Solution - TD Feuille 5 - Résiduels et minimisation des automates

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

Solution de l'exercice 1 :

On donne les résiduels pour chaque langage. On commence par $L_1 = a^*b^*$ qui a trois résiduels :

$(\epsilon)^{-1}L_1 = L_1 = R_0$

$a^{-1}R_0 = a^*b^* = R_0$

$b^{-1}R_0 = b^* = R_1$

$(ba)^{-1}L_1 = a^{-1}R_1 = \emptyset = R_2$

$b^{-1}R_1 = b^* = R_1$

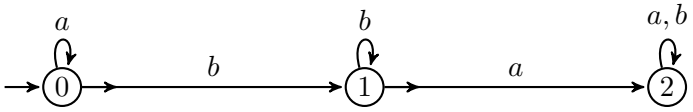
Soit w un mot :

— Si $w \in a^*$ alors $w^{-1}L_1 = L_1 = a^*b^*$. Par exemple $(aa)^{-1}L_1 = a^*b^*$.

— Si $w \notin a^*$ et $w \in a^*b^*$ alors $w^{-1}L_1 = b^*$. Par exemple $(ab)^{-1}L_1 = b^*$.

— Sinon (si $w \in (a+b)^*ba(a+b)^*$) alors $w^{-1}L_1 = \emptyset$.

L'automate minimal de a^*b^* a trois états, correspondant aux 3 résiduels :



On passe maintenant à $L_2 = a^*b + b^*a$. Ce second langage possède 7 résiduels, $R_0, \ldots, R_6$:

$(\epsilon)^{-1}L_2 = L_2 = R_0$

$a^{-1}R_0 = a^*b + \epsilon = R_1$

$b^{-1}R_0 = \epsilon + b^*a = R_2$

$(aa)^{-1}L_2 = a^{-1}R_1 = a^*b = R_3$

$(ab)^{-1}L_2 = b^{-1}R_1 = \epsilon = R_4$

$(ba)^{-1}L_2 = a^{-1}R_2 = \epsilon = R_4$

$(bb)^{-1}L_2 = b^{-1}R_2 = b^*a = R_5$

$(aaa)^{-1}L_2 = a^{-1}R_3 = a^*b = R_3$

$(aab)^{-1}L_2 = b^{-1}R_3 = \epsilon = R_4$

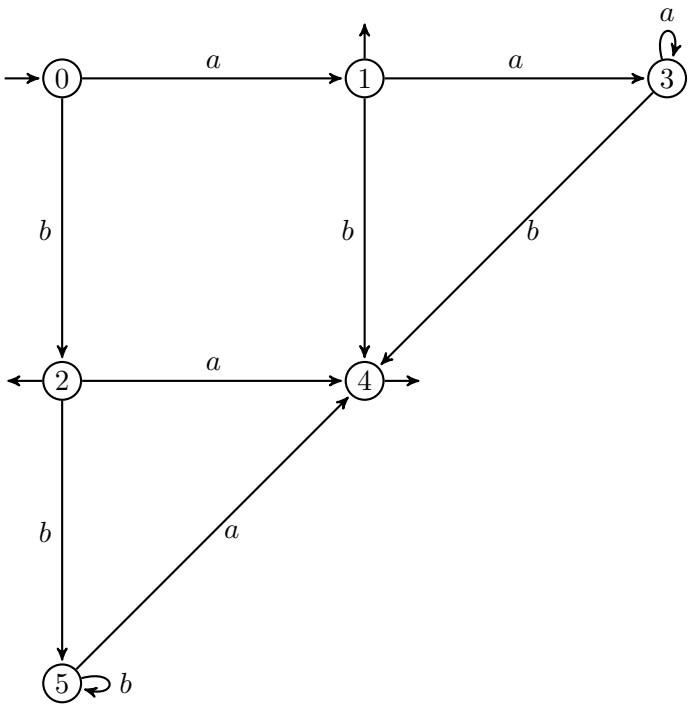
$(aba)^{-1}L_2 = a^{-1}R_4 = \emptyset = R_6$

$(abb)^{-1}L_2 = b^{-1}R_4 = \emptyset = R_6$

$(bba)^{-1}L_2 = a^{-1}R_5 = \epsilon = R_4$

$(bbb)^{-1}L_2 = b^{-1}R_5 = b^*a = R_5$

L'automate minimal de L_2 (sans l'état puits 6, qui correspond à R_6) :



Solution de l'exercice 3 :

On souhaite comparer les quatres langages. Pour cela 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 langages sont donc égaux si ils ont le même automate minimal (modulo renommage des états).

