Bases formelles du TAL
DM sur les ϵ-transitions

Pierre-Léo Bégay

À me rendre le 6 mars 2020

1 ϵ-transitions

1.1 Définitions

On donne parfois la définition suivante d'un AFND :

$$A = \langle Q, \Sigma, q_0, F, \delta \rangle$$

Q ensemble fini d'états

Σ l'alphabet (ensemble de lettres)

q_0 l'état initial

$F \subseteq Q$, les états terminaux

δ fonction de $(Q \times (\Sigma \cup \{\epsilon\}))$ dans 2^Q

Par rapport à la définition du cours, on revient à un seul état initial et qu'on permet d'étiqueter des transitions par ϵ. Ces transitions, appelées ϵ-transitions, sont *gratuites*, par contraste avec les transitions normales qui *consomment* une lettre chaque fois qu'on les emprunte. La notion d'acceptation est sinon la même que pour les AFND qu'on a déjà vus.

1.2 Exemples

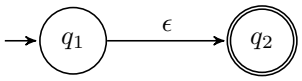

Figure 1: Automate A_1

Dans l'automate A_1, aucune transition par lettre n'est possible, ce qui empêche d'accepter tout mot autre que le mot vide. Ce dernier est cependant reconnu car on peut emprunter *gratuitement* l'unique transition et atterrir dans un état terminal.

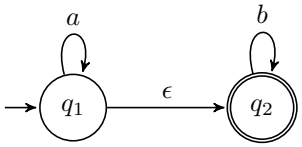

Figure 2: Automate A_2

L'automate A_2 reconnait quant à lui le langage a^*b^*. En effet, on peut boucler avec des a sur q_1 puis, une fois qu'on a fini, on passe gratuitement à q_2 (sans consommer de a ou de b) où on peut boucler avec des b jusqu'à avoir fini le mot.

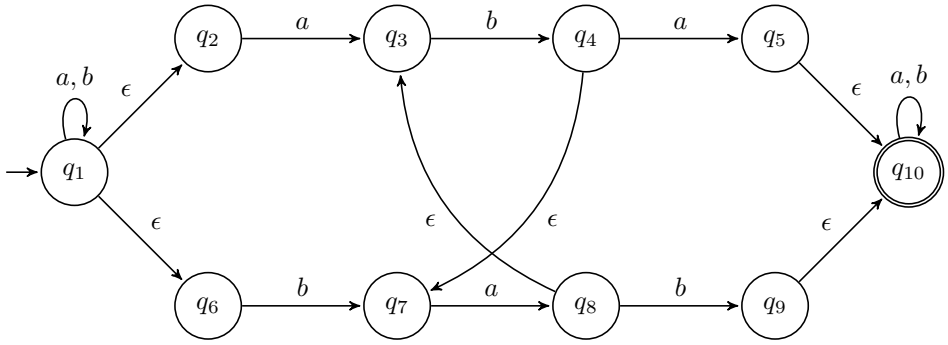

Figure 3: Automate A_3

Enfin, l'automate A_3, proche d'un qu'on a vu en cours, reconnaît quant à lui le langage $\Sigma^*aba\Sigma^* + \Sigma^*bab\Sigma^*$ (les deux ϵ-transitions en croix ne permettent pas d'accepter plus de mots).

2 Lecture d'automates avec ϵ-transitions

Décrivez les langages reconnus par les automates A_4, A_5 et A_6 à l'aide d'une expression rationelle. Essayez de justifier, au moins informellement, votre réponse.

