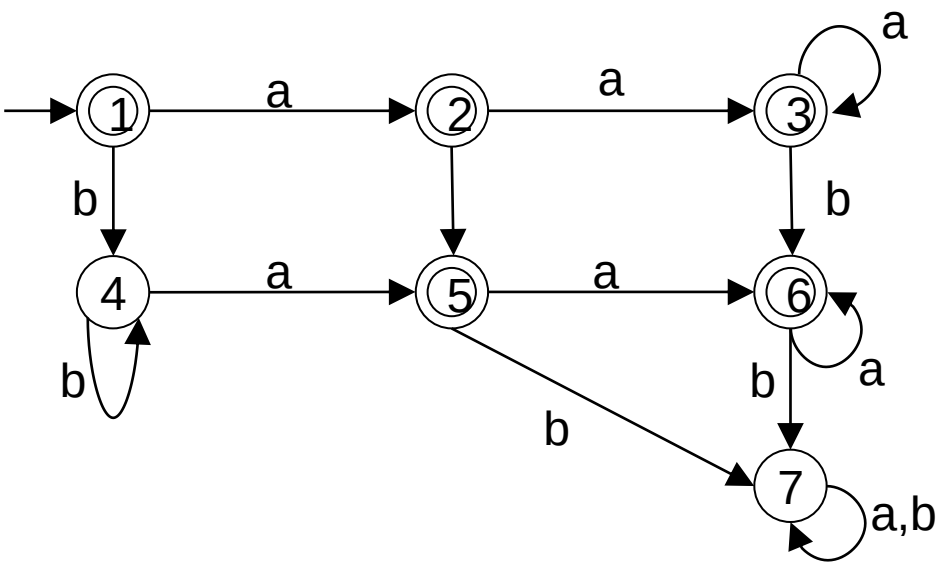
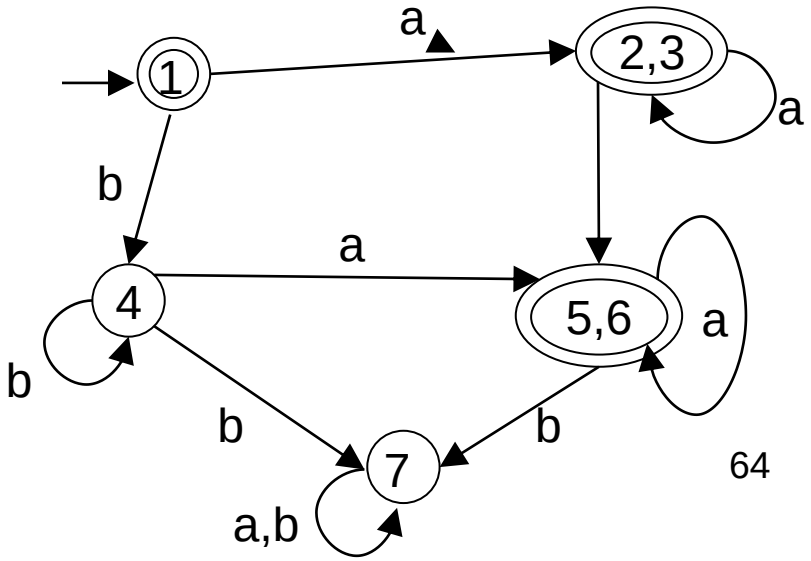
Minimisation d'AEF: Exemple



≡0	≡0,1	≡0,1,2	≡0,1,2,3
1 2 3 5 6	2 3	2 3	2 3
	1 5 6	1	1
		5 6	5 6
4 7	4	4	4
	7	7	7

