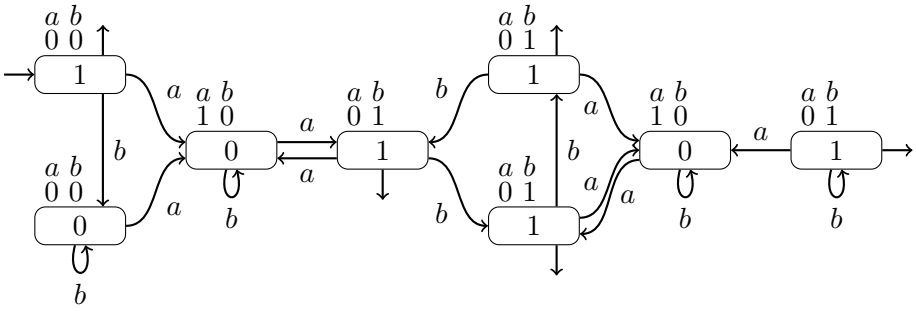
Solution - TD Feuille 3 - Minimisation des automates et Résiduels

Informatique Théorique 2 - Unité J1INPW11
Licence 3 - Université Bordeaux 1

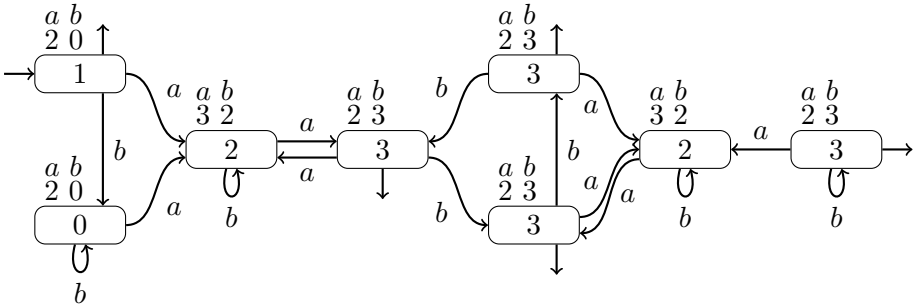
Solution de l'exercice 1 :

On observe que pour cet exercice, les deux automates à minimiser sont déjà déterministe. En conséquence on peut sauter l'étape de déterminisation et directement passer à la minimisation. On va détailler l'algorithme de Moore en faisant un schéma pour chaque étape :

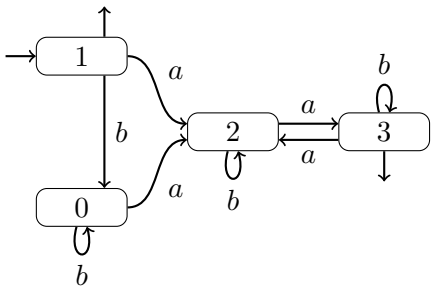
Automate $\mathcal{A}_1$. On commence par $\mathcal{A}_1$, la partition initiale est donnée par les états finaux (ce sont les seul état qui acceptent le mot vide) :



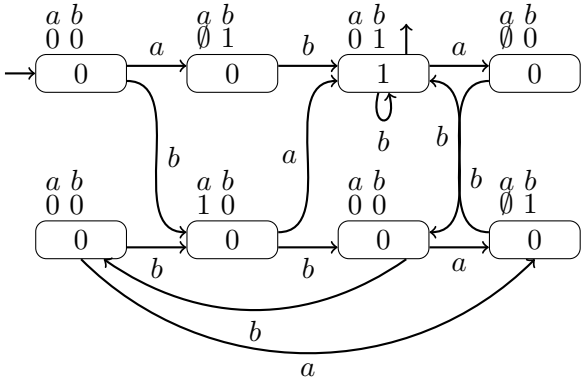
On met la partition à jour :



