# TD 2 - Machines de Turing

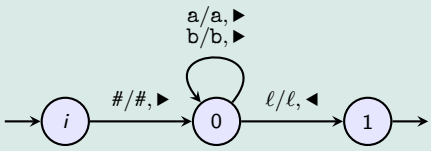
Dans ce TD, on considère des machines opérant sur un ruban semi-infini (indexé par les entiers naturels). On note # le symbole case vide, et on suppose que la configuration initiale associée au mot $w$ d'une machine ayant comme état initial $q_0$ est $q_0\#w$ : autrement dit le pointeur est initialement sur la première case du ruban, qui est initialement laissée vide.

## Première partie – Manipulation du ruban

### Exercice 1. Déplacements sur le ruban

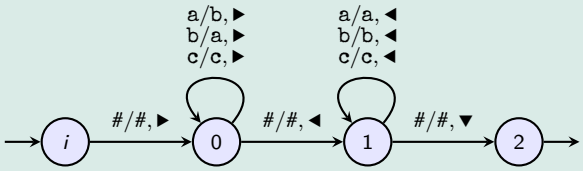
**Question 1.** - *Avancer jusqu'à un symbole* - Soit $\Sigma = \{\mathtt{a}, \mathtt{b}, \ell\}$ un alphabet d'entrée. Écrire une machine de Turing $A_\ell$ qui avance la tête de lecture jusqu'à la première occurence du symbole $\ell$, puis recule pour se placer sur la case immédiatement à gauche. Cette machine refuse les mots qui ne contiennent pas le symbole $\ell$.

**Solution :**




**Question 2.** - *Remplacer* - Soit $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ l'alphabet d'entrée. Écrire une machine $R_{\mathtt{a},\mathtt{b}}$ qui échange les $\mathtt{a}$ et les $\mathtt{b}$ du mot d'entrée (en laissant les $\mathtt{c}$ inchangés) et qui s'arrête sur la première case du ruban.

**Solution :**

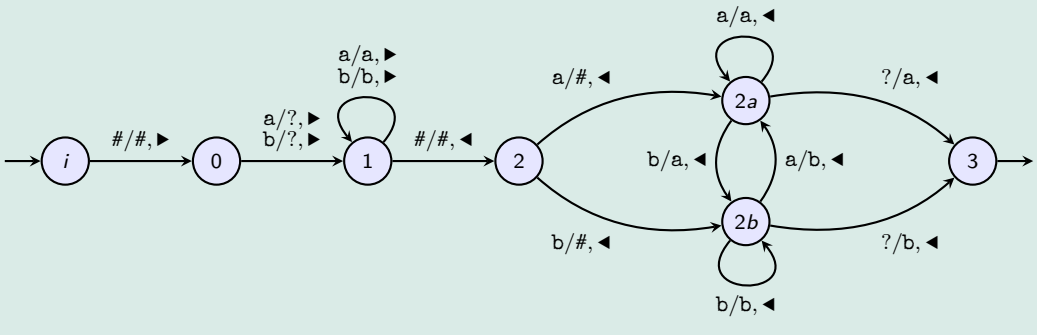



### Exercice 2. Copies et effacements

On fixe pour cet exercice l'alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. Les symboles $\ell$ et $w$ seront utilisés respectivement pour dénoter un symbole de $\Sigma$ et un mot de $\Sigma^\star$.

**Question 1.** - *Effacer* - Écrire une machine $E$ qui, en partant d'un ruban $\underline{\#}\ell w$, s'arrête avec le ruban $\underline{\#}w$.

