Bases formelles du TAL
Correction du partiel

Pierre-Léo Bégay

13 mars 2020

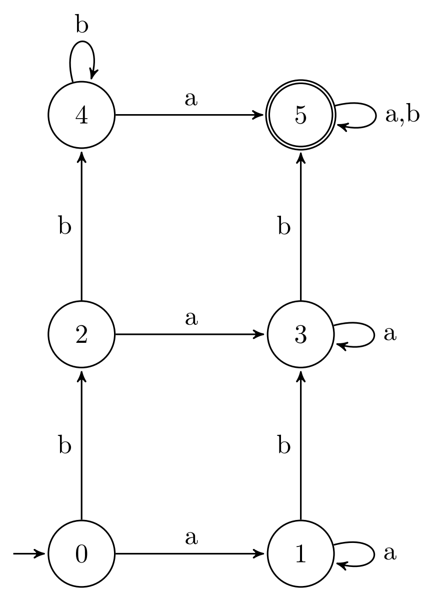
Dans tout ce partiel, on utilise l'alphabet $\Sigma = \{a, b\}$. Vous devrez justifier vos réponses, à moins qu'on puisse reconnaître les algorithmes utilisés (via par exemple le nom des états de vos automates).

Exercice 1 [6 points]

Soient $L_1 = \{w \in \Sigma^* \mid |w|_a > 0 \text{ et } |w|_b > 1\}$ et $L_2 = \{w \in \Sigma^* \mid |w|_a > 1 \text{ et } |w|_b > 0\}$.

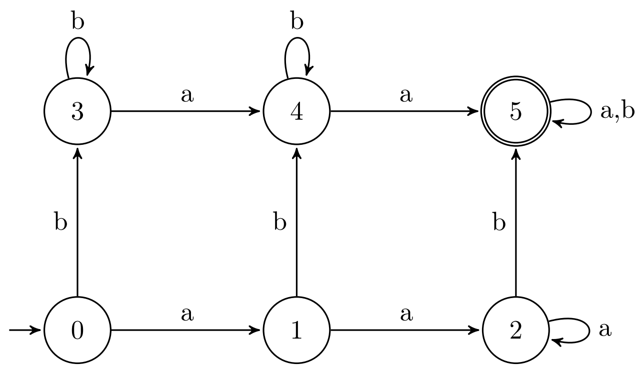
Question 1 [1] Donnez un automate fini reconnaissant L_1.

Correction On compte les a et les b jusqu'à 1 et 2 avec les états, disposés en matrice (les a horizontalement, les b verticalement).



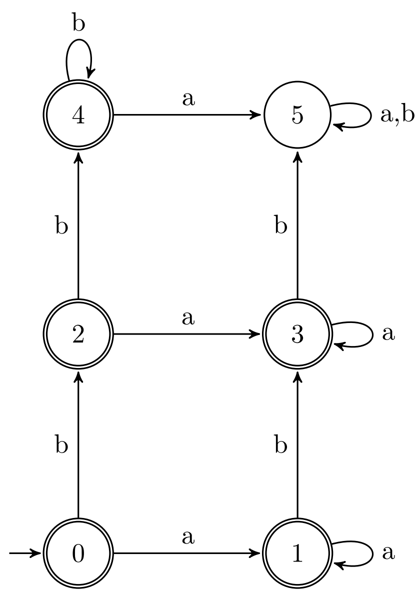
Question 2 [0,5] Donnez un automate fini reconnaissant L_2.

Correction Même astuce, en inversant les limites :



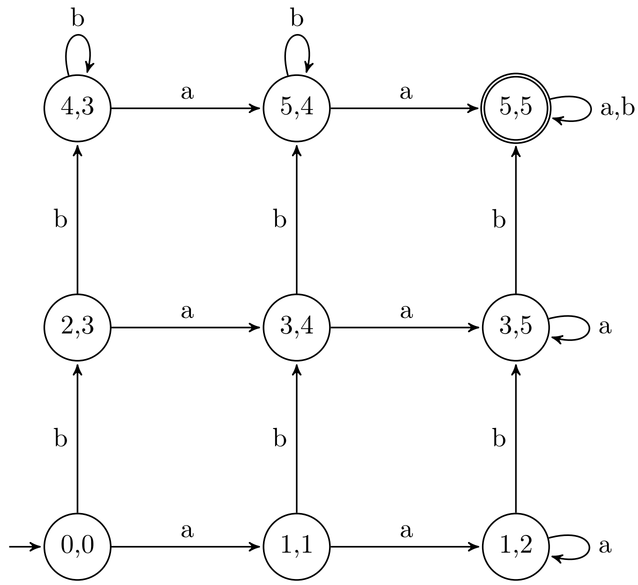
Question 3 [0,5] Donnez un automate fini reconnaissant $\overline{L_1}$.

Correction L'automate de la question 1 étant déjà déterministe et complet, il suffit d'intervertir ses états terminaux et non-terminaux :



Question 4 [2] Donnez un automate fini reconnaissant $L_1 \cap L_2$.

