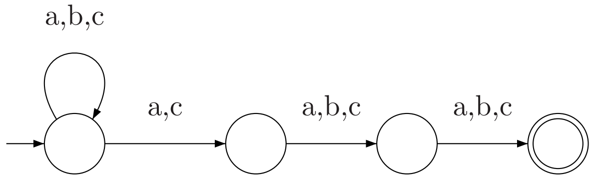
1 Révisions : Automates à états finis

1.1 Automates déterministes complets

Soit l'alphabet $X = \{a, b, c\}$. Donner des automates déterministes complets reconnaissant les langages suivants :

1. L'ensemble des mots dont la longueur est un multiple de 3.
2. L'ensemble des mots dans lesquels le motif *ab*, s'il apparaît, est suivi de *ccc*.
3. L'ensemble des mots se terminant par *b*.
4. L'ensemble des mots ne se terminant pas par *b*.
5. L'ensemble des mots non vides ne se terminant pas par *b*.
6. L'ensemble des mots contenant au moins un *a* et dont la première occurence de *a* n'est pas suivie par un *c*.
7. L'ensemble des mots comportant au moins 3 lettres et dont la troisième lettre à partir de la fin est un *a* ou un *c*.
   Correction : Erreur dans l'énoncé, cf. la version déterministe est assez difficile à obtenir directement. La version non déterministe est triviale :

   Rem : On peut effectivement remarquer que pour L˜(le langage miroir de L, i.e toutes les chaînes de L mais en ordre inverse), un automate déterministe est facile à trouver. Plus généralement, à partir d'un automate à états finis A (pas forcément déterministe), reconnaissant le langage L, on construit l'automate miroir A˜ (i.e. reconnaissant L˜) avec les étapes suivantes :
   (i) transformer A pour qu'il n'ait plus qu'un seul état final (si plusieurs au départ, ajouter un nouvel état, qui devient le seul état final, et ajouter des transitions epsilon des anciens états finaux vers ce nouvel état final).
   (ii) inverser les sens de toutes les transitions
   (iii) transformer l'état initial en état final et vice-versa
   On peut remarquer que A peut ne pas être déterministe même si A l'est, et vice-versa.
8. Le langage $L_2 = \{u \in X^*/\ \nexists v, w \in X^* \text{ t.q. } u = vbacaw\}$.

1.2 Automate non déterministe

Proposer un automate et une expression rationnelle pour le langage de tous les mots de $\{a, b, c\}^*$ dont *cac* est un sous-mot[1].

1.3 Complément d'un automate

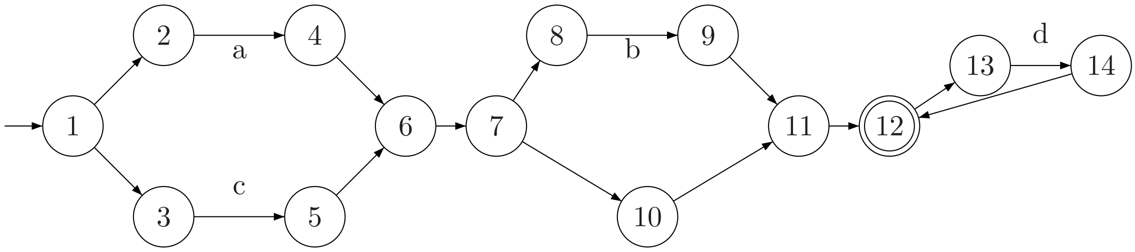
Donner l'algorithme passant d'un automate fini à un automate reconnaissant le complément. L'algorithme suppose un automate déterministe et complet. Justifier de manière informelle

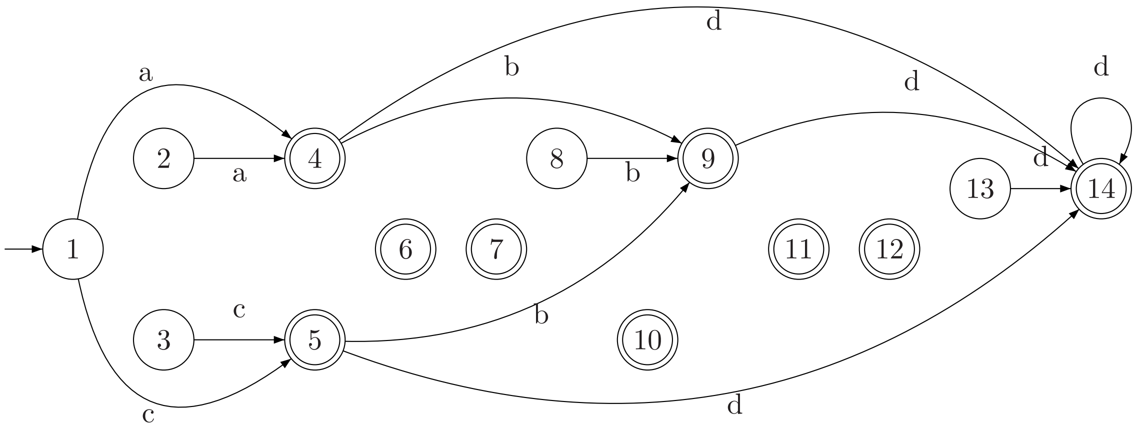
[1]Un *sous-mot* de *u* est une sous-suite de lettres — non nécessairement contiguë — de *u*. À distinguer d'un *facteur*. Exemple : *pis* est un sous-mot de *produits*.

