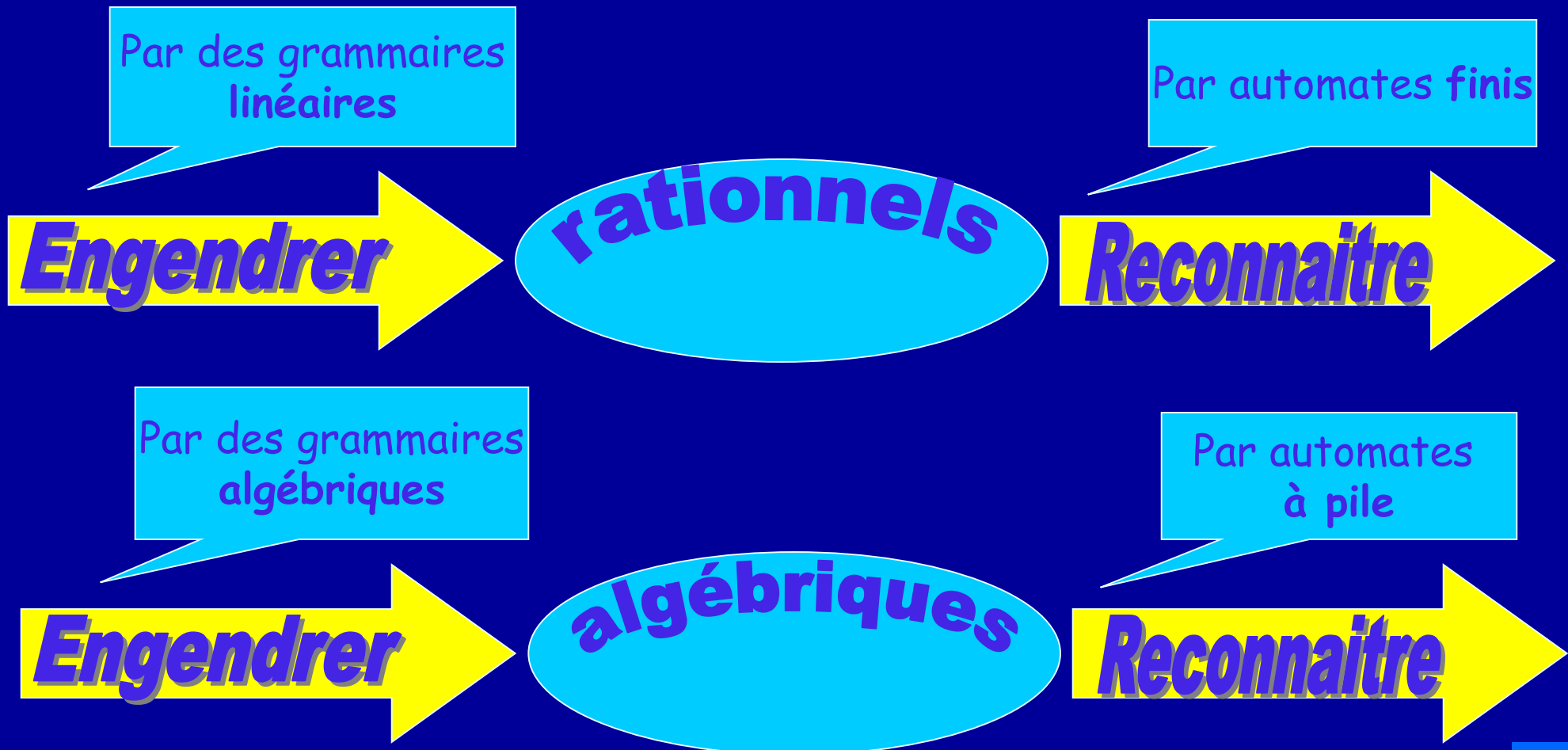
# Automates à pile

# Rationnels et Algébriques



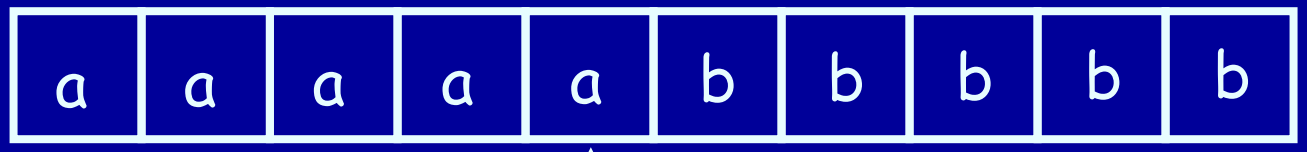
# Automate Fini vs Automate à Pile

- Comparables aux AFND auxquels on a ajouté un composant supplémentaire : Une PILE
- La pile procure une mémoire additionnelle à celle finie de l 'unité de contrôle.
- On peut reconnaître les langages algébriques.

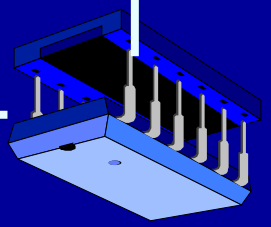
**Un langage est algébrique SSI il est reconnu par un automate à pile**

- Permet de montrer l 'algébricité d 'un langage donné en sus de la construction d'une grammaire algébrique.

# Schéma d'un Automate à pile




Tête de lecture



Pile
B
A

Unité de contrôle

- Lecture/écriture de symboles dans la pile
- Opérations:
  - Empiler des symboles
  - Dépiler un symbole => Lire le symbole de sommet
- Permet le stockage de nombre illimité de symboles

# Rappel

- Pourquoi un AF ne peut reconnaître $\{a^n b^n : n \geq 0\}$ ?
  - Impossibilité de mémoriser la valeur de n avec ses seuls états
  - On peut toujours trouver un mot trop grand qui « sature » la mémoire de l 'AF

- Alors qu'avec un AP peut mémoriser n dans sa pile et vérifier qu'il y a autant de a que de b.

# L'idée

- **Scénario** :
  - À la lecture d'un a
    - Empiler un symbole
  - À la lecture du premier b
    - Dépiler un symbole et changer d'état
  - À la lecture d'un b
    - Dépiler un symbole
  - À la fin, la pile est vide.

# Reconnaissance

- Comment AP accepte l'entrée ?
- À la fin de la lecture de l'entrée,
  - On est dans un état particulier
  - La pile est vide
- Deux manières de reconnaître un mot :
  - En atteignant un état de reconnaissance
  - Lorsque la pile est vide
- On verra l'équivalence de ces deux critères de reconnaissance.

# Définition $(Q, \Sigma, \Gamma, \delta, i, Z, T)$

- $Q$ : ensemble fini d'états
- $\Sigma$ : alphabet fini des symboles d'entrée
- $\Gamma$ : alphabet fini des symboles de pile
- $i$ : état initial
- $Z \in \Gamma$ : symbole de fond de pile
- $T \subseteq Q$ : ensemble des états terminaux
- $\delta$ : $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$

- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$

Lit l'état courant, le symbole courant, le sommet de la pile

- Renvoie un nouvel état, empile un mot
- La fonction de transition prend en compte
  - L 'état de l 'unité de contrôle
  - Le caractère lu par la tête de lecture
  - **Le** symbole au sommet de la pile
- Et doit pouvoir
  - Changer l 'état courant de l 'UC
  - Empiler **des** symboles

**Remarque** : L'image d 'un triplet (état, lettre, sommet de pile) est un sous-ensemble de $Q \times \Gamma^*$; on a donc un modèle a priori **non déterministe**

# Déterminisme

Un AP est déterministe lorsque, à chaque instant, il n 'y a pas plus d'une transition applicable.

Plus formellement,

$\forall q \in Q$, $\forall X \in \Gamma$, si $\delta(q,\varepsilon,X) \neq \varnothing$, alors $\forall a \in \Sigma$, $\delta(q,a,X) = \varnothing$

Retire le choix entre un déplacement indépendant de l'entrée et un déplacement consommant un symbole d 'entrée

$\forall q \in Q$, $\forall X \in \Gamma$, $\forall a \in \Sigma \cup \{\varepsilon\}$, $|\delta(q,a,X)| < 2$

Empêche d 'avoir un choix de déplacement pour un triplet donné.