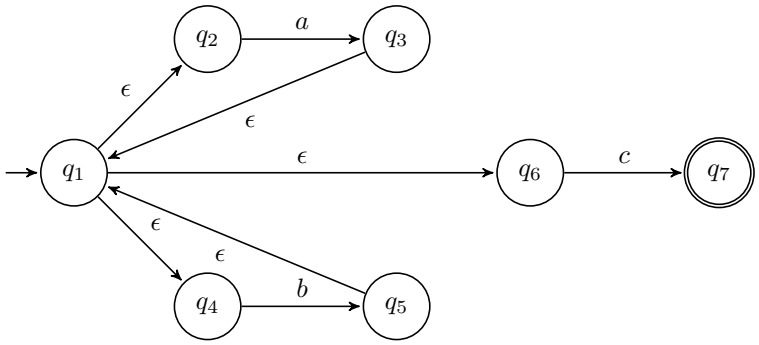

Figure 4: Automate A_4

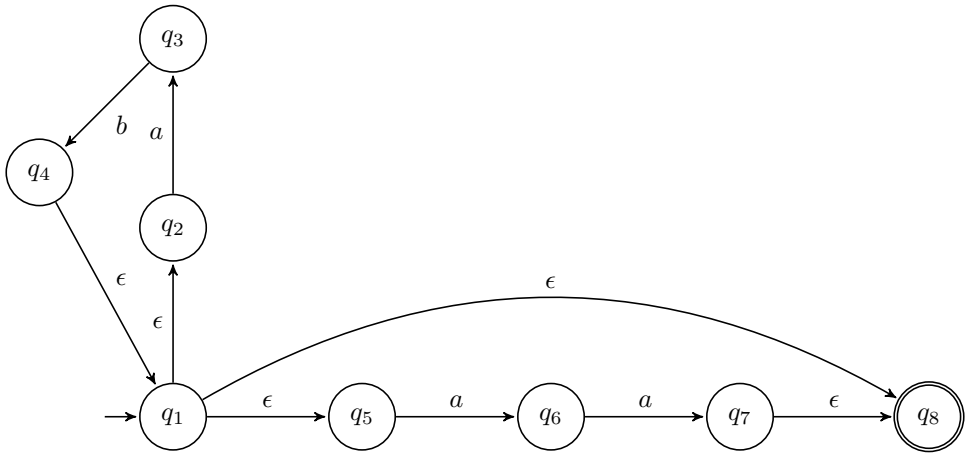

Figure 5: Automate A_5

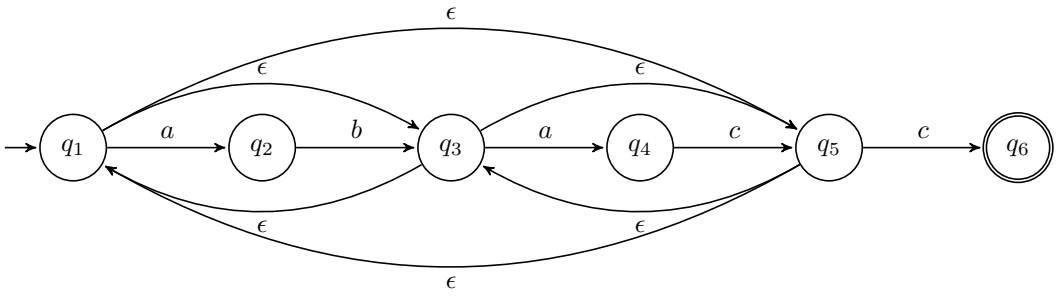

Figure 6: Automate A_6

Correction Dans A_4, le cycle $q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{a} q_3 \xrightarrow{\epsilon} q_1$ permet de boucler sur l'état initial q_1 en lisant un a. De même, le cycle $q_1 \xrightarrow{\epsilon} q_4 \xrightarrow{b} q_5 \xrightarrow{\epsilon} q_1$ permet lui de boucler en lisant un b. On peut donc lire autant de a et de b qu'on veut au début du mot. Ensuite, le segment $q_1 \xrightarrow{\epsilon} q_6 \xrightarrow{c} q_7$ amène à l'état terminal en lisant un c. Une fois arrivé, on ne peut plus rien lire. Le langage reconnu est donc $(a+b)^*c$

Dans A_5, le circuit $q_1 \xrightarrow{\epsilon} q_2 \xrightarrow{a} q_3 \xrightarrow{b} q_4 \xrightarrow{\epsilon} q_1$ permet de boucler sur q_1 en lisant ab. Une fois qu'on a fini de boucler, on peut soit lire aa en parcourant $q_1 \xrightarrow{\epsilon} q_5 \xrightarrow{a} q_6 \xrightarrow{a} q_7 \xrightarrow{\epsilon} q_8$, soit aller directement de q_1 à q_8 avec une ϵ-transition. Une fois arrivé, on ne peut plus rien lire. Le langage reconnu est donc $(ab)^*((aa)+\epsilon)$

Dans A_6, on peut utiliser les ϵ-transitions pour circuler librement entre q_1, q_3 et q_5 (notez que les transitions entre q_1 et q_5 sont inutiles). Via $q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_3$ on lit ab, et $q_3 \xrightarrow{a} q_4 \xrightarrow{c} q_5$ nous fait lire ac. On peut donc lire autant de fois ab et ac qu'on veut, avant de lire un c en faisant $q_5 \xrightarrow{c} q_6$, où on ne peut plus rien faire. Le langage reconnu est donc $((ab)+(ac))^*c$.