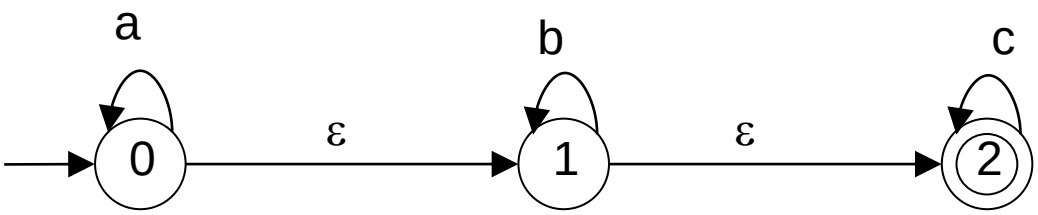


Automates finis avec ε-transitions

- Un **ε-AFN** est un quintuplet A = (Q,Σ,δ,q0, F) où:
  - Q est l'ensemble fini d'états
  - Σ est un vocabulaire (d'entrée) fini
  - δ **: Q × Σ∪{ε} → $\mathcal{P}$(Q)** est une fonction dite de transition
  - q0∈ Q est l'état initial
  - F ⊆ Q est l'ensemble des états terminaux (ou d'acceptation)

Automates finis avec ε-transitions

- Exemple



- **Configuration** : couple (q, w) avec $q \in Q$ et $w \in \Sigma^*$ (q représente l'état courant et w le mot qui reste à lire sur le ruban)(sans changement)

- **Succession immédiate** : (modifiée)

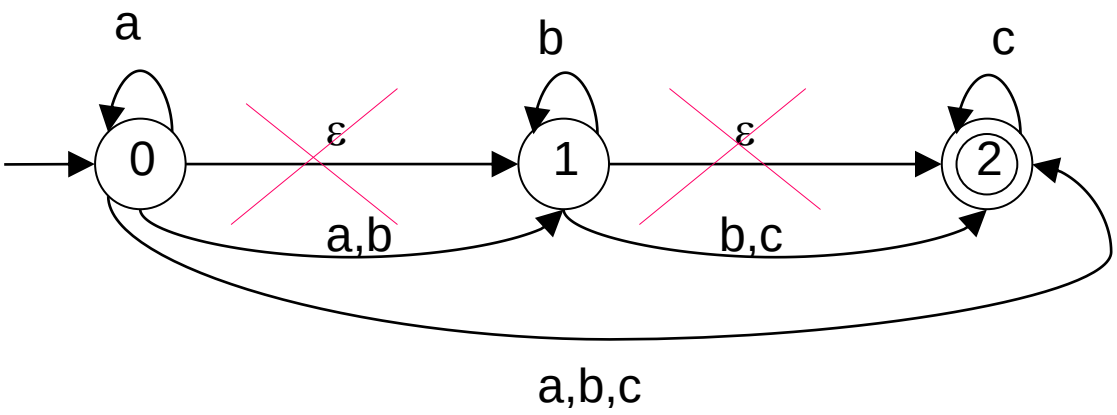
(q',w') peut suivre immédiatement (q,w):

(q, w) ➔ (q', w')

si et seulement si

  - w = vw' avec $v \in \Sigma \cup \{\varepsilon\}$ (si $v=\varepsilon$ alors w = w')
  - (q, v, q') $\in R$

Elimination des ε-transitions



Elimination des ε-transitions

- Soit A=(Q,Σ,δ,q0, F) un ε-AFND. On construit un automate ε-el(A)=(Q,Σ, ε-el(δ),q0, ε-el(F)) qui reconnaît L(A).

- On définit inductivement la relation $\Rightarrow$:
  - q $\Rightarrow$ q et
  - Si q $\Rightarrow$ q' et q'' $\in$ δ(q', ε) alors q $\Rightarrow$ q''

- On définit ε-el(δ) par:
  q' $\in$ δ(q, a) ssi il existe q1 q2 $\in$ Q tels que q2 $\in$ δ(q1, a) et q2 $\Rightarrow$ q'

- L'ensemble des états accepteurs ε-el(F) est défini par:
  - ε-el(F)=F$\cup${q0} si q0 $\Rightarrow$ q' $\in$F
  - ε-el(F)= F sinon.