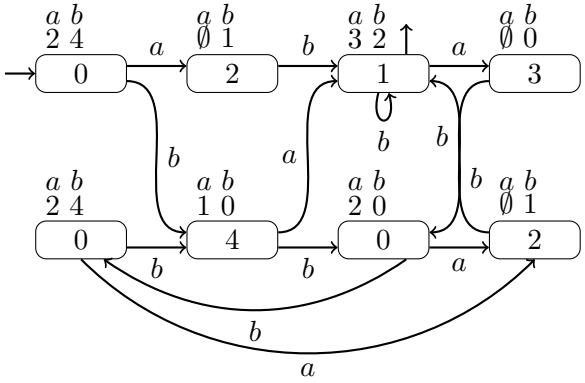
On observe que la partition n'est plus raffinée, on a donc bien les classes de Nérode. En fusionnant les états qui se trouvent dans la même classe on obtient l'automate minimal suivant :



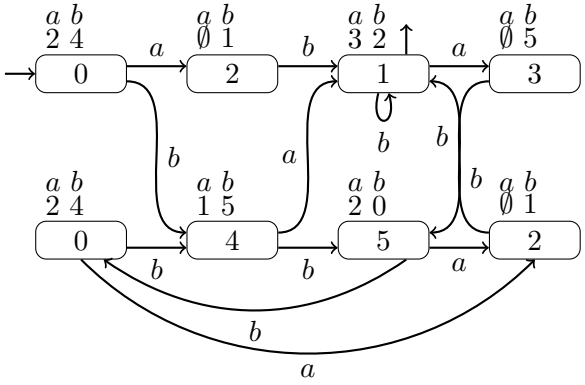
Automate $\mathcal{A}_2$. On passe maintenant à l'automate $\mathcal{A}_2$ dont voici la partition initiale :



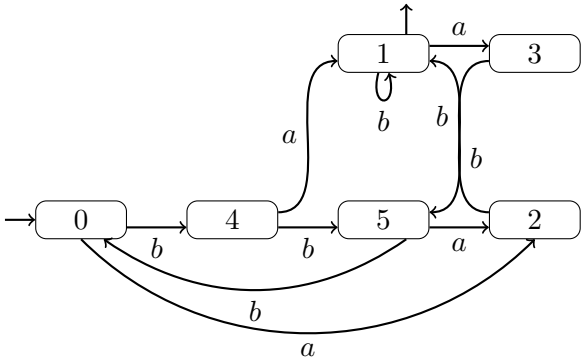
On met à jour la partition :



On met à nouveau à jour la partition :



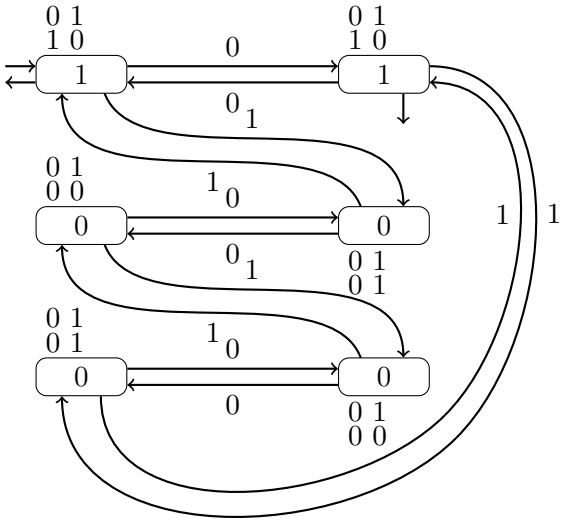
La partition est maintenant stable. Elle décrit donc les classes de Nérode. En fusionnant les états équivalents on obtient l'automate minimal suivant :



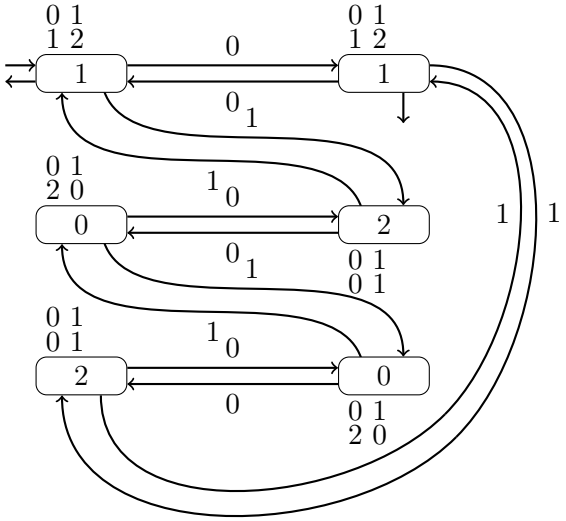
Solution de l'exercice 2 :

Le premier automate (lecture du mot de gauche à droite) est déjà minimal. Le moyen le plus simple de d'en apercevoir est d'utiliser l'algorithme du double renversement : l'automate étant déterministe et co-déterministe (l'automate miroir est lui même déterministe), cet algorithme laisse l'automate inchangé.