Correction En appliquant l'algorithme vu en cours :



Soient $L_3 = \{w \in \Sigma^* \mid |w|_a > 0 \textbf{ ou } |w|_b > 1\}$ et $L_4 = \{w \in \Sigma^* \mid |w|_a > 1 \textbf{ ou } |w|_b > 0\}$.

Question 5 [2] Donnez un automate fini, **complet**, **déterministe** et **minimal** reconnaissant $\overline{L_3 \cup L_4}$.

Correction Les informations qu'on veut retenir pour $L_3 \cup L_4$ sont les mêmes que dans la question 4, tout ce qui change est l'ensemble des états terminaux. Il fallait en effet les rendre tous terminaux, sauf l'initial, puis inverser et de minimiser. On pouvait également se compte que, pour tous ensembles E_1 et E_2, $\overline{E_1 \cup E_2} = \overline{E_1} \cap \overline{E_2}$. De plus, par un raisonnement similaire :

$$\overline{L_3} = \{w \in \Sigma^* \mid |w|_a \leq 0 \textbf{ et } |w|_b \leq 1\}$$

et

$$\overline{L_4} = \{w \in \Sigma^* \mid |w|_a \leq 1 \textbf{ et } |w|_b \leq 0\}$$

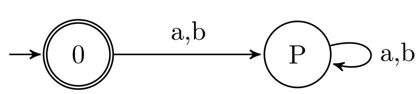
On a donc

$$\overline{L_3} \cap \overline{L_4} = \{w \in \Sigma^* \mid |w|_a \leq 1 \textbf{ et } |w|_b \leq 0 \textbf{ et } |w|_a \leq 0 \textbf{ et } |w|_b \leq 1\}$$

En élimant les chevauchements :

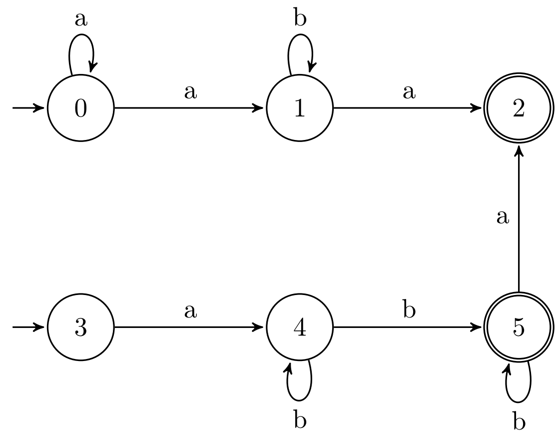
$$\overline{L_3} \cap \overline{L_4} = \{w \in \Sigma^* \mid |w|_a \leq 0 \textbf{ et } |w|_b \leq 0\} = \{\epsilon\}$$

Le plus petit automate déterministe et complet reconnaissant ce langage est bien sûr :

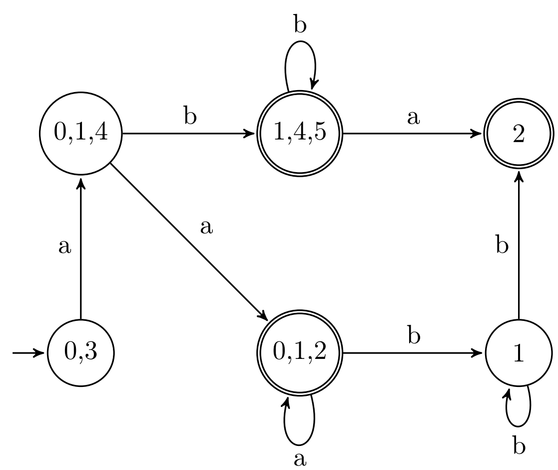


Exercice 2 [3 points]

Déterminisez l'automate suivant :



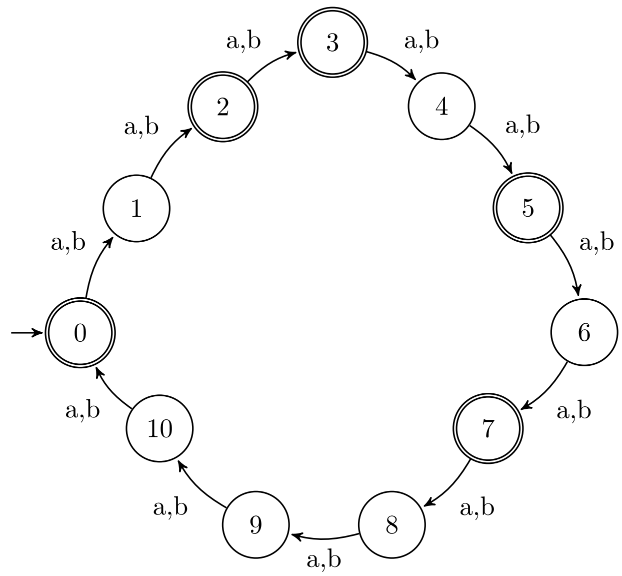
Correction En appliquant l'algorithme vu en cours :



Remarque L'automate initial et sa version déterminisée ont autant d'états l'un que l'autre, et même autant de transitions !