Déterminisation des ε-AFN

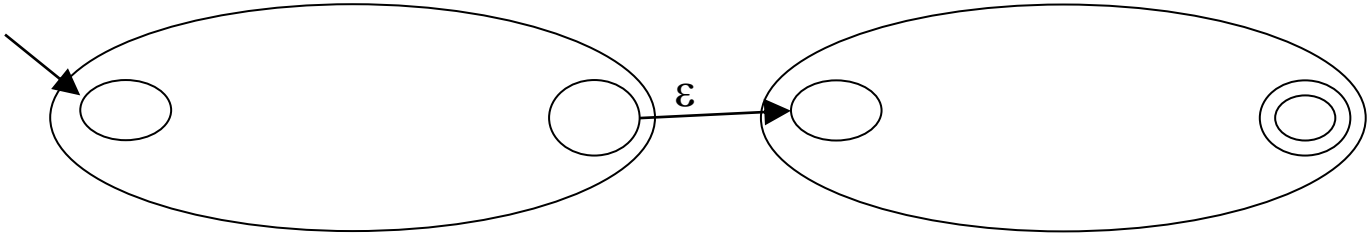
- **ε-fermeture**:

  On appelle ε-fermeture de l'ensemble d'états T= {e_1,e_2, ….,e_n}, l'ensemble des états accessibles depuis un état e_i de T par des ε-transitions

- **Principe de déterminisation**:
  1. Partir de l' ε-fermeture de l'état initial
  2. Rajouter dans la table de transition toutes les ε-fermetures des nouveaux états produits avec leurs transitions
  3. Recommencer 2 jusqu'à ce qu'il n'y ait plus de nouveaux états
  4. Tous les états contenant au moins un état terminal deviennent terminaux