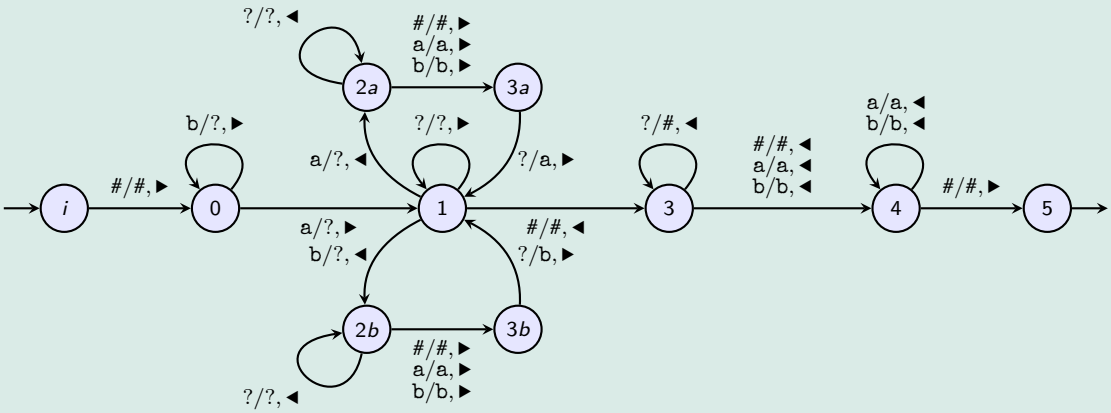
**Solution :**



**Question 2.** - *Effacer jusqu'à un symbole* - Écrire une machine $E_\ell$ qui efface toutes les lettres de son entrée jusqu'au premier $\ell \in \Sigma$ inclus.

**Solution :**

On donne la solution suivante dans le cas où $\ell = \mathtt{a}$.

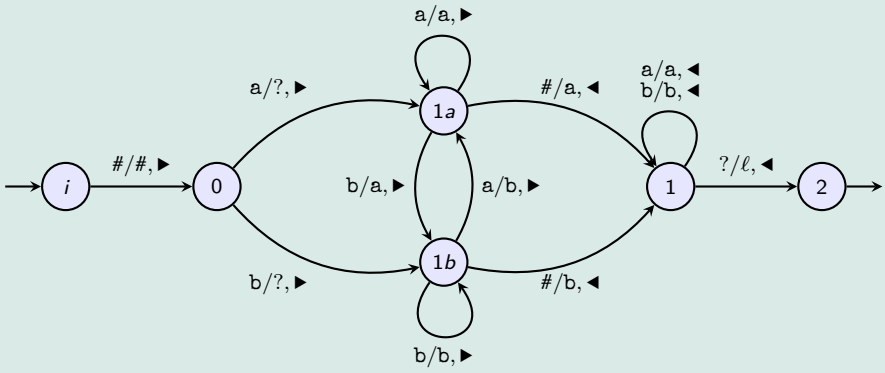



Le cas $\ell = \mathtt{b}$ se traite de la même manière, en échangeant $\mathtt{a}$ et $\mathtt{b}$ dans les deux transitions sortant de l'état 0. Ainsi on effacera tous les $\mathtt{a}$ jusqu'à croiser le premier $\mathtt{b}$.

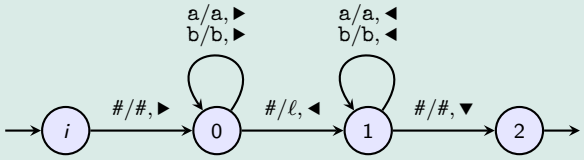
**Question 3.** - *Insérer* - Écrire trois machines $I_\ell$, $I'_\ell$, et $I''_\ell$ qui prennent en entrée un mot $w \in \Sigma^\star$ (donc un ruban $\underline{\#}w$) et s'arrêtent respectivement avec les rubans $\underline{\#}\ell w$, $\underline{\#}w\ell$, et $\#w\ell\underline{\#}$.