# Déterminisme

- Si, pour les AF, les AFD et les AFND acceptent les mêmes langages, ce n 'est plus vrai pour les AP

- Exemple :

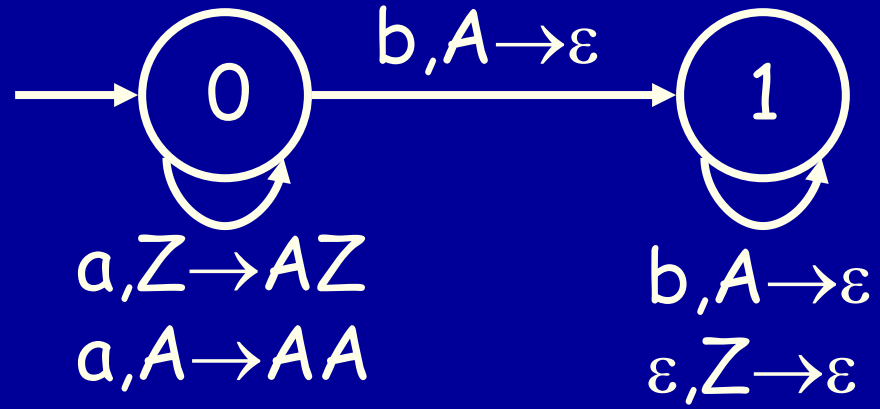
  Le langage $\{ww^R : w \in \{0,1\}^+\}$ (palindromes)

  Est accepté par un AP non déterministe

  N'est pas accepté par aucun AP déterministe

# Exemple d'AP déterministe

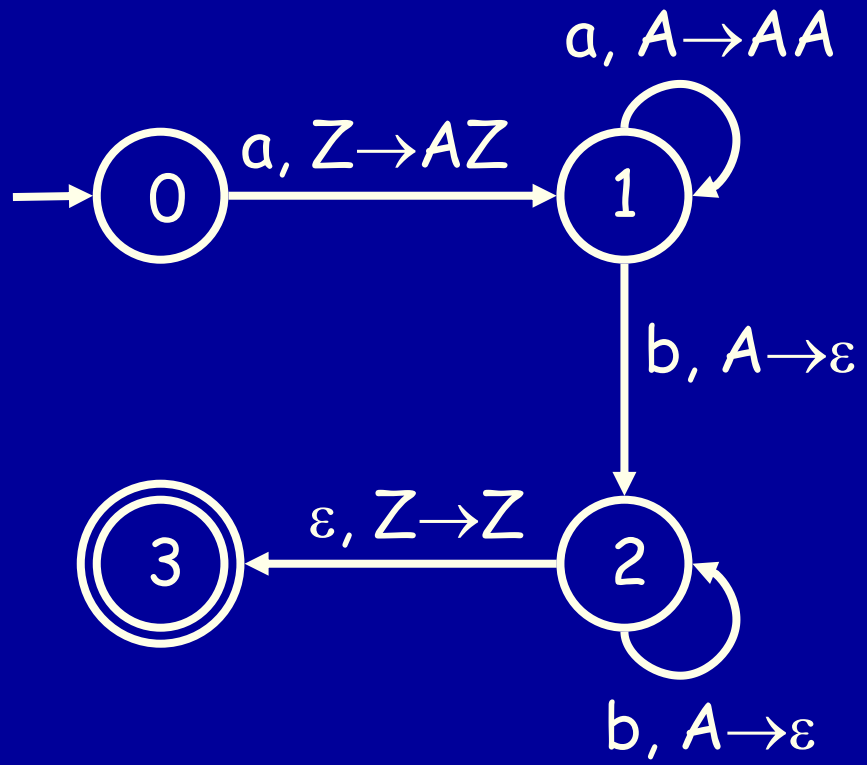
| état | lecture | pile | nouv. état | pile |
|---|---|---|---|---|
| 0 | a | Z | 0 | AZ |
| 0 | a | A | 0 | AA |
| 0 | b | A | 1 | ε |
| 1 | b | A | 1 | ε |
| 1 | ε | Z | 1 | ε |




Reconnaît $\{a^n b^n : n>0\}$ par pile vide

# Exemple d'AP déterministe

| état | lecture | pile | nouv. état | pile |
|---|---|---|---|---|
| 0 | a | Z | 1 | AZ |
| 1 | a | A | 1 | AA |
| 1 | b | A | 2 | ε |
| 2 | b | A | 2 | ε |
| 2 | ε | Z | 3 | ε |



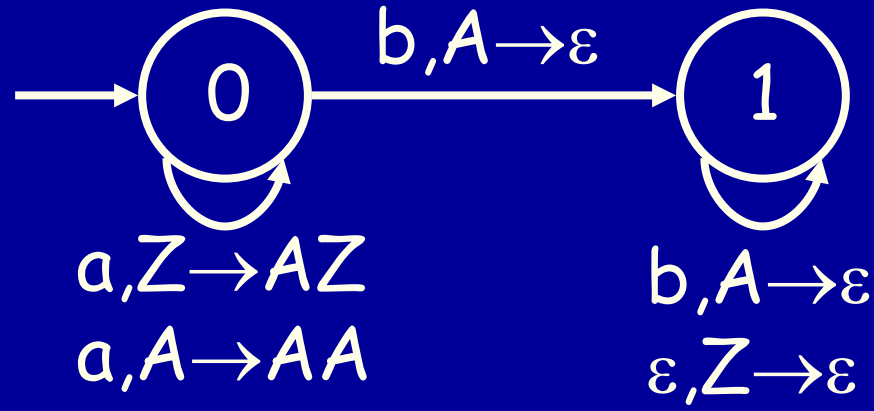

Reconnaît $\{a^n b^n : n>0\}$ par état final (état 3)

# Configuration et dérivation

- Une configuration représente
  - L'état courant, la partie du mot qui reste à lire, le contenu de la pile

**Exemple :**

- (0,aaabbb,Z) est la configuration initiale de notre exemple
- (0,aabbb,AZ) est la configuration après une transition



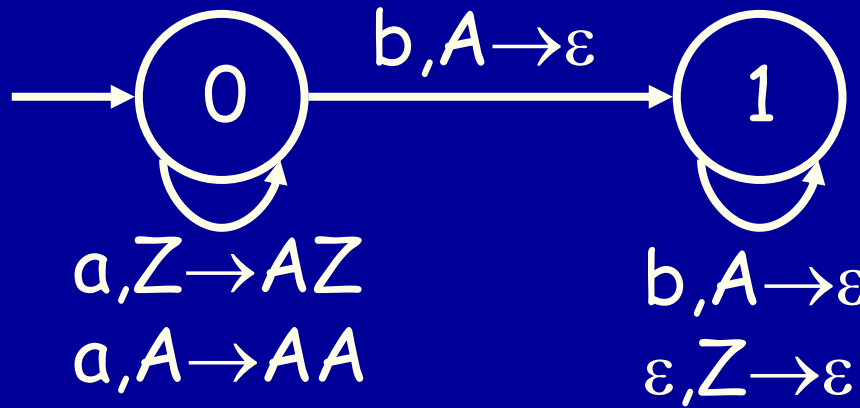
# Configuration et dérivation

- Une dérivation correspond à l'application de la fonction de transition
  - Entrée : une configuration
  - Sortie : {configurations} (l 'AP est a priori non déterministe)

$$(e,aw,P\gamma)\rightarrow(e',w,\alpha\gamma) \quad \text{Si } (e',\alpha)\in \delta(e,a,P)$$

**Exemple :**

- (0,aaabbb,Z)
- (0,aabbb,AZ)
- (0,abbb,AAZ)



# Lecture d'un mot

Lecture = suite des configurations prises par l 'AP

- Commence
  - Dans un état initial
  - Symbole de fond de pile
- Termine soit
  - Dans un état terminal
  - Avec la pile vide

| état | lecture | pile | nouv. état | pile |
|---|---|---|---|---|
| 0 | a | Z | 0 | AZ |
| 0 | a | A | 0 | AA |
| 0 | b | A | 1 | ε |
| 1 | b | A | 1 | ε |
| 1 | ε | Z | 1 | ε |

$(0,aabb,Z)\rightarrow(0,abb,AZ)\rightarrow(0,bb,AAZ)\rightarrow(1,b,AZ)\rightarrow(1,\varepsilon,Z)\rightarrow(1,\varepsilon,\varepsilon)$

$(0,aabbb,Z)\rightarrow(0,abbb,AZ)\rightarrow(0,bbb,AAZ)\rightarrow(1,bb,AZ)\rightarrow(1,b,Z)$

# Dérivation (réussie) de mots

**Réussie** : On étend la notion de dérivation aux mots

- $(i,w,Z) \rightarrow^* (e,\varepsilon,\varepsilon)$ par pile vide
- $(i,w,Z) \rightarrow^* (f,\varepsilon,\gamma)$ par état final

**Exemple** : $(0,aabb,Z) \rightarrow^* (1,\varepsilon,\varepsilon)$ accepte par pile vide

# Dérivation (ratée) de mots

**Ratée** : il n 'existe pas de lecture réussie

- Soit parce que pour chaque lecture :
  - On n'arrive pas à lire entièrement le mot (fonction de transition non définie)
  - On n'arrive pas soit
    - À terminer avec la pile vide
    - À terminer dans un état de reconnaissance

Exemple :

- $(0,aabbb,Z) \rightarrow^* (1,b,Z)$ (Pas de transition définie)

