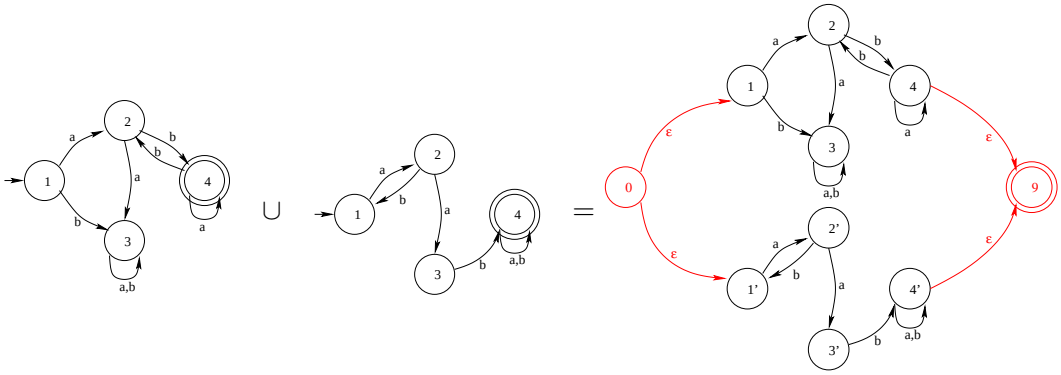
1.4.3 Propriétés de fermeture

Comme on a défini des opérations (*internes*) sur les langages, on peut envisager des opérations internes sur les automates. Voici les plus courantes.

1.4.3.1 Union

Pour réaliser une automate qui reconnaît l'union de deux langages, il suffit de faire en sorte que le nouvel automate comprenne tous les chemins du premier automate et tous ceux du second. Un moyen simple de procéder est de créer un nouvel état initial, duquel partent des ε-transitions vers les états initiaux des deux automates à réunir, et de créer un nouvel état d'acceptation, qui sera la cible par une ε-transition de tous les (anciens) états d'acceptation des deux automates. Le résultat est bien sûr non déterministe. Noter qu'on peut aussi garder les états d'acceptation sans un créer de nouveau.

Exemple :



Formulation mathématique : cf. la section "Théorème de Kleene".

1.4.3.2 Concaténation, étoile

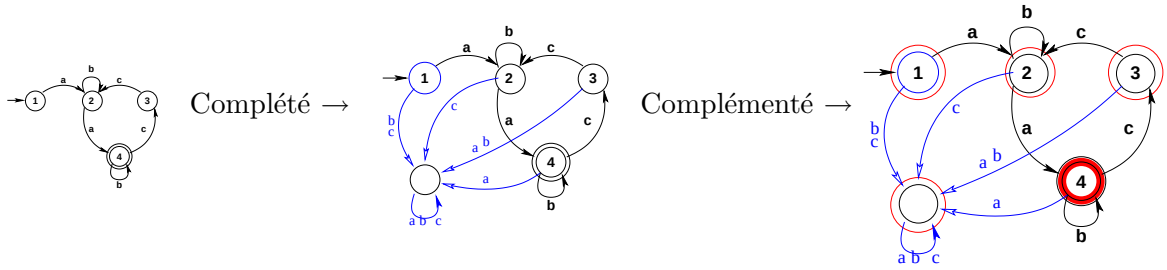
Il est facile d'imaginer, sur la même base que précédemment, comment créer un automate réalisant la concaténation de deux automates, ou l'étoile d'un automate. C'est en fait exactement ce que l'on fait dans l'algorithme de traduction d'une expression rationnelle en automate.

1.4.3.3 Complémentation

Le complément d'un langage $\mathcal{L}_1$ est l'ensemble de tous les mots du monoïde qui n'appartiennent pas à ce langage. En terme d'automate, il s'agit donc de tous les mots qui n'ont pas de chemin aboutissant à un état final dans l'automate reconnaissant $\mathcal{L}_1$.

L'algorithme pour construire le complément d'un automate est relativement intuitif : il suffit que tous les états d'échec de l'automate initial deviennent des états de réussite, et réciproquement. Pratiquement, il suffit de rendre terminaux les états non terminaux et réciproquement (on échange Q et $Q \setminus F$). Mais attention, il est nécessaire que tous les chemins possibles soient présent dans l'automate, et donc qu'il soit **complet** ; de même il est nécessaire que l'automate initial soit **déterministe**.

exemple :



En exercice, le lecteur est invité à se figurer ce qui se produit lorsque ces contraintes ne sont pas vérifiées, et que l'on applique l'algorithme (échange de Q et $Q \setminus F$).

1.4.3.4 Intersection

Théorie : on sait que $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$. On pourrait donc utiliser les algorithmes précédents. Mais il y a une autre méthode, moins fastidieuse (ne pas oublier que la complémentation nécessite d'abord une déterminisation).

Intuitivement, l'idée est de parcourir "en parallèle" les deux automates, et de ne garder que les chemins qui existent dans les deux automates. Pour cela, les états du nouvel automate sont des couples (q_i, q_j), où q_i appartient au premier automate et q_j au second. Pour chaque lettre de transition, on crée le nouvel état-couple atteint, et on continue.

	a	b
→ 1	2	4
2	4	3
← 3	3	3
4	4	4

	a	b
↔ 1	2	5
2	5	3
3	4	5
4	1	4
5	5	5

	a	b
→ (1,1)	(2,2)	(4,5)
(2,2)	(4,5)	(3,3)
(4,5)	(4,5)	(4,5)
(3,3)	(3,4)	(3,5)
(3,4)	(3,1)	(3,4)
← (3,1)	(3,2)	(3,4)
(3,2)	(3,4)	(3,3)
(3,5)	(3,5)	(3,5)

Les mêmes contraintes que précédemment s'appliquent : on part de deux automates **déterministes complets**. À noter aussi qu'un tel algorithme, comme l'algorithme de déterminisation, a le mérite de ne pas conserver les états non atteint depuis l'état initial.

1.5 Théorèmes d'équivalence

1.5.1 Le théorème triangulaire

On a établi (Kleene y a contribué) une ensemble de résultats d'équivalence que l'on peut résumer de la façon suivante[2]

$$\mathcal{L}_{\text{Rec}} = \mathcal{L}_{\text{Rat}} = \mathcal{L}_{\text{Reg}}$$

où

– $\mathcal{L}_{\text{Rec}}$ est la classe des langages **rec**onnaissables par un automate à nombre fini d'états ;
– $\mathcal{L}_{\text{Rat}}$ est la classe des langages que l'on peut décrire avec une expression **rat**ionnelle ;
– $\mathcal{L}_{\text{Reg}}$ est la classe des langages engendrés par une grammaire **rég**ulière.

On symbolise en général ce résultat sous la forme du triangle représenté à la figure 1.2.

[2]Le théorème de Kleene correspond à l'équation : $\mathcal{L}_{\text{Rec}} = \mathcal{L}_{\text{Rat}}$.