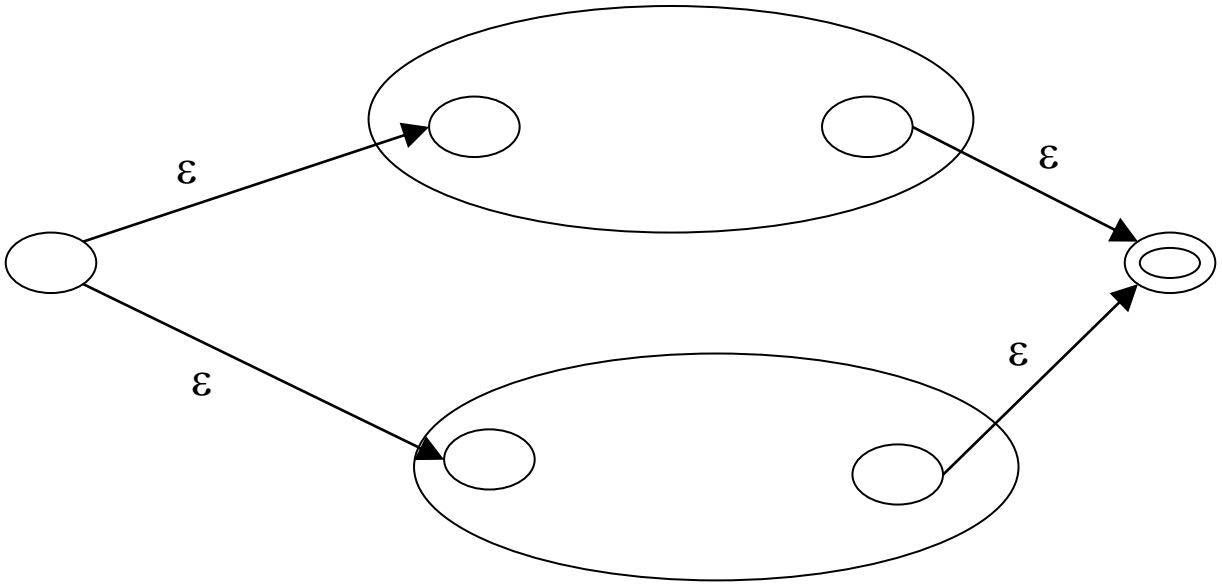
Propriétés de clôture

- Fermeture par concaténation



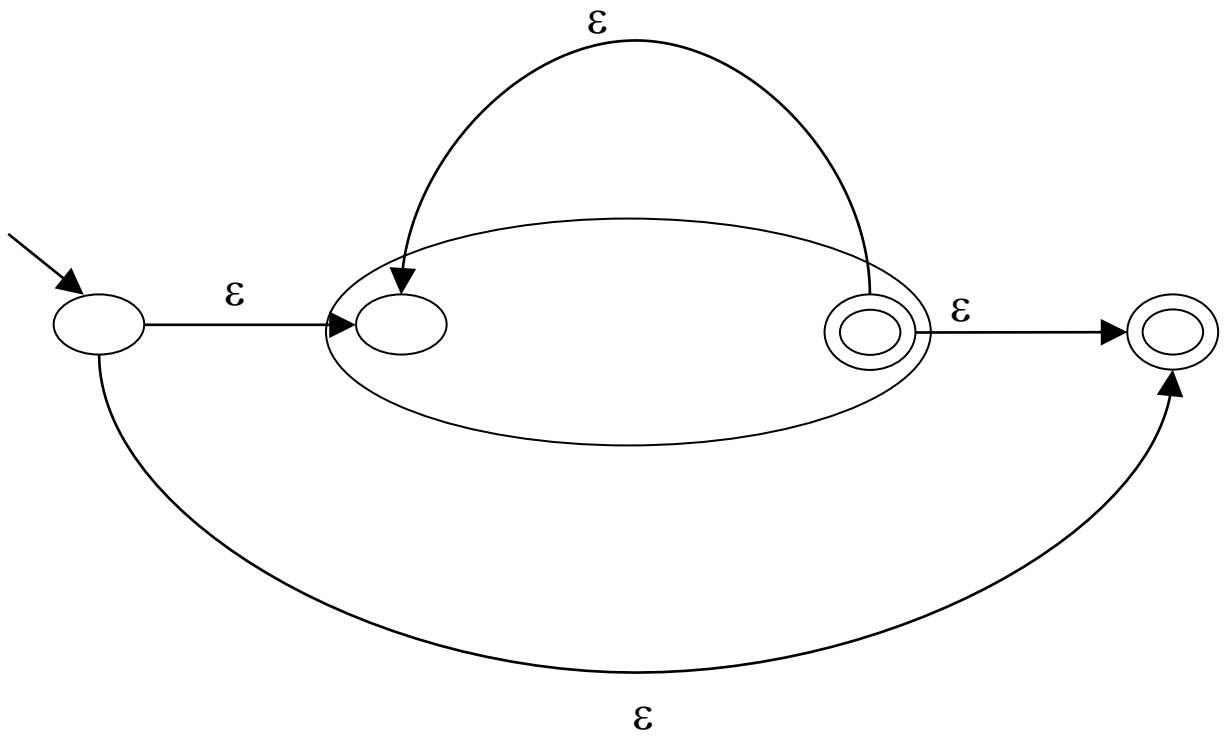
Propriétés de clôture

- Fermeture par union



Propriétés de clôture

- Fermeture par l'opération *



Propriétés de clôture

- Fermeture par complémentation
  - On complète l'automate sur son alphabet Σ
  - On inverse les états accepteurs et non accepteurs

Langages réguliers

Expressions régulières

- Les **expressions régulières** servent à **désigner** les langages réguliers
  - $\varnothing$, ε et toute lettre $x \in A$ sont des expressions régulières sur A,
  - Si α et β sont des expressions régulières sur A, alors:
    - $\alpha+\beta$ et $\alpha\beta$ sont des expressions régulières sur A
    - $(\alpha)^*$ est aussi une expression régulière sur A