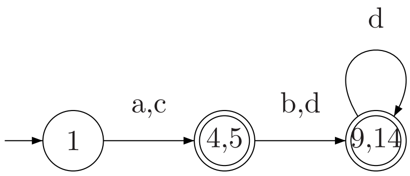
ces deux contraintes.
Correction : Pour un automate fini déterministe complet, il suffit de garder les mêmes états, les mêmes transitions, mais d'inverser la propriété "état final" : les états non finaux deviennent finaux et vice-versa.
Supposons que l'automate A ne soit pas complet. Alors il existe au moins un mot x pour lequel aucun parcours n'existe dans A. Donc $x \notin L_A$, et donc $\in \overline{L_A}$. Or d'après la procédure de complémentation ci-dessus, il n'existe pas dans à $\overline{A}$ de parcours pour x, et donc $x \notin \overline{L_A}$. Contradiction.
Dans le cas d'un automate non déterministe, la procédure peut sous certaines conditions fonctionner, mais pas toujours. Un cas d'échec est le suivant : supposons qu'il existe un mot x pour lequel il existe au moins deux parcours p et q contradictoires dans A, i.e. p terminant sur un état d'acceptation, mais pas q. Alors $x \in A$, par l'existence de p. Mais dans $\overline{A}$, p ne termine pas par un état d'acceptation, alors que q oui. Donc, du fait de q, $x \in \overline{A}$. Contradiction.

2 Expressions rationnelles vers Automates

1. Donner un automate qui reconnaît le langage décrit par les expressions régulières suivantes, (i) en appliquant la méthode systématique (p.20 dupoly), (ii) par une méthode directe, donnant un automate le plus simple possible.
   – (a|c)(b|ε)d*
   – (a|b)(a|b)*
2. Suppression des ε-transitions : étudiez l'algorithme de suppression des ε-transitions (p.13 du poly). Appliquer cet algorithme à l'automate obtenu par la méthode systématique pour (a|c)(b|ε)d*.
   Correction :
   Voici :
   - l'automate obtenu par méthode systématique,
   - la matrice représentant $\varepsilon+$,
   - l'automate obtenu par suppression des ε-transitions (sauf que l'on n'a pas reporté des transitions partant d'états inaccessibles à partir de l'état initial)
   - et l'automate obtenu par minimisation, que l'on aurait pu trouver directement







$\varepsilon+$	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1		x	x											
2														
3														
4						x	x	x		x	x	x	x	
5						x	x	x		x	x	x	x	
6							x	x		x	x	x	x	
7								x		x	x	x	x	
8														
9											x	x	x	
10											x	x	x	
11												x	x	
12													x	
13														
14												x	x	

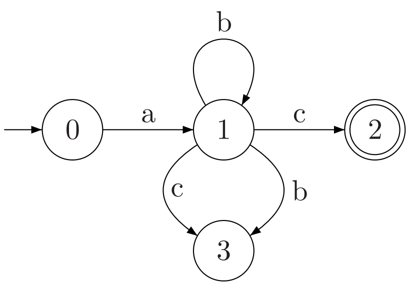
3 Minimisation

1. Minimiser l'automate dont la table de transition est la suivante ($X = \{a, b\}$; les états sont désignés par des lettres majuscules ; seul H est un état d'acceptation) :

δ	A	B	C	D	E	F	G	H
a	B	G	A	C	H	C	G	G
b	F	C	C	G	F	G	E	C

Rappel : **principe de la minimisation** : Intuitivement (!) on peut fusionner deux états q et r si à partir de ces deux états, le même langage est reconnu. Ce qui peut se formuler par : pour tout mot x de X^*, reconnaître x à partir de q ou de r amène soit à un état final pour les deux cas, soit à un état non final pour les deux cas.
Plus formellement, on peut fusionner sans effet sur le langage reconnu, des états dits "indistinguables" :
q et r sont indistinguables dans un automate fini déterministe complet, dont l'ensemble des état finaux est noté F, si et seulement si $\forall x \in X^*, \delta^*(q,x) \in F \Leftrightarrow \delta^*(r,x) \in F$
Deux états sont distinguables ssi ils sont pas indistinguables. On peut définir récursivement la distinguabilité :
En effet on peut montrer facilement que deux états q et r sont distinguables ssi :
(i) l'un est final et l'autre pas : $\varepsilon \in$ au langage de l'un, mais pas de l'autre
(ii) ou $\exists a \in X$ / les états $\delta(q,a)$ et $\delta(r,a)$ sont distinguables
C'est cette récurrence que capture l'algorithme de minimisation vu en cours.