**Solution :**

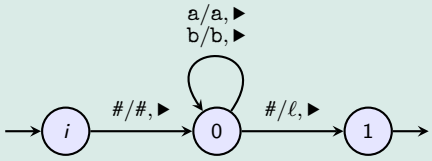
Machine $I_\ell$




Machine $I'_\ell$

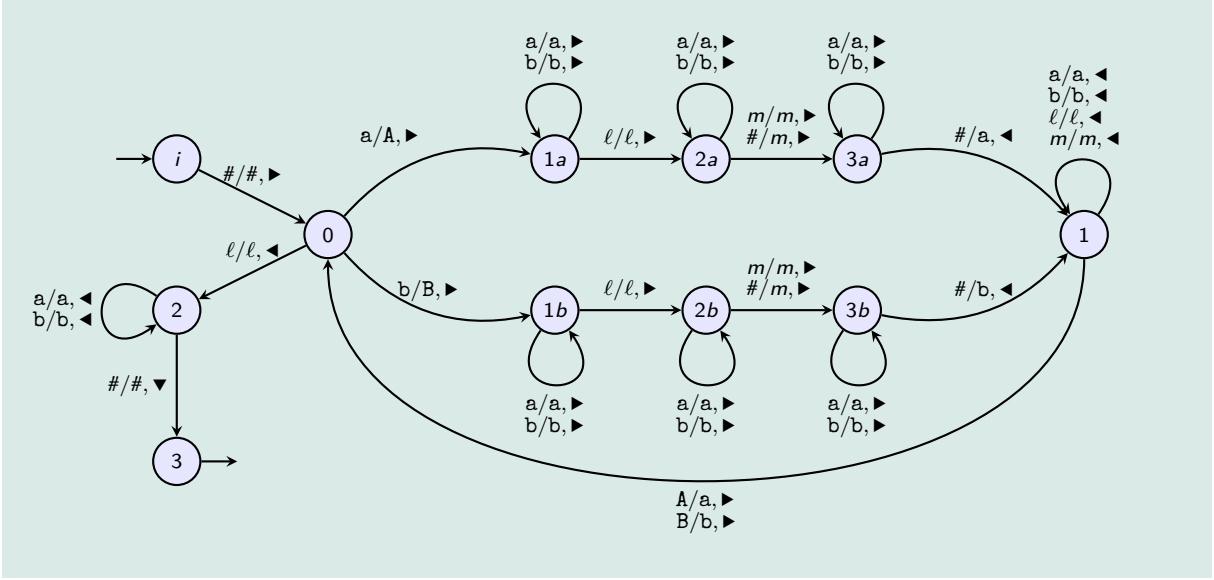



Machine $I''_\ell$



**Question 4.** - *Copier* - Soit $\Gamma$ un alphabet de travail contenant $\Sigma$, et $\ell, m \in \Gamma \setminus (\Sigma \cup \{\#\})$ deux symboles de travail. Écrire une machine $C_\ell^m$ qui prend en entrée un mot $w_1 \ell w_2$, avec $w_1, w_2 \in \Sigma^\star$, et s'arrête avec le ruban $\underline{\#} w_1 \ell w_2 m w_1$.

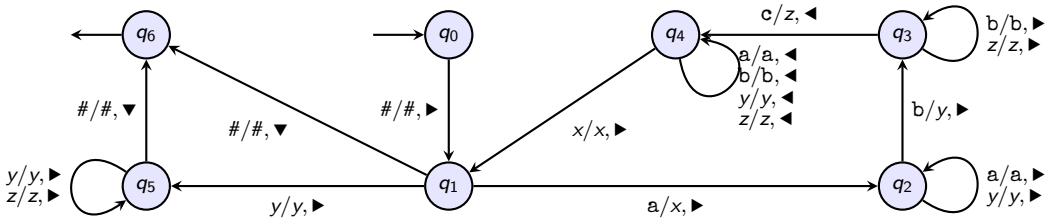
**Solution :**




## Deuxième partie – Reconnaissance de langages

### Exercice 3. Langage mystère

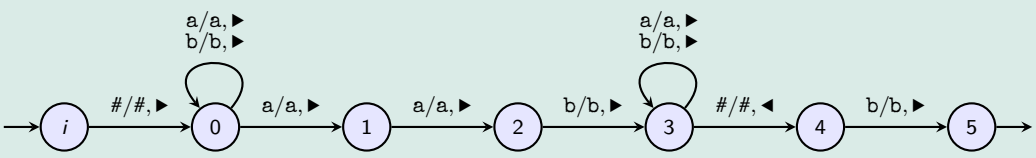
Quel est le langage sur l'alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ reconnu par la machine suivante ?




### Exercice 4. Langages réguliers

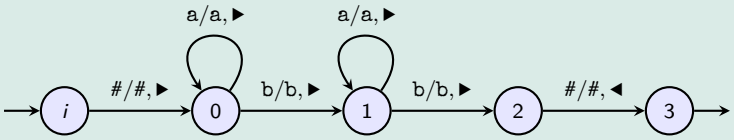
**Question 1.** - *Sous-mot* - Écrire une machine de Turing acceptant les mots sur l'alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ qui contiennent le sous-mot $\mathtt{aab}$ et se terminent par un $\mathtt{b}$ (et refusant tous les autres mots).