3 Elimination d'ϵ-transitions

On propose une méthode pour éliminer les ϵ-transitions s'appuyant sur la fonction ϵ^+ de type $Q \to 2^Q$ définie de la façon suivante :

1. Si $q_j \in \delta(q_i, \epsilon)$, alors $q_j \in \epsilon^+(q_i)$
2. Si $q_j \in \epsilon^+(q_i)$ et $q_k \in \delta(q_j, \epsilon)$, alors $q_k \in \epsilon^+(q_i)$

Figure 7: Définition de ϵ^+

A partir d'un automate non-déterministe avec ϵ-transitions $\langle Q, \Sigma, q_0, F, \delta \rangle$, on génère un automate non-déterministe équivalent sans ϵ-transitions $\langle Q, \Sigma, q_0, F', \delta' \rangle$ avec l'algorithme suivant :

```
1:  F' := F
2:  for all q_i ∈ Q do
3:      for all c ∈ Σ do δ'(q_i, c) := δ(q_i, c)
4:      end for
5:  end for
6:  for all q_i ∈ Q tels que ε+(q_i) ≠ ∅ do
7:      for all q_j ∈ ε+(q_i) do
8:          for all c ∈ Σ et q_r ∈ Q tels que q_r ∈ δ(q_j, c) do
9:              δ'(q_i, c) := δ'(q_i, c) ∪ {q_r}
10:         end for
11:         if q_j ∈ F then
12:             F' := F' ∪ {q_i}
13:         end if
14:     end for
15: end for
```

Figure 8: Algorithme d'élimination des ϵ-transitions

Question 1 Pour chaque automate (A_1 à A_6), calculez la fonction ϵ^+ en vous servant de la définition en figure 7. Vous devriez donner l'image de chaque état, par exemple $\epsilon^+(q_1) = \{q_1, q_2\}$, $\epsilon^+(q_2) = \emptyset$ etc.

Remarque On corrige en même temps les questions 1 et 2.

Question 2 Pour chaque automate (A_1 à A_6), appliquez l'algorithme de la figure 8. Vous pouvez dessiner le résultat. Pas besoin de détailler tous les calculs.

Correction pour A_1

$\epsilon^+(q_1) = \{q_2\}$[1]

[1] En rouge on note les états terminaux, pour appliquer la règle des lignes 11/12 de l'algorithme.

$\epsilon^+(q_2) = \emptyset$

Figure 9: Automate A'_1

Notez que l'état q_2 est devenu inaccessible. Techniquement il fait toujours partie de l'automate (l'ensemble d'états n'est pas modifié par l'algorithme), mais on pourrait bien sûr raisonnablement le 'jeter'.

Correction pour A_2

$\epsilon^+(q_1) = \{q_2\}$

$\epsilon^+(q_2) = \emptyset$

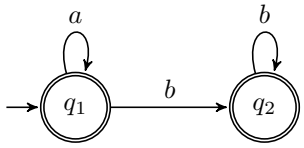

Figure 10: Automate A'_2

Correction pour A_3

$\epsilon^+(q_1) = \{q_2, q_6\}$

$\epsilon^+(q_2) = \emptyset$

$\epsilon^+(q_3) = \{\}$

$\epsilon^+(q_4) = \{q_7\}$

$\epsilon^+(q_5) = \{q_{10}\}$

$\epsilon^+(q_6) = \emptyset$

$\epsilon^+(q_7) = \emptyset$

$\epsilon^+(q_8) = \{q_3\}$

$\epsilon^+(q_9) = \{q_{10}\}$

$\epsilon^+(q_{10}) = \emptyset$

Notez que l'automate A'_3 n'est ni minimal, ni déterministe (t qu'il y a encre des états inaccessibles).

