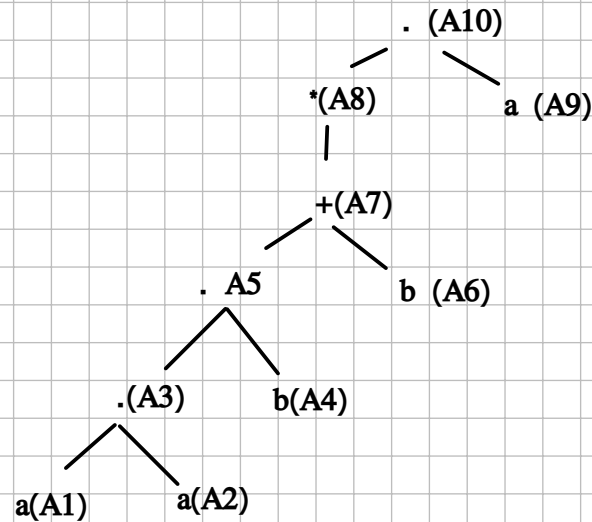


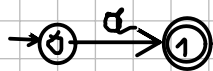
Expression de départ

$((a.a).b+b)^*.a$

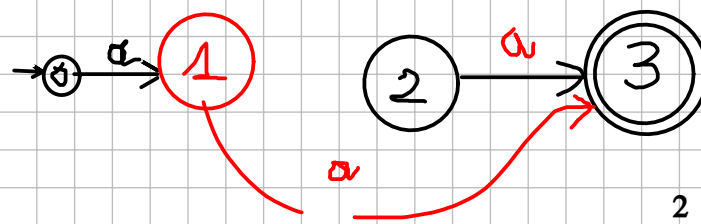
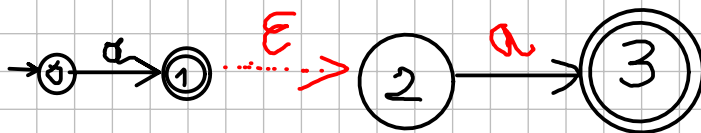
l'arbre de calcul correspondant à cette expression est



Toutes les feuilles de l'arbre (A1, A2, A4, A6, A9) sont de la forme un état initial, une lettre, un état final. Par exemple A1 est :



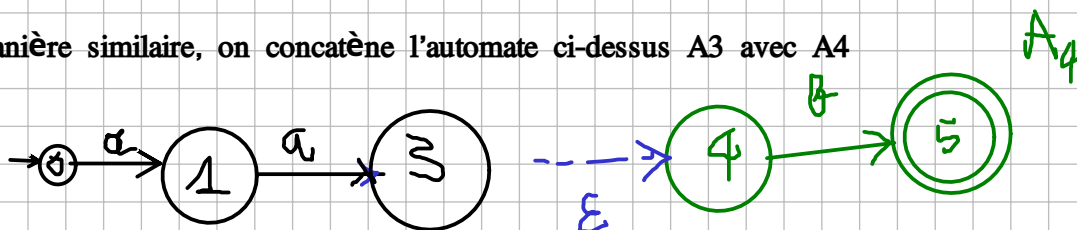
Et A2 a la même forme, en remplaçant 0 et 1 par de nouveaux états, disons 2 et 3.

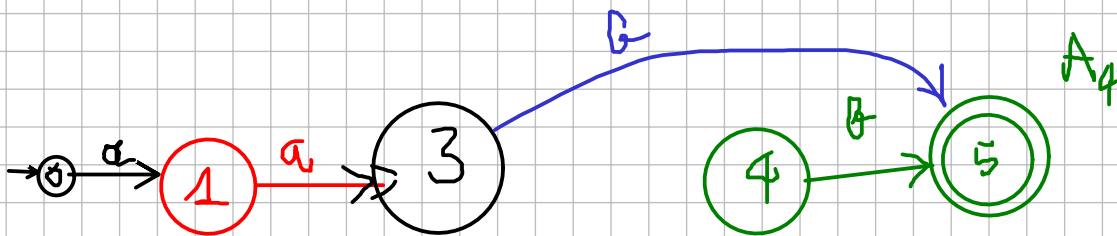


2 pas accessible je peux l'enlever

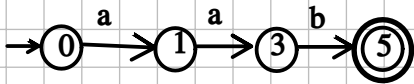


De manière similaire, on concatène l'automate ci-dessus A3 avec A4

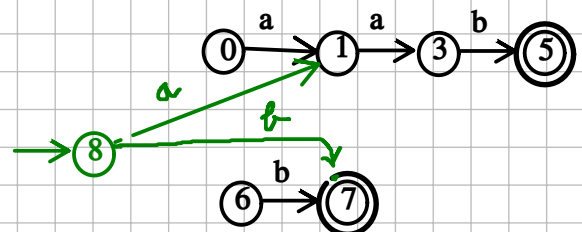
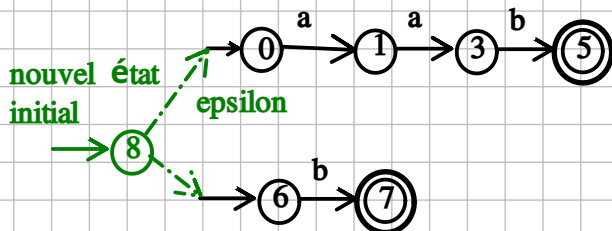
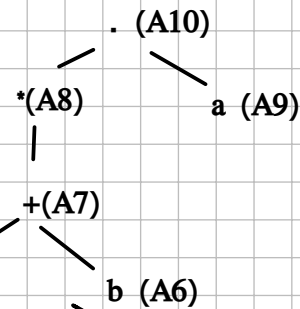
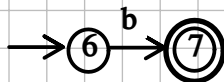