**Solution :**



**Question 2.** Écrire une machine de Turing acceptant le langage $[\![\mathtt{a}^\star\mathtt{b}\mathtt{a}^\star\mathtt{b}]\!]$.

**Solution :**

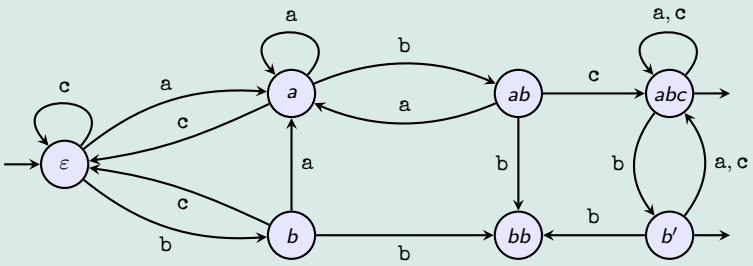



**Question 3.** Écrire une machine de Turing acceptant le langage

$$\left\{w \in \{\mathtt{a},\mathtt{b},\mathtt{c}\}^\star \;\middle|\; w \text{ contient } \mathtt{abc} \text{ mais pas } \mathtt{bb}\right\}.$$

**Solution :**

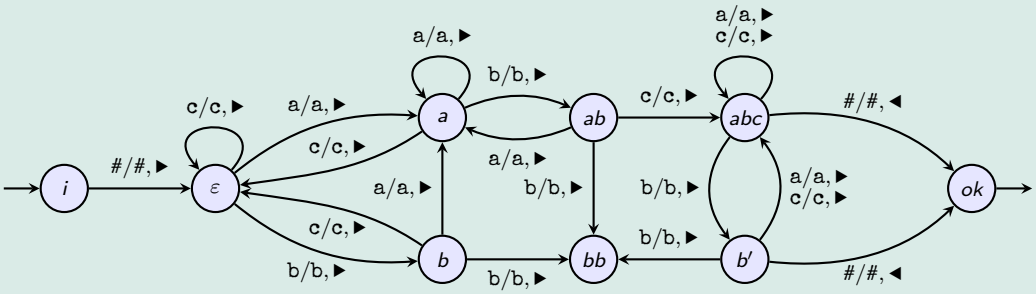
Le langage demandé est un langage régulier, on peut donc le reconnaître avec un automate à états finis. Voici un tel automate :



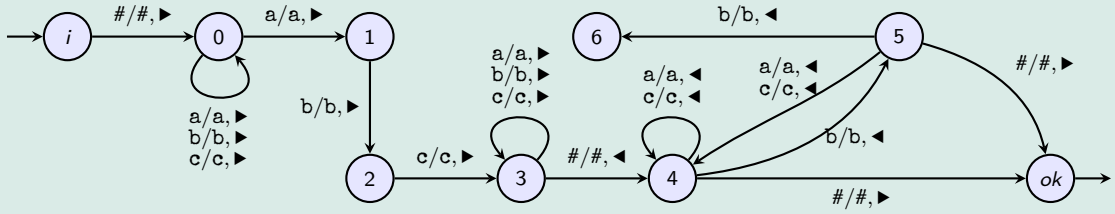

On peut ensuite transformer cet automate en machine de Turing, en applicant les modifications suivantes :

- on ajoute un nouvel état initial $i$ et un nouvel état final $ok$ ;
- chaque transition $p \xrightarrow{x} q$ est remplacée par $p \xrightarrow{x/x,\blacktriangleright} q$ ;
- on ajoute une transition $i \xrightarrow{\#/\#,\blacktriangleright} \varepsilon$, et pour chaque état final de l'automate $q \in F$, on ajoute une transition $q \xrightarrow{\#/\#,\blacktriangleleft} ok$.