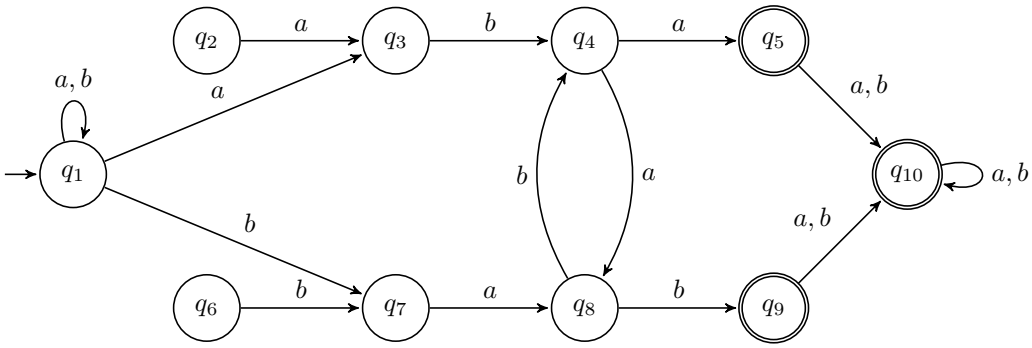

Figure 11: Automate A'_3

Correction pour A_4

$\epsilon^+(q_1) = \{q_2, q_4, q_6\}$

$\epsilon^+(q_2) = \emptyset$

$\epsilon^+(q_3) = \{q_1, q_2, q_4, q_6\}$

$\epsilon^+(q_4) = \emptyset$

$\epsilon^+(q_5) = \{q_1, q_2, q_4, q_6\}$

$\epsilon^+(q_6) = \emptyset$

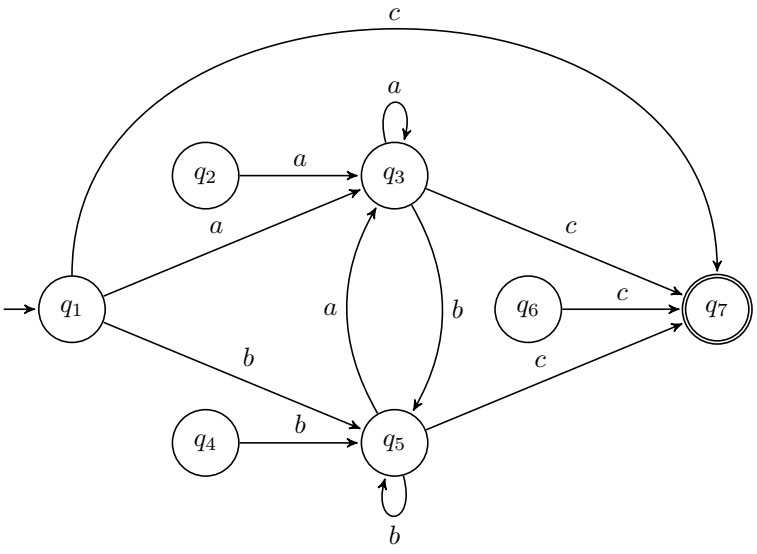

Figure 12: Automate A'_4

Correction pour A_5

$\epsilon^+(q_1) = \{q_2, q_5, q_8\}$

$\epsilon^+(q_2) = \emptyset$

$\epsilon^+(q_3) = \emptyset$

$\epsilon^+(q_4) = \{q_1, q_2, q_5, q_8\}$

$\epsilon^+(q_5) = \emptyset$

$\epsilon^+(q_6) = \emptyset$

$\epsilon^+(q_7) = \{q_8\}$

$\epsilon^+(q_9) = \emptyset$

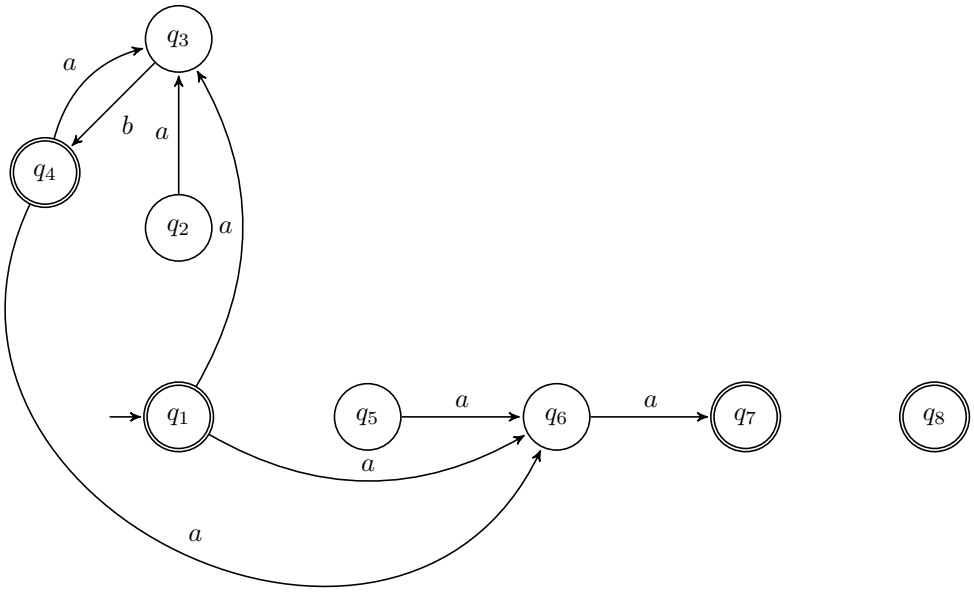

Figure 13: Automate A'_5

Correction pour A_6

$\epsilon^+(q_1) = \{q_1, q_3, q_5\}$

$\epsilon^+(q_2) = \emptyset$

$\epsilon^+(q_3) = \{q_1, q_3, q_5\}$

$\epsilon^+(q_4) = \emptyset$

$\epsilon^+(q_5) = \{q_1, q_3, q_5\}$

$\epsilon^+(q_6) = \emptyset$

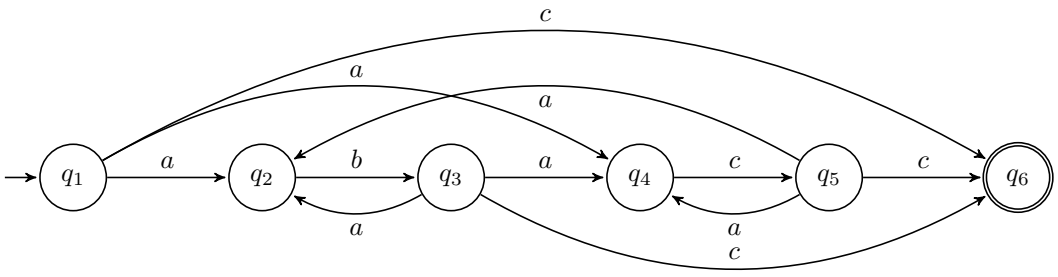

Figure 14: Automate A'_6

Question 3 Essayez d'expliquer en français la fonction ϵ^+ et l'algorithme comme si vous vouliez me convaincre qu'ils font correctement leur boulot (ce qui est le cas)[2]. Vous pouvez vous aider d'exemples, soit tirés de la question précédente, soit originaux.

