Bases formelles du TAL
Correction de l'examen

Pierre-Léo Bégay

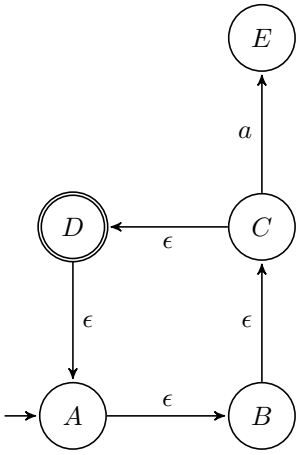
10 mai 2019

La plupart des questions peuvent se faire de plusieurs façons, pas toujours aussi efficaces les unes que les autres. Corollairement, il n'est jamais nécessaire d'avoir fait une question pour répondre à la suivante, bien que ça puisse aider (notamment dans l'exercice 4, que vous devriez lire en entier avant d'attaquer).

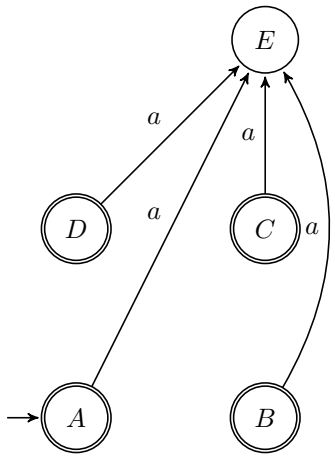
Toutes vos réponses devraient être justifiées, soit en mentionnant un algorithme (dont on devrait pouvoir un minimum reconnaître l'utilisation), soit en faisant le lien entre la donnée et le résultat.

Exercice 1 [3,5 points]

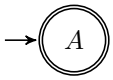
Donnez un automate sans ϵ-transition équivalent à



Première correction Puisque les états A, B et D peuvent accéder *gratuitement* à C, et que C va en E via a, il faut ajouter des transitions de A, B et D vers E avec a. De plus, A, B et C vont *gratuitement* en D, qui est terminal. Ils doivent donc également être. On obtient donc au final :



Deuxième correction On remarque que le seul mot reconnu par l'automate est ϵ. En effet, le seul état terminal est D, et le seul moyen de l'obtenir depuis A est de faire des tours via les ϵ-transitions. A partir du moment où on lit une lettre, au mieux on va en E, et on ne peut plus rien faire (pas même accepter). L'automate suivant, sans ϵ-transition, permet donc de reconnaître le même langage :



Exercice 2 [4,5 points]

Soit la grammaire suivante :

- $S \to CDS \mid \epsilon$
- $C \to aC \mid \epsilon$
- $D \to bD \mid \epsilon$

Question 1 [3] Donnez, sous la forme d'une expression rationnelle, le langage reconnu par la grammaire.

Correction On analyse d'abord les symboles qui ne dépendent pas d'autres :

- C génère une série - potentiellement vide - de a avant de s'arrêter, et engendre donc le langage a^*
- De même, D génère b^*

S génère une série - potentiellement vide - de CD, c'est-à-dire de a^*b^*, avant de s'arrêter. Le langage engendré est donc $(a^*b^*)^*$.

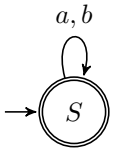
Remarque Une autre façon - plus longue - de répondre aurait été de faire d'abord la question 2, puis de raisonner sur l'automate équivalent.

Question 2 [1,5] Donnez une grammaire de type 3 qui engendre le même langage.

Première correction Dans la première question, on a vu que le langage engendré est $(a^*b^*)^*$. On peut remarquer que cette expression rationnelle est équivalente à $(a+b)^*$. En effet, elle permet d'engendrer n'importe quelle suite de a et de b, c'est-à-dire n'importe quel mot. On peut donc utiliser la grammaire suivante :

- $S \to aS \mid bS \mid \epsilon$

Notez que cette grammaire est directement équivalente à l'automate suivant qui reconnaît trivialement tout mot :



Deuxième correction Même sans avoir répondu à la question 1, on peut essayer, bon gré mal gré, de "simuler" la grammaire originale dans une de type 3 : on voit que tout C est suivi d'un D, lui-même suivi d'un S. Le plan général va donc être que S engendre un C, qui engendre un D, qui engendre un S.