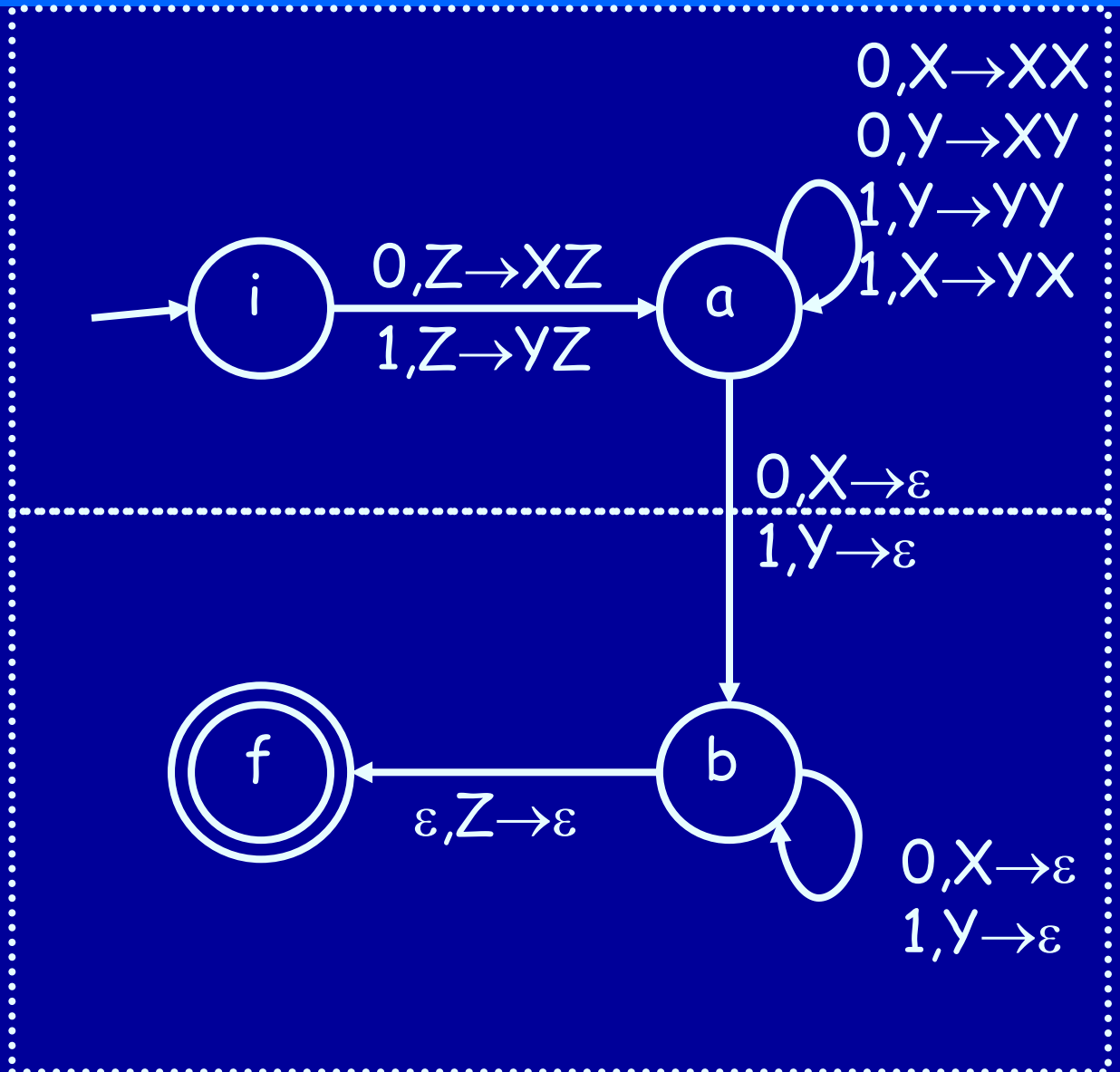
# Dérivation de mots

Pour $M=(Q,\Sigma,\Gamma,\delta,i,Z,T)$ un *AP*,

le langage accepté est

- $L_{PV}(M)=\{w\in\Sigma^*:(i,w,Z)\rightarrow^*(p,\varepsilon,\varepsilon),\ p\in Q\}$
- $L_{EF}(M)=\{w\in\Sigma^*:(i,w,Z)\rightarrow^*(p,\varepsilon,\gamma),\ p\in T,\ \gamma\in\Gamma^*\}$

# Exemple



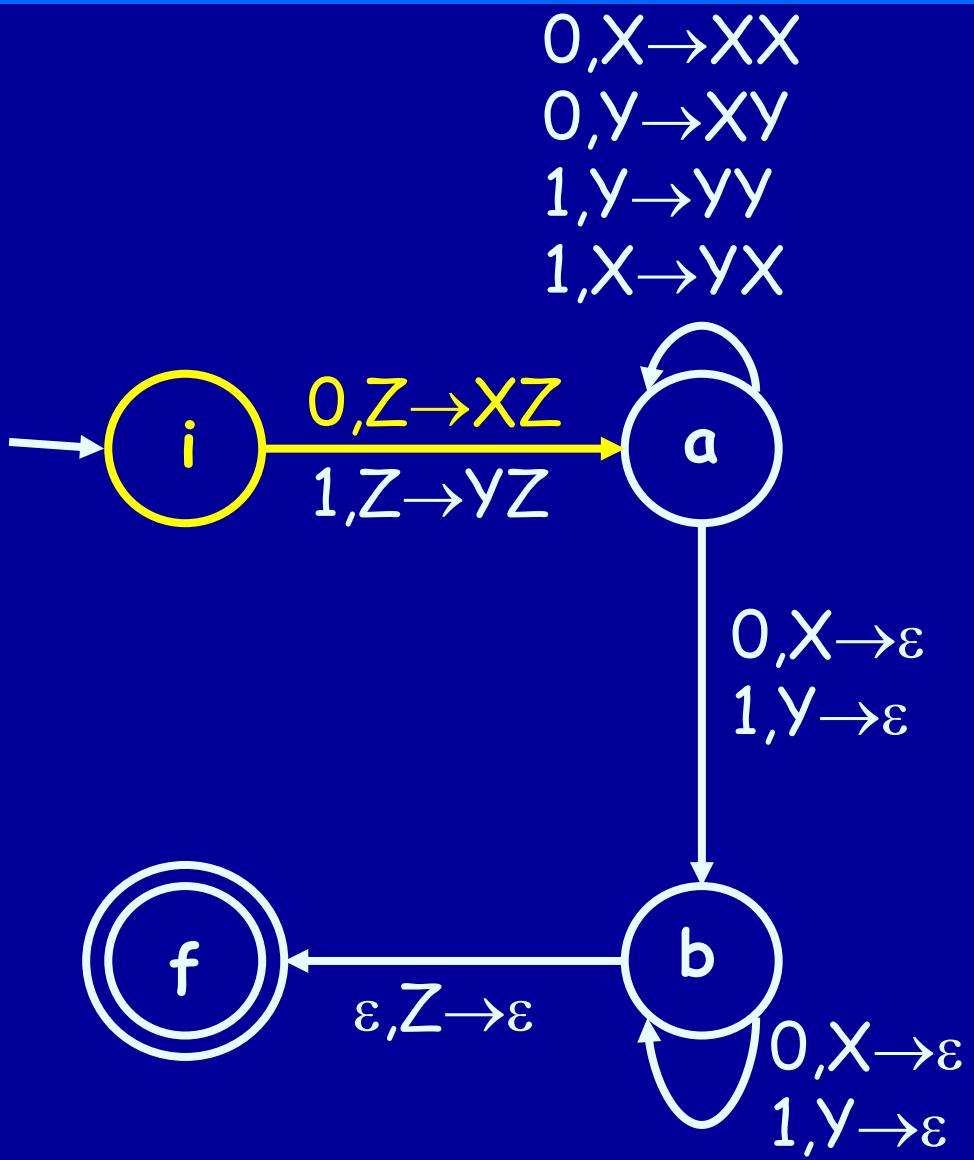

Le langage

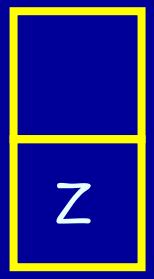
$\{ ww^R : w \in \{0,1\}^+ \}$

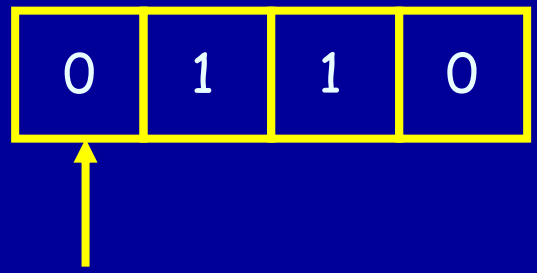
# Exemple

- Le langage { $ww^R : w \in \{0,1\}^+$ }
- Scénario de fonctionnement
  - Sur la première moitié du mot (état moitié 1)
    - Lecture d 'un 0, empiler X, Lecture d 'un 1, empiler Y
  - Sur la seconde moitié (état moitié 2)
    - Lecture d 'un 0, dépiler X, Lecture d 'un 1, dépiler Y
  - Si pile vide accepte
- Le problème est de déterminer le milieu !
  - Résolu par le non déterminisme
    - « On fait » toutes les lectures possibles
    - Si une réussit, on accepte