Correction La fonction ϵ^+ associe à chaque état q_i l'ensemble des états qui sont accessible par une ϵ-transition (règle 1 de la fonction) ou plusieurs (la **récursion** de la règle 2). Par exemple, dans la correction de l'élimination des ϵ-transitions de l'automate A_5, les états q_2, q_5 et q_8 sont intégrés à $e^+(q_4)$ par la deuxième règle de la définition de la fonction car on peut y accéder depuis q_4 en deux coups d'ϵ

L'automate produit (ou renvoyé) par l'algorithme a le même ensemble d'états et le même initial que l'automate donné en entrée. Ce qui change, ce sont donc les transitions et les états terminaux.

Les transitions Déjà, la boucle des lignes 2 à 4 de l'algorithme permet de *garder* dans le nouvel automate toutes les 'non-ϵ-transitions' données[3]. Toute transition par a, b etc dans l'automate original apparaîtra donc également dans la version sans ϵ-transition.

La seconde boucle s'intéresse à chaque triplet d'états q_i, q_j et q_r tels que $q_i \xrightarrow{\epsilon}{}^{+} q_j \xrightarrow{c} q_r$, c'est à dire tels que q_i peut atteindre q_j avec une ou plusieurs ϵ-transitions et q_j va en q_r avec une transition *normale* avec c.