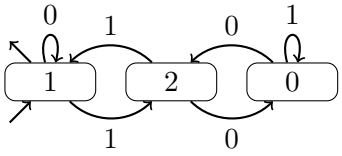
On va confirmer ce résultat une seconde fois en appliquant l'algorithme de Moore au second automate (lecture du mot de droite à gauche) pour constater qu'on obtient bien le miroir du premier automate. Voici la parition initiale en états finaux et non finaux :



On met à jour la partition :



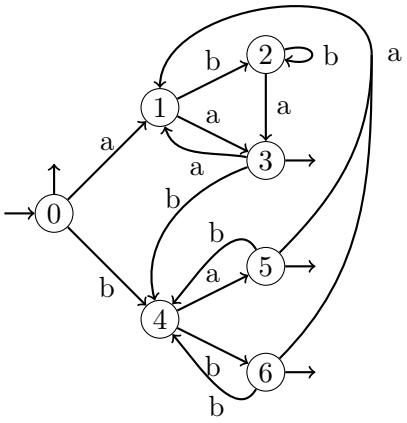
La partition n'étant plus raffinée elle représente les classes de Nérode, on obient donc l'automate minimal en fusionnant les états équivalents. On constate bien qu'on obtient l'automate miroir du premier automate.



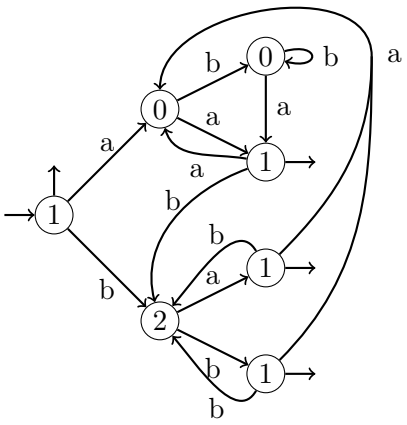
Solution de l'exercice 3 :

On souhaite comparer les quatres langages. Pour celà on commence par calculer l'automate minimal de chaque langage. Il suffira ensuite de comparer ces automates. En effet l'automate minimal est un objet canonique ne dépendant que du langage, deux lanages sont donc égaux si ils ont le même automate minimal (modulo renommage des états).

Expression Rationnelle $(ab^*a + b(a + b))^*$. On commence par construire un automate par la méthode de Glushkov :



On souhaite maintenant construire l'automate minimal du langage. Pour celà il faut d'abord déterminiser puis minimiser l'automate ci-dessus. Par chance on a déjà un automate déterministe, on peut donc directement passer à l'algorithme de Moore. On donne ci-dessous la partition finale obtenue après stabilisation de l'algorithme :



On termine en fusionnant les états équivalents pour obtenir l'automate minimal :

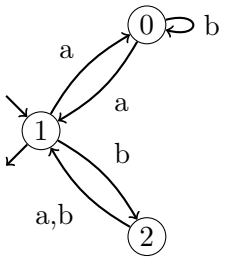
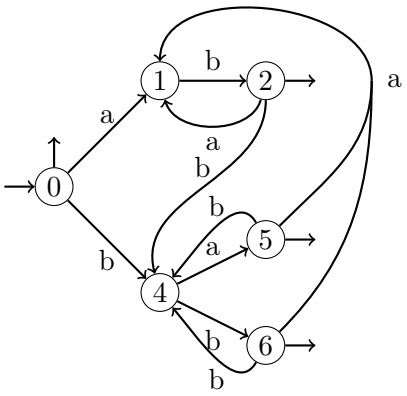
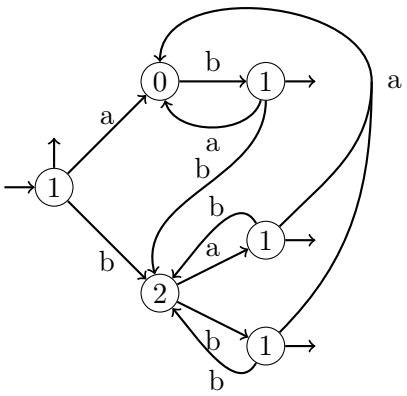