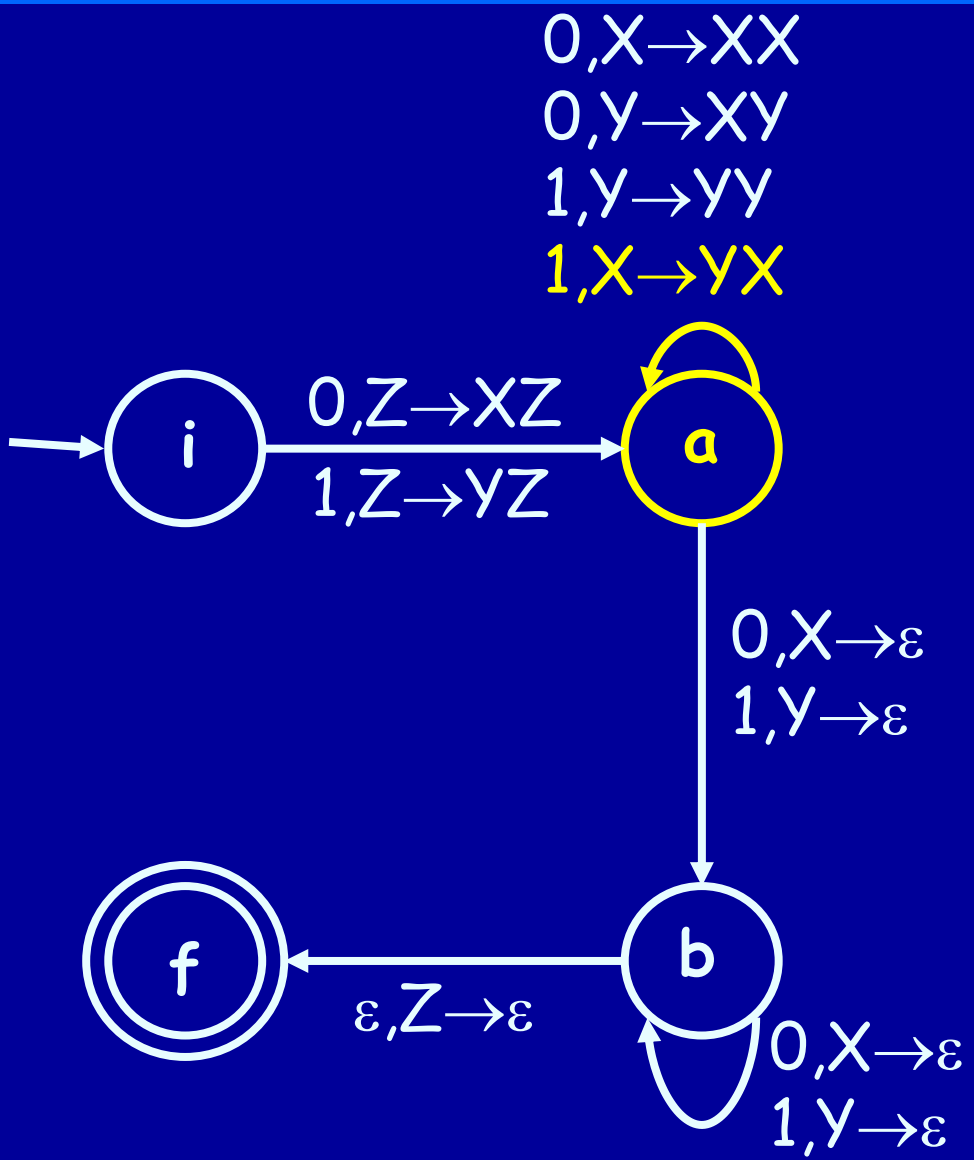
# Exemple



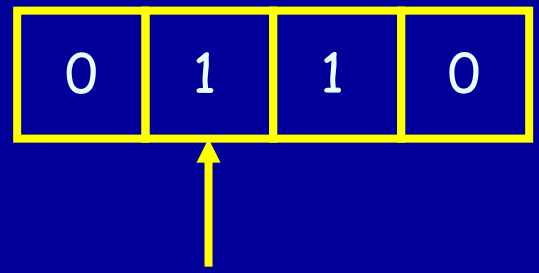



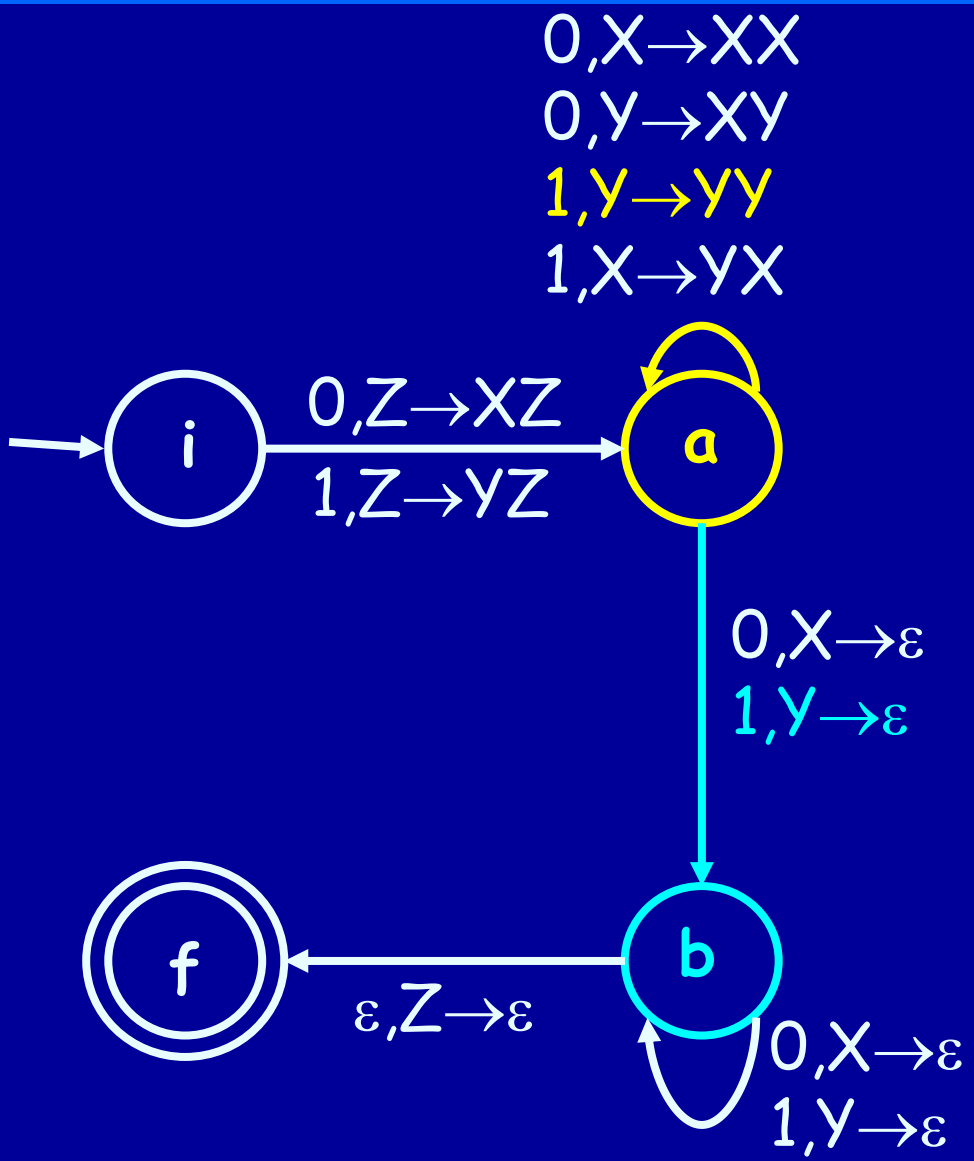
# Exemple

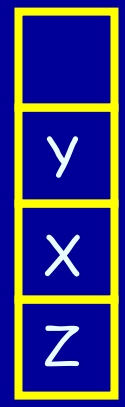


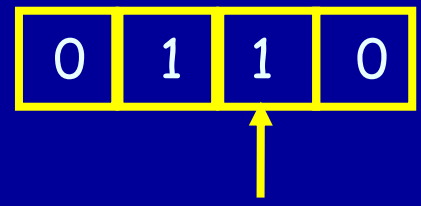


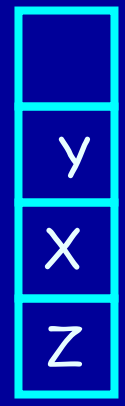


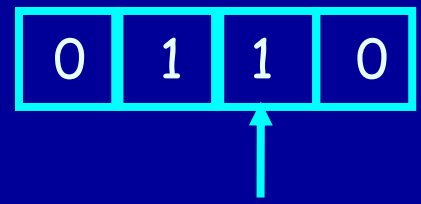
# Exemple


0,X→XX
0,Y→XY
1,Y→YY
1,X→YX
i