Bien sûr, S ne peut pas générer *directement* un C (les grammaires de type 3 n'acceptent pas les règles de la forme $A \to B$), on va donc permettre à S de générer le "contenu" de C (pensez à l'élimination de cycles dans une type 2). S doit donc pouvoir générer aC.

De plus, C peut générer ϵ, on doit donc également permettre à S de générer immédiatement un D, ou en tout cas son contenu bD. Au lieu de générer ϵ, C devrait également simuler D en générant bD.

Puisque un C comme un D peuvent être suivis par un S, ils doivent également le simuler. Or, le "vrai contenu" de S c'est ... C ou D ! On obtient donc la grammaire un peu débile - mais correcte et de type 3 - suivante :

- $S \to aC \mid bD \mid \epsilon$
- $C \to aC \mid bD \mid \epsilon$
- $D \to bD \mid aC \mid \epsilon$

Remarque Bon, on est d'accord que c'est très avec les mains et qu'il valait mieux utiliser la première question.

Exercice 3 [6 points]

Soit G la grammaire

- $S \to AB$
- $A \to DD \mid aA \mid DaD$
- $B \to bB \mid E \mid b$
- $C \to BA \mid aD$

- $D \to AbA \mid \epsilon$
- $E \to Fa \mid aE$
- $F \to Ea \mid aF$

Question 1 [4] Nettoyez $\mathcal{G}$ (pas de symbole inutile, pas d'ϵ-production, pas de cycle).

Correction Par souci d'optimisation, on s'occupe d'abord des symboles inutiles :

Symboles inaccessibles On calcule les symboles accessibles :

- $N_0 = \{S\}$
- $N_1 = \{S, A, B\}$
- $N_2 = \{S, A, B, D, E\}$
- $N_3 = \{S, A, B, D, E, F\}$
- $N_4 = \{S, A, B, D, E, F\} = N_3$

C est le seul symbole inaccessible, on peut donc le supprimer.

Symboles sans production On calcule les symboles avec production :

- $N_0 = \emptyset$
- $N_1 = \{B, D\}$
- $N_2 = \{A, B, D\}$
- $N_3 = \{S, A, B, D\}$
- $N_4 = \{S, A, B, D\} = N_3$

E et F sont sans production, on peut donc les supprimer. La grammaire est donc désormais

- $S \to AB$
- $A \to DD \mid aA \mid DaD$
- $B \to bB \mid b$
- $D \to AbA \mid \epsilon$

Elimination d'ϵ-transition Une seule ϵ-transition ici, mais attention, puisque A peut se réécrire en DD, il est aussi concerné ! Par contre, rien ne se réécrit strictement en un mélange de A et D. Il faut donc "potentiellement effacer" tous les A et les D dans la grammaire, pour obtenir :

- $S \to AB \mid B$
- $A \to DD \mid D \mid aA \mid a \mid DaD \mid aD \mid Da$ ($\mid a$)
- $B \to bB \mid b$
- $D \to AbA \mid bA \mid Ab \mid b$

Elimination de cycles Maintenant qu'on a éliminé les ϵ-transitions, il suffit d'éliminer les productions singulières (les $A \to B$). On doit ajouter le contenu de D dans A, ainsi que celui de B dans S :

- $S \to AB \mid bB \mid b$
- $A \to DD \mid AbA \mid bA \mid Ab \mid b \mid aA \mid a \mid DaD \mid aD \mid Da$
- $B \to bB \mid b$
- $D \to AbA \mid bA \mid Ab \mid b$

Question 2 [2] Soit L le langage engendré par $\mathcal{G}$. Donnez une grammaire qui engendre L^*.

Correction Le plus simple est de construire au-dessus de la grammaire précédente. De façon générale, si on a une grammaire d'axiome S, on peut créer une grammaire engendrant le même langage en rajoutant un symbole S', nouvel axiome, ainsi que les deux règles suivantes :

- $S' \to SS' \mid \epsilon$

En l'occurrence, on obtient donc la grammaire suivante :

- $S' \to SS' \mid \epsilon$
- $S \to AB$
- $A \to DD \mid aA \mid DaD$
- $B \to bB \mid b$
- $D \to AbA \mid \epsilon$

Notez que rien n'interdisait d'utiliser la version non-propre de la grammaire.