Relation de désignation

- $\varnothing$ désigne $\varnothing$,
- ε désigne $\{\varepsilon\}$
- et toute lettre $x \in A$ désigne $\{x\}$
- Si α désigne A et β désigne B, alors
  - $\alpha+\beta$ désigne $A \cup B$
  - $\alpha\beta$ désigne A.B
  - $(\alpha)^*$ désigne $(A)^*$

Notation: Si e est une expression régulière, on note par L(e) le langage désigné par e

Relation de désignation

Exemple:

Ensemble des mots alternant des 0 et des 1

(ε+ 1)(01)*(ε+ 0)

Langages réguliers

- Un langage L est **régulier** ssi il existe une expression régulière e telle que L(e)=L
- **Théorème de Kleene**:

  Soit Σ un alphabet et L $\subset \Sigma^*$

  L est régulier ssi L est reconnu par un automate d'états fini.

  Preuve:

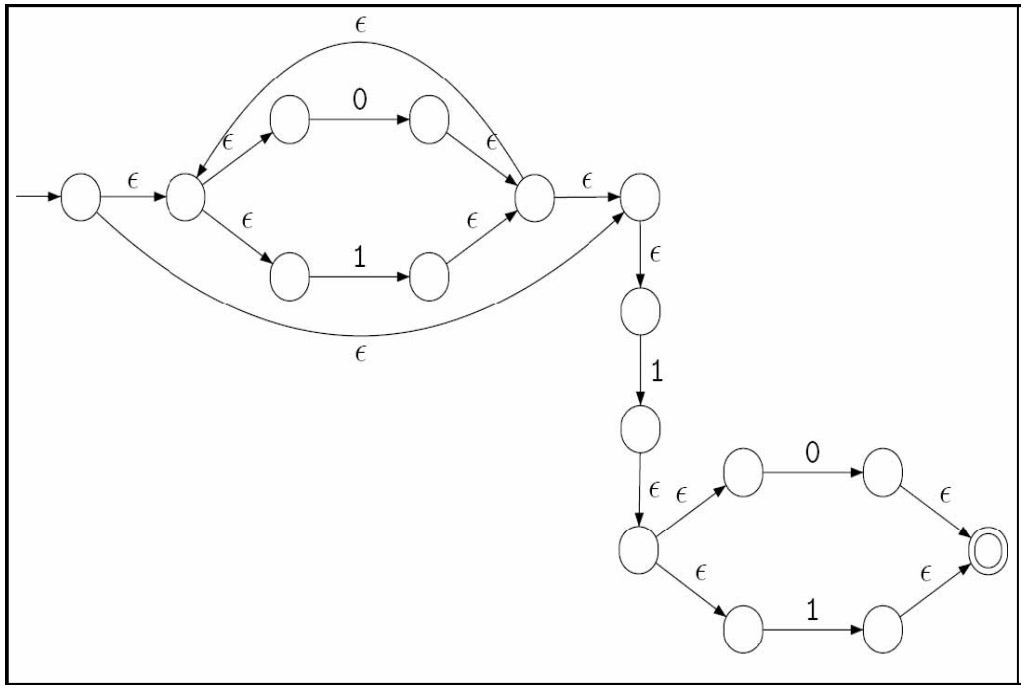
  – Il existe un algorithme qui transforme une expression régulière en un automate fini équivalent (propriétés de clôture des automates)

  – Il existe un algorithme qui transforme un automate fini en une expression régulière équivalente.) (lemme d'Arden)

Construction d'un AFN à partir d'une expression régulière

- Automate associé à l'expression régulière :

  (0 + 1)*1(0 + 1)