On peut simplifier en enlevant l'état 4 qui n'est plus accessible.
On obtient donc l'automate A5 suivant.

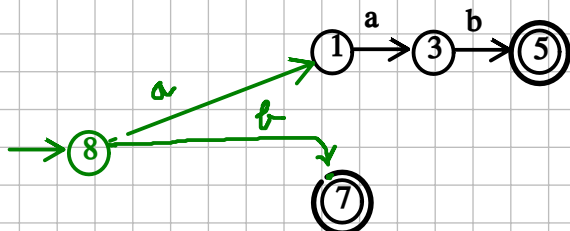


on continue la construction
en remontant dans l'arbre
Il faut faire l'union (opération +)
avec l'automate A6 ci-dessous
pour construire A7

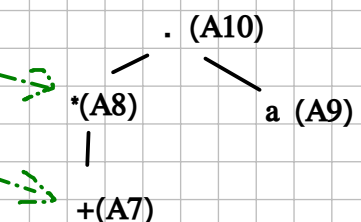


NB. 0 et n6 ne sont pas accessibles
on peut les enlever

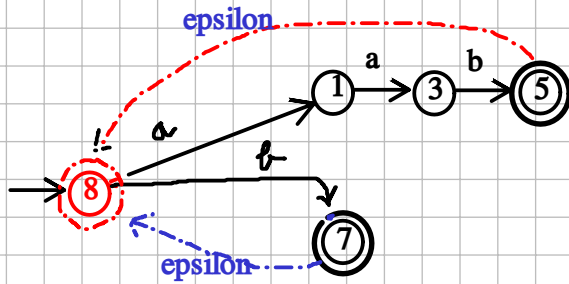
On obtient donc pour A7



On continue en prenant l'étoile de A7 pour obtenir A8



Pas besoin de standardiser A7 car pas de boucle ou cycle revenant sur 8.

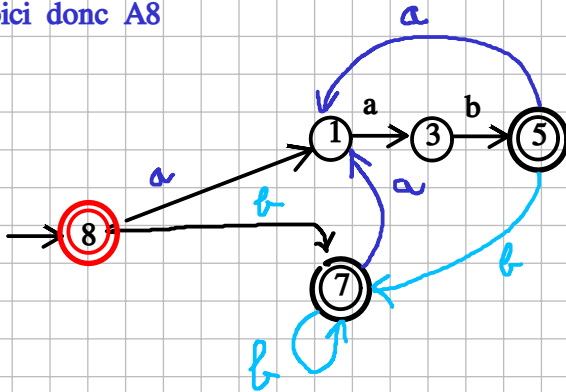


8 devient acceptant

(car epsilon appartient toujours à l'étoile d'un langage)

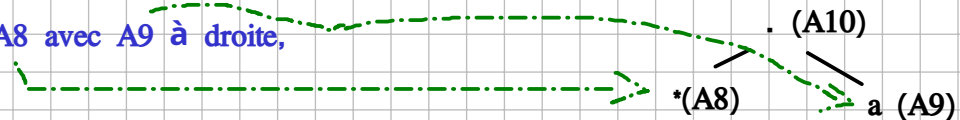
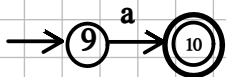
et on branche la fin de A7 (ici 5 et 7 sur le début 8).

voici donc A8

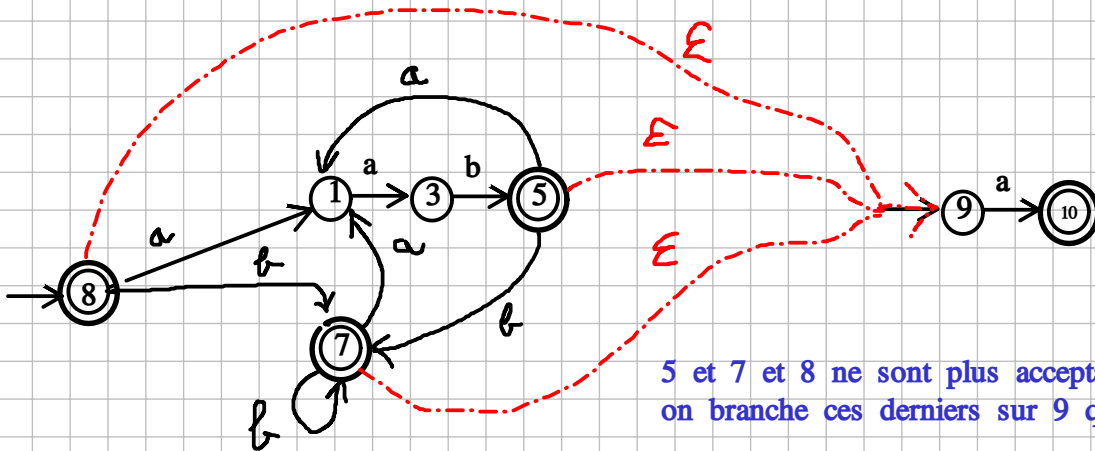


fin de l'arbre de l'expression:

