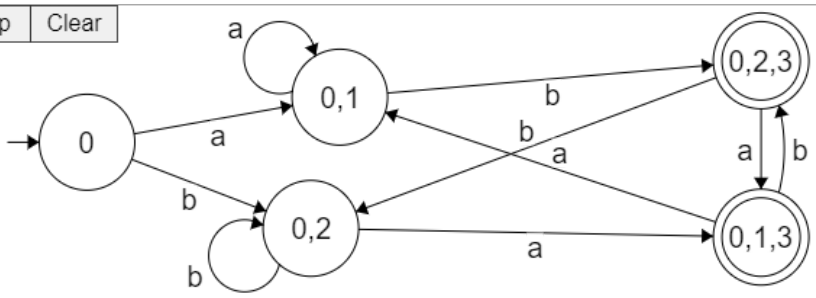
Dans votre réponse, les numéros/noms donnés aux états ne sont pas importants.

(votre réponse n'est pas évaluée durant le contrôle, donc toute réponse allume vert)

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

Help Clear



	Test	Résultat attendu	Résultat obtenu	
✔	`dfa_s_is_complete`	`True`	`True`	✔
✔	`dfa_s_c._states`	`[0, 1, 2, 3, 4]`	`[0, 1, 2, 3, 4]`	✔
✔	`dfa_s_c._accept_states`	`[3, 4]`	`[3, 4]`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 0 in k))`	`(1, 2)`	`(1, 2)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 1 in k))`	`(1, 3)`	`(1, 3)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 2 in k))`	`(4, 2)`	`(4, 2)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 3 in k))`	`(4, 2)`	`(4, 2)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 3 in k))`	`(4, 2)`	`(4, 2)`	✔

Tous les tests ont été réussis ! ✔

Ici il s'agit de donner le résultat de l'algorithme de déterminisation de l'AFI donné.

Chacun des états du DFA obtenu, est un sous-ensemble de l'AFI donné, ainsi :

1. **état 0** initial = ensemble des états initiaux de l'AFI donné : {0}

2. **état 1** = ensemble des états atteints par les différentes transitions (0,'a') dans l'AFI : {0,1}
3. **état 0 =** ensemble des états atteints par les différentes transitions (0,'b') dans l'AFI : {0}

4. **état 1 =** ensemble des états atteints par les différentes transitions ({0,1},'a') dans l'AFI : {0,1}
5. **état 2** = ensemble des états atteints par les différentes transitions ({0,1},'b') dans l'AFI : {0,2}

6. **état 3** = ensemble des états atteints par les différentes transitions ({0,2},'a') dans l'AFI : {0,1,3}
7. **état 0** = ensemble des états atteints par les différentes transitions ({0,2},'b') dans l'AFI : {0}

8. **état 1 =** ensemble des états atteints par les différentes transitions ({0,1,3},'a') dans l'AFI : {0,1}
9. **état 2** = ensemble des états atteints par les différentes transitions ({0,1,3},'b') dans l'AFI : {0,2}

Votre réponse est correcte si et seulement si votre DFA renuméroté selon le même principe, est celui qui vient d'être construit.
On peut noter que ce DFA est **LE DFA complet minimum** qui reconnait le langage reconnu par l'AFI donné, à savoir (a+b)*aba (expression régulière qui se lit, sur cet exemple, plus naturellement sur l'AFI que sur le DFA).

▸ **Montrer / masquer la solution de l'auteur de la question (Python3)**

Correct

Note pour cet envoi : 2,00/2,00.

Question 5

Correct

Note de 2,00 sur 2,00

Donner l'automate **déterministe complet minimal** qui reconnait le langage (ab*a+b)*

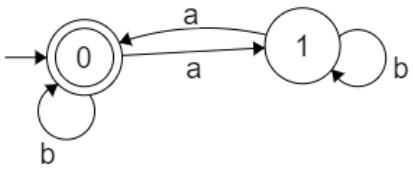
Dans votre réponse, les numéros/noms donnés aux états ne sont pas importants.