Expression rationnelle $(ab^*a + b(a + b))^*$.
Pour calculer l'automate minimal de cette expression rationnelle on peut utiliser deux méthodes : calculer les résiduels du langage et ensuite l'automate (minimal) des résiduels, ou bien construire un automate par la méthode de Glushkov, et ensuite le minimiser.

1. Soit L le langage décrit par l'expression rationnelle ci-dessus, le calcul des reśiduels de L donne :

   $(\epsilon)^{-1}L = L = R_0$
   $a^{-1}R_0 = b^*a = R_1$
   $b^{-1}R_0 = (a + b)L = R_2$
   $a^{-1}R_1 = R_0$
   $b^{-1}R_1 = R_1$
   $a^{-1}R_2 = R_0$
   $b^{-1}R_2 = R_0$

   Ce qui donne l'automate :

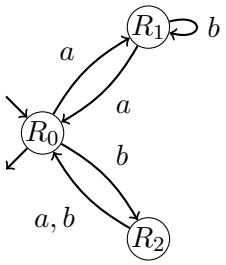
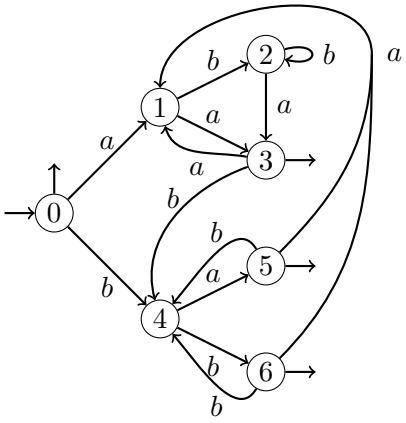


FIGURE 1 – Automate minimal pour $(ab^*a + b(a + b))^*$

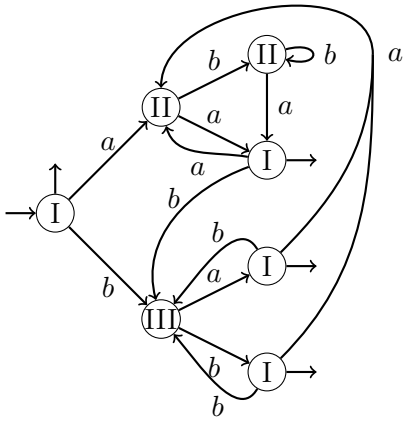
2. Construction d'un automate par la méthode de Glushkov :



On souhaite maintenant construire l'automate minimal du langage. Pour cela il faut d'abord déterminiser puis minimiser l'automate ci-dessus. Par chance on a déjà un automate déterministe (et complet !), on peut donc directement appliquer l'algorithme de minimisation.

état	0	1	2	3	4	5	6
ϵ	I	II	II	I	II	I	I
a	II	I	I	II	I	II	II
b	II	II	II	II	I	II	II
bilan	I	II	II	I	III	I	I
a	II	I	I	II	I	II	II
b	III	II	II	III	II	III	III
bilan	I	II	II	I	III	I	I

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 déjà dessiné à la figure 1 :

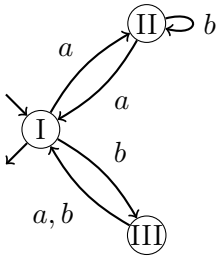


FIGURE 2 – Automate minimal pour $(ab^*a + b(a + b))^*$

Expression rationnelle $(ab + b(a + b))^*$.

1. Soit L' le langage décrit par l'expression rationnelle ci-dessus, le calcul des résiduels de L' donne :

   $(\epsilon)^{-1}L' = L' = R_0$
   $a^{-1}R_0 = b = R_1$
   $b^{-1}R_0 = (a + b)L' = R_2$
   $b^{-1}R_1 = R_0$
   $a^{-1}R_2 = R_0$
   $b^{-1}R_2 = R_0$
   $a^{-1}R_1 = \emptyset$

   Ce qui donne l'automate (non complet) de la figure suivante.
2. Construction d'un automate pour $(ab + b(a + b))^*$ par la méthode de Glushkov :