Pour chacun de ces triplets $q_i \xrightarrow{\epsilon}{}^{+} q_j \xrightarrow{c} q_r$, on ajoute (ligne 9) la transition $q_i \xrightarrow{c} q_r$. Le sens de cette règle est que quand, depuis un état, on peut *se préparer* à une *vraie* transition avec une série d'ϵ-transitions, alors on peut la faire directement sans bouger au préalable.

Etats terminaux De plus, si un état non-terminal peut atteindre un état terminal via des ϵ-transitions, il accepte le mot vide. Il faut donc *garder* ça même après l'élimination des transitions, d'où le fait qu'on ajoute ces états à F' (lignes 11/12).

[2]Notez que rien n'est gratuit dans l'algorithme et que chaque *morceau* a un sens. Vous devriez donc tout mentionner.

[3]Pour rappel, ϵ ne fait pas partie de Σ. Il est *rajouté* à la liste des symboles pouvant étiqueter une transition dans la définition du δ d'un automate avec ϵ-transitions

Version plus technique Soit le mot $w = c_1c_2...c_n$ accepté par l'automate avec ϵ-transitions (appelé A). Il existe donc dans cet automate un chemin pour le mot en alternant *vraies transitions* et (potentiellement) une ou plusieurs ϵ-transitions. Le chemin est donc de la forme :

$$q_{u_1} \xrightarrow{\epsilon}{}^{*} q_{u_2} \xrightarrow{c_1} q_{u_3} \xrightarrow{\epsilon}{}^{*} q_{u_4} \xrightarrow{c_2} q_{u_5} \ \cdots \ q_{u_{2n-2}} \xrightarrow{\epsilon}{}^{*} q_{u_{2n-1}} \xrightarrow{c_n} q_{u_{2n+1}} \xrightarrow{\epsilon}{}^{*} q_{u_{2n+2}}$$

avec $q_{u_{2n+2}} \in F$.

Pour rappel, chaque $q_{u_i} \xrightarrow{\epsilon}{}^{*} q_{u_{i+1}} \xrightarrow{c} q_{u_{i+2}}$ représente un passage de q_{u_i} à $q_{u_{i+1}}$ avec aucune ou un nombre strictement positif d'ϵ-transitions puis à $q_{u_{i+2}}$ en lisant c.

Dans le premier cas, on a $q_{u_i} = q_{u_{i+1}}$ (puisqu'on n'a pas bougé), on peut donc enlever la (non-)transition du chemin et passer directement de q_{u_i} à $q_{u_{i+2}}$ avec c.

Dans le second cas, $q_{u_{i+1}} \in \epsilon^+(q_{u_i})$. On a donc ajouté la transition $q_{u_i} \xrightarrow{c} q_{u_{i+2}}$ à l'automate produit par l'automate (A').

Le chemin donné plus haut correspond donc, dans l'automate produit, à :

$$q_{u_1} \qquad \xrightarrow{c_1} q_{u_3} \qquad \xrightarrow{c_2} q_{u_5} \ \cdots \ q_{u_{2n-2}} \qquad \xrightarrow{c_n} q_{u_{2n+1}}$$

De plus, puisque $q_{u_{2n+1}} \in \epsilon^+(q_{2n+2})$, $q_{u_{i+1}}$ est un état terminal dans l'automate produit (s'il ne l'était pas déjà dans l'original). La lecture du mot s'arrête donc sur un état acceptant. Le mot w, qui était accepté dans l'automate initial, est donc également accepté dans l'automate produit par l'algorithme.

Pour faire les choses bien, il faudrait également montrer que l'algorithme n'ajoute pas des mots acceptés, c'est à dire qu'il n'existe pas de w accepté par A' et pas par A. La démonstration est un peu plus lourde que la précédente mais en utilisant un raisonnement par l'absurde on retrouve au fond le même principe.

En gros, on suppose qu'un tel mot existe. Il existe donc un chemin dans A' qui accepte w. Si toutes les transitions du chemin étaient déjà présentes dans A et que le dernier état était à l'origine terminal, alors l'exact même chemin aurait accepté w dans A. Au moins une de deux propositions est donc fausse.

