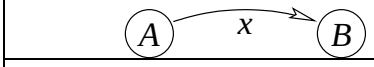
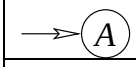
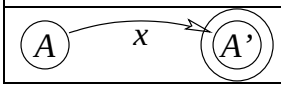
1.4.2 Grammaires et automates

1.4.2.1 Principe

Rappel une grammaire **régulière** (dite aussi linéaire) est une grammaire dont toutes les règles de production sont sous l'une des formes suivantes[2] : $A \to xB$
$A \to x$
$A \to \varepsilon$

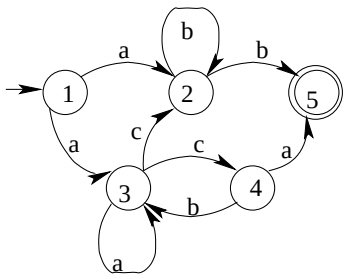
Note On peut toujours proposer une définition sans ε-production, ou plutôt sans autre ε-production qu'une règle $S \to \varepsilon$, où S est l'axiome, et S est inaccessible.

Le principe de correspondance entre automates et grammaires régulières est très intuitif : il correspond à l'observation que chaque transition dans un automate produit exactement un symbole, de même que chaque dérivation dans une grammaire régulière. Le tableau 1.2 résume cette correspondance, qui donne les bases des algorithmes dans les deux sens.

	$A \to xB$
	Axiome = A
	$B \to \varepsilon$
 (A' nouvel état)	$A \to x$

TAB. 1.2 – Correspondances automate ↔ grammaire régulière

L'application de l'algorithme est illustrée à la figure 1.3 avec un automate non déterministe.



$$
\begin{array}{lcl}
S_1 & \to & \mathrm{a}S_2 \\
 & | & \mathrm{a}S_3 \\
S_2 & \to & \mathrm{b}S_2 \\
 & | & \mathrm{b}S_5 \\
S_3 & \to & \mathrm{c}S_2 \\
 & | & \mathrm{c}S_4 \\
 & | & \mathrm{a}S_3 \\
S_4 & \to & \mathrm{b}S_3 \\
 & | & \mathrm{a}S_5 \\
S_5 & \to & \varepsilon
\end{array}
$$

FIG. 1.3 – Exemple Rec → Reg (automate non déterministe)

1.4.2.2 Automates → grammaires régulières

On peut partir d'un automate quelconque (non déterministe, non complet), il suffit de considérer une à une toutes les transitions et de produire les règles correspondantes d'après le tableau 1.2.

Si l'automate contient des transitions vides, on peut bien sûr s'en débarrasser (algorithme déjà vu), ou bien les traduire en productions singulières ( devient $A \to B$). Mais il faut ensuite supprimer les productions singulières (qui ne sont pas permises dans une grammaire régulière), avec un algorithme qui ressemble beaucoup à l'algorithme de suppression des ε-productions dans un automate.

[2] Où, conformément aux conventions habituelles, A et B sont des non-terminaux, et x est un symbole terminal. La définition donnée ici correspond à une grammaire linéaire/régulière **gauche**. Définition analogue possible à droite.

1.4.2.3 Grammaires régulières → automates

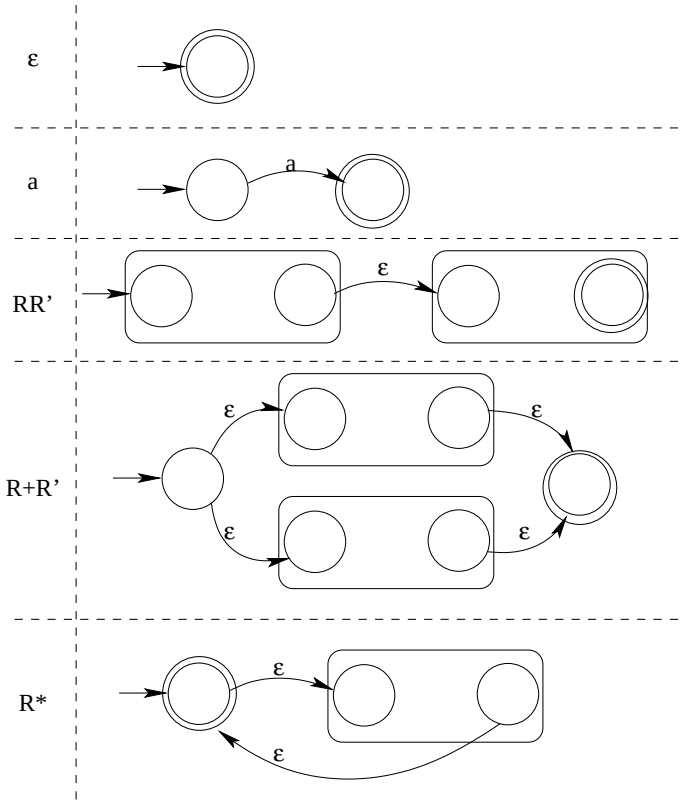
Partant d'une grammaire régulière, il suffit de créer un état pour chaque non-terminal, et de "traduire" chaque règle de production en utilisant le même tableau de correspondance. Le seul cas un peu particulier concerne les règles de la forme $A \to x$, pour lesquelles il suffit de remarquer que ce sont des règles terminales (la dérivation s'arrête nécessairement dès qu'une production de cette forme est déclenchée). Il faut créer un nouvel état, terminal (A' dans le tableau). Pour comprendre cette correspondance, on peut observer que la dérivation $A \to x$ est équivalente à une dérivation avec les deux règles $A \to xA'$, et $A' \to \varepsilon$.

1.4.3 Automates et expressions rationnelles

1.4.3.1 Expression rationnelle → Automate

On peut montrer (voir section "Propriétés de fermeture") que la *réunion*, la *concaténation*, et l'*étoile* peuvent être définis sur les automates ; il est donc possible, par exemple, de construire un automate qui reconnaît $L_1 \cup L_2$ par la réunion de l'automate qui reconnaît L_1 et de l'automate qui reconnaît L_2. Ces considérations permettent de définir facilement un algorithme de "traduction" d'une expression rationnelle quelconque en un automate reconnaissant le même langage.

Voici cet algorithme, spécifié d'abord sous forme mathématique, puis sous la forme d'un tableau de correspondance graphique, dans le même esprit que le tableau 1.2 donné plus haut (mais orienté cette fois-ci).



Tab. 1.3 – D'une expression rationnelle vers un automate