FIGURE 1 – Automate minimal pour $(ab^*a + b(a + b))^*$

Expression Rationnelle $(ab + b(a + b))^*$. On commence par construire un automate par la méthode de Glushkov :



Encore une fois, l'automate que nous donne la méthode de Glushkov est déjà déterministe. On peut donc directement appliquer l'algorithme de Moore sans passer par l'étape de déterminisation. On obtient la partition suivante à la fin de l'algorithme :



On fusionne ensuite les états équivalents ce qui donne l'automate minimal suivant :

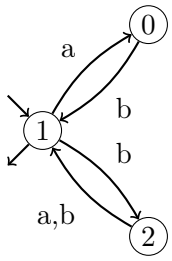
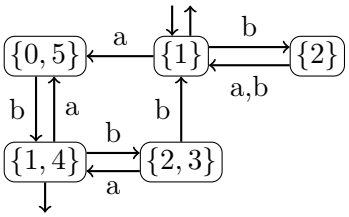
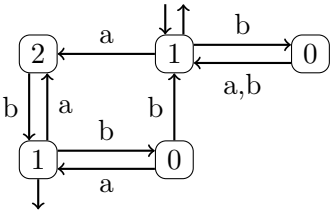


FIGURE 2 – Automate minimal pour $(ab + b(a + b))^*$

Automate $\mathcal{A}_3$. Pour minimiser $\mathcal{A}_3$, on doit d'abord le déterminiser. On donne ci dessous le résultat de l'algorithme de déterminisation :



On peut maintenant appliquer l'algorithme de Moore. La partition finale obtenue est représentée ci-dessous :



L'automate minimal s'obtient ensuite en fusionnant les états équivalents :

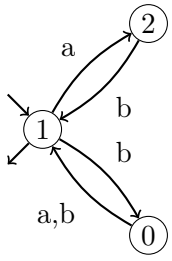
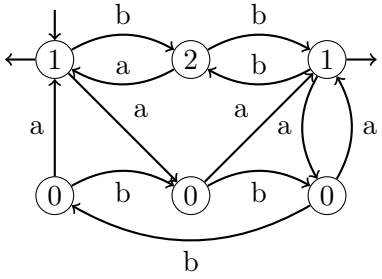


FIGURE 3 – Automate minimal pour $\mathcal{A}_3$

Automate $\mathcal{A}_4$. L'automate $\mathcal{A}_4$ est déjà déterministe, on passe donc directement à l'algorithme de Moore. La partition obtenue est la suivante :



Après fusion on obtient l'automate minimal suivant :

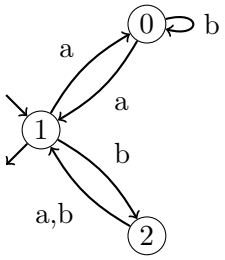


FIGURE 4 – Automate minimal pour $\mathcal{A}_4$

Maintenant que nous avons construit l'automate minimal pour chacun des quatre langages, on peut les comparer. On constate que modulo renommage des états les langages de $\mathcal{A}_3$ et $(ab + b(a + b))^*$ ont le même automate minimal et sont donc égaux. Il en va de même pour les langages de $\mathcal{A}_4$ et $(ab^*a + b(a + b))^*$.

Solution de l'exercice 4 :

1. On a :
   - $L_1^{-1} \cdot L_2 = \{\epsilon, a, b, ab, ba\}$.
   - $L_2^{-1} \cdot L_1 = \{\epsilon, a, bab, abab, ababab\}$.
   - $\epsilon^{-1} \cdot L_1 = L_1$.
   - $[(a + b)^*]^{-1} \cdot L_1 = \{\epsilon, a, b, aa, ab, ba, bab, abab, babab, ababab, aababab\}$.
   - $\emptyset^{-1} \cdot L_1 = \emptyset$.
2. Si $\epsilon \in L_1$ alors $L_2 \subseteq L_1^{-1} \cdot L_2$. Si $L_2 = \emptyset$ alors $L_1^{-1} \cdot L_2 = \emptyset$.