Si il est faux que toutes les transitions étaient déjà dans A, au moins une ne l'était pas. Or, si des transitions $q_i \xrightarrow{c} q_j$ apparaissent dans A', c'est que q_j était accessible depuis q_i avec des ϵ-transitions puis c. On peut donc 'retrouver' toutes ces transitions dans A.

S'il est faux que le dernier état du chemin était terminal dans A, alors il l'est devenu. Or, la seule façon dont l'algorithme peut un état terminal est s'il peut atteindre un terminal avec une série d'ϵ-transitions. On peut donc, dans A, rajouter des ϵ-transitions au chemin pour atteindre un état terminal.

Tout chemin acceptant w dans A' peut donc être transformé en un chemin acceptant dans A acceptant le même mot. L'algorithme n'ajoute donc pas de nouveaux mots.

Conclusion Tout mot accepté par A est accepté par A', et inversement. L'algorithme est donc conforme à sa spécification.

4 Formalisation

Question 4 Donnez une formalisation de l'acceptation d'un mot dans le contexte des AFND avec ϵ-transtion en adaptant la définition de δ^* donnée dans le cours.

Correction On adapte la formalisation des AFND vue en cours :

- $\delta^*(q, \epsilon^4) = \{q\} \cup \epsilon^+(q)$
- $\delta^*(q, a.w) = \bigcup_{q' \in \{q\} \cup \epsilon^+(q)} \bigcup_{q'' \in \delta(q', a)} \delta^*(q'', w)$

La première disjonction $\bigcup$ permet d'utiliser des ϵ-transitions pour passer de q à un des états q' accessibles gratuitement. La deuxième disjonction renvoie l'ensemble des états accessibles via une transition normale depuis l'état choisi.

Pour l'acceptation, on veut partir du seul état initial i, lire le mot donné et vérifier qu'on a atteint un état terminal ou qu'on peut en atteindre un gratuitement avec des ϵ-transitions. On dit donc qu'un AFND avec ϵ-transitions accepte un mot w ssi.

$$\exists q_f \in (\delta^*(i, w) \cap F)$$

Question bonus Les AFND avec ϵ-transitions sont-ils plus ou moins expressifs[5] que ceux vus en cours, où on pouvait avoir plusieurs états initiaux ?

Correction Par rapport aux AFND du cours, les AFND avec ϵ-transitions ont les ϵ-transitions en plus et la possibilité d'avoir plusieurs états initiaux en moins. Puisqu'il existe un algorithme (justifié !) d'élimination des ϵ-transitions qui produit un automate reconnaissant le même langage, l'utilisation d'ϵ-transitions ne permet pas de reconnaître des langages que les AFND classiques ne peuvent pas représenter.

Il faut maintenant regarder si l'impossibilité d'avoir plusieurs états initiaux nous fait perdre de l'expressivité. Ce n'est pas le cas, parce que les ϵ-transitions permettent de simuler la présence de plusieurs états initiaux. En effet, si on souhaite par exemple que les états q_1, q_2 et q_3 soient initiaux, il suffit d'ajouter un état q_i initial, avec des ϵ-transitions de q_i vers les trois états sus-nommés.

Les changements entre AFND avec et sans ϵ-transitions n'altèrent donc pas l'expressivité des modèles.

Remarque On aurait aussi pu contraindre les $AFND$ avec ϵ-transitions en n'autorisant qu'un seul état terminal. En effet, soit un automate avec trois états terminaux q_1, q_2 et q_3, on pouvait les rendre "normaux" mais créer un unique état terminal q_f avec des ϵ-transitions de q_1, q_2 et q_3 vers q_f. En utilisant une astuce similaire, on aurait également pu obliger le déterminisme sur les transitions normales.

Remarque bis Puisque les ϵ-transitions ne changent pas l'expressivité des AFND, on mélange en pratique les définitions, en autorisant plusieurs états initiaux, plusieurs transitions pour une même lettre et les ϵ-transitions.

[4] Attention, il s'agit ici du mot vide, pas d'une ϵ-transition !

[5] cad. permettent-ils de décrire plus ou moins de langages ?