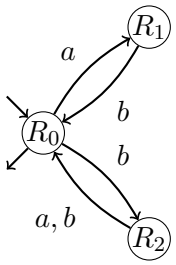
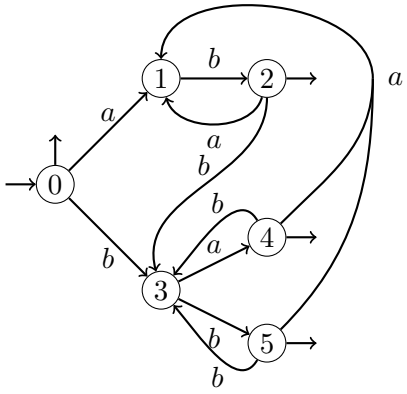


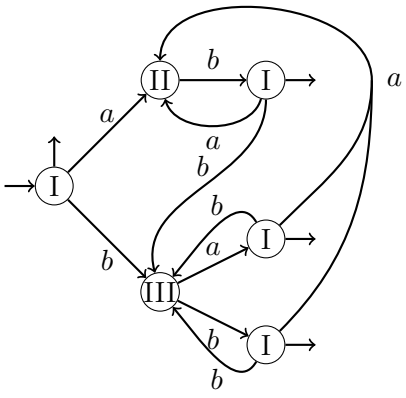
FIGURE 3 – Automate minimal pour $(ab + b(a + b))^*$



Encore une fois, l'automate que nous donne la méthode de Glushkov est déjà déterministe. On peut donc directement appliquer l'algorithme de minimisation (après avoir complété !) sans passer par l'étape de déterminisation.

état	0	1	2	3	4	5	$\emptyset$
ϵ	I	II	I	II	I	I	II
a	II	II	II	I	II	II	II
b	II	I	II	I	II	II	II
bilan	I	II	I	III	I	I	IV
a	II	IV	II	I	II	II	IV
b	III	I	III	I	III	III	IV
bilan	I	II	I	III	I	I	IV

On obtient la partition suivante à la fin de l'algorithme :



On fusionne ensuite les états équivalents ce qui donne l'automate (non complet) minimal suivant déjà dessiné à la figure 3.

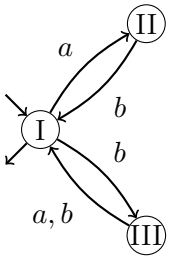
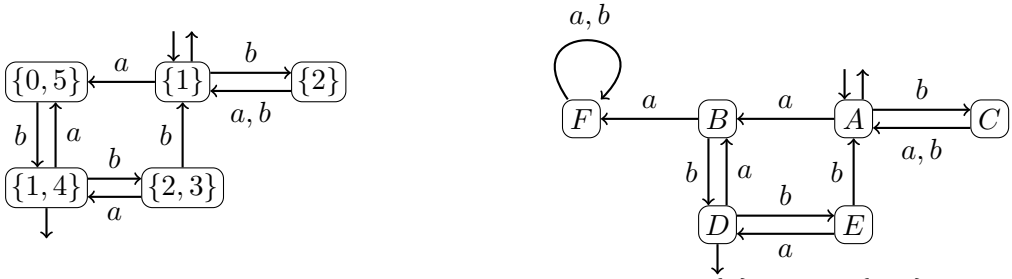


FIGURE 4 – Automate minimal pour $(ab + b(a + b))^*$

Automate $\mathcal{A}_1$. Pour minimiser $\mathcal{A}_1$, on doit d'abord le déterminiser. On donne ci dessous le résultat de l'algorithme de déterminisation :



(A droite dans la figure précédante on a renommé les états : $\{1\} \to A$, $\{0, 5\} \to B$, $\{2\} \to C$, $\{1, 4\} \to D$, $\{2, 3\} \to E$ et $\emptyset \to F$ - le dernier est état puits, avec transition $B \xrightarrow{a} F$.)

On peut maintenant appliquer l'algorithme de minimisation :

— La partition initiale est $\{A, D\}$ et $\{B, C, E, F\}$.

— Avec la lettre a : B, F vont dans $\{B, C, E, F\}$ et C, E vont dans $\{A, D\}$. On sépare donc B, F de C, E et on obtient la partition :

$$\{A, D\} \quad | \quad \{B, F\} \quad | \quad \{C, E\}$$

— Avec la lettre b : B va dans $\{A, D\}$ et F reste dans $\{B, F\}$. On sépare donc B de F en obtenant :

$$\{A, D\} \quad | \quad \{B\} \quad | \quad \{F\} \quad | \quad \{C, E\}$$

La partition finale obtenue est stable et donne l'automate minimal représenté ci-dessous, obtenu en fusionnant A, D et C, E (sans représenter l'état puits F) :