Exercice 3 [3 points]

Minimisez l'automate ci-dessous. Si vous présentez votre réponse sous forme d'arbre, indiquez l'ordre dans lequel vous avez traité les classes.



Correction On pose $C_1 = \{0, 2, 3, 5, 7\}$ et $C_2 = \{1, 4, 6, 8, 9, 10\}$. On peut immédiatement séparer C_2 en $C_3 = \{1, 4, 6, 10\}$ et $C_4 = \{8, 9\}$. C_4 peut alors être séparé en $\{8\}$ et $\{9\}$.

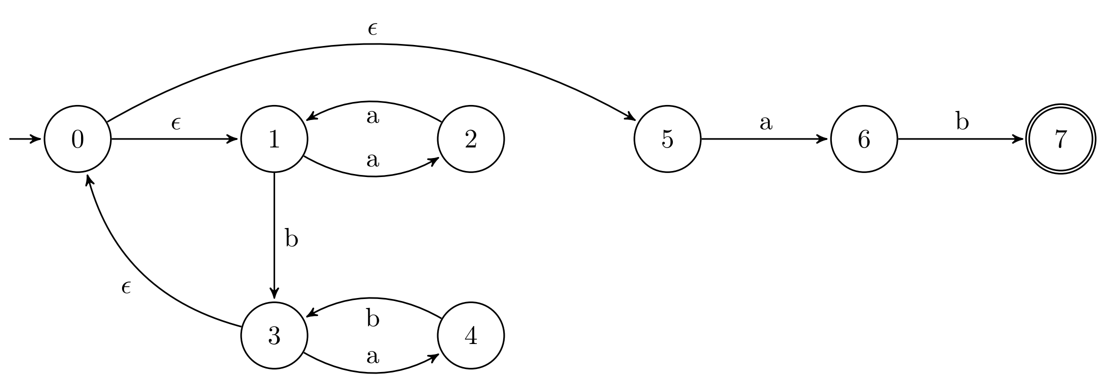
On sépare maintenant C_1 en $\{2\}$, $\{7\}$ et $C_5 = \{0, 3, 5\}$. C_3 peut donc maintenant être séparé en $\{1\}$, $\{6\}$, $C_6 = \{4, 10\}$. C_5 peut maintenant être totalement éclaté en $\{0\}$, $\{3\}$ et $\{5\}$, ce qui permet ensuite de casser C_6 en $\{4\}$ et $\{10\}$.

L'automate était donc déjà minimal !

Exercice 4 [3 points]

Donnez un automate fini reconnaissant le langage dénoté par l'expression $((aa)^*b(ab)^*)^*ab$.

Correction Un exemple parmi d'autres :



Exercice 5 [5 points]

Question 1 [1,5] Comment s'assurer qu'un automate représente le langage vide ? Dit autrement, proposez un critère simple, et si possible visuel, permettant de vérifier qu'un automate ne reconnaît aucun mot.

Correction Il est sans doute tentant de répondre "Quand il n'y a pas d'état terminal dans l'automate", mais ce n'est pas suffisant. En effet, même l'automate suivant ne reconnaît aucun mot :

Un automate ne reconnaît aucun si et seulement si il n'existe aucun chemin allant d'un état initial à un état terminal.

Question 2 [2,5] Soient L_1 et L_2 deux langages reconnus, respectivement, par les automates A_1 et A_2. Comment construire un automate reconnaissant le langage $L_1 \backslash L_2$? Vous pourrez illustrer votre réponse avec des schémas et / ou vous appuyer sur le cours.

Correction $L_1 \backslash L_2 = L_1 \cap \overline{L_2}$. On peut donc simplement calculer A'_2 le complémentaire de A_2, puis faire l'intersection de A_1 et A'_2. On a vu en cours comment faire les deux.

On pouvait cependant retrouver la transformation sans se rendre compte de l'égalité rappelée ci-dessus, et proposer de reprendre la construction de l'intersection de A_1 et A_2 pour simuler les deux automates, mais changer les états terminaux en "terminal à gauche, pas terminal à droite". Il fallait cependant dans ce cas penser à rappeler de compléter et déterminiser A_2.

Question 3 [1] Soient e_1 et e_2 deux expressions rationnelles quelconques. Proposez une méthode pour vérifier que e_1 représente un sous-ensemble d'e_2, cad. que tout mot appartenant au langage dénoté par e_1 appartient également à celui dénoté par e_2.

Soit L_1 et L_2 les langages reconnus par e_1 et e_2, respectivement. On utilise par exemple l'algorithme de Glushkov pour construire A_1 et A_2, les automates qui reconnaissent L_1 et L_2, puis la question 1 pour construire A_3 l'automate de $L_1 \backslash L_2$. L_1 est un sous-ensemble de L_2 ssi. A_3 n'accepte aucun mot, ce qu'on peut vérifier avec la méthode de la question 2.