(votre réponse n'est pas évaluée durant le contrôle, donc toute réponse allume vert)

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse

Help Clear



	Test	Résultat attendu	Résultat obtenu	
✔	`dfa_s_is_complete`	`True`	`True`	✔
✔	`dfa_s_c._states`	`[0, 1]`	`[0, 1]`	✔
✔	`dfa_s_c._accept_states`	`[0]`	`[0]`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 0 in k))`	`(1, 0)`	`(1, 0)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 1 in k))`	`(0, 1)`	`(0, 1)`	✔

Tous les tests ont été réussis ! ✔

Il s'agit ici de donner l'unique DFA complet minimum à partir de l'expression régulière (ab*a+b)* .
Sans chercher à appliquer l'algorithme vu en cours/TD (ce qui est bien sûr possible), on *voit* que (ab*a+b)* est l'itération (*) de 2 boucles :

1. ab*a d'une part : une transition 'a', une boucle 'b' sur l'état atteint, un *retour* 'a' vers l'état de départ
2. b d'autre part.

Ce qui donne le DFA ci-dessous, ce DFA est *clairement* complet (de chacun des 2 états, il y a une transition 'a' et une transition 'b') et minimal (moins que 2 états, c'est 1 état et le DFA complet ayant 1 état reconnait (avec {a,b} pour alphabet) (a+b)* si l'unique état est état d'acceptation, et l'ensemble vide sinon).

▸ **Montrer / masquer la solution de l'auteur de la question (Python3)**

Correct

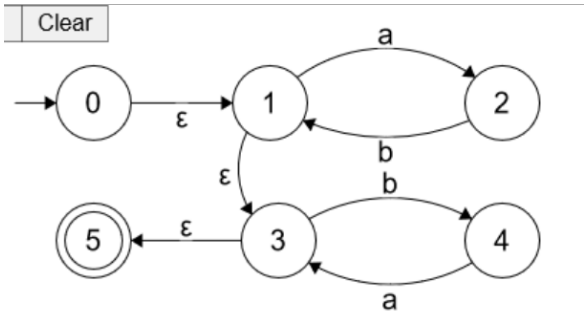
Note pour cet envoi : 2,00/2,00.

Question 6

Correct

Note de 2,00 sur 2,00

En partant de l'automate avec ε-transitions, suivant :

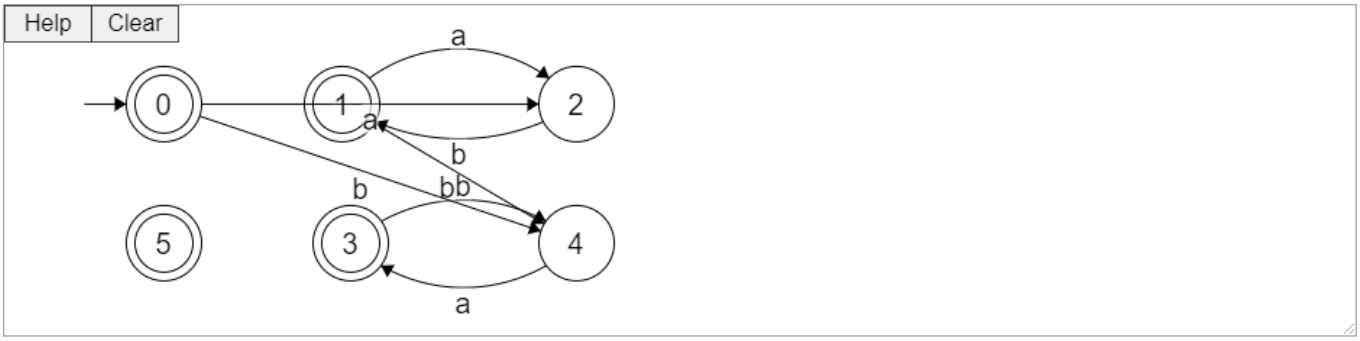


donner l'automate obtenu par application de l'algorithme de suppression des ε-transitions vu en cours/TD.