Exercice 4 [6 points + bonus]

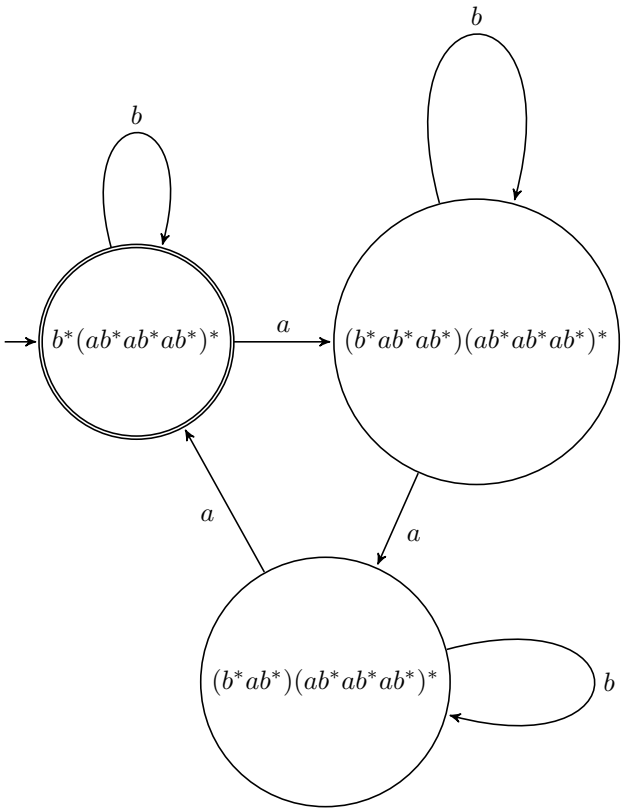
Soit l'expression rationnelle $e = b^*(ab^*ab^*ab^*)^*$

Question 1 [4] Donnez un automate reconnaissant le langage décrit par e.

Première correction On pourrait utiliser la construction de Thomson, mais ce n'est pas le plus simple pour la deuxième question (puisque Thomson génère un automate en général non-déterministe, alors qu'on a besoin du déterminisme pour la complémentation). On va donc choisir l'automate des résiduels :

$$\begin{aligned} b^{-1}e &= b^{-1}bb^*(ab^*ab^*ab^*)^* \\ &= b^*(ab^*ab^*ab^*)^* = e \\ a^{-1}e &= a^{-1}(ab^*ab^*ab^*)^* \\ &= a^{-1}(ab^*ab^*ab^*)(ab^*ab^*ab^*)^* \\ &= (b^*ab^*ab^*)(ab^*ab^*ab^*)^* \end{aligned}$$

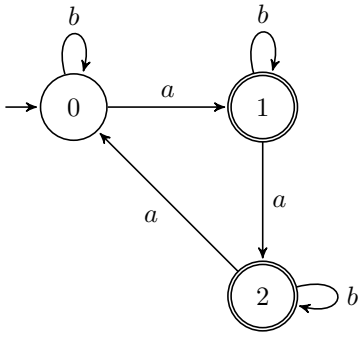
Et ainsi de suite. On obtient au final



Deuxième correction On peut remarquer que l'expression rationnelle correspond au langage des mots qui ont un nombre de a qui est un multiple de 3. En ayant ceci en tête, on peut retrouver le même automate (modulo les noms des états).

Question 2 [2] Donnez un automate reconnaissant le complémentaire du langage décrit par e.

Correction En utilisant la première question ou la compréhension de l'expression, on obtient :



Question bonus Donnez une expression rationnelle décrivant le complémentaire du langage décrit par e.

Correction On peut le faire en utilisant la question précédente et l'algorithme de McNaughton et Yamada. Il est également possible de se dire que le complémentaire du langage des mots avec un nombre de a multiple de 3, c'est ceux avec $3n+1$ ou $3n+2$ a. On peut s'en assurer avec l'expression (sous-optimale) $(b^*ab^*ab^*ab^*)^*b^*ab^*ab^*$