0,Z→XZ
1,Z→YZ
a
0,X→ε
1,Y→ε
f
b
ε,Z→ε
0,X→ε
1,Y→ε

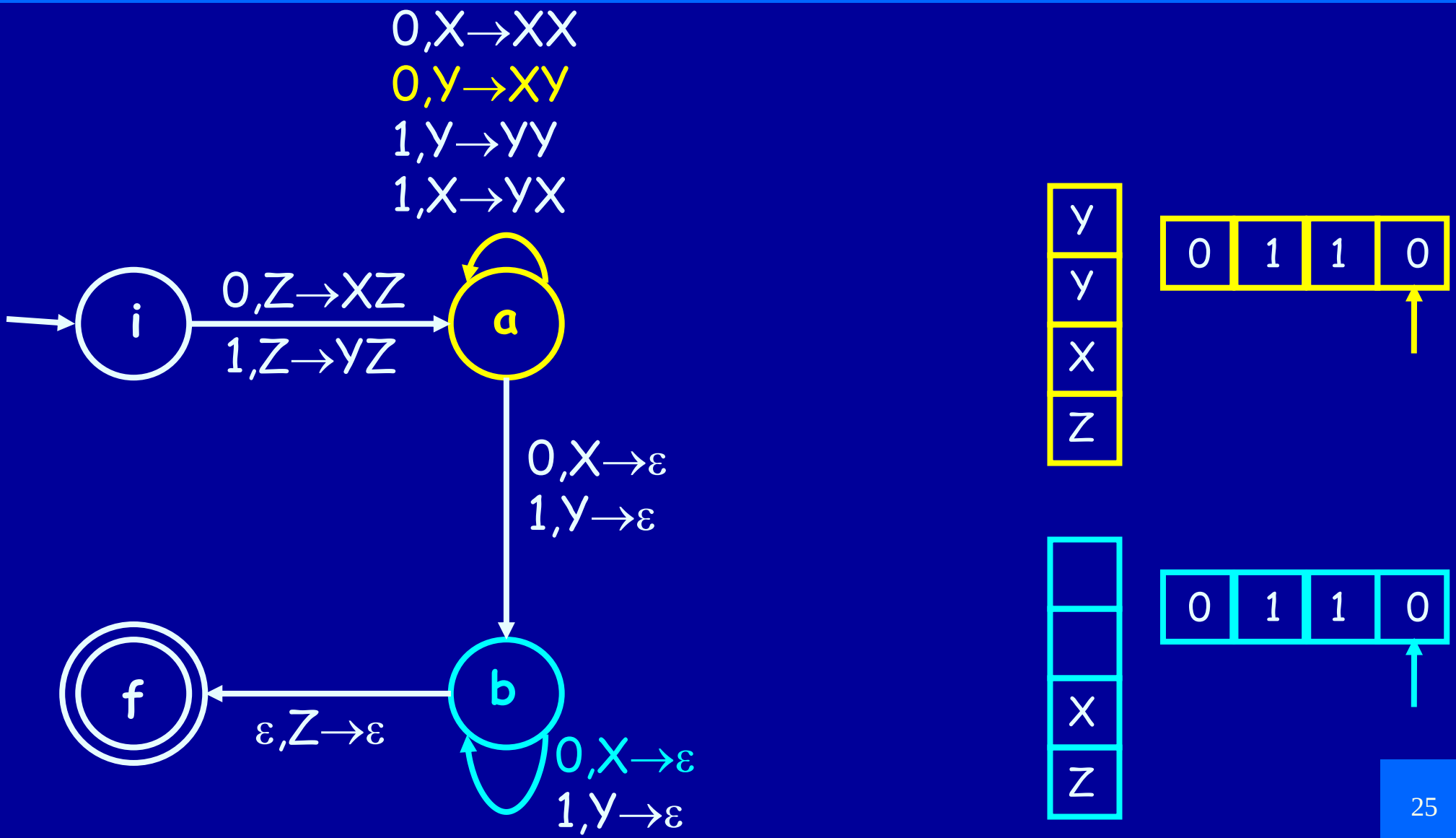

Y
X
Z


0
1
1
0

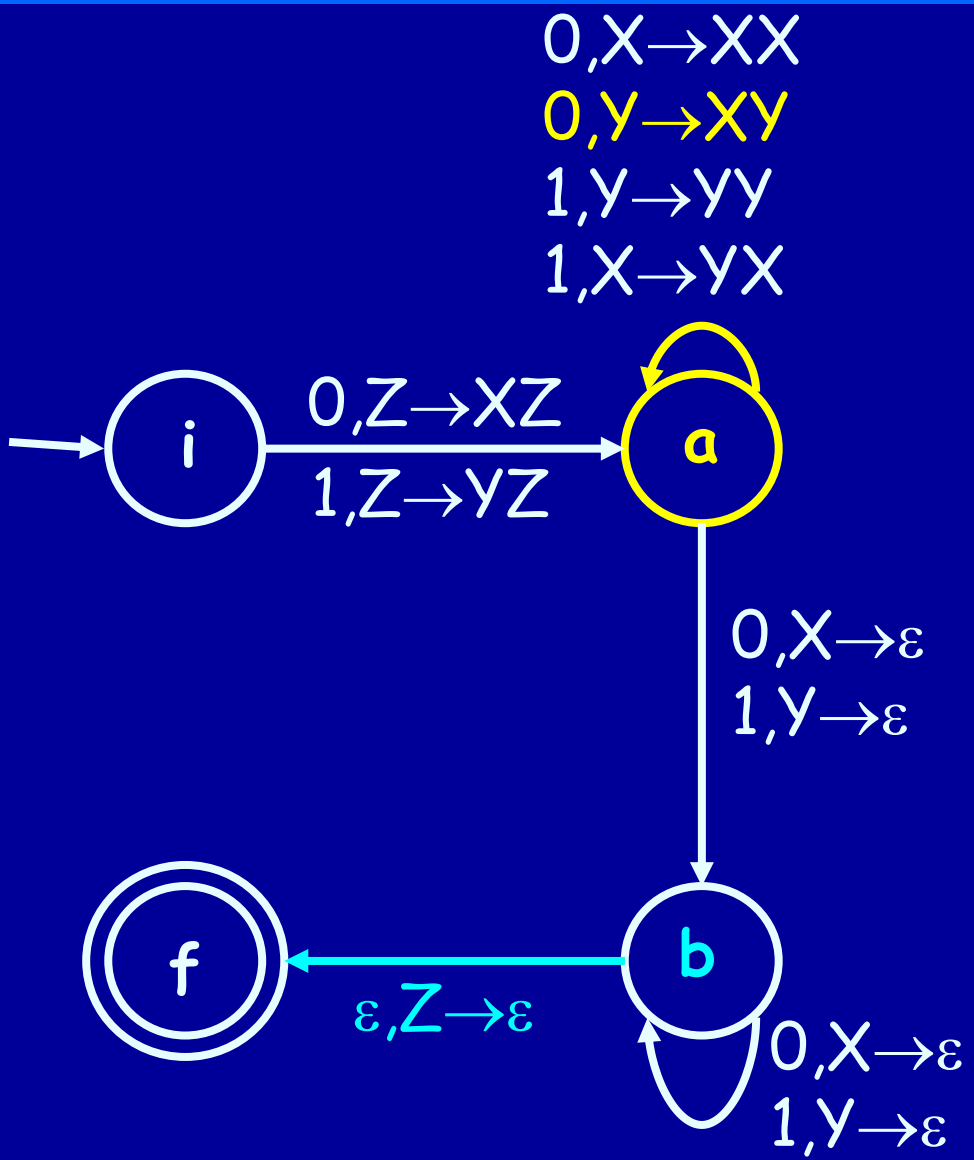

Y
X
Z

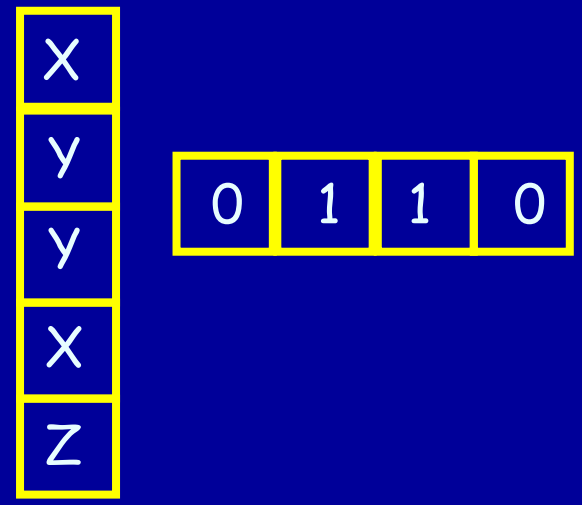

0
1
1
0

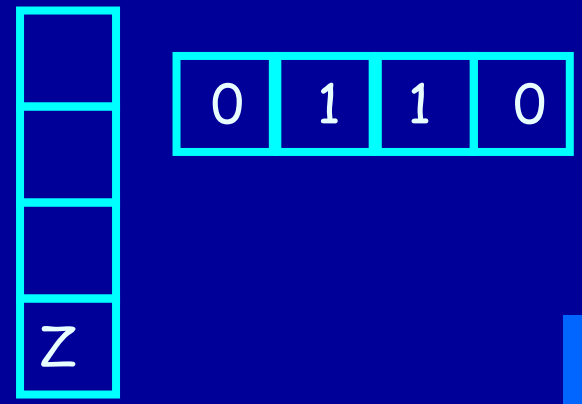
# Exemple


0,X→XX
0,Y→XY
1,Y→YY
1,X→YX
i
0,Z→XZ
1,Z→YZ
a
0,X→ε
1,Y→ε
f
b
ε,Z→ε
0,X→ε
1,Y→ε
Y
Y
X
Z
0 1 1 0
0 1 1 0