(votre réponse n'est pas évaluée durant le contrôle, donc toute réponse allume vert)

Réponse : (régime de pénalités : 0 %)

Réinitialiser la réponse



	Test	Résultat attendu	Résultat obtenu	
✔	`dfa_s_c._states`	`[0, 1, 2, 3, 4, 5]`	`[0, 1, 2, 3, 4, 5]`	✔
✔	`dfa_s_c._accept_states`	`[0, 4, 5]`	`[0, 4, 5]`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 0 in k))`	`(1, 2)`	`(1, 2)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 1 in k))`	`(3, 4)`	`(3, 4)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 2 in k))`	`(5, 3)`	`(5, 3)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 3 in k))`	`(3, 3)`	`(3, 3)`	✔
✔	`tuple((dfa_s_c._state_transitions.get(k) for k in dfa_s_c._state_transitions if 4 in k))`	`(1, 2)`	`(1, 2)`	✔

Tous les tests ont été réussis ! ✔

Pour cet algorithme de suppression des ε-transitions, on prend tout état (l'ordre dans lequel ces états sont traités n'a pas d'impact sur le résultat obtenu) ayant au moins une ε-transition sortante :

1. état 0 : on atteint 1 par une ε-transition, et il y a une transition (1,'a'):2, donc on ajoute la transition **(0,'a'):2**
2. état 0 : on atteint 3 par une suite de 2 ε-transitions, et il y a une transition (3,'b'):4, donc on ajoute la transition **(0,'b'):4**
3. état 0 : on atteint 5 par une suite de 3 ε-transitions, mais aucune transition ne sort de 5, donc on n'ajoute pas de transition, mais comme 5 est état d'acceptation, **0 devient état d'acceptation**
4. on a fini pour 0, donc **on supprime l'ε-transition sortant de 0**
5. état 1 : on atteint 3 par une ε-transition, et il y a une transition (3,'b'):4, donc on ajoute la transition **(1,'b'):4**

6. état 1 : on atteint 5 par une suite de 2 ε-transitions, mais aucune transition ne sort de 5, donc on n'ajoute pas de transition, mais comme 5 est état d'acceptation, **1 devient état d'acceptation**
7. on a fini pour 1, donc **on supprime l'ε-transition sortant de 1**
8. état 3 : on atteint 5 par une ε-transition, mais aucune transition ne sort de 5, donc on n'ajoute pas de transition, mais comme 5 est état d'acceptation, **3 devient état d'acceptation**
9. on a fini pour 3, donc **on supprime l'ε-transition sortant de 3**

L'état 5 devient inaccessible, donc sera supprimé par l'algo. de renumérotation (si vous ne l'avez pas supprimé dans votre réponse) qui ajoutera un *état poubelle* (qui aura le numéro 3) car l'automate obtenu par l'algo. est un DFA qui (ici) n'est pas complet (mais si vous l'avez complété la réponse est correcte).

▸ **Montrer / masquer la solution de l'auteur de la question (Python3)**

Correct

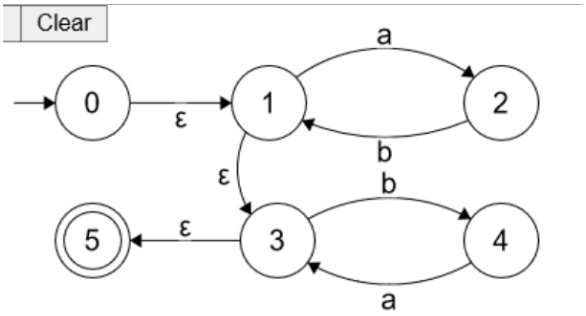
Note pour cet envoi : 2,00/2,00.

Question 7

Correct

Note de 2,00 sur 2,00

En partant de l'automate suivant :



donner l'expression régulière obtenue par application de l'algorithme *Transformer un AFD en expression régulière* vu en cours/TD. Ici l'automate de départ n'est pas un AFD (il contient des ε-transitions), mais ça ne change rien à l'application de l'algorithme.

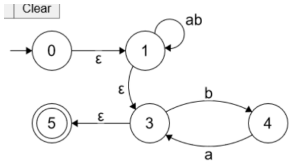
Réponse : (ab)*(ba)*

(ab) * ba

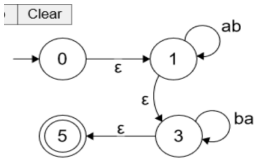
Mots mal placés

Il s'agit ici de passer d'un automate à une expression régulière.
On *voit* que pour aller de 0 à 5, il faut aller en 1 (en ne lisant rien), ensuite boucler sur 1 avec ab, puis aller en 3 (en ne lisant rien) puis boucler sur 3 avec ba, puis aller en 5 (en ne lisant rien). Ce qui donne le langage (ab)*(ba)*