On obtient ainsi la machine suivante :



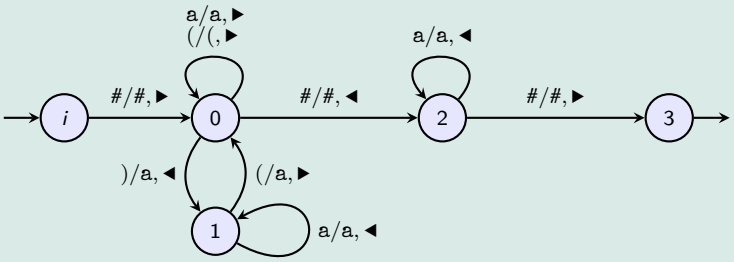

On peut également reconnaître ce langage avec une machine de Turing qui n'est pas la traduction d'un automate. La machine suivante commence par lire le mot d'entrée de gauche à droite sur l'état 0, en déclanchant de manière non-déterministe le test du motif $\mathtt{abc}$, passant ainsi par les états 1 et 2, avant de finir la lecture de l'entrée dans l'état 3. Atteindre l'état 3 est possible pour tous les mots de la forme $u\mathtt{abc}v$, c'est à dire ceux qui contiennent le motif recherché. Ensuite, la machine relit le mot, cette fois-ci de droite à gauche, et cherche de manière déterministe à détecter le motif $\mathtt{bb}$. La détection de ce motif mène à l'état 6, qui se comporte comme un état « puit » : cet état n'est pas final, et aucune transition ne permet de le quitter. Le mots contenant ce motif sont donc rejetés. À l'inverse, les mots ne contenant pas se motif seront lus en entier par une exécution visitant les états 4 et 5 un certain nombre de fois, et se terminant dans l'état final $ok$.




## Exercice 5. Langages hors-contextes

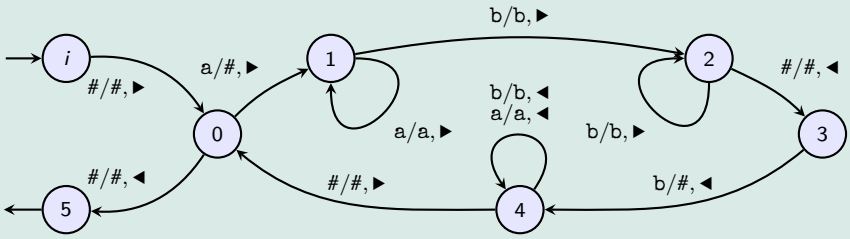
**Question 1.** Écrire une machine de Turing acceptant les mots bien parenthèsés sur l'alphabet $\Sigma = \{\mathtt{a}, (, )\}$, c'est à dire le langage décrit par la grammaire $S \to \varepsilon \mid \mathtt{a} \mid SS \mid (S)$.

**Solution :**




**Question 2.** Écrire une machine de Turing acceptant le langage $\{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$.

**Solution :**

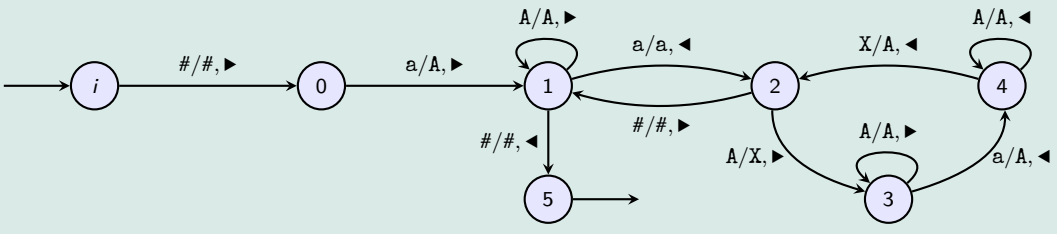



## Exercice 6.

On considère le langage $L := \left\{\mathtt{a}^{2^n} \;\middle|\; n \in \mathbb{N}\right\}$.