Solution de l'exercice 5 :

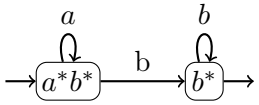
1. On donne les résiduels pour chaque langage. On commence par $L_1 = a^*b^*$ qui a trois résiduels. Soit w un mot :
   - Si $w \in a^*$ alors $R_w = L_1 = a^*b^*$. Par exemple $R_\epsilon = a^*b^*$.
   - Si $w \in a^*b^*$ alors $R_w = b^*$. Par exemple $R_a = b^*$.
   - Sinon $R_w = \emptyset$.

   On passe maintenant à $L_2 = a^*ba + b^*aba$. Ce second langage possède neuf résiduels. Soit w un mot :
   - Si $w = \epsilon$, $R_\epsilon = L_2 = a^*ba + b^*aba$.
   - Si $w \in aa^*$ alors $R_w = a^*ba$.

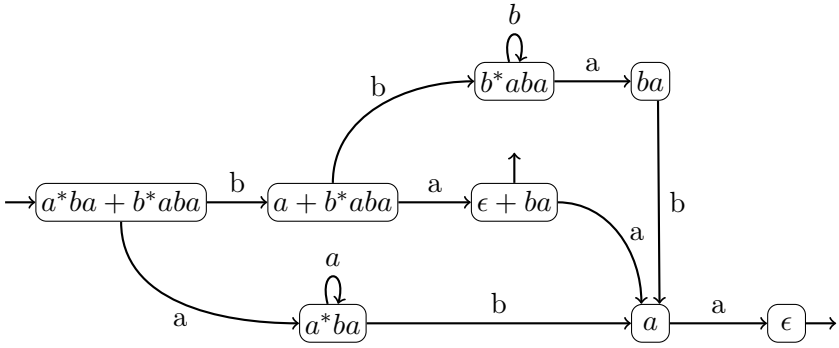
– Si $w \in aa^*b + b^*ab$, alors $R_w = \{a\}$.
– Si $w = b$, alors $R_b = a + b^*aba$.
– Si $w = ba$, alors $R_{ba} = \epsilon + ba$.
– Si $w \in bbb^*$, alors $R_w = b^*aba$.
– Si $w \in bbb^*a$, alors $R_w = ba$.
– Si $w \in L_2$, alors $R_w = \epsilon$.
– Sinon $R_w = \emptyset$.

Pour le langage $L_3 = \{a^n b^n \mid n \geq 0\}$, on va voir qu'on a une infinité de résiduels. Soit w un mot :
– Si il existe $i \geq 0$ tel que $w = a^i$ alors $R_{a^i} = \{a^j b^k \mid i + j = k\}$.
– Si il existe $i > j \geq 0$ tel que $w = a^i b^j$ alors $R_w = \{b^{i-j}\}$.
– Sinon $R_w = \emptyset$.

2. On sait qu'un langage est régulier si et seulement si il a un nombre fini de résiduels. On peut donc d'ores et déjà conclure que seuls L_1 et L_2 sont réguliers. De plus on sait que l'automate minimal d'un langage régulier a pour ensemble d'états l'ensemble des résiduels du langage, chaque mot étant accepté par l'état correspondant à son résiduel. Le calcul effectué à la question précédente nous permet donc de donner les automates minimaux de L_1 et L_2. On commence avec L_1, on étiquette chaque état avec son résiduel. C'est-à-dire que l'étiquette de chaque état représente l'ensemble des mots qui sont acceptés par l'automate en partant de cet état :



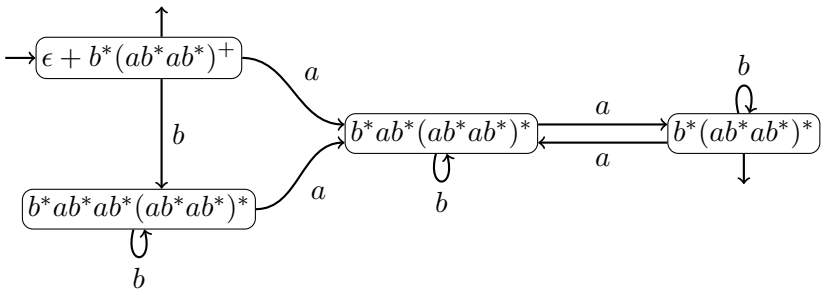
On donne maintenant l'automate des résiduels de L_2 :