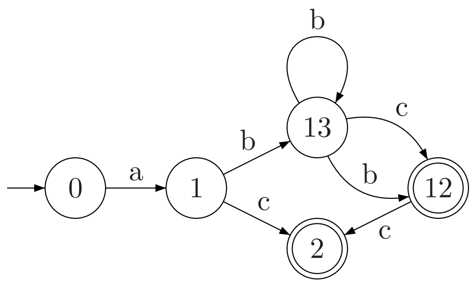
2. Proposer un automate **minimal** qui reconnaisse le même langage que l'expression rationnelle $a(b|bc)^*c$. (Il peut être intéressant de passer par une étape de déterminisation.)
Correction : Automate non déterministe (méthode non systématique) :



Déterminisation et complétion (on rajoute le puits 5, et tous les trous de la table sont remplis par 5) :

	a	b	c
$\to$ 0	1	5	5
1	5	1,3	2
$\leftarrow$ 2	5	5	5
3	5	5	5
1,3	5	1,3	1,2
$\leftarrow$ 1,2	5	1,3	2

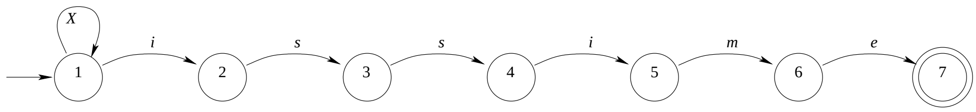
Ce qui donne :



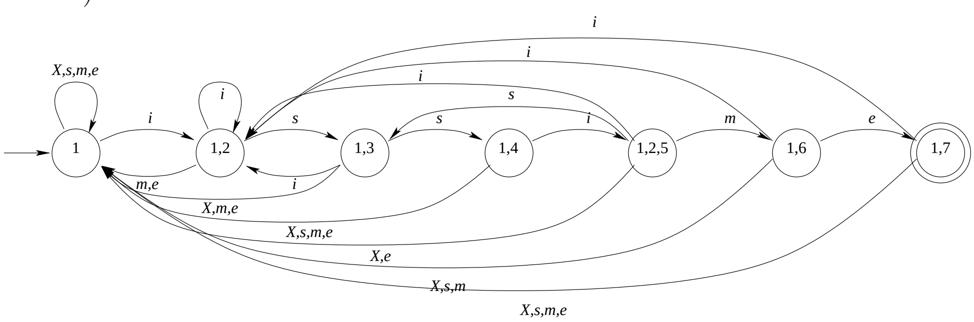
Minimisation : l'algorithme conduit à séparer tous les états, donc l'automate déterminisé est minimal.

	0	1	2	13	12
5	X2	X1	X0	X1	X0
12	X0	X0	X1	X0	
13	X1	X2	X0		
2	X0	X0			
1	X1				

3. Soit $X = \{a, b, c, \ldots, z\}$. Proposer un automate **déterministe** et **minimal** qui reconnaisse le langage $X^*issime$[2]. Peut-on proposer une généralisation sur le nombre minimal d'états d'un automate reconnaissant X^*u pour $u \in X^*$, en fonction de la longueur de u ?
Correction : Version non déterministe (X représente tout l'alphabet) :



Version déterministe (et aussi complète) (X représente l'alphabet privé de i, m, s, e) :



On peut vérifier facilement que cette version est minimale : chaque itération de l'algorithme sépare l'état le plus à droite de la classe de tous les autres.

Le suffixe u à reconnaître doit donner lieu à un chemin complet, de longueur égale à $|u|$. La reconnaissance du reste du mot, quant à elle, ne "coûte" qu'un état. Le nombre minimal d'états pour reconnaître un langage de la forme X^*u est donc $|u|+1$.

2 Lemme de pompage

Montrer grâce au lemme de pompage que le langage L des mots dont la longueur est un nombre premier n'est pas un langage rationnel.
Correction : Supposons que L est rationnel. Alors il vérifie le lemme de pompage. Soit k l'entier spécifié par ce lemme. Soit $x \in L/|x| \geq k$. Par le lemme, $\exists u, v, w/x = uvw$ et $|u| + |v| + |w|$ est un nombre premier. Et $\forall i > 0, uv^iw \in L$. C'est vrai en particulier pour $i = |u| + |w|$, donc $y = uv^{|u|+|w|}w \in L$, et donc sa longueur est un nombre premier. Or $|y| = |u| + (|v| * (|u| + |w|)) + |w| = (|v| + 1) * (|u| + |w|)$, qui n'est pas un nombre premier. Contradiction. Donc L n'est pas rationnel.

[2]Mots formés d'un mot quelconque de X^* suivi des lettres i, s, s, i, m, e.