**Question 1.** Proposer une machine reconnaissant $L$.

**Solution :**



**Question 2.** Montrer que ce langage n'est pas régulier.

**Solution :**

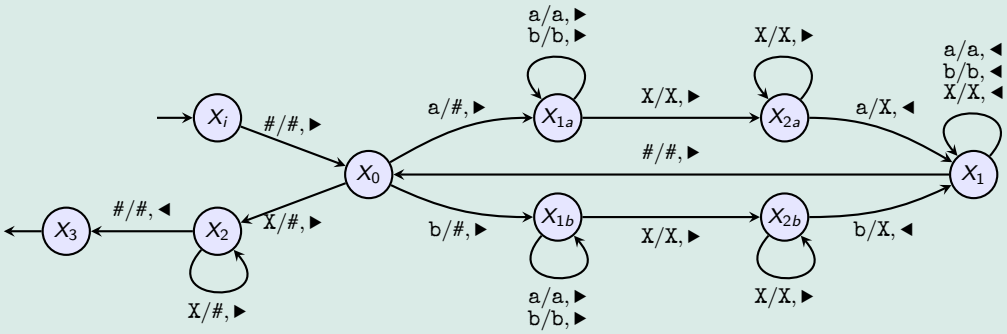
On utilise le lemme de l'étoile, avec la fonction $\varphi : n \mapsto a^{2^{n+1}}$. Clairement, $|\varphi(n)| = 2^{n+1} > n$. Considérons une décomposion $a^{2^{n+1}} = u_1\,u_2\,u_3$ avec $u_2 \neq \varepsilon$ et $|u_1 u_2| \leqslant n$. Je note $|u_2| = k$. Comme $u_2 \neq \varepsilon$ et $|u_2| \leqslant |u_1 u_2| \leqslant n$, on a $0 < k \leqslant n$. Or $u_1\,(u_2)^0\,u_3 = u_1 u_3 = a^{2^{n+1}-k}$. On remarque que comme $k > 0$, on a $2^{n+1} - k < 2^{n+1}$. De plus, comme $k \leqslant n < 2^n$, on a $2^{n+1} - k > 2^{n+1} - 2^n = 2^n$. Donc $2^{n+1} - k$ est strictement compris entre deux puissances de 2 successives, et donc $u_1\,(u_2)^0\,u_3 \notin L$.

## Exercice 7. Les carrés

Les machines de Turing peuvent reconnaître plus que les langages hors-contexte. C'est le cas par exemple du langage $L := \{ww \mid w \in \{\mathtt{a},\mathtt{b}\}^\star\}$ des carrés. On peut montrer, par exemple avec le lemme de la double étoile, que ce langage n'est pas hors-contexte : il ne peut être défini à l'aide d'une grammaire hors-contexte.

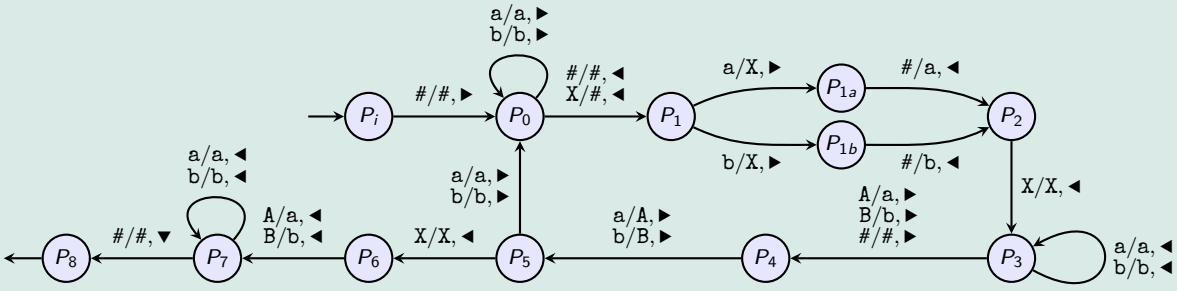
**Question 1.** Écrire une machine de Turing reconnaissant le langage $w\mathtt{X}w$, où $w \in \{\mathtt{a},\mathtt{b}\}^\star$.