Solution de l'exercice 6 :

On sait que pour un langage régulier L chaque résiduel correspond à un état de l'automate minimal (plus le résiduel vide si l'automate n'est pas complet). En particulier pour chaque état q, le résiduel associé est égal à l'ensemble des mots acceptés par l'automate en partant de q. C'est aussi le résiduel R_w pour tout mot w dont le calcul arrive dans l'état q sur l'automate minimal. L'automate minimal donne donc une description complète des résiduels du langage associé.

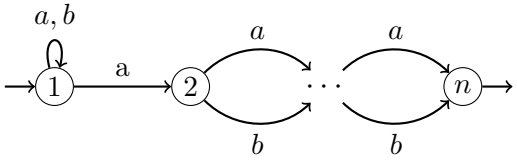
On illustre cette propriété en reprenant l'automate minimal du langage de $\mathcal{A}_1$ et en remplaçant l'étiquette de chaque état par une expression rationnelle pour le résiduel correspondant :



Observons que l'automate est complet, il n'y a donc pas de résiduel vide pour ce langage.

Solution de l'exercice 7 :

1. On donne au automate non déterministe à n états pour L :



2. On souhaite montrer qu'il n'existe pas d'automate déterministe ayant moins de 2^{n-1} états pour le langage L. On va donc montrer que l'automate minimal a plus de 2^{n-1}. On rappelle que chaque état de l'automate minimal correspond à un résiduel non vide du langage. Il nous suffit donc de montrer que L possède plus de 2^{n-1} résiduels. C'est ce que nous allons faire.

   Observons tout d'abord que L est le langage des mots dont la $(n-1)$-ème lettre en partant de la droite est un a.

   Considérons l'ensemble $\{1, 2, \ldots, n-1\}$, à chaque sous ensemble $E \subseteq \{1, 2, \ldots, n-1\}$ on associe un mot w_E de la façon suivante :

   – w_E a $n-1$ lettres.
   – si $i \in E$ alors la i-ème lettre de w_E en partant de la droite est un a.
   – si $i \notin E$ alors la i-ème lettre de w_E en partant de la droite est un b.

   Par exemple $w_{\{1\}} = b^{n-2}a$ et $w_{\{2,n-1\}} = ab^{n-4}ab$. On va montrer que si $E \neq E'$ alors $R_{w_E} \neq R_{w_{E'}}$.

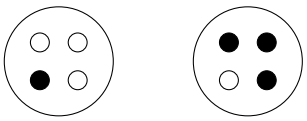
   Si $E \neq E'$ alors il existe forcément $i \in \{1, 2, \ldots, n\}$ tel que $i \in E$ et $i \notin E'$ ou tel que $i \in E'$ et $i \notin E$. Les deux cas étant symétriques, on suppose qu'on est dans le premier. Par définition, w_E contient un a à la i-ème position en partant de la droite et $w_{E'}$ contient un b à la i-ème position en partant de la droite. Il en découle que $w_E a^{n-1-i} \in L$ et $w_{E'} a^{n-1-i} \notin L$, donc $a^{n-1-i} \in R_{w_E}$ et $a^{n-1-i} \notin R_{w_{E'}}$. On en déduit que $R_{w_E} \neq R_{w_{E'}}$

   De plus pour tout $E \subseteq \{1, 2, \ldots, n-1\}$, on constate que $a^{n-1} \in R_{w_E}$. Donc L a au moins autant de résiduels non vides qu'il y a de sous-ensembles $E \subseteq \{1, 2, \ldots, n-1\}$, c'est-à-dire 2^{n-1}.

