0.1 Traduction récursive d'une expression rationnelle en un automate

1. Au mot vide ε, on associe l'automate $\langle X, \{q_0\}, \{q_0\}, \{q_0\}, \emptyset\rangle$
2. À l'expression rationnelle x ($x \in X$), on associe l'automate
$$\langle X, \{q_0, q_1\}, \{q_0\}, \{q_1\}, \{(q_0, x, q_1)\}\rangle$$
3. Soit R une expression rationnelle, associée à l'automate $\langle X, Q_R, I_R, F_R, \delta_R\rangle$;
à R^*, on associe l'automate $\langle X, Q_R \cup \{Q_0\}, \{Q_0\}, \{Q_0\}, \delta'_R\rangle$[1],
où $\delta'_R = \delta_R \cup \bigcup_{q\in I_R}(Q_0, \varepsilon, q) \cup \bigcup_{q\in F_R}(q, \varepsilon, Q_0)$
4. Soient R et S deux expressions rationnelles auxquelles ont été associés respectivement $\langle X, Q_R, I_R, F_R, \delta_R\rangle$ et $\langle X, Q_S, I_S, F_S, \delta_S\rangle$, dont on suppose que tous les états sont distincts ($Q_S \cap Q_R = \emptyset$).
   (a) À RS on associe l'automate
$$\left\langle X, Q_R \cup Q_S, I_R, F_S, \delta_R \cup \delta_S \cup \bigcup_{q\in F_R}\bigcup_{q'\in I_S}(q, \varepsilon, q')\right\rangle$$
   (b) À $R|S$ on associe l'automate
$$\left\langle X, Q_R \cup Q_S, Q_0, F_R \cup F_S, \delta_R \cup \delta_S \cup \bigcup_{q\in I_R\cup I_S}(Q_0, \varepsilon, q)\right\rangle$$

Voir le tableau 1 pour un récapitulatif.

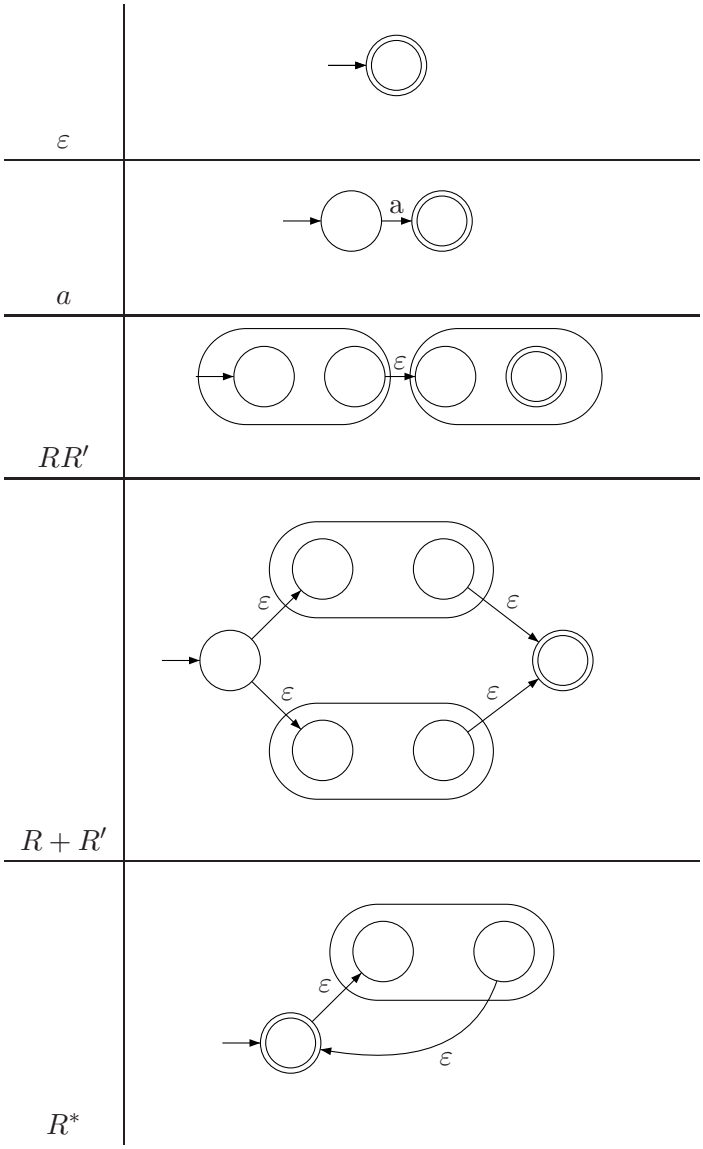
0.2 Automate → expression rationnelle

Les automates généralisés que nous allons manipuler vérifient les contraintes suivantes :
- L'état initial possède une transition vers tous les autres états (éventuellement une transition "non passsante", étiquetée par $\emptyset$) ;
- Aucun état n'a de transition vers l'état initial
- Il existe un et un seul état d'acceptation,
  - distinct de l'état initial,
  - qui n'a aucune transition vers les autres états
  - qui est atteint par tous les autres états
- Tous les états (sauf initial et acceptation) possèdent une et une seule transition vers chacun des autres états.

La première étape de l'algorithme, qui consiste à **transformer l'automate initial en automate généralisé**, revient à ajouter un état initial et un état final vérifiant les contraintes précédentes (reliés par des ε-transitions aux états initial et finals de l'automate initial) ; puis à faire en sorte que tous les états soient reliés à tous les états, soit par une transition marquée $\emptyset$ (lorsqu'il n'existe pas de chemin entre les deux états), soit par une transition unique portant l'étiquette de l'automate initial (ou l'union des étiquettes s'il y a plusieurs transitions entre deux états). Il est facile de vérifier que l'automate généralisé reconnaît le même langage (les transitions $\emptyset$ sont "non passantes").

La seconde étape est une **réduction itérative du nombre d'états** de l'automate généralisé, jusqu'à obtenir un automate n'ayant que deux états, et une transition qui sera étiquetée par l'expression rationnelle correspondant au langage reconnu. Il est clair que cette réduction doit conserver à chaque étape le langage reconnu.

[1] Q_0 est un nouvel état t.q. $Q_0 \notin Q_R$.



Tab. 1 – D'une expression rationnelle vers un automate

Chaque étape de cette réduction consiste en la **suppression d'un état** en réarrangeant l'automate de façon à reconnaître le même langage.

Soit q_e l'état à supprimer, il faut considérer **tous** les couples d'états (q_1, q_2), et faire en sorte que les transitions allant de q_1 à q_2 en passant ou non par q_e soient "synthétisées" sur une seule transition représentant tous les chemins possibles. Cette élimination est représentée par la figure 1.

Il est important de noter que l'élimination d'un état conduit à considérer **tous** les couples d'états de l'automate, y compris les couples (q_i, q_i) (mais en tenant compte de la définition d'un automate généralisé, donc le couple (q_0, q_i) est considéré, mais pas le couple $(q_i, q_0,)$ (où q_0 est l'état initial)).

FIG. 1 – Elimination d'un état, Algorithme de McNaughton & Yamada