**Solution :**



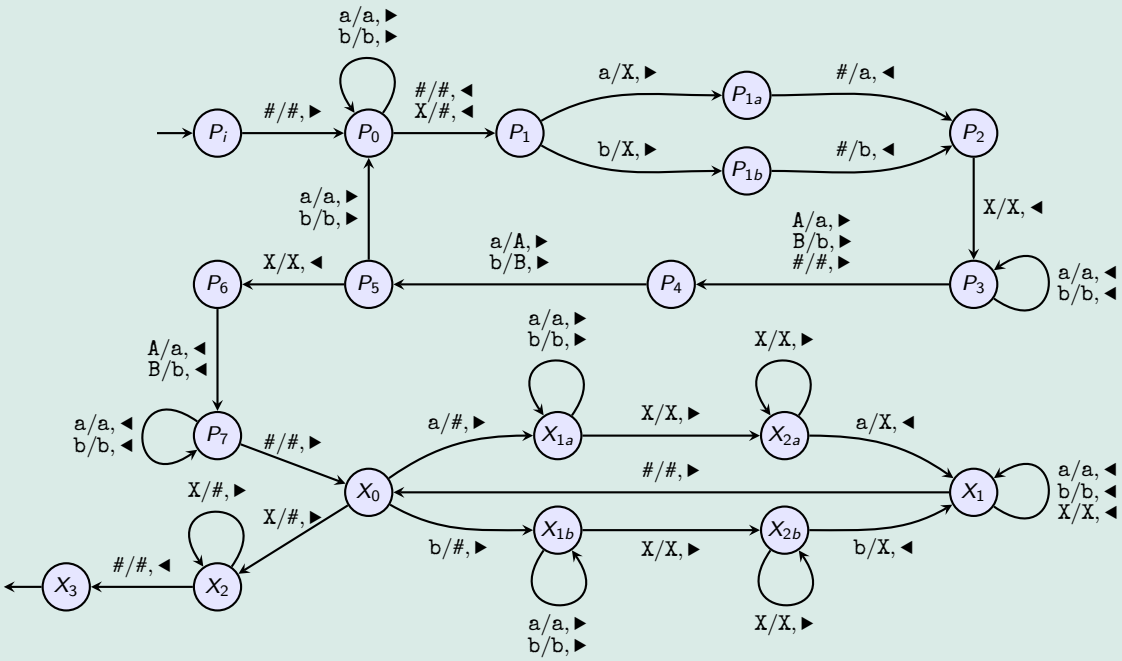

**Question 2.** Écrire une machine de Turing qui rejette les mots de longueur impaire, et pour les mots de longueur paire insère le symbole $\mathtt{X}$ entre les deux moitiés du mot. Par exemple, le mot $\mathtt{abcd}$ est transformé en $\mathtt{abXcd}$.

**Solution :**



**Question 3.** Combiner les machines des deux questions précédentes pour en produire une reconnaissant $L$.

**Solution :**




**Question 4.** Montrer que ce langage n'est pas rationnel.

**Solution :**

On utilise le lemme de l'étoile.
Soit un entier $n$. Le mot $w = \mathtt{a}^n\mathtt{b}\mathtt{a}^n\mathtt{b}$ est un carré, et appartient donc au langage $L$.
En revanche pour tout découpage $w = u_1\,u_2\,u_3$ tel que $|u_1u_2| < n$ et $u_2 \neq \varepsilon$, on observe que le $u_1u_2$ ne contient que des $\mathtt{a}$.
Par conséquent, le mot $u_1\,(u_2)^0\,u_3 = u_1\,u_3$ est de la forme $\mathtt{a}^k\mathtt{b}\mathtt{a}^n\mathtt{b}$, avec $k < n$. Ce mot n'est donc pas un carré, et en temps que tel n'appartient pas à $L$.
Si $L$ était régulier, d'après le lemme de l'étoile il existerait un certain entier $N$ tel que pour tout mot $w \in L$ de longueur supérieure à $N$, il existe un découpage $w = u_1\,u_2\,u_3$ avec $|u_1u_2| < n$ et $u_2 \neq \varepsilon$, tel que pour tout entier $k \in \mathbb{N}$ on aie $u_1\,(u_2)^k\,u_3 \in L$. On a montré plus haut une famille de mots arbitrairement longs pour lesquels un tel découpage est impossible, ce qui prouve que $L$ n'est pas régulier.

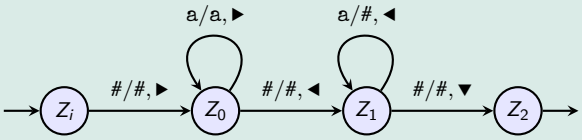
## Troisième partie – Arithmétique

### Exercice 8. Calculs unaires

On se donne un alphabet $\{\mathtt{a}, \mathtt{b}\}$. On code l'entier $n$ sur cet alphabet par le mot $\mathtt{a}^n$, c'est à dire le mot constitué de $n$ fois la lettre $\mathtt{a}$. On pourra également coder une séquence finie d'entiers $\langle n_1, \ldots, n_m \rangle$ par le mot $\mathtt{a}^{n_1}\mathtt{b}\mathtt{a}^{n_2} \cdots \mathtt{b}\mathtt{a}^{n_m}$.