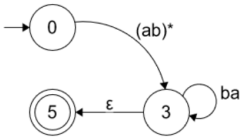
En appliquant l'algorithme, on peut traiter d'abord les états *les plus simples* (c'est-à-dire avec le moins de transitions incidentes), donc on peut supprimer 2 (sachant que 0 et 5 ne sont pas à supprimer, le résultat final étant une unique transition étiqueté par une exp. rég. du langage) ce qui donne :



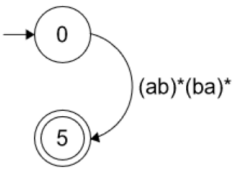
puis on supprime 4 :



puis on supprime 1 :



où ε(ab)*ε a été remplacé par (ab)* (on n'écrit pas ε quand il est inutile)
et enfin on supprime 3 :



D'où le résultat : (ab)*(ba)*

La meilleure réponse est :
(ab)*(ba)*

Question 8

Correct

Note de 1,00 sur 1,00

Le nombre d'états initiaux d'un automate fini **déterministe** est :
(veuillez saisir exactement une réponse, cette question pourra donner des pouillèmes de points négatifs)

- 0 ou 1 (c'est-à-dire que certains AFD, mais pas tous, ont 0 état initial et les autres AFD ont exactement 1 état initial)
- exactement 1 (c'est-à-dire que tout AFD a exactement 1 état initial) ✔
- au moins 1 (c'est-à-dire que certains AFD, mais pas tous, ont 1 état initial et les autres AFD ont plusieurs états initiaux)
- Aucune des autres réponses

Votre réponse est correcte.

La réponse correcte est :
exactement 1 (c'est-à-dire que tout AFD a exactement 1 état initial)

Question 9

Correct

Note de 1,00 sur 1,00

Le nombre d'états d'acceptation d'un automate fini **déterministe** est :
(veuillez saisir exactement une réponse, cette question pourra donner des pouillèmes de points négatifs)

- 0 ou 1 (c'est-à-dire que certains AFD, mais pas tous, ont 0 état d'acceptation et les autres AFD ont exactement 1 état d'acceptation)
- exactement 1 (c'est-à-dire que tout AFD a exactement 1 état d'acceptation)
- au moins 1 (c'est-à-dire que certains AFD, mais pas tous, ont 1 état d'acceptation et les autres AFD ont plusieurs états d'acceptation) ✔
- Aucune des autres réponses

Votre réponse est correcte.

Un DFA peut avoir 1 ou plusieurs états d'acceptation, mais il peut aussi n'avoir aucun état d'acceptation (pour reconnaître l'ensemble vide). Donc la bonne réponse est *Aucune des autres réponses* (il faut bien que ça arrive de temps en temps). Mais comme le cas où il n'y a aucun état d'acceptation est quand même *atypique*, la réponse (la plus souvent donnée) *au moins 1* a été comptée juste.

Les réponses correctes sont : au moins 1 (c'est-à-dire que certains AFD, mais pas tous, ont 1 état d'acceptation et les autres AFD ont plusieurs états d'acceptation),
Aucune des autres réponses

Question 10

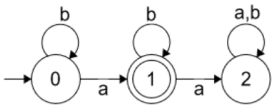
Correct

Note de 1,00 sur 1,00

Le nombre d'états de l'automate **déterministe minimal complet** qui reconnaît le langage des mots sur l'alphabet {a,b} contenant exactement un a est (répondre en écrivant un entier en base dix, par exemple 47) :

Réponse : 3

Le DFA complet minimum est :



Le DFA reduit aux états 0 et 1, reconnait bien le langage b*ab*, mais il n'est pas complet.

La réponse correcte est : 3

Question 11

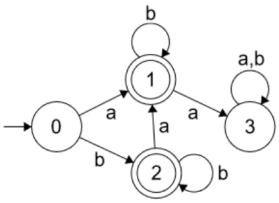
Correct

Note de 1,00 sur 1,00

On travaille sur l'alphabet {a,b}, donner le nombre d'états de l'automate **déterministe minimal complet** qui reconnaît le langage b*(a+b)b* (répondre en écrivant un entier en base dix, par exemple 635) :

Réponse : 4

Le DFA complet minimum qui reconnait b*(a+b)b* est :



Le DFA sans l'état 3 reconnait bien b*(a+b)b*, mais il n'est pas complet.