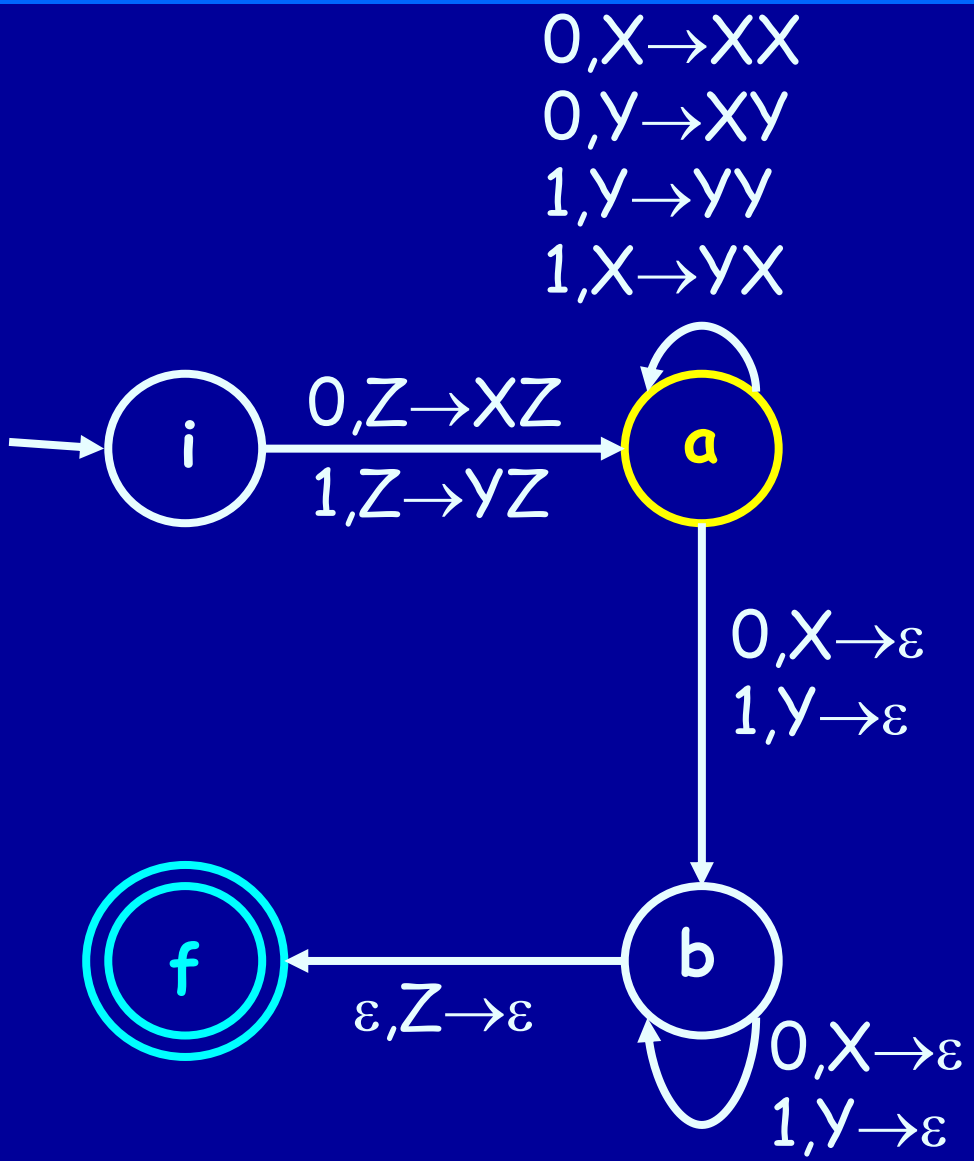
# Exemple

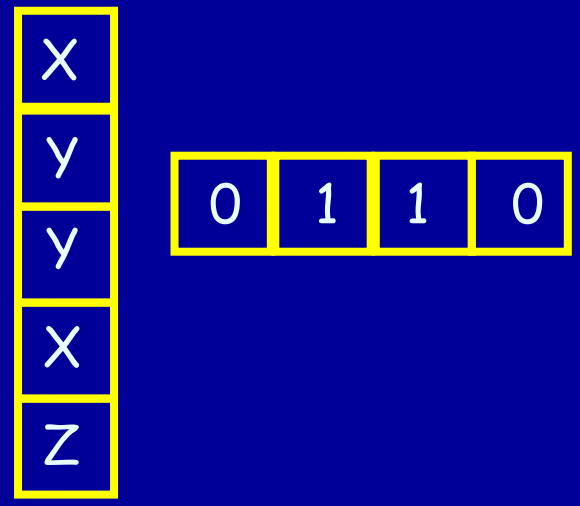

0,X→XX
0,Y→XY
1,Y→YY
1,X→YX
i
0,Z→XZ
1,Z→YZ
a
0,X→ε
1,Y→ε
f
ε,Z→ε
b
0,X→ε
1,Y→ε

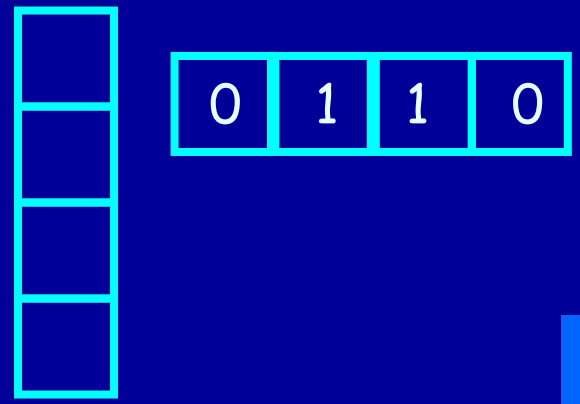

X
Y
Y
X
Z
0 1 1 0


Z
0 1 1 0

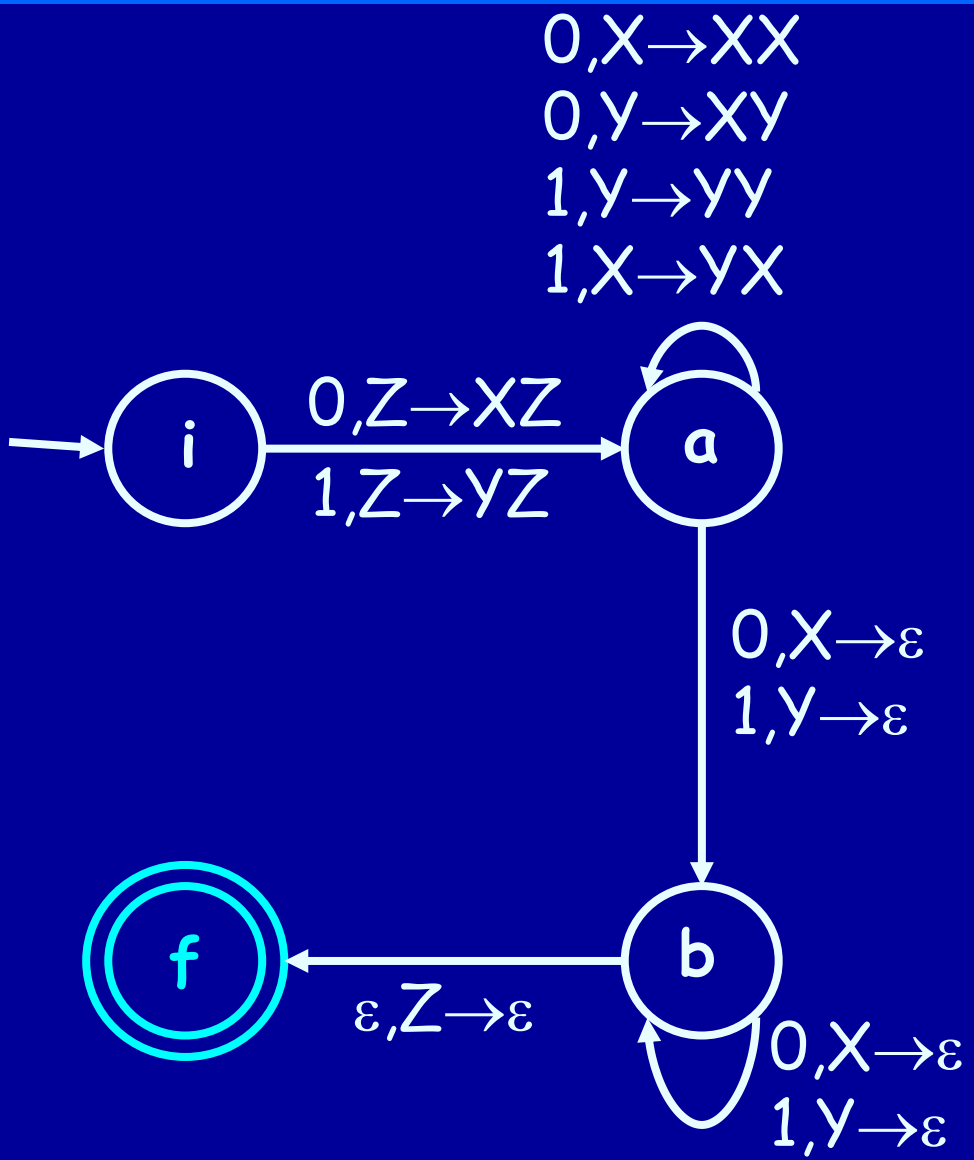
# Exemple

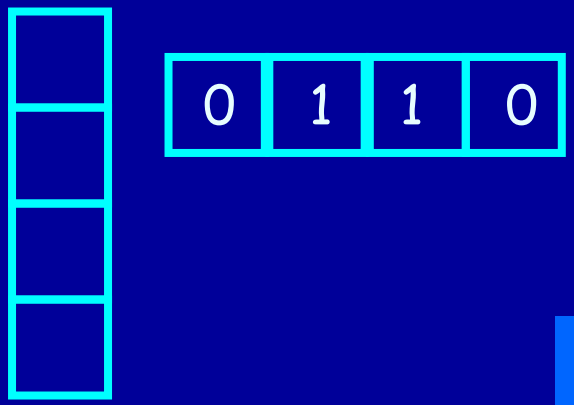

0,X→XX
0,Y→XY
1,Y→YY
1,X→YX
i
0,Z→XZ
1,Z→YZ
a
0,X→ε
1,Y→ε
f
b
ε,Z→ε
0,X→ε
1,Y→ε


X
Y
Y
X
Z
0 1 1 0

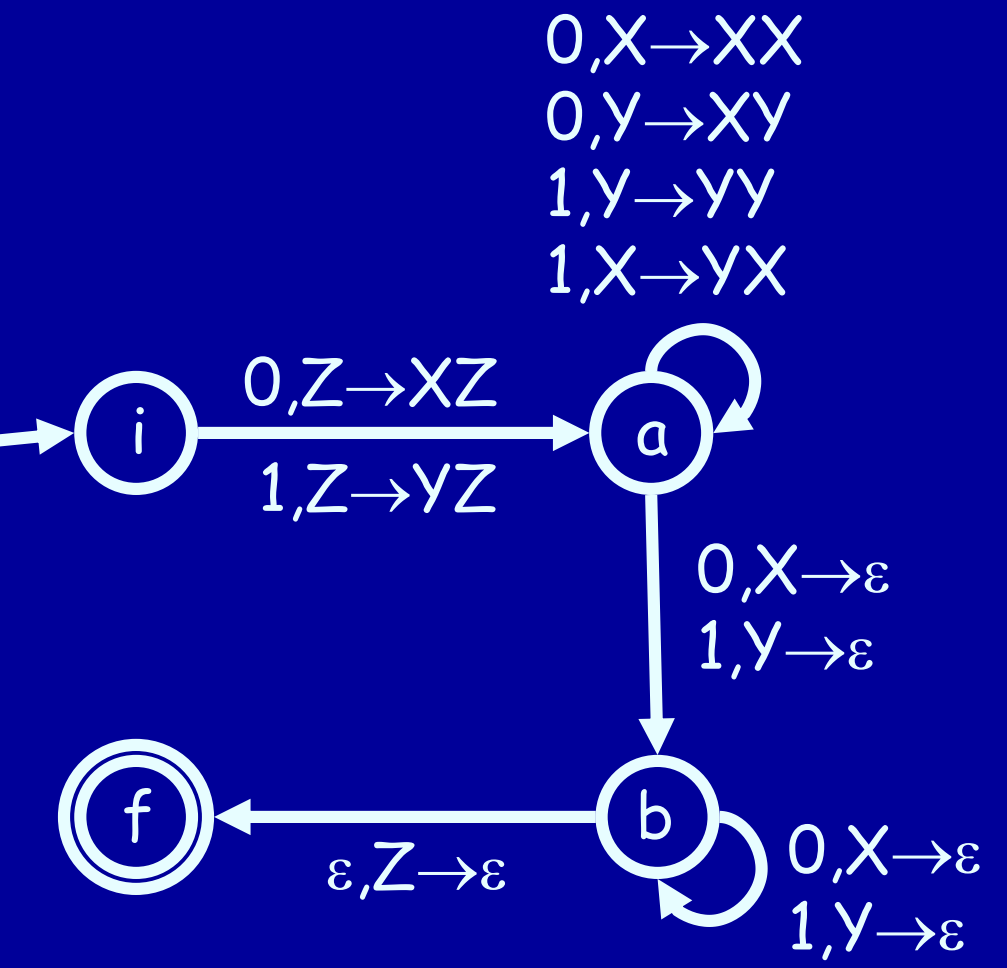

0 1 1 0

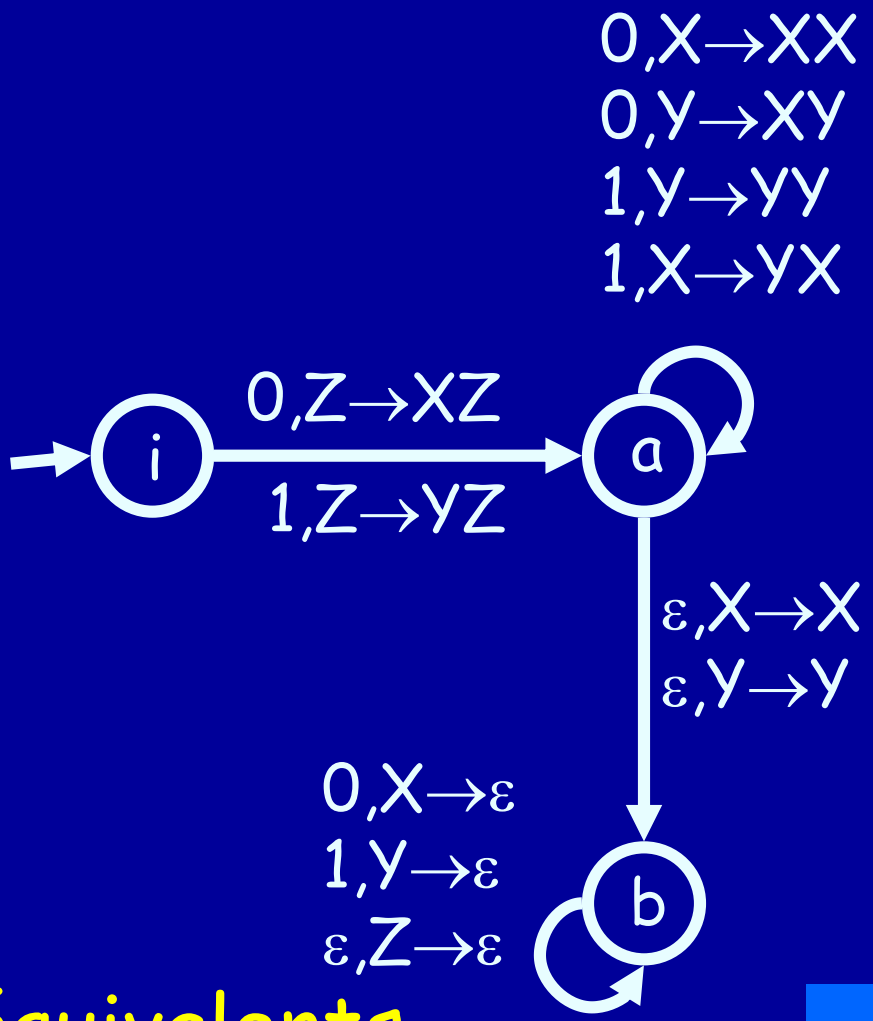
# Exemple



# Exemple




Deux AP équivalents

# Remarque

- Un des AP précédents reconnaît par état final.
- L'autre reconnaît par pile vide.
- Mais tous deux reconnaissent le même langage.

- Ce n'est pas un cas particulier

**Théorème** [PV=EF] : Tout ce qui est reconnu par pile vide est reconnu par état final et réciproquement

# EF→PV

- **Idée** : M' simule le fonctionnement de M jusqu'à la fin. Ensuite, M' termine le travail en vidant tout ce qui reste dans la pile.

$$M=(Q,\Sigma,\Gamma,\delta,i,Z,T)$$

$$M'=(Q\cup\{i',e\},\Sigma,\Gamma\cup\{Z'\},\delta',i',Z',\varnothing)$$

# EF→PV

$$M'=(Q\cup\{i',e\},\Sigma,\Gamma\cup\{Z'\},\delta',i',Z',\varnothing)$$

- État e: sert à effacer le contenu de la pile
- État i': sert à passer dans l'état de M avec un symbole de pile différent au départ Z'
- Z': nouveau symbole de pile pour que M' n'accepte pas accidentellement si M vide sa pile trop tôt