Propriétés des langages réguliers

Théorème

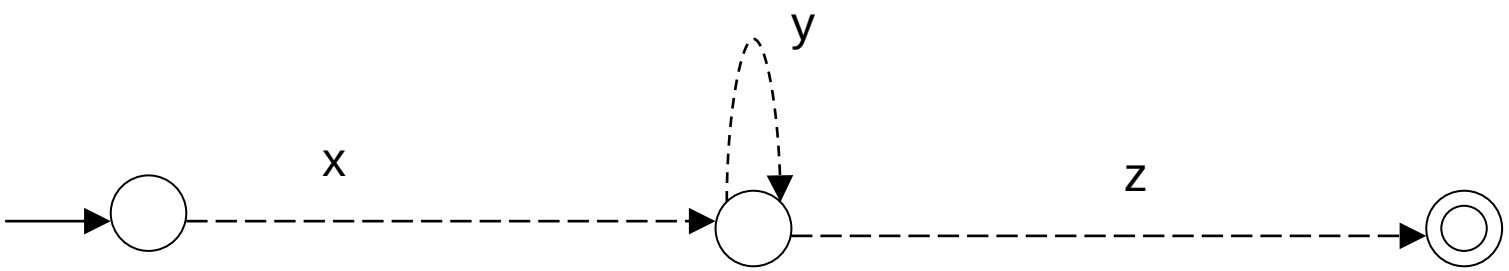
L'ensemble des langages réguliers est fermé par:

- la réunion
- l'intersection
- la complémentation
- l'image miroir
- l'opération *
- l'opération +

Lemme de pompage

Soit L un langage régulier. Alors, il existe $n \in N$ tel que pour tout mot $w \in L$ avec $|w| \geq n$, on peut trouver $x,y,z \in \Sigma^*$ tels que $w=xyz$ et

- $y \neq \varepsilon$
- $|xy| \leq n$
- Pour tout $k \in N$, $xy^kz \in L$



Limites des langages réguliers

- Il existe des langages non réguliers
- On utilise le lemme de pompage pour montrer par l'absurde, que $\{a^n b^n | n \geq 0\}$ est non régulier

Grammaires

Grammaires

- Une grammaire G est un quadruplet <V, Σ, S, R> où:
  - V : **vocabulaire non terminal**
  - Σ : **vocabulaire terminal**
  - S$\in$ V : **axiome** ou **symbole initial**
  - R : **règles** (ou **productions**)

- Une règle est un couple (α,β) qu'on note en général :

$$\alpha \rightarrow \beta$$

où : $\alpha \in (V \cup \Sigma)^* - \{\varepsilon\}$

et $\beta \in (V \cup \Sigma)^*$

Grammaires

Exemple1

- $V = \{S, A, B\}$
- $\Sigma = \{0, 1\}$
- $S \in V$: axiome
- R :

  $S \rightarrow 0A1B$

  $1B \rightarrow 1ABB$

  $1A \rightarrow A1$

  $1B \rightarrow 11$

  $0A \rightarrow 00$

Grammaires

Exemple2

- V = { <prop.>, <impl.>, <terme>, <fact.>, <prop. sec.>, <prop. pri.>}
- $\Sigma = \{(,)\} \cup \{p, q, r , . . .\} \cup \{\neg,\leftrightarrow,\rightarrow,\wedge,\vee\}$
- R est donné sous la forme BNF suivante :

  <prop.> ::= <impl.> | <prop.> $\leftrightarrow$ <impl.>

  <impl.> ::= <terme> | <impl.> $\rightarrow$ <terme>

  <terme> ::= <fact.> | <terme> $\vee$ <fact.>

  <fact.> ::= <prop. sec.> | <fact.> $\wedge$ <prop. sec.>

  <prop. sec.> ::= <prop. prim.> | $\neg$ <prop. prim.>

  <prop. prim.> ::= (<prop.>) | p (avec p $\in$ P)

Hiérarchie de Chomsky

type	restrictions
Type 3 ou linéaire à droite	Si toutes les règles sont de la forme: $A \rightarrow aB$ ou $A \rightarrow a$ ou $A \rightarrow \varepsilon$
Type 2 ou hors-contexte	Si toutes les règles sont de la forme $A \rightarrow \alpha$
Type 1 ou sous-contexte	Pour tout $\alpha \rightarrow \beta$, avec $\lvert\alpha\rvert \leq \lvert\beta\rvert$
Type 0 ou générale	Pour tout $\alpha \rightarrow \beta$