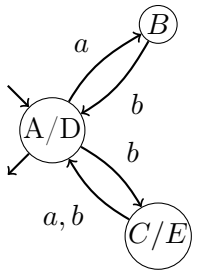
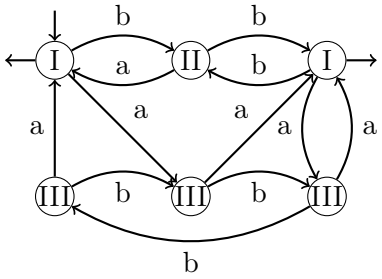


FIGURE 5 – Automate minimal pour $\mathcal{A}_1$

Automate $\mathcal{A}_2$. L'automate $\mathcal{A}_2$ est déjà déterministe, on passe donc directement à l'algorithme de minimisation. La partition obtenue est la suivante :



Après fusion on obtient l'automate minimal suivant :

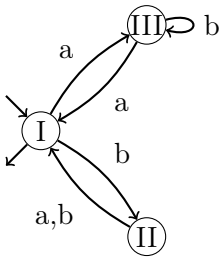


FIGURE 6 – Automate minimal pour $\mathcal{A}_2$

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 $\mathcal{A}_1$ et $(ab + b(a + b))^*$ ont le même automate minimal et sont donc égaux. Il en va de même pour les langages de $\mathcal{A}_2$ et $(ab^*a + b(a + b))^*$.

Solution de l'exercice 4 :

1. $L = \text{PAL} \cap a^*ba^* = \{a^nba^n \mid n \geq 0\}$.

2. Avec les résiduels : si $u = a^k$ alors le résiduel de L par u, donc $u^{-1}L$, est égal à $\{a^{n-k}ba^n \mid n \geq k\}$. Les résiduels $(a^k)^{-1}L$ sont donc deux à deux différents, ce qui montre que L a une infinité de résiduels. On conclut que L n'est pas régulier.

   Avec le lemme de pompage :
   — Soit $N \geq 0$ quelconque.
   — Nous choisissons $x = a^N ba^N$. Ce mot x appartient à L et est de longueur au moins N.
   — Soit $x = uvw$ une décomposition quelconque de x telle que $v \neq \epsilon$ et $|uv| < N$. Cela entraine que $u = a^i$, $v = a^j$ et $w = a^{N-i-j}ba^N$ avec $j \neq 0$.
   — Nous choisissons $k = 0$ et observons que

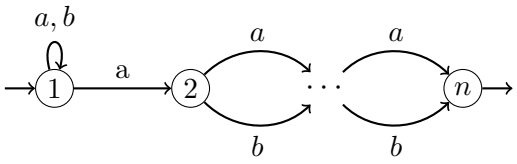
$$u\,v^0\,w = a^i\,a^{N-i-j}ba^N = a^{N-j}ba^N \notin L$$

   On a donc appliqué avec succès la contre-posée du lemme de pompage, ce qui entraine que L n'est pas régulier.

3. Si PAL était régulier, alors L serait aussi régulier (car l'intersection de deux langages réguliers est un langage régulier). Mais on vient de voir que L n'est pas régulier. Donc PAL n'est pas régulier.

Solution de l'exercice 5 :

1. On donne un automate non déterministe à n états pour L_n :



2. On montre qu'il n'existe pas d'automate déterministe ayant moins de 2^{n-1} états qui accepte le langage L_n. On va donc montrer que l'automate minimal pour L_n a plus de 2^{n-1} états. On rappelle que chaque état de l'automate minimal correspond à un résiduel du langage. Il nous suffit donc de montrer que L_n possède au moins 2^{n-1} résiduels.

   Observons tout d'abord que L_n 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 tout automate déterministe qui accepte L_n ne peut pas arriver dans le même état après avoir lu w_E et $w_{E'}$, respectivement. Autrement dit, les résiduels $(w_E)^{-1}L_n$ et $(w_{E'})^{-1}L_n$ sont différents.

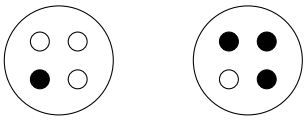
   Si $E \neq E'$ alors il existe forcément $i \in \{1, 2, \ldots, n\}$ tel que $i \in E$ et $i \notin E'$ ou tel que $i \in E'$ et $i \notin E$. Les deux cas étant symetriques, on suppose qu'on est dans le premier. Par définition, w_E contient un a à la i-ème position en partant de la droite et $w_{E'}$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

un b à la i-ème position en partant de la droite. Il en découle que $w_E a^{n-1-i} \in L_n$ et $w_{E'} a^{n-1-i} \notin L_n$, donc $a^{n-1-i} \in (w_E)^{-1} L_n$ et $a^{n-1-i} \notin (w_{E'})^{-1} L_n$. On en déduit que $(w_E)^{-1} L_n \neq (w_{E'})^{-1} L_n$.