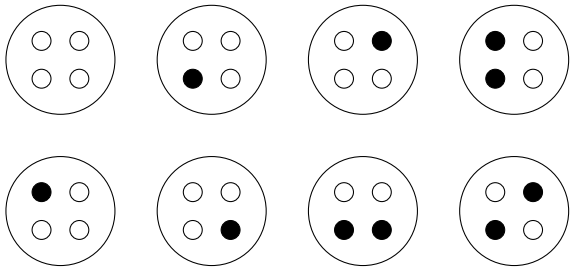
Solution de l'exercice 8 :

De manière générale il y a seize configurations possibles du plateau. On peut cependant réduire ce nombre en constatant que certaines conditions sont équivalentes. Tout d'abord,

puisque la condition de victoire impose seulement d'avoir tous les verres dans le même sens mais n'impose rien sur ce sens, on constate que retourner les quatres verres laisse le barman dans une situation équivalente pour le reste du jeu. Par exemple les deux configurations suivantes sont équivalentes :



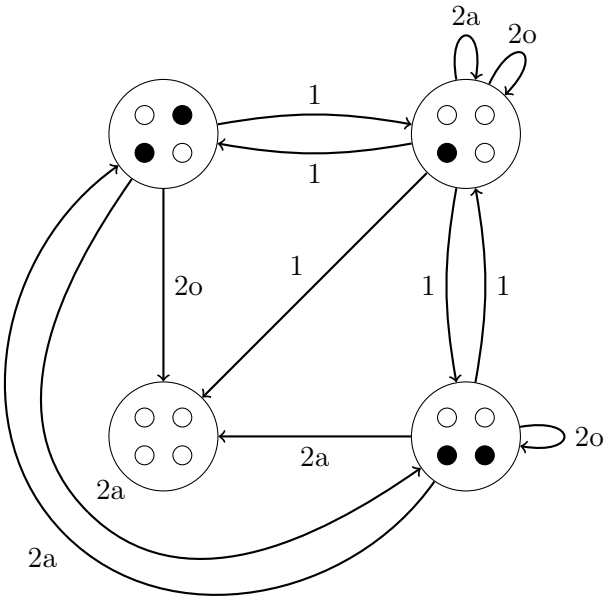
Grâce à cette observation, on divise déjà le nombre de configurations par deux ce qui nous laisse les huit configurations suivantes :



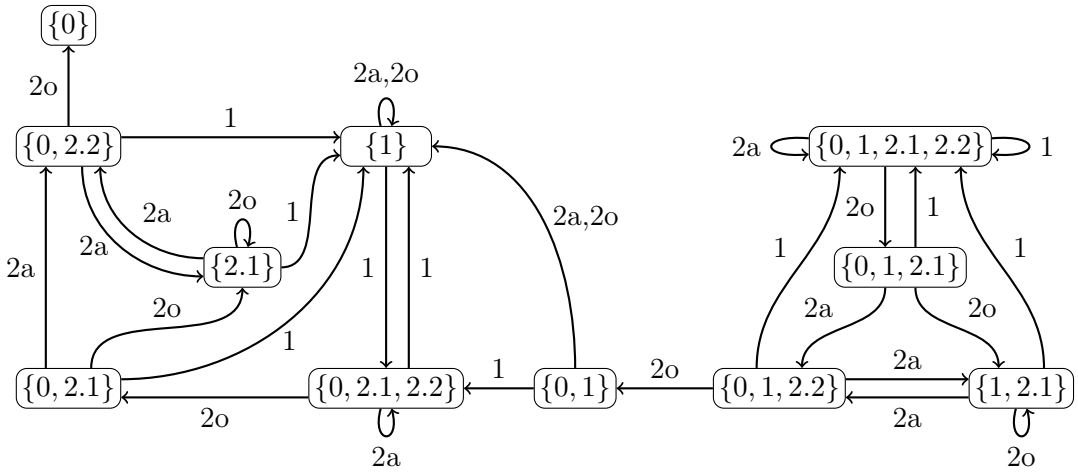
De plus, le contrôle de la rotation est laissé à l'adversaire, le barman ne sait donc pas quelle face lui est présentée. En particulier celà signifie que si il décide de retourner un verre il ne contrôle pas lequel il retourne. Si il décide de retourner deux verres, il n'a que deux choix : retourner deux verres adjacents (sur une même face) ou deux verres opposés (sur une même diagonale). Chacune de ces actions peut avoir plusieurs conséquences suivant les verres qui sont en pratique retournés, mais ces ensembles de conséquences ne dépendent pas de la rotation du plateau. On peut donc réduire encore le nombre de configurations équivalentes pour le barman à quatre :