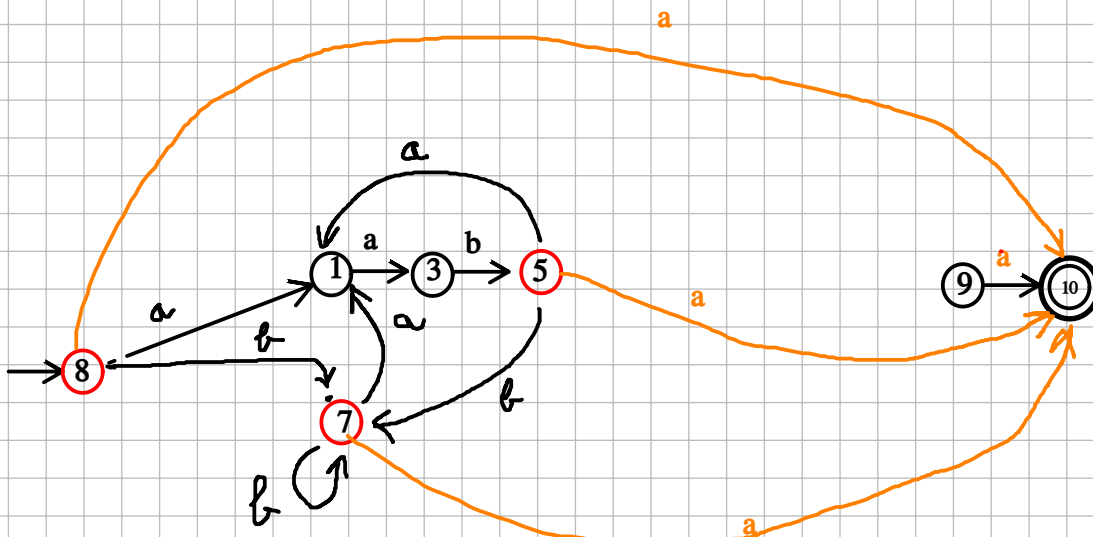
Il faut concaténer A8 avec A9 à droite, A9 étant :



On prépare la concaténation comme ci-dessous.



5 et 7 et 8 ne sont plus acceptants (car 9 ne l'est pas) on branche ces derniers sur 9 qui n'est plus initial.

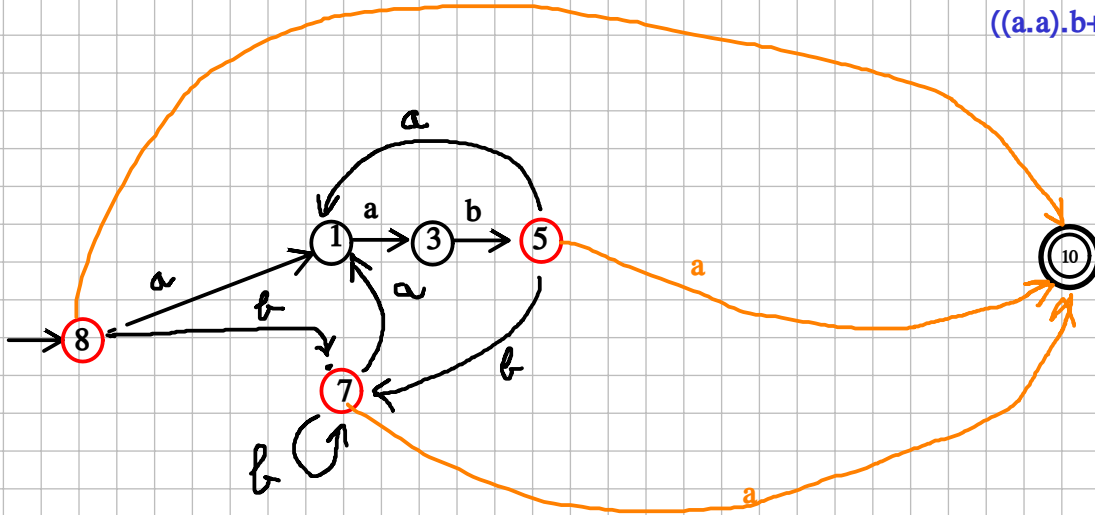


9 pas accessible on peut l'enlever

On a calculé A10 qui calcule par construction, le langage spécifié par la regexp du départ.

a

$((a.a).b+b)^*.a$



Attention dans l'arbre, on peut permuter des sous-arbres si l'opération est un +
(ça ne change pas le résultat, juste le numéro qu'on donne aux états)
mais pas pour l'opération .