5 Propriétés de clôture

Question 5 Etant donnés des AFND (version ϵ) représentant deux expressions rationnelles quelconques e_1 et e_2, expliquez, en vous aidant de schémas, comment construire des automates reconnaissant les expressions $e_1 + e_2$, $e_1.e_2$ et e_1^*.

Question bonus Répondre à la question précédente en utilisant la formalisation des automates.

5.1 $e_1 + e_2$

On a A_1 et A_2 qui reconnaissent e_1 et e_2, respectivement. On a bien sûr envie de mettre les deux automates "l'un à côté de l'autre", mais on se retrouverait alors avec deux états initiaux. Soient i_1 et i_2 les états initiaux de A_1 et A_2, on les rend normaux et on ajoute un nouvel état q_i initial, avec des ϵ-transitions vers i_1 et i_2;

Formalisation Soient $A_1 = \langle Q_1, \Sigma_1, i_1, F_1, \delta_1 \rangle$ et $A_2 = \langle Q_2, \Sigma_2, i_2, F_2, \delta_2 \rangle$, on renvoie

$$\langle Q_1 \cup Q_2 \cup \{q_i\}, \Sigma_1 \cup \Sigma_2, q_i, F_1 \cup F_2, \delta' \rangle$$

où

$$\begin{cases} \delta'(q,a) = \delta_1(q,a) & \text{si } q \in Q_1, \\ \delta'(q,a) = \delta_2(q,a) & \text{si } q \in Q_2, \\ \delta'(q,\epsilon) = \delta_1(q,\epsilon) & \text{si } q \in Q_1, \\ \delta'(q,\epsilon) = \delta_2(q,\epsilon) & \text{si } q \in Q_2, \\ \delta'(q_i,\epsilon) = \{i_1, i_2\}, \\ \delta'(q_i,a) = \emptyset \end{cases}$$

5.2 $e_1.e_2$

On a A_1 et A_2 qui reconnaissent e_1 et e_2, respectivement. Le but est de faire une lecture dans A_1 puis une dans A_2. Pour ça, on met les automates à la suite l'un de l'autre, en ajoutant des ϵ-transitions de chaque état terminal de A_1 vers l'état initial de A_2. Les états terminaux de A_1 ne le sont plus, tandis que l'état initial de A_2 devient également "normal".

Formalisation Soient $A_1 = \langle Q_1, \Sigma_1, i_1, F_1, \delta_1 \rangle$ et $A_2 = \langle Q_2, \Sigma_2, i_2, F_2, \delta_2 \rangle$, on renvoie

$$\langle Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, i_1, F_2, \delta' \rangle$$

où

$$\begin{cases} \delta'(q,a) = \delta_1(q,a) & \text{si } q \in Q_1 \\ \delta'(q,a) = \delta_2(q,a) & \text{si } q \in Q_2 \\ \delta'(q,\epsilon) = \delta_2(q,\epsilon) & \text{si } q \in Q_2 \\ \delta'(q,\epsilon) = \delta_1(q,\epsilon) & \text{si } q \in Q_1 \backslash F_1, \\ \delta'(q,\epsilon) = \delta_1(q,\epsilon) \cup \{i_2\} & \text{si } q \in Q_1 \cap F_1 \end{cases}$$

5.3 e_1^*

On a A_1 qui reconnaît e_1. Le but est d'ajouter un état S qui va permettre de faire des "tours de manège" dans A_1. S est donc initial, et terminal (car e_1^* accepte forcément le mot vide), a une ϵ-transition vers l'état anciennement initial et en reçoit une depuis tout état anciennement terminal (cf. le polycopié pour plus de détails).

Formalisation Soit $A_1 = \langle Q_1, \Sigma_1, i_1, F_1, \delta_1 \rangle$, on renvoie

$$\langle Q_1 \cup \{S\}, \Sigma_1, S, \{S\}, \delta' \rangle$$

où

$$\begin{cases} \delta'(q,a) = \delta_1(q,a) & \\ \delta'(q,\epsilon) = \delta_1(q,\epsilon) & \text{si } q \in Q_1 \backslash F_1, \\ \delta'(q,\epsilon) = \delta_1(q,\epsilon) \cup \{S\} & \text{si } q \in Q_1 \cap F_1, \\ \delta'(S,\epsilon) = \{i_1\} & \\ \delta'(S,a) = \emptyset & \end{cases}$$