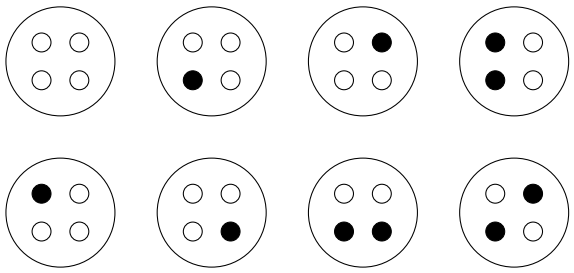
Donc L_n a au moins autant de résiduels qu'il y a de sous-ensembles $E \subseteq \{1, 2, \ldots, n-1\}$, c'est-à-dire 2^{n-1}. Ca implique que tout automate déterministe qui accepte L_n a au moins 2^{n-1} états.

Solution de l'exercice 6 :

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 :

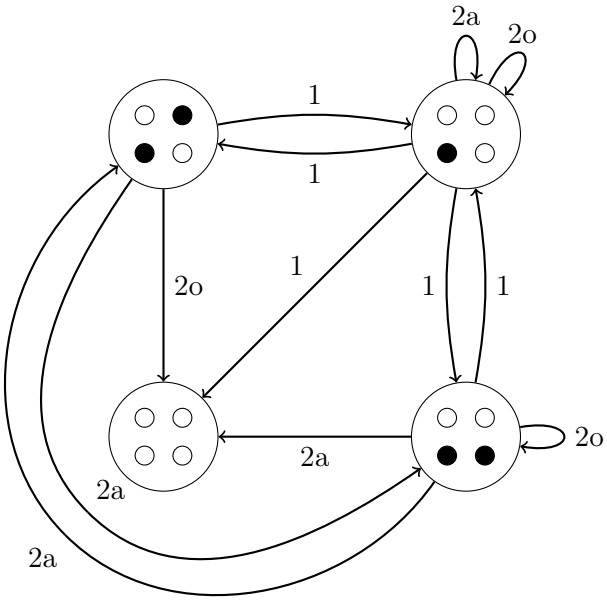


De plus, le contrôle de la rotation est laissé à l'adversaire, le barman ne sait donc pas quelle face lui est présentée. En particulier celà signifie que si il décide de retourner un verre il ne contrôle pas lequel il retourne. Si il décide de retourner deux verres, il n'a que deux choix : retourner 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 :

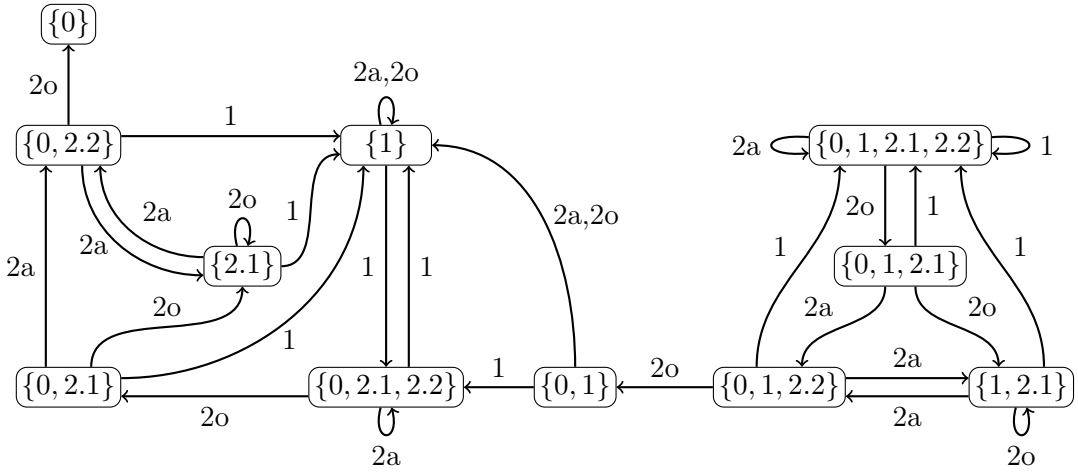
Configuration 0 Configuration 1 Configuration 2.1 Configuration 2.2

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.