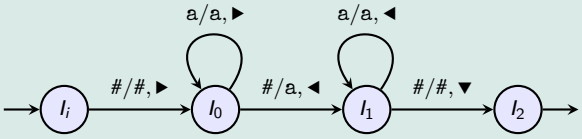
**Question 1.** Écrire une machine qui « calcule zéro », c'est à dire qu'elle prend en entrée le codage d'un entier et produit en sortie le codage de 0.

**Solution :**



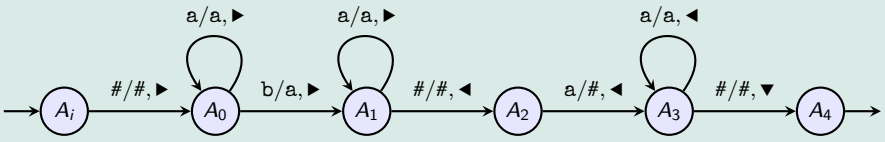

**Question 2.** Écrire une machine qui incrémente son entrée d'une unité, c'est à dire qu'elle prend en entrée le codage d'un entier $n$ et produit en sortie le codage de $n+1$.

**Solution :**




**Question 3.** Écrire une machine qui calcule l'addition, c'est à dire qu'elle prend en entrée le codage d'une paire d'entiers $\langle n, m \rangle$ et produit en sortie le codage de $n+m$.

**Solution :**




### Exercice 9. Calculs binaires

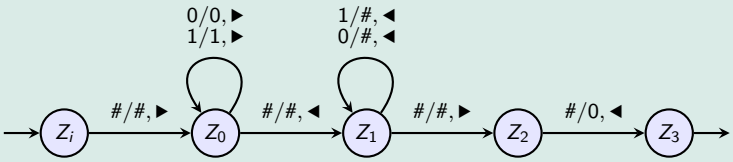
On se donne l'alphabet $\{0, 1, |\}$. On code l'entier $n$ en binaire sur cet alphabet, avec le bit de poids fort à droite. Par exemple, les 8 premiers entiers sont codés comme suit :

| | | | |
|---|---|---|---|
| $0 \mapsto 0$ | $2 \mapsto 01$ | $4 \mapsto 001$ | $6 \mapsto 011$ |
| $1 \mapsto 1$ | $3 \mapsto 11$ | $5 \mapsto 101$ | $7 \mapsto 111$. |

On note $[n]_2$ le code de l'entier $n$. On pourra également coder une séquence finie d'entiers $\langle n_1, \ldots, n_m \rangle$ par le mot $[n_1]_2 \,|\, [n_2]_2 \cdots |\, [n_m]_2$. Par exemple la paire $\langle 2, 5 \rangle$ est représentée par $01|101$.

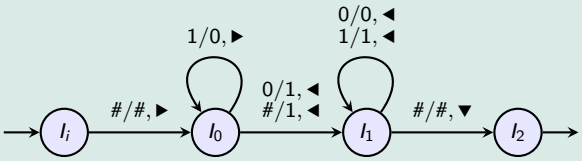
**Question 1.** Écrire une machine qui « calcule zéro », c'est à dire qu'elle prend en entrée le codage d'un entier et produit en sortie le codage de 0.

**Solution :**




**Question 2.** Écrire une machine qui incrémente son entrée d'une unité, c'est à dire qu'elle prend en entrée le codage d'un entier $n$ et produit en sortie le codage de $n+1$.

**Solution :**




**Question 3.** Écrire une machine qui calcule l'addition, c'est à dire qu'elle prend en entrée le codage d'une paire d'entiers $\langle n, m \rangle$ et produit en sortie le codage de $n+m$.

**Question 4.** Écrire une machine qui calcule la multiplication, c'est à dire qu'elle prend en entrée le codage d'une paire d'entiers $\langle n, m \rangle$ et produit en sortie le codage de $n \times m$.