La réponse correcte est : 4

Question 12

Correct

Note de 2,00 sur 2,00

L et M étant 2 langages, pour chacun des langages ci-dessous, dire si il est :

- (dans tous les cas) rationnel
- (dans tous les cas) non rationnel
- parfois rationnel parfois non rationnel (ça dépend de L et de M)
- aucune des 3 propositions précédentes n'est vraie

Si L et M sont rationnels alors L ∩ M est	rationnel ✔
Si L est rationnel et M est non rationnel alors L ∩ M est	parfois rationnel parfois non rationnel ✔
Si L est un langage rationnel et M un langage tel que M ⊆ L alors M est	parfois rationnel parfois non rationnel ✔
Si L et M sont rationnels alors L.M = {u.v | u ∈ L et v ∈ M} est	rationnel ✔
Si L est rationnel alors {u.u | u ∈ L} est	parfois rationnel parfois non rationnel ✔

Votre réponse est correcte.

La classe (= l'ensemble) des langages rationnels est close pour les opérations ensemblistes *usuelles (?)*, telles que union, intersection, concaténation, * (vu en cours/TD). C'est-à-dire ici, que si L et M sont rationnels, alors L ∩ M est rationnel et L.M est rationnel.

Mais dès que l'un (a fortiori les 2) des langages n'est pas rationnel, alors les résultat peut être rationnel ou pas. Par exemple :

- si L=a* (rationnel) et M={a^nb^n : n ≥ 0} (non rationnel) : L ∩ M = {ε} (rationnel)
- si L=(a+b)* (rationnel) et M={a^nb^n : n ≥ 0} (non rationnel) : L ∩ M = {a^nb^n : n ≥ 0} (non rationnel)

Pour la dernière question, si L est rationnel que dire du langage {u.u | u ∈ L}, c'est un des rares cas où la rationalité n'est pas (toujours) conservée. Par exemple :

- si L=a* (rationnel), alors {u.u | u ∈ L} = {a^na^n : n ≥ 0} = {a^{2n} : n ≥ 0} = (a^2)* (rationnel)
- si L=a*b (rationnel), alors {u.u | u ∈ L} = {a^nba^nb : n ≥ 0} (non rationnel)

La réponse correcte est :
Si L et M sont rationnels alors L ∩ M est → rationnel,

Si L est rationnel et M est non rationnel alors L ∩ M est → parfois rationnel parfois non rationnel,

Si L est un langage rationnel et M un langage tel que M ⊆ L alors M est → parfois rationnel parfois non rationnel,

Si L et M sont rationnels alors L.M = {u.v | u ∈ L et v ∈ M} est → rationnel,

Si L est rationnel alors {u.u | u ∈ L} est → parfois rationnel parfois non rationnel

Question 13

Correct

Note de 2,00 sur 2,00

Pour chacun des langages ci-dessous, dire si il est :

- rationnel
- non rationnel
- ça dépend

Le langage $\{a^nb^n : n \geq 0\}$ est — non rationnel ✔

Le langage $a^*\{a^nb^n : n \geq 0\}b^*$ est — rationnel ✔

Le langage $\{(ab)^n : n \geq 0\}$ est — rationnel ✔

Le langage $\{a^naa^n : n \geq 0\}$ est — rationnel ✔

Votre réponse est correcte.

Le langage $\{a^nb^n : n \geq 0\}$ n'est pas rationnel (vu en cours/Td et c'est THE example de langage non rationnel).

Mais $a^*\{a^nb^n : n \geq 0\}b^* = a^*b^*$ est rationnel (ce n'est pas parce qu'une écriture est *compliquée* qu'il n'y en a pas de plus simple).

Le langage $\{(ab)^n : n \geq 0\} = (ab)^*$ est rationnel (idem).

Le langage $\{a^naa^n : n \geq 0\}=\{a^{2n+1} : n \geq 0\}=(a^2)^*a$ est rationnel (idem).

La réponse correcte est : Le langage $\{a^nb^n : n \geq 0\}$ est → non rationnel,
Le langage $a^*\{a^nb^n : n \geq 0\}b^*$ est → rationnel,

Le langage $\{(ab)^n : n \geq 0\}$ est → rationnel,

Le langage $\{a^naa^n : n \geq 0\}$ est → rationnel

Question 14

Correct

Note de 1,00 sur 1,00

L étant un langage, pour chacun des langages ci-dessous, dire si il est :

- (dans tous les cas) rationnel
- (dans tous les cas) non rationnel
- parfois rationnel parfois non rationnel (ça dépend de L)
- aucune des 3 propositions précédentes n'est vraie

Si L est rationnel alors L* est	rationnel ✔
Si L est non rationnel alors L* est	parfois rationnel parfois non rationnel ✔
Si L* est rationnel alors L est	parfois rationnel parfois non rationnel ✔
Si L* est non rationnel alors L est	non rationnel ✔

Votre réponse est correcte.