# EF→PV

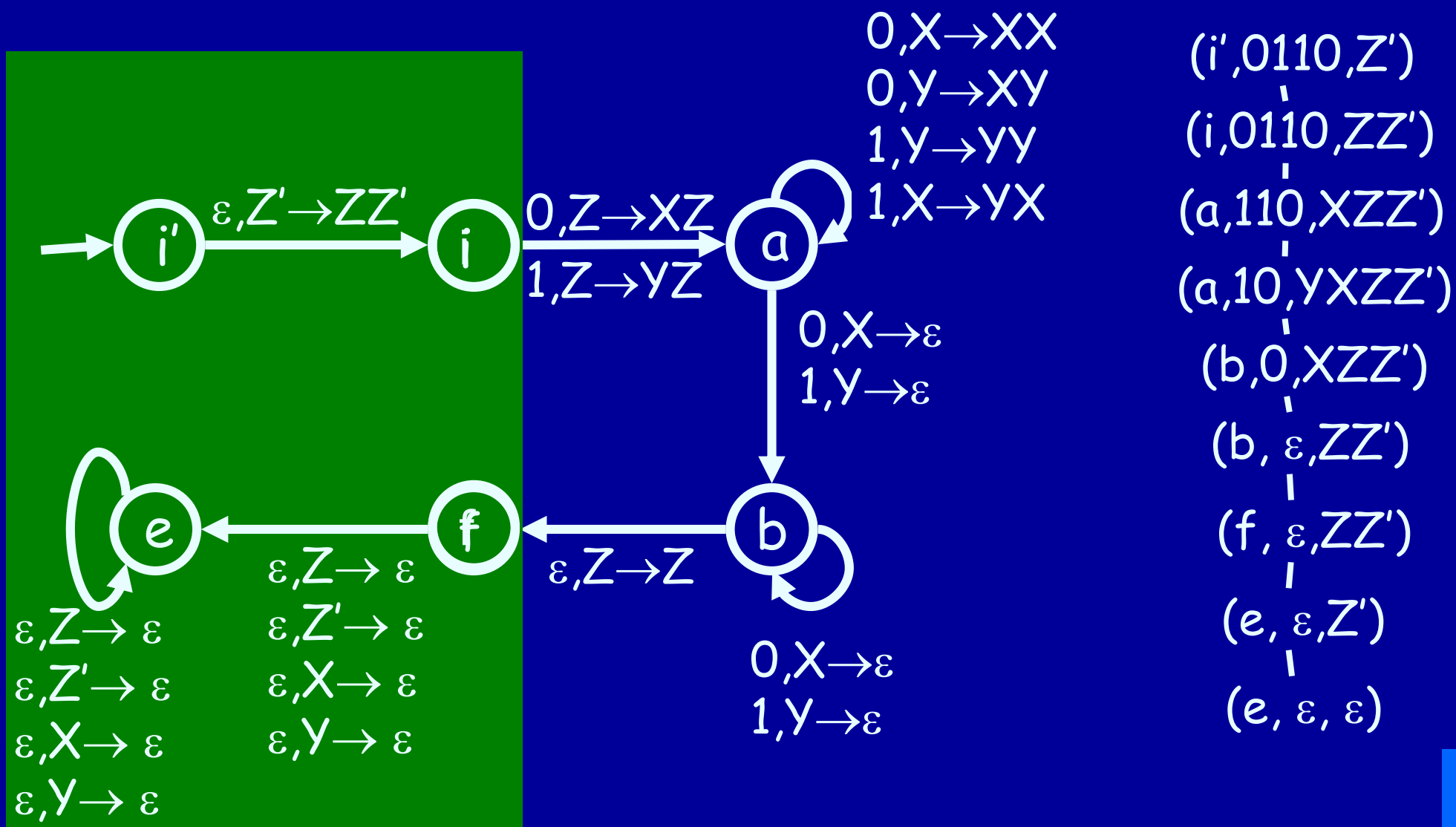
- $\delta'$ simule le fonctionnement de $\delta$ et, de plus
  - $\delta'(i',\varepsilon,Z')=(i,ZZ')$

  initialise M' comme M avec Z' en fond de pile
  - $\forall f\in T,\ \forall\ \gamma\in\Gamma\cup\{Z'\},\ \{(e,\ \varepsilon)\}\in\delta'(f,\varepsilon,\gamma\ )$

  si état final, on <u>peut</u> (ND) passer en effacement
  - $\forall\ \gamma\in\Gamma\cup\{Z'\},\ \delta'(e,\varepsilon,\gamma\ )=(e,\ \varepsilon)$

  effacement de la pile

# Exemple




(i',0110,Z')
(i,0110,ZZ')
(a,110,XZZ')
(a,10,YXZZ')
(b,0,XZZ')
(b, ε,ZZ')
(f, ε,ZZ')
(e, ε,Z')
(e, ε, ε)

# PV→EF

- Idée : l'automate à état final M' doit simuler le fonctionnement de l 'automate à pile vide M et détecter quand M efface le contenu de la pile. Il doit alors entrer dans un état final.

$M=(Q,\Sigma,\Gamma,\delta,i,Z,\varnothing)$

$M'=(Q\cup\{i',f\},\Sigma,\Gamma\cup\{Z'\},\delta',i',Z',\{f\})$

# PV→EF

$$M'=(Q \cup \{i',f\},\Sigma,\Gamma \cup \{Z'\},\delta',i',Z',\{f\})$$

- État f: état final de M'
- État i': sert à passer dans l'état de M avec un symbole de pile différent au départ Z'
- Z': nouveau symbole de pile pour que M' puisse passer dans son état final à l 'apparition de Z' et accepter l 'entrée.

# PV→EF

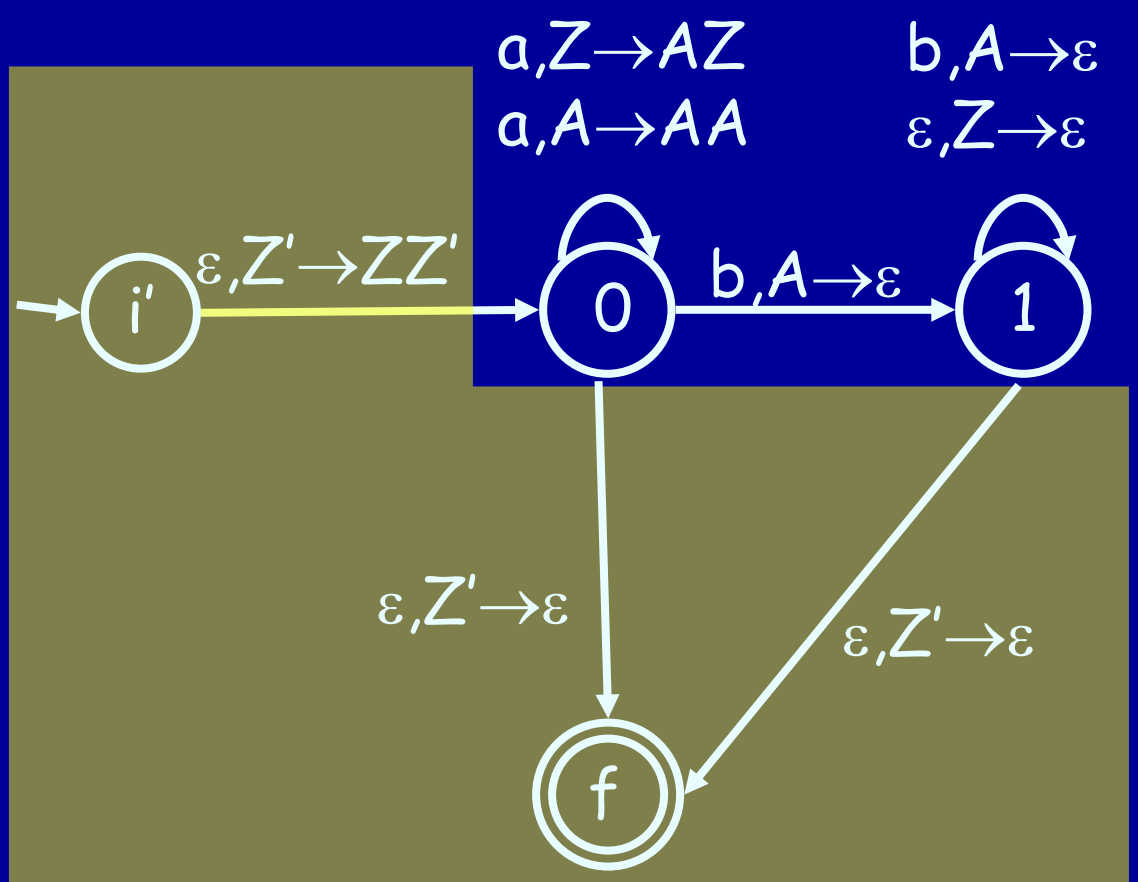
- $\delta'$ simule le fonctionnement de $\delta$ et, de plus
  - $\delta'(i',\varepsilon,Z')=(i,ZZ')$

  Initialise M' comme M avec Z' en fond de pile

  - $\forall q \in Q, \{(f,\varepsilon)\} \in \delta'(q,\varepsilon,Z')$

  *À l'apparition du symbole de fond de pile ajouté, M' peut alors passer dans son état final f et accepter l 'entrée. On ne considère que les états de M et non ceux de M'.*

# Exemple de $\{a^n b^n : n>0\}$




~~Reconnaît par pile vide~~

Reconnaît par état final

(i',aabb,Z')

(0,aabb,ZZ')

(0,abb,AZZ')

(0,bb,AAZZ')

(1,b,AZZ')

(1,ε,ZZ')

(1,ε,Z')

(f,ε,ε)