Dérivations

- Un mot y **dérive immédiatement** d'un mot x si et seulement si il existe une règle r: $\alpha \rightarrow \beta$ et deux mots g et d de V* tels que :

$$x = g\ \alpha\ d \quad \text{et} \quad y = g\ \beta\ d$$

On le note: $x \Rightarrow_G y$

(ou $x \Rightarrow y$ quand il n'y a pas d'ambiguïté)

Dérivations

- Une dérivation est l'utilisation de la fermeture réflexive et transitive de $\Rightarrow$, notée $\Rightarrow^*$
- $\Rightarrow^+$ est la fermeture transitive non réflexive de $\Rightarrow$
- dérivation à gauche : dériver en premier le non-terminal le plus à gauche
- dérivation à droite : dériver en premier le non-terminal le plus à droite

Dérivations

Exemple:

Soit G = ({S}, {0, 1}, S, {(S $\rightarrow$0S1), (S $\rightarrow$ 01)}).

S $\Rightarrow$ 0S1 $\Rightarrow$00S11 $\Rightarrow$ 000S111 $\Rightarrow$ 000111

Donc S $\Rightarrow$* 000111

Remarque: S $\Rightarrow$* S (0 pas)

Langage engendré par une grammaire

- Le langage engendré par une grammaire G, noté L(G), est l'ensemble des mots terminaux dérivant de S. Formellement :

  $L(G) = \{x \in V_T^* | S \Rightarrow_G^* x\}$

- Exemple:

  Soit G = ({S}, {0, 1}, S, {(S, 0S1), (S, 01)}).

  On montre que: $L(G) = \{0^n 1^n | n \in N\}$

- Deux grammaires G et G_0 sont dites équivalentes si et seulement si elles engendrent le même langage:

  $L(G) = L(G_0)$

Typologie des langages

- Un langage $L \subseteq T^*$ est de type i s'il existe une grammaire G=(N,T,P,S) de type i avec L=L(G)

- **Théorème**

  Soit T_i l'ensemble des langages de type i

$$T_3 \subseteq T_2 \subseteq T_1 \subseteq T_0$$

Typologie des langages

- $\{a^n b^n c^n | n \in N\}$ est de type 1

  $S \rightarrow aAbc | \varepsilon$

  $A \rightarrow aAbC | \varepsilon$

  $Cb \rightarrow bC$

  $Cc \rightarrow cc$

- $\{a^n b^n | n \in N\}$ est de type 2

  $S \rightarrow aSb | \varepsilon$

- $\{a^n b^m | n,m \in N\}$ est de type 3

  $S \rightarrow aA | bB | \varepsilon$

  $A \rightarrow aA | \varepsilon$

  $B \rightarrow bB | \varepsilon$

Typologie des langages

Exemple 1

$S \rightarrow aS$
$S \rightarrow aA$
$A \rightarrow bA$
$A \rightarrow b$

S => aA => ab
S => aS => aaS => aaaS => aaaaA => aaaab
S => aS => aaA => aab
S => aS => aaA => aabA => aabbA => aabbb

$L(G) = \{a^n b^m;\ n,m \geq 1\} = L(aa^*bb^*)$

Typologie des langages

Exemple 2:

$S \rightarrow aS$
$S \rightarrow bA$
$S \rightarrow \varepsilon$
$A \rightarrow bA$
$A \rightarrow \varepsilon$

S => aS => a; S => aS => aaS => aa
S => aS => aaS => aaaS => aaabA => aaab
S => aS => aaS => aaaS => aaabA => aaabbA => aaabb

L(G) = L(a*b*)

Equivalence grammaire régulière et AFN

- **Théorème**

  Un langage est régulier si et seulement s'il est généré par une grammaire régulière.

- Preuve:
  - Conversion grammaire régulière $\rightarrow$ automate
  - Conversion automate $\rightarrow$ grammaire régulière