L'étoile d'un langage L rationnel est rationnel (vu en cours/TD), mais si L n'est pas rationnel tout est possible. Par exemple :

- si $L=\{a^nb^n : n \geq 0\}$ alors L* n'est pas rationnel
- si $L=\{a^nb^n : n \geq 0\} \cup \{a,b\}$, L n'est pas rationnel, mais $L^*=(a+b)^*$ qui est rationnel.

Donc si L* est rationnel, L peut être rationnel (car L rationnel => L* rationnel) ou pas (cas2 ci-dessus).

Mais si L* n'est pas rationnel alors L n'est pas rationnel (contraposée de L rationnel => L* rationnel).

La réponse correcte est :
Si L est rationnel alors L* est → rationnel,

Si L est non rationnel alors L* est → parfois rationnel parfois non rationnel,

Si L* est rationnel alors L est → parfois rationnel parfois non rationnel,

Si L* est non rationnel alors L est → non rationnel

Question 15

Correct

Note de 1,00 sur 1,00

Pour chacun des langages ci-dessous, dire si il est :

- rationnel
- non rationnel
- ça dépend

Le langage des mots sur l'alphabet {a,b}, de longueur impaire et dont le nombre de a est égal au nombre de b est — rationnel ✔

Le langage des mots sur l'alphabet {a,b,c}, de longueur impaire et dont le nombre de a est égal au nombre de b est — non rationnel ✔

Le langage des mots sur l'alphabet {a,b} dont le nombre de facteurs ab est égal au nombre de facteurs ba est — rationnel ✔

Votre réponse est correcte.

Finalement, cette question n'était pas bienvenue (ce n'est pas la seule sans doute), elle était de trop et trop *casse-tête*. (en même temps, comme c'était sans point négatif et qu'on avait plus de temps, on pouvait répondre au feeling ou au pif)

Q1 : le langage des mots sur l'alphabet {a,b}, de longueur impaire et dont le nombre de a est égal au nombre de b est ... l'ensemble vide (donc **rationnel**) (si le nombre de a égal le nombre de b et qu'il n'y a que des a et des b, la longueur du mot est paire). (un peu piège comme question)

Q2 : mais s'il y a aussi des c alors l'ensemble n'est pas vide, et par exemple avec les mots du type $a^n c b^n$ on vérifierait (mais bien sûr pas dans le temps imparti) que ce langage n'est **pas rationnel**.

Q3 : pour ce qui est du langage des mots sur l'alphabet {a,b} dont le nombre de facteurs ab est égal au nombre de facteurs ba, il est égal à $\mathbf{(a^+b^+)^*a^+ + (b^+a^+)^*b^+ + \varepsilon}$. En effet tout mot ne contenant que des 'a' ou que des 'b' convient (il n'a aucun facteur 'ab' ni 'ba'). Et tout mot contenant au moins un 'a' et au moins un 'b' peut être vu comme une succession de suites non vides de 'a' suivies de suites non vides de 'b', et chaque fois qu'on passe d'une suite non vide de 'a' à une suite non vide de 'b', cela fait un facteur 'ab' (idem avec 'ba' en inversant les rôle de 'a' et 'b'), et donc pour avoir le même nombre de 'ab' que de 'ba', si on commence par 'a', il faut finir pas 'a' (idem pour 'b'). Le langage est donc **rationne**l. (mieux valait répondre au hasard, on avait une chance sur 2 de gagner)

La réponse correcte est :
Le langage des mots sur l'alphabet {a,b}, de longueur impaire et dont le nombre de a est égal au nombre de b est → rationnel,

Le langage des mots sur l'alphabet {a,b,c}, de longueur impaire et dont le nombre de a est égal au nombre de b est → non rationnel,

Le langage des mots sur l'alphabet {a,b} dont le nombre de facteurs ab est égal au nombre de facteurs ba est → rationnel