Configuration 0 Configuration 1 Configuration 2.1 Configuration 2.2

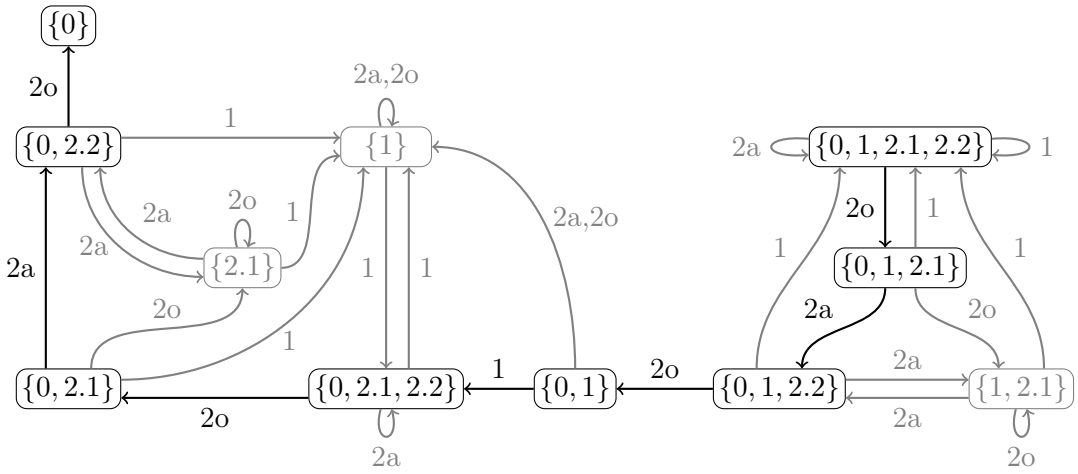
Comme on l'a dit plus haut, le barman a trois actions à sa disposition : retourner un seul verre (action 1), retourner deux verres adjacents (action $2a$) ou retourner deux verres opposés (action $2o$). On représente les conséquences possibles de chaques action pour chaque configuration dans l'automate suivant, les états sont les quatre configurations et l'alphabet est l'ensemble des actions possibles $\{1, 2a, 2o\}$:



Notons qu'il n'y a aucune action possible depuis la configuration gagnante puisque le barman a gagné (et s'arrête donc de jouer). On souhaite savoir si le barman a une stratégie gagnante, c'est à dire une suite d'actions qui, quelle que soit la configuration initiale, mène à coup sûr dans la configuration gagnante. Pour celà on va déterminiser l'automate. En effet si on trouve un chemin de l'état $\{0, 1, 2.1, 2.2\}$ jusqu'à l'état $\{0\}$ dans le déterminisé celà signifie qu'il existe une séquence d'actions qui nous mène à coup sûr en 0 ou dans un état bloquant (0 qui est le seul état bloquant).



Observons maintenant que depuis l'état $\{0, 1, 2.1, 2.2\}$, en lisant le mot $2o{\cdot}2a{\cdot}2o{\cdot}1{\cdot}2o{\cdot}2a{\cdot}2o$, on arrive dans l'état $\{0\}$. Autrement dit, dans notre automate original, partant de n'importe quelle situation, si le barman joue la séquence $2o \cdot 2a \cdot 2o \cdot 1 \cdot 2o \cdot 2a \cdot 2o$, soit il se retrouve à un moment dans état bloquant (i.e. la configuration gagnante qui est le seul état bloquant), soit il arrive dans la configuration gagnante après le dernier coup. Dans tous les cas le barman est sûr d'atteindre la configuration gagnante en jouant cette séquence, c'est donc une stratégie gagnante.


{0}
2o
{0, 2.2}
1
{1}
2a,2o
2a
2o
1
2a
{2.1}
2a
2o
1
1
1
2a,2o
2a
{0, 2.1}
1
{0, 2.1, 2.2}
1
{0, 1}
2o
{0, 1, 2.2}
2o
1
2a
{0, 1, 2.1, 2.2}
1
1
{0, 1, 2.1}
1
2a
2o
2a
{1